Clojush Tutorial

Lee Spector, lspector@hampshire.edu

This tutorial shows how to use Clojush, a Clojure implementation of the Push programming language and the PushGP genetic programming system, as a library in your own projects. To download the full system and for instructions for running the examples that are included with it, see the Clojush github page.

This tutorial was created with Gorilla REPL. To run this worksheet, or to use the code that it contains in some other worksheet or project, you must include the Clojush dependency in your project's `project.clj` file. This tutorial was created with the dependency of [clojush "2.0.60"].

First, we create a namespace for the code that we will run:

```
(ns tutorial-worksheet)
```

```
nil
```

These expressions provide access to *all* Clojush namespaces:

```
(use 'clojush.ns)
(use-clojush)
```

```
nil
```

Clojush is an implementation of the Push programming for evolutionary computation, and of the PushGP genetic programming system that evolves Push programs.

Push programs operate on data stacks, with a separate stack for each data type.

A "Push state" is a complete set of data stacks, and the `make-push-state` function returns a Push state with all stacks empty:

```
(make-push-state)
```

```
{:exec nil, :code nil, :integer nil, :float nil, :boolean
nil, :char nil, :string nil, :zip nil, :vector_integer
nil, :vector_float nil, :vector_boolean nil,
:vector_string nil, :input nil, :output nil, :auxiliary
nil, :tag nil, :return nil, :environment nil, :genome
nil}
```

What you see here are all of the data stacks included by default in the current version of Clojush.

You can ignore most of them for now. The `:exec` stack is the most important one, since it stores a program while it is being run.

For the sake of this tutorial we'll only be using `:exec` and a few others, so here's a function that shows us just the ones we want:

```
(defn stks
  "Return a map with just a few of the stacks in a Push
state."
  [push-state]
  (select-keys push-state
               [:exec :code :integer :float :boolean]))
```

```
#'tutorial-worksheet/stks
```

```
(stks (make-push-state))
```

```
{:exec nil, :code nil, :integer nil, :float nil, :boolean
nil}
```

Now let's actually run a Push program.

Push is a stack-based language, which means that when the interpreter sees a value it just pushes it on the appropriate stack, and when it sees an instruction it pops any needed arguments from the stacks, executes the instruction, and pushes results back on the appropriate stacks.

Here we run the Push program `(1 2 integer_add)` on an initially empty Push state and return the resulting Push state (with just the stacks we care about):

```
(stks (run-push '(1 2 integer_add)
                (make-push-state)))
```

```
{:exec (), :code nil, :integer (3), :float nil, :boolean
nil}
```

You can see that `3` is on the `:integer` stack at the end because `1+2=3`, and because the program `(1 2 integer_add)` essentially says to first push `1` onto the `:integer` stack, then `2`, and then execute the `integer_add` instruction, which pops two integers and then pushes their sum back on the `:integer` stack.

If you give `run-push` a `true` third argument then it will print the (complete) Push state at each execution step. For this simple example we first see the entire program pushed on to the `:exec` stack. Then we see that the interpreter processes the program list (as it processes all lists) by pushing its elements individually back onto the `:exec` stack (notice that a layer of parentheses is removed in the first step). Then we see the `1` being processed (notice that it moves to the `:integer` stack), and then the `2`, and finally `integer_add`. Then the `:exec` stack is empty and execution terminates:

```
(stks (run-push '(1 2 integer_add)
                (make-push-state)
                true))
```

```
State after 0 steps:
:exec = ((1 2 integer_add))
:code = nil
:integer = nil
:float = nil
:boolean = nil
```

```
:char = nil
:string = nil
:zip = nil
:vector_integer = nil
:vector_float = nil
:vector_boolean = nil
:vector_string = nil
:input = nil
:output = nil
:auxiliary = nil
:tag = nil
:return = nil
:environment = nil
:genome = nil

State after 1 steps (last step: (...)):
:exec = (1 2 integer_add)
:code = nil
:integer = nil
:float = nil
:boolean = nil
:char = nil
:string = nil
:zip = nil
:vector_integer = nil
:vector_float = nil
:vector_boolean = nil
:vector_string = nil
:input = nil
:output = nil
:auxiliary = nil
:tag = nil
:return = nil
:environment = nil
:genome = nil

State after 2 steps (last step: 1):
:exec = (2 integer_add)
:code = nil
:integer = (1)
:float = nil
:boolean = nil
:char = nil
:string = nil
:zip = nil
:vector_integer = nil
:vector_float = nil
:vector_boolean = nil
:vector_string = nil
:input = nil
:output = nil
:auxiliary = nil
:tag = nil
:return = nil
:environment = nil
:genome = nil

State after 3 steps (last step: 2):
:exec = (integer_add)
:code = nil
:integer = (2 1)
```

```
:float = nil
:boolean = nil
:char = nil
:string = nil
:zip = nil
:vector_integer = nil
:vector_float = nil
:vector_boolean = nil
:vector_string = nil
:input = nil
:output = nil
:auxiliary = nil
:tag = nil
:return = nil
:environment = nil
:genome = nil

State after 4 steps (last step: integer_add):
:exec = ()
:code = nil
:integer = (3)
:float = nil
:boolean = nil
:char = nil
:string = nil
:zip = nil
:vector_integer = nil
:vector_float = nil
:vector_boolean = nil
:vector_string = nil
:input = nil
:output = nil
:auxiliary = nil
:tag = nil
:return = nil
:environment = nil
:genome = nil
```

```
{:exec (), :code nil, :integer (3), :float nil, :boolean
nil}
```

To summarize, when the interpreter sees a value it pushes it onto the appropriate stack. When it sees an instruction it pops the needed arguments and pushes the results onto the appropriate stacks. If there are not sufficient arguments on the stacks then the instruction **does nothing**. And when the interpreter sees a list it simply pushes the contents of the list back onto the `:exec` stack individually.

That's pretty much all you need to know about how the Push interpreter works. Some instructions can do complicated things, including manipulating code on the `:exec` stack (which can create loops and conditionals, etc.), but the Push interpreter just follows the simple rules sketched here and all instructions take values from stacks and push results from stacks.

Here is a slightly more complicated example:

```
(stks (run-push '(5 1.23 integer_add (4) integer_sub
                  5.67 float_mult)
                (make-push-state)))
```

```
{:exec (), :code nil, :integer (1), :float (6.9741),
:boolean nil}
```

Notice that the integer and float operations are interleaved, that `integer_add` has no effect because there is only one integer on the stack when it is executed, and that the parentheses around the `4` have no effect.

For every type, there are standard stack manipulation instructions including `dup` (which duplicates the top element), `pop` (which throws away the top element), `swap` (which exchanges the top two elements), `rot` (which rotates the top three elements, inserting the top element beneath the next two), and `flush` (which empties the stack).

Here's a simple example with `integer_dup`:

```
(stks (run-push '(5 integer_dup integer_mult)
                (make-push-state)))
```

```
{:exec (), :code nil, :integer (25), :float nil, :boolean
nil}
```

These instructions work on the `:exec` stack too, which lets us do things like this:

```
(stks (run-push '(5 10 20 exec_dup integer_mult)
                (make-push-state)))
```

```
{:exec (), :code nil, :integer (1000), :float nil,
:boolean nil}
```

This might be confusing, because while most Push instructions operate on the things that come before them in the program (looking like "postfix" operations), the `exec_dup` instruction looks like it operates on something that comes after it (like a "prefix" operation).

But this makes sense when you consider how the Push interpreter executes programs, using the `:exec` stack: when `exec_dup` is being executed, `integer_mult` will be on top of the `:exec` stack, so that is what will be duplicated (and subsequently executed).

Now we can also see how parentheses can matter for program execution:

```
(stks (run-push '(5 exec_dup (3 integer_mult))
                (make-push-state)))
```

```
{:exec (), :code nil, :integer (45), :float nil, :boolean
nil}
```

What's going on here is that `exec_dup` duplicates the entire `(3 integer_mult)` expression, so that whole thing ends up getting executed twice.

Manipulating the `:exec` stack provides a lot of power, because a program can grow or change itself as it runs.

Even more fancy code-self-manipulation stuff can be done using the `:code` stack, for which there are many high-level list-processing instructions.

Here's a simple example:

```
(stks (run-push '(code_quote 20 code_quote 2 code_quote
                  (2 3 integer_mult) code_subst
                  code_do)
                (make-push-state)))
```

```
{:exec (), :code (), :integer (60), :float nil, :boolean
nil}
```

What happened here is that `code_subst` substituted `20` for `2` in a piece of code, `(2 3 integer_mult)`, and then `code_do` executed it (by copying it to the `:exec` stack). This is a pretty silly example, but it demonstrates how code it's possible to manipulate code and then execute the manipulated code.

Some instructions may take data from multiple stacks, or push results onto multiple stacks. A simple example of the former are the `eq` instructions, which are implemented for many types. These take two items of the specified type and push `true` on the `:boolean` stack if they are equal, or `false` if they are not:

```
(stks (run-push '(1 2 integer_eq)
                (make-push-state)))
```

```
{:exec (), :code nil, :integer (), :float nil, :boolean
(false)}
```

```
(stks (run-push '(1 1 integer_eq)
                (make-push-state)))
```

```
{:exec (), :code nil, :integer (), :float nil, :boolean
(true)}
```

What can you do with those boolean values? Well, there are boolean instructions like `boolean_and`, `boolean_or`, and `boolean_not`, but there are also handy things like `exec_if`:

```
(stks (run-push '(1 2 integer_eq exec_if 123 99.6)
                (make-push-state)))
```

```
{:exec (), :code nil, :integer (), :float (99.6),
:boolean ()}
```

What happened there? After executing `integer_eq` there was a `false` on top of the `:boolean` stack. The `exec_if` instruction takes the top element of the `:boolean` stack and if it's false then it also removes the top element of the `:exec` stack (which in this case was `123`). If it's `true` then it leaves the top element of the `:exec` stack but removes the *second* element. This means that only one of them will be executed, depending on the answer of the previous boolean expression. Here each branch of the conditional expression was just a single number (one, an integer, and the other, a float), but you could just as well have complex programs in each branch, in parentheses.

Just for kicks, and to show that it can be done, here's a pretty messy program that computes the factorial of a number (`8` in the example) using recursion on the `:code` stack.

```
(stks (run-push '(code_quote
                   (code_quote (integer_pop 1)
                    code_quote (code_dup integer_dup
                                1 integer_sub code_do
                                integer_mult)
                    integer_dup 2 integer_lt code_if)
                  code_dup
                  8
                  code_do)
                (make-push-state)))
```

```
{:exec (), :code ((code_quote (integer_pop 1) code_quote
(code_dup integer_dup 1 integer_sub code_do integer_mult)
integer_dup 2 integer_lt code_if)), :integer (40320),
:float nil, :boolean ()}
```

How does it work? It first pushes a bunch of stuff (let's call this the *recursive definition*) onto the `:code` stack, then duplicates it, then pushes `8` onto the `:integer` stack, and then finally executes `code_do` which will move the top copy of the recursive definition from the `:code` stack to the `:exec` stack, from where it will then be executed. I won't walk through the rest of the process in full detail, but the key thing to notice is that the recursive definition first pushes two bodies of code to the `:code` stack, one for the "base case" of the recursion and one for the "recursive case," and that it then compares the top `:integer` to 2 to decide which of the branches to execute. You can also notice that the first thing in the recursive branch is another call to `code_dup`, which ensures that there will be another copy of the recursive definition on the `:code` stack for further recursion if necessary.

This provides an example of recursion on the `:code` stack, but happily, there are also much simpler ways to compute factorials in Push, for example:

```
(stks (run-push '(1 8 exec_do*range integer_mult)
                (make-push-state)))
```

```
{:exec (), :code nil, :integer (40320), :float nil,
:boolean nil}
```

This example uses `exec_do*range`, which is an instruction that loops a body of code that it finds on the `:exec` stack, pushing a counter onto the integer stack before each loop body execution. There are other, related functions too, like `exec_do*count`, `exec_while`, etc.

Since we can do looping and recursion, it's possible to produce code that will run for a very long time, possibly infinitely. Push handles this by imposing an execution step limit:

```
(stks (run-push '(0 true exec_while
                  (1 integer_add true))
                (make-push-state)))
```

```
{:exec (1 integer_add true exec_while (1 integer_add
true)), :code nil, :integer (29), :float nil, :boolean
()}
```

This program pushes `0` on the `:integer` stack, then `true` on the `:boolean` stack, and then starts running a "while" loop that will keep executing as long as it finds `true` on top of the `:boolean` stack. Since the body of the loop itself pushes another `true` (along with adding `1` to the top integer), it will never stop.

But it did stop! And the top integer only got up to `29`. What's going on?

The answer is that it hit the step limit, and returned the state at that point.

The step limit is stored in an atom that we can query this way:

```
@global-evalpush-limit
```

```
150
```

So currently it will just run for 150 steps. As for why that gets us only up to `29`, that has to do with the steps required for the other parts of the program and the mechanics of `exec_while`, which does its work by pushing stuff to the `:exec` stack and executing it.

We can change the step limit like this:

```
(reset! global-evalpush-limit 1000)
```

```
1000
```

And then it'll get further before stopping:

```
(stks (run-push '(0 true exec_while
                   (1 integer_add true))
                 (make-push-state)))
```

```
{:exec (1 integer_add true exec_while (1 integer_add
true)), :code nil, :integer (199), :float nil, :boolean
()}
```

Incidentally, you can tell whether a call to run-push terminated normally or by hitting the step limit by looking at the `:termination` of the returned Push state. We've been hiding this with our `stks` function, but here's that same call without `stks`, where you can see that the `:termination` is `:abnormal`:

```
(run-push '(0 true exec_while (1 integer_add true))
          (make-push-state))
```

```
{:exec (1 integer_add true exec_while (1 integer_add
true)), :code nil, :integer (199), :float nil, :boolean
(), :char nil, :string nil, :zip nil, :vector_integer
nil, :vector_float nil, :vector_boolean nil,
:vector_string nil, :input nil, :output nil, :auxiliary
nil, :tag nil, :return nil, :environment nil, :genome
nil, :termination :abnormal}
```

All of the Push programs we've seen so far have been completely self-contained in the sense that the data they process is encoded directly within them. How can you give a program *inputs*?

One way is to push them onto stacks before you run the program. The `push-item` utility function makes this easy. For example:

```
(stks (run-push '(float_dup float_mult
                  3.141592 float_mult)
                (push-item 2.5
                           :float
                           (make-push-state))))
```

```
{:exec (), :code nil, :integer nil, :float (19.63495),
:boolean nil}
```

Here we pushed `2.5` onto the `:float` stack before we ran the Push program. The push program squared the initial value (by dulicating it and then multiplying the two copies), and then multiplied it by π. So it computed the area of a circle from a radius. You could call the same program on different initial states to compute different circle areas.

Sometimes it's even more convenient to have instructions that push inputs. That way the program can refer to them whenever it wants, without keeping track of where the initially-pushed items are on the stacks.

This is the job of Clojush's "input instructions," which work with the `:input` stack. The idea is that you push the inputs onto the `:input` stack (rather than the `:float` or other stacks), and then instructions like `in1`, `in2`, etc. can be used to push those values.

Here's a simple example:

```
(run-push '(in1 in1 float_mult 3.141592 float_mult)
          (push-item 2.5 :input (make-push-state)))
```

```
{:exec (), :code nil, :integer nil, :float (19.63495),
:boolean nil, :char nil, :string nil, :zip nil,
:vector_integer nil, :vector_float nil, :vector_boolean
nil, :vector_string nil, :input (2.5), :output nil,
:auxiliary nil, :tag nil, :return nil, :environment nil,
:genome nil, :termination :normal}
```

Here again I didn't use `stks` because I wanted to show the `:input` stack, so you can see that it contains one item, `2.5`. The `:input` stack is a stack like any other, but there aren't any instructions defined for it except the input instructions `in1`, `in2`, etc. So it never changes, aside from when you manually stick stuff on it before running your

Push program.

There are a *lot* more instructions and features in Clojush, but this is enough to get see how things work in general, and it's time to move on to using the `pushgp` function to evolve Push programs.

The `pushgp` function takes a single argument which should be a map containing key/value pairs for any or all of the arguments listed in Clojush's pushgp.clj file.

The only argument that is absolutely necessary is `:error-function` which should be a function that takes a Push program and returns a vector of errors (with lower being better and `0` being perfect). The reason it's a vector and not a single value is because you'll often want to run your program on a bunch of inputs, and this lets you return all of the individual test case errors, which might later be combined in various ways, depending on other options. If you have only one test case you should still have your error function return this in a vector.

In the example below I've specied an `:error-function` that evaluates the program for integer inputs from `0` to `9`, and for each input there's an error indicating how far the output is from $x^3 - 2x^2 - x$, where the input is x. If there's nothing on the `:integer` stack after the program runs (in which case `top-item` will return `:no-stack-item` rather than a number), then the error will be `1000` (so, really bad).

The other arguments I've specified in the example below are `:atom-generators`, `:population-size`, and `:use-single-thread`.

The `:atom-generators` argument specifies what's in the "primordial ooze"; that is, what ingredients can occur in evolving programs. By default this will be all known instructions, including some that generate random values, and you usually don't want all of those. So you'll usually want to provide your own value for this too. In addition to defined Push instructions, the `:atom-generators` can include functions of zero arguments. When one of these is picked from the ooze to be included in a program (either when `pushgp` creates the initial population or during mutation) it will be called and the result of the call is what will actually be included. In the example below I've included `(fn [] (lrand-int 10))`, which means that it will be possible for integers from `0` to `9` to show up in evolving programs.

(Why did I use `lrand-int` instead of Clojure's normal `rand-int`? This stems from our attempts to improve Clojush's performance when running in multi-threaded mode; we use "thread-local" random number generators and provide `lrand`, `lrand-int`, `lrand-nth`, and `lshuffle` that use those. That is preferred, but the ordinary Clojure functions will also work.)

For `:population-size` I supplied the value `100`, simply because the default of `1000` runs a little more slowly that I wanted for a tutorial.

Finally, I included `:use-single-thread true`. Why? By default, Clojush will run in a multi-threaded mode and use all available cores, which should make it run faster. But if you do something wrong, and cause an exception to be raised in one of the threads, Clojush will exit unceremoniously and without providing helpful feedback. (And if you're working in Gorilla REPL it will kill the server and you'll have to restart it, and you might have to jump through hoops to avoid losing unsaved changes to your worksheet.) The reasons for this are too boring to recount here, but the bottom line is that `:use-single-thread true` will prevent the exit, and you'll see normal error messages. I recommend running in this mode until you're confident that everything is working, and then run with `:use-single-thread false` (which is the default if you don't provide this argument at all) for better performance.

The following call to pushgp is the last thing in this tutorial. It's **long** because Clojush prints **lots** of info about the run as it proceeds, so that we can study how it is working and try to improve it. Many of the things printed won't make much sense without reading either more of the code base or publications linked to http://pushlanguage.org. But you can see the final result of the run near the bottom of the file.

```
(pushgp
  {:error-function
   (fn [program]
     (vec
       (for [input (range 10)]
         (let [output (->> (make-push-state)
                           (push-item input :input)
                           (run-push program)
                           (top-item :integer))]
           (if (number? output)
             (Math/abs (float (- output
                                 (- (* input
                                       input
                                       input)
                                    (* 2 input input)
                                    input))))
             1000)))))
   :atom-generators (list (fn [] (lrand-int 10))
                          'in1
                          'integer_div
                          'integer_mult
```

```
                              'integer_add
                              'integer_sub)
    :population-size 100
    :use-single-thread true})
```

```
Registered instructions: #{code_atom code_car print_newline
integer_sub integer_inc boolean_stackdepth return_exec_pop
vector_integer_eq autoconstructive_integer_rand boolean_pop
genome_close_inc string_fromchar vector_string_shove
zip_yankdup genome_new vector_float_yankdup exec_yankdup
vector_integer_shove integer_yankdup string_flush boolean_swap
zip_empty exec_shove vector_boolean_yank code_eq exec_y
boolean_yank integer_eq genome_silence string_butlast
code_contains string_conjchar code_do*count vector_float_last
genome_pop string_substring integer_mult code_length
vector_integer_dup boolean_or code_position boolean_empty
zip_fromcode print_vector_string vector_boolean_swap
return_frominteger vector_float_pushall char_iswhitespace
code_cdr exec_do*vector_integer integer_rand
vector_string_replacefirst string_first vector_boolean_first
exec_do*while exec_string_iterate string_indexofchar
vector_float_replace integer_fromstring code_list code_swap
char_frominteger genome_gene_randomize
vector_integer_emptyvector vector_string_eq
vector_float_butlast exec_empty zip_end? exec_fromzipnode
string_shove vector_boolean_pushall zip_insert_left_fromcode
exec_rot vector_string_concat vector_float_indexof code_pop
vector_string_subvec vector_integer_swap code_subst char_pop
return_string_pop zip_yank exec_dup vector_integer_butlast
vector_float_rest vector_string_flush boolean_fromfloat
code_fromziprights float_sin boolean_flush char_isdigit
float_lte exec_fromziproot vector_integer_empty print_code
vector_string_stackdepth string_reverse exec_k
vector_integer_yank float_frominteger char_rot print_char
vector_integer_stackdepth vector_boolean_concat boolean_xor
integer_gte genome_yankdup vector_float_shove
vector_integer_take code_quote string_replacefirst
return_fromstring exec_fromziplefts vector_integer_yankdup
boolean_shove float_lt vector_string_dup
vector_string_occurrencesof vector_integer_replace zip_branch?
vector_float_reverse float_mod vector_float_subvec string_last
print_boolean boolean_rot vector_string_rest integer_div
vector_float_remove integer_fromfloat integer_lte
code_fromzipchildren environment_end vector_integer_rot
integer_mod string_concat vector_string_butlast genome_swap
code_null exec_do*count vector_float_emptyvector
vector_string_yankdup integer_rot float_yankdup
vector_string_rot zip_replace_fromexec vector_string_take
integer_add vector_integer_occurrencesof integer_shove
genome_dup return_code_pop char_swap integer_max
return_fromexec code_wrap return_float_pop code_flush
genome_yank zip_shove vector_integer_flush
vector_integer_subvec vector_boolean_indexof vector_float_pop
vector_string_remove vector_integer_contains zip_remove
code_append vector_float_eq vector_integer_conj string_eq
zip_leftmost code_yankdup code_rot integer_stackdepth float_max
vector_boolean_set zip_append_child_fromexec zip_next
vector_float_conj zip_fromexec string_take zip_left
zip_replace_fromcode char_stackdepth return_fromchar genome_eq
vector_integer_replacefirst float_stackdepth code_fromziproot
```

```
float_fromchar float_gt boolean_dup float_fromboolean
code_fromzipnode genome_rot vector_float_replacefirst
vector_boolean_conj vector_boolean_dup vector_integer_indexof
vector_string_swap exec_eq string_emptystring string_swap
integer_yank exec_while float_empty print_vector_boolean
integer_min exec_swap genome_rotate integer_fromchar
vector_string_yank string_stackdepth code_do*range
string_replacechar char_allfromstring vector_integer_rest
vector_boolean_length char_yank vector_float_empty
code_fromfloat genome_parent2 return_fromcode string_pop
float_eq vector_boolean_empty zip_insert_child_fromexec
vector_string_last string_nth code_do* return_zip_pop
vector_string_pop zip_rot vector_integer_nth exec_do*range
exec_if char_shove zip_down zip_insert_left_fromexec
code_frominteger vector_boolean_remove vector_integer_remove
boolean_invert_first_then_and genome_flush print_string
integer_fromboolean char_yankdup code_do vector_string_first
boolean_frominteger string_setchar vector_integer_last
char_isletter genome_gene_dup vector_integer_concat
print_integer code_map boolean_eq float_gte return_fromfloat
genome_gene_copy string_occurrencesofchar
string_replacefirstchar print_float boolean_rand integer_flush
float_shove string_replace char_dup float_pop char_eq
vector_float_nth vector_string_conj integer_gt
return_integer_pop float_sub vector_integer_length
vector_float_set vector_string_indexof vector_boolean_rest
code_dup vector_boolean_shove zip_eq float_min boolean_not
float_mult float_fromstring genome_unsilence code_if
vector_integer_pop vector_boolean_last exec_do*times zip_pop
zip_rightmost float_dec vector_float_contains
genome_gene_copy_range environment_new exec_do*vector_string
code_nthcdr string_empty char_empty exec_pop vector_integer_set
autoconstructive_boolean_rand vector_float_rot string_yankdup
exec_do*vector_float string_removechar code_extract
vector_string_replace vector_float_first genome_parent1
return_tagspace char_flush vector_float_occurrencesof
vector_string_emptyvector float_add code_stackdepth exec_s
zip_insert_right_fromexec float_dup vector_string_nth
zip_stackdepth vector_integer_reverse print_vector_integer
char_fromfloat code_do*times code_noop zip_swap code_yank
integer_lt vector_boolean_eq genome_stackdepth
code_fromziplefts noop_open_paren string_containschar
string_yank char_rand zip_flush vector_boolean_rot float_swap
exec_fromziprights vector_string_pushall vector_string_set
vector_boolean_flush exec_noop code_size
vector_boolean_stackdepth vector_integer_pushall
vector_boolean_reverse integer_swap string_split
vector_boolean_contains string_fromboolean return_boolean_pop
vector_float_dup vector_boolean_replace integer_dup
vector_boolean_nth vector_string_length string_rest
zip_insert_child_fromcode float_tan string_rot string_rand
exec_yank string_parse_to_chars integer_pop integer_empty
vector_float_flush vector_float_yank
noop_delete_prev_paren_pair print_exec
zip_append_child_fromcode genome_gene_delete code_empty
float_inc zip_right vector_float_length float_rand integer_dec
string_contains return_fromboolean vector_float_concat
vector_float_stackdepth exec_do*vector_boolean
vector_integer_first genome_shove code_rand print_vector_float
float_rot return_char_pop vector_string_contains
vector_boolean_occurrencesof genome_empty zip_prev
```

```
genome_toggle_silent vector_string_reverse zip_dup code_cons
code_member exec_stackdepth float_flush boolean_and
vector_boolean_butlast string_length float_cos
string_frominteger exec_flush vector_string_empty exec_when
vector_float_swap genome_close_dec code_insert
vector_boolean_pop float_div zip_insert_right_fromcode
code_fromboolean vector_boolean_take code_shove
environment_begin vector_float_take
boolean_invert_second_then_and code_container code_nth
vector_boolean_subvec float_yank zip_up
vector_boolean_emptyvector vector_boolean_replacefirst
string_fromfloat vector_boolean_yankdup string_dup
boolean_yankdup exec_fromzipchildren}
Starting PushGP run.
Clojush version = version number unavailable
Hash of last Git commit = unavailable
GitHub link = unavailable
alignment-deviation = 10
alternation-rate = 0.01
atom-generators = (#object[tutorial_worksheet$eval9554$fn__9571
0x621fbefa tutorial_worksheet$eval9554$fn__9571@621fbefa] in1
integer_div integer_mult integer_add integer_sub)
autoconstructive = false
autoconstructive-boolean-rand-enrichment = -1
autoconstructive-integer-rand-enrichment = 1
close-increment-rate = 0.2
close-parens-probabilities = [0.772 0.206 0.021 0.001]
csv-columns = [:generation :location :total-error :push-
program-size]
csv-log-filename = log.csv
decimation-ratio = 1
decimation-tournament-size = 2
epigenetic-markers = [:close]
error-function = #object[tutorial_worksheet$eval9554$fn__9555
0x729ad0de tutorial_worksheet$eval9554$fn__9555@729ad0de]
error-threshold = 0
evalpush-limit = 150
evalpush-time-limit = 0
final-report-simplifications = 1000
genetic-operator-probabilities = {:reproduction 0.0, :uniform-
deletion 0.0, :uniform-close-mutation 0.0, :alternation 0.7,
[:make-next-operator-revertable :uniform-silence-mutation] 0.0,
[:alternation :uniform-mutation] 0.2, :uniform-mutation 0.1,
:uniform-silence-mutation 0.0, :autoconstruction 0.0}
json-log-filename = log.json
json-log-program-strings = false
lexicase-leakage = 0.1
log-fitnesses-for-all-cases = false
maintain-ancestors = false
max-error = 1000
max-generations = 1001
max-genome-size-in-initial-program = 50
max-point-evaluations = 1.0E101
max-points = 100
meta-error-categories = []
normalization = :none
parent-reversion-probability = 1.0
parent-selection = :lexicase
pass-individual-to-error-function = false
pop-when-tagging = true
population-size = 100
```

```
print-ancestors-of-solution = false
print-behavioral-diversity = false
print-cosmos-data = false
print-csv-logs = false
print-error-frequencies-by-case = false
print-errors = true
print-history = false
print-homology-data = false
print-json-logs = false
print-selection-counts = false
print-timings = false
problem-specific-report =
#object[clojush.pushgp.report$default_problem_specific_report
0x3fd573d3
clojush.pushgp.report$default_problem_specific_report@3fd573d3]
random-seed = -18 115 68 -96 67 64 58 82 -115 -122 82 45 54 24
74 52
replace-child-that-exceeds-size-limit-with = :random
report-simplifications = 100
return-simplified-on-failure = false
reuse-errors = true
save-initial-population = false
silent-instruction-probability = 0.2
tag-limit = 10000
top-level-pop-code = false
top-level-push-code = false
total-error-method = :sum
tournament-size = 7
trivial-geography-radius = 0
uniform-close-mutation-rate = 0.1
uniform-deletion-rate = 0.01
uniform-mutation-constant-tweak-rate = 0.5
uniform-mutation-float-gaussian-standard-deviation = 1.0
uniform-mutation-int-gaussian-standard-deviation = 1
uniform-mutation-rate = 0.01
uniform-mutation-string-char-change-rate = 0.1
uniform-mutation-tag-gaussian-standard-deviation = 100
uniform-silence-mutation-rate = 0.1
use-single-thread = true

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Generating initial population...
Processing generation: 0
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 0
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction 6} {:close 1, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_add} {:close 2, :instruction integer_sub} {:close 1,
:instruction in1} {:close 0, :instruction integer_mult} {:close
1, :instruction integer_mult} {:close 1, :instruction 2}
{:close 0, :instruction in1} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_mult})
Lexicase best program: (5 4 in1 integer_sub in1 integer_mult
integer_sub integer_div 6 in1 integer_sub integer_sub
integer_add integer_add integer_sub in1 integer_mult
integer_mult 2 in1 integer_sub integer_add integer_mult)
Lexicase best partial simplification: (5 4 in1 integer_sub in1
integer_mult integer_sub 6 in1 integer_sub integer_sub in1
integer_mult 2 in1 integer_sub integer_add)
Lexicase best errors: [2.0 0.0 4.0 10.0 18.0 28.0 40.0 54.0
70.0 88.0]
Lexicase best number of elite cases: 3
Lexicase best total error: 314.0
Lexicase best mean error: 31.4
Lexicase best size: 24
Percent parens: 0.042
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction integer_add}
{:close 0, :instruction integer_mult} {:close 1, :instruction
in1} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_div}
{:close 0, :instruction integer_div} {:close 0, :instruction
integer_sub} {:close 1, :instruction integer_mult} {:close 0,
:instruction integer_div} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_add} {:close 0, :instruction 6} {:close 0, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_div}
{:close 0, :instruction integer_div} {:close 0, :instruction
in1} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_div}
{:close 0, :instruction integer_sub})
Zero cases best program: (integer_add integer_mult in1
integer_sub integer_add integer_div integer_div integer_sub
integer_mult integer_div integer_mult integer_add integer_add 6
integer_div in1 integer_add integer_div integer_div in1
integer_add integer_mult integer_div integer_sub)
Zero cases best partial simplification: (in1 6 integer_div in1
integer_add in1 integer_add)
Zero cases best errors: [0.0 4.0 6.0 0.0 20.0 60.0 125.0 223.0
359.0 539.0]
Zero cases best number of elite cases: 2
Zero cases best number of zero cases: 2
Zero cases best total error: 1336.0
Zero cases best mean error: 133.6
Zero cases best size: 25
Percent parens: 0.040
--- Lexicase Population Statistics ---
Count of elite individuals by case: (46 1 1 2 1 1 1 1 1 1)
Population mean number of elite cases: 0.56
Count of perfect (error zero) individuals by case: (46 1 0 2 0
0 0 0 0 0)
Population mean number of perfect (error zero) cases: 0.49
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction 6} {:close 1, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_sub}
```

```
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_add} {:close 2, :instruction integer_sub} {:close 1,
:instruction in1} {:close 0, :instruction integer_mult} {:close
1, :instruction integer_mult} {:close 1, :instruction 2}
{:close 0, :instruction in1} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_mult})
Best program: (5 4 in1 integer_sub in1 integer_mult integer_sub
integer_div 6 in1 integer_sub integer_sub integer_add
integer_add integer_sub in1 integer_mult integer_mult 2 in1
integer_sub integer_add integer_mult)
Partial simplification: (5 4 in1 integer_sub in1 integer_mult
integer_sub 6 in1 integer_sub integer_sub in1 integer_mult 2
in1 integer_sub integer_add)
Errors: [2.0 0.0 4.0 10.0 18.0 28.0 40.0 54.0 70.0 88.0]
Total: 314.0
Mean: 31.4
Genome size: 23
Size: 24
Percent parens: 0.042
--- Population Statistics ---
Average total errors in population: 2235.44
Median total errors in population: 1406.0
Error averages by case: (60.56 53.2 53.59 58.62 84.81 135.64
218.56 339.07 505.89 725.5)
Error minima by case: (0.0 0.0 1.0 0.0 6.0 15.0 40.0 6.0 16.0
88.0)
Average genome size in population (length): 26.08
Average program size in population (points): 27.06
Average percent parens in population: 0.060
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 1
Genome diversity (% unique Plush genomes):       1.0
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 1
Syntactic diversity (% unique Push programs):    1.0
Total error diversity:                           0.48
Error (vector) diversity:                        0.5
--- Run Statistics ---
Number of program evaluations used so far: 3100
Number of point (instruction) evaluations so far: 1008561
--- Timings ---
Current time: 1457558109109 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 0
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 1
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 1
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction integer_mult} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_mult} {:close 1,
:instruction 6} {:close 0, :instruction integer_mult} {:close
0, :instruction 9} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_div} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_sub} {:close 1, :instruction 1} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 1, :instruction in1} {:close 0, :instruction
integer_mult} {:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_mult} {:close 0,
:instruction integer_mult} {:close 0, :instruction in1} {:close
0, :instruction integer_add} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction in1} {:close 0, :instruction integer_div} {:close
0, :instruction in1} {:close 0, :instruction integer_sub})
Lexicase best program: (in1 integer_add integer_div in1
integer_mult integer_sub integer_add integer_div integer_mult
integer_mult integer_mult 6 integer_mult 9 integer_sub
integer_div integer_div integer_sub integer_add integer_sub
integer_sub integer_mult in1 integer_sub 1 integer_add
integer_mult in1 integer_mult in1 integer_add integer_mult
integer_sub integer_div integer_mult integer_mult in1
integer_add integer_sub integer_div in1 integer_div in1
integer_sub)
Lexicase best partial simplification: (in1 in1 integer_mult 6
integer_mult 9 integer_sub in1 integer_sub 1 integer_add in1
integer_mult in1 integer_add in1 integer_add in1 integer_div
in1 integer_sub)
Lexicase best errors: [0.0 0.0 16.0 36.0 54.0 64.0 60.0 36.0
14.0 96.0]
Lexicase best number of elite cases: 3
Lexicase best total error: 376.0
Lexicase best mean error: 37.6
Lexicase best size: 45
Percent parens: 0.022
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction integer_add}
{:close 0, :instruction integer_mult} {:close 1, :instruction
in1} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_div}
{:close 0, :instruction integer_div} {:close 0, :instruction
integer_sub} {:close 1, :instruction integer_mult} {:close 0,
:instruction integer_div} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_add} {:close 0, :instruction 6} {:close 0, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_div}
{:close 0, :instruction integer_div} {:close 0, :instruction
in1} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_div}
{:close 0, :instruction integer_sub})
Zero cases best program: (integer_add integer_mult in1
integer_sub integer_add integer_div integer_div integer_sub
```

```
integer_mult integer_div integer_mult integer_add integer_add 6
integer_div in1 integer_add integer_div integer_div in1
integer_add integer_mult integer_div integer_sub)
Zero cases best partial simplification: (in1 6 integer_div in1
integer_add in1 integer_add)
Zero cases best errors: [0.0 4.0 6.0 0.0 20.0 60.0 125.0 223.0
359.0 539.0]
Zero cases best number of elite cases: 2
Zero cases best number of zero cases: 2
Zero cases best total error: 1336.0
Zero cases best mean error: 133.6
Zero cases best size: 25
Percent parens: 0.040
--- Lexicase Population Statistics ---
Count of elite individuals by case: (43 22 5 10 1 2 1 1 1 1)
Population mean number of elite cases: 0.87
Count of perfect (error zero) individuals by case: (43 22 0 10
0 0 0 0 0 0)
Population mean number of perfect (error zero) cases: 0.75
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction 6} {:close 1, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_add} {:close 2, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 2, :instruction in1} {:close
3, :instruction 5} {:close 0, :instruction integer_add} {:close
0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult} {:close 1, :instruction 2} {:close 0,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_add} {:close 0, :instruction
integer_mult})
Best program: (5 4 in1 integer_sub in1 integer_mult integer_sub
integer_div 6 in1 integer_sub integer_sub integer_add
integer_add integer_sub integer_add in1 5 integer_add
integer_add integer_mult in1 integer_mult integer_mult 2 in1
integer_sub integer_add integer_mult)
Partial simplification: (5 4 in1 integer_sub in1 integer_mult
integer_sub 6 in1 integer_sub integer_sub integer_add in1 5
integer_add integer_add in1 integer_mult 2 in1 integer_sub
integer_add)
Errors: [2.0 6.0 10.0 14.0 18.0 22.0 26.0 30.0 34.0 38.0]
Total: 200.0
Mean: 20.0
Genome size: 29
Size: 30
Percent parens: 0.033
--- Population Statistics ---
Average total errors in population: 984.97
Median total errors in population: 1162.0
Error averages by case: (1.98 13.25 25.02 32.85 43.04 62.53
93.95 138.59 220.03 353.73)
Error minima by case: (0.0 0.0 1.0 0.0 4.0 15.0 26.0 1.0 14.0
38.0)
Average genome size in population (length): 29.31
Average program size in population (points): 30.31
Average percent parens in population: 0.038
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 21
Genome diversity (% unique Plush genomes):       0.46
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 21
Syntactic diversity (% unique Push programs):    0.46
Total error diversity:                           0.38
Error (vector) diversity:                        0.41
--- Run Statistics ---
Number of program evaluations used so far: 3200
Number of point (instruction) evaluations so far: 1037871
--- Timings ---
Current time: 1457558109771 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 1
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 2
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 2
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction integer_mult} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_mult} {:close 1,
:instruction 6} {:close 0, :instruction integer_mult} {:close
0, :instruction 9} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_div} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_add}
{:close 1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_div} {:close 0, :instruction
in1} {:close 0, :instruction integer_div})
Lexicase best program: (in1 integer_add integer_div in1
integer_mult integer_sub integer_add integer_div integer_mult
integer_mult integer_mult 6 integer_mult 9 integer_sub
integer_div integer_div integer_sub integer_add integer_sub
integer_sub integer_mult in1 integer_mult integer_add
integer_mult integer_sub integer_div integer_mult integer_mult
in1 integer_add integer_sub integer_div in1 integer_div)
Lexicase best partial simplification: (in1 in1 integer_mult 6
integer_mult 9 integer_sub in1 integer_mult in1 integer_add in1
```

```
integer_div)
Lexicase best errors: [0.0 0.0 18.0 40.0 60.0 72.0 70.0 48.0
0.0 80.0]
Lexicase best number of elite cases: 3
Lexicase best total error: 388.0
Lexicase best mean error: 38.8
Lexicase best size: 37
Percent parens: 0.027
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction in1} {:close
0, :instruction integer_add} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 1, :instruction
integer_div} {:close 0, :instruction integer_mult} {:close 1,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 1, :instruction 6} {:close 0,
:instruction integer_mult} {:close 0, :instruction 9} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_mult} {:close 1,
:instruction in1} {:close 0, :instruction integer_mult} {:close
0, :instruction integer_add} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_div} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_mult} {:close 0, :instruction
in1} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_div})
Zero cases best program: (in1 integer_add integer_div in1
integer_mult integer_sub integer_add integer_div integer_mult
integer_mult integer_mult 6 integer_mult 9 integer_sub
integer_div integer_div integer_sub integer_add integer_sub
integer_sub integer_mult in1 integer_mult integer_add
integer_mult integer_sub integer_div integer_mult integer_mult
in1 integer_add integer_sub integer_div in1 integer_div)
Zero cases best partial simplification: (in1 in1 integer_mult 6
integer_mult 9 integer_sub in1 integer_mult in1 integer_add in1
integer_div)
Zero cases best errors: [0.0 0.0 18.0 40.0 60.0 72.0 70.0 48.0
0.0 80.0]
Zero cases best number of elite cases: 3
Zero cases best number of zero cases: 3
Zero cases best total error: 388.0
Zero cases best mean error: 38.8
Zero cases best size: 37
Percent parens: 0.027
--- Lexicase Population Statistics ---
Count of elite individuals by case: (62 26 6 5 1 1 1 8 1 1)
Population mean number of elite cases: 1.12
Count of perfect (error zero) individuals by case: (62 26 0 5 0
0 0 0 1 0)
Population mean number of perfect (error zero) cases: 0.94
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction 6} {:close 1, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_add} {:close 2, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 2, :instruction in1} {:close
3, :instruction 5} {:close 0, :instruction integer_add} {:close
0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult} {:close 1, :instruction 2} {:close 0,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_add} {:close 0, :instruction
integer_mult})
Best program: (5 4 in1 integer_sub in1 integer_mult integer_sub
integer_div 6 in1 integer_sub integer_sub integer_add
integer_add integer_sub integer_add in1 5 integer_add
integer_add integer_mult in1 integer_mult integer_mult 2 in1
integer_sub integer_add integer_mult)
Partial simplification: (5 4 in1 integer_sub in1 integer_mult
integer_sub 6 in1 integer_sub integer_sub integer_add in1 5
integer_add integer_add in1 integer_mult 2 in1 integer_sub
integer_add)
Errors: [2.0 6.0 10.0 14.0 18.0 22.0 26.0 30.0 34.0 38.0]
Total: 200.0
Mean: 20.0
Genome size: 29
Size: 30
Percent parens: 0.033
--- Population Statistics ---
Average total errors in population: 1088.08
Median total errors in population: 870.0
Error averages by case: (1.05 8.64 21.26 34.15 49.92 73.83
108.6 159.71 246.74 384.18)
Error minima by case: (0.0 0.0 1.0 0.0 2.0 3.0 23.0 1.0 0.0
2.0)
Average genome size in population (length): 33.67
Average program size in population (points): 34.67
Average percent parens in population: 0.031
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 18
Genome diversity (% unique Plush genomes):       0.38
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 18
Syntactic diversity (% unique Push programs):    0.38
Total error diversity:                           0.34
Error (vector) diversity:                        0.35
--- Run Statistics ---
Number of program evaluations used so far: 3300
Number of point (instruction) evaluations so far: 1071541
--- Timings ---
Current time: 1457558110405 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 2
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
```

```
Processing generation: 3
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 3
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction integer_mult} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_mult} {:close 1,
:instruction 6} {:close 0, :instruction integer_mult} {:close
0, :instruction 9} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_div} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_add}
{:close 1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_div} {:close 0, :instruction
in1} {:close 0, :instruction integer_div})
Lexicase best program: (in1 integer_add integer_div in1
integer_mult integer_sub integer_add integer_div integer_mult
integer_mult integer_mult 6 integer_mult 9 integer_sub
integer_div integer_div integer_sub integer_add integer_sub
integer_sub integer_mult in1 integer_mult integer_add
integer_mult integer_sub integer_div integer_mult integer_mult
in1 integer_add integer_sub integer_div in1 integer_div)
Lexicase best partial simplification: (in1 in1 integer_mult 6
integer_mult 9 integer_sub in1 integer_mult integer_mult in1
integer_add in1 integer_div)
Lexicase best errors: [0.0 0.0 18.0 40.0 60.0 72.0 70.0 48.0
0.0 80.0]
Lexicase best number of elite cases: 3
Lexicase best total error: 388.0
Lexicase best mean error: 38.8
Lexicase best size: 37
Percent parens: 0.027
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction in1} {:close
0, :instruction integer_add} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 1, :instruction
integer_div} {:close 0, :instruction integer_mult} {:close 1,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 1, :instruction 6} {:close 0,
:instruction integer_mult} {:close 0, :instruction 9} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_mult} {:close 1,
:instruction in1} {:close 0, :instruction integer_mult} {:close
0, :instruction integer_add} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_div} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_mult} {:close 0, :instruction
in1} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_div})
Zero cases best program: (in1 integer_add integer_div in1
integer_mult integer_sub integer_add integer_div integer_mult
integer_mult integer_mult 6 integer_mult 9 integer_sub
integer_div integer_div integer_sub integer_add integer_sub
integer_sub integer_mult in1 integer_mult integer_add
integer_mult integer_sub integer_div integer_mult integer_mult
in1 integer_add integer_sub integer_div in1 integer_div)
Zero cases best partial simplification: (in1 in1 integer_mult 6
integer_mult 9 integer_sub in1 integer_mult in1 integer_add in1
integer_div)
Zero cases best errors: [0.0 0.0 18.0 40.0 60.0 72.0 70.0 48.0
0.0 80.0]
Zero cases best number of elite cases: 3
Zero cases best number of zero cases: 3
Zero cases best total error: 388.0
Zero cases best mean error: 38.8
Zero cases best size: 37
Percent parens: 0.027
--- Lexicase Population Statistics ---
Count of elite individuals by case: (49 10 1 6 7 7 1 11 5 13)
Population mean number of elite cases: 1.10
Count of perfect (error zero) individuals by case: (49 10 1 6 0
0 0 0 5 0)
Population mean number of perfect (error zero) cases: 0.71
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_div} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 2, :instruction in1} {:close 3, :instruction 5} {:close
0, :instruction integer_add} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_mult} {:close 1,
:instruction in1} {:close 0, :instruction integer_mult} {:close
1, :instruction integer_mult} {:close 1, :instruction 2}
{:close 0, :instruction in1} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_mult})
Best program: (integer_sub integer_add integer_mult integer_sub
integer_add integer_add integer_sub integer_add 5 4 in1
integer_sub in1 integer_mult integer_sub integer_div 6 in1
integer_sub integer_sub integer_add integer_add integer_sub
integer_add in1 5 integer_add integer_add integer_mult in1
integer_mult integer_mult 2 in1 integer_sub integer_add
```

```
integer_mult)
Partial simplification: (integer_sub integer_add 5 4 in1
integer_sub in1 integer_mult integer_sub 6 in1 integer_sub
integer_sub in1 5 integer_add integer_add in1 integer_mult 2
in1 integer_sub integer_add)
Errors: [2.0 6.0 10.0 14.0 18.0 22.0 26.0 30.0 34.0 38.0]
Total: 200.0
Mean: 20.0
Genome size: 37
Size: 38
Percent parens: 0.026
--- Population Statistics ---
Average total errors in population: 1270.75
Median total errors in population: 1008.0
Error averages by case: (2.91 18.67 34.18 49.51 73.11 105.0
115.53 174.76 277.38 419.7)
Error minima by case: (0.0 0.0 0.0 0.0 2.0 3.0 22.0 1.0 0.0
2.0)
Average genome size in population (length): 34.54
Average program size in population (points): 35.54
Average percent parens in population: 0.029
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 13
Genome diversity (% unique Plush genomes):       0.41
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 13
Syntactic diversity (% unique Push programs):    0.41
Total error diversity:                           0.38
Error (vector) diversity:                        0.39
--- Run Statistics ---
Number of program evaluations used so far: 3400
Number of point (instruction) evaluations so far: 1106081
--- Timings ---
Current time: 1457558111028 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 3
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 4
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 4
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction integer_mult} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_mult} {:close 1,
:instruction 6} {:close 0, :instruction integer_mult} {:close
0, :instruction 9} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_div} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
```

```
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_add}
{:close 1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_div} {:close 0, :instruction
in1} {:close 0, :instruction integer_div})
Lexicase best program: (in1 integer_add integer_div in1
integer_mult integer_sub integer_add integer_div integer_mult
integer_mult integer_mult 6 integer_mult 9 integer_sub
integer_div integer_div integer_sub integer_add integer_sub
integer_sub integer_mult in1 integer_mult integer_add
integer_mult integer_sub integer_div integer_mult integer_mult
in1 integer_add integer_sub integer_div in1 integer_div)
Lexicase best partial simplification: (in1 in1 integer_mult 6
integer_mult 9 integer_sub in1 integer_mult in1 integer_add in1
integer_div)
Lexicase best errors: [0.0 0.0 18.0 40.0 60.0 72.0 70.0 48.0
0.0 80.0]
Lexicase best number of elite cases: 3
Lexicase best total error: 388.0
Lexicase best mean error: 38.8
Lexicase best size: 37
Percent parens: 0.027
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction in1} {:close
0, :instruction integer_add} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 1, :instruction
integer_div} {:close 0, :instruction integer_mult} {:close 1,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 1, :instruction 6} {:close 0,
:instruction integer_mult} {:close 0, :instruction 9} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_mult} {:close 1,
:instruction in1} {:close 0, :instruction integer_mult} {:close
0, :instruction integer_add} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_div} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_mult} {:close 0, :instruction
in1} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_div})
Zero cases best program: (in1 integer_add integer_div in1
integer_mult integer_sub integer_add integer_div integer_mult
integer_mult integer_mult 6 integer_mult 9 integer_sub
integer_div integer_div integer_sub integer_add integer_sub
integer_sub integer_mult in1 integer_mult integer_add
integer_mult integer_sub integer_div integer_mult integer_mult
in1 integer_add integer_sub integer_div in1 integer_div)
Zero cases best partial simplification: (in1 in1 integer_mult 6
integer_mult 9 integer_sub integer_add in1 integer_mult in1
integer_add in1 integer_div)
```

```
Zero cases best errors: [0.0 0.0 18.0 40.0 60.0 72.0 70.0 48.0
0.0 80.0]
Zero cases best number of elite cases: 3
Zero cases best number of zero cases: 3
Zero cases best total error: 388.0
Zero cases best mean error: 38.8
Zero cases best size: 37
Percent parens: 0.027
--- Lexicase Population Statistics ---
Count of elite individuals by case: (63 22 8 12 9 9 12 8 11 4)
Population mean number of elite cases: 1.58
Count of perfect (error zero) individuals by case: (63 22 8 12
0 0 0 0 11 0)
Population mean number of perfect (error zero) cases: 1.16
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction integer_mult} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_mult} {:close 1,
:instruction 6} {:close 0, :instruction integer_mult} {:close
0, :instruction 9} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_div} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_sub} {:close 1, :instruction 1} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 1, :instruction in1} {:close 0, :instruction
integer_mult} {:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_mult} {:close 0,
:instruction integer_mult} {:close 0, :instruction in1} {:close
0, :instruction integer_add} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction in1} {:close 0, :instruction integer_div})
Best program: (in1 integer_add integer_div in1 integer_mult
integer_sub integer_add integer_div integer_mult integer_mult
integer_mult 6 integer_mult 9 integer_sub integer_div
integer_div integer_sub integer_mult integer_mult in1
integer_sub 1 integer_add integer_mult in1 integer_mult in1
integer_add integer_mult integer_sub integer_div integer_mult
integer_mult in1 integer_add integer_sub integer_div in1
integer_div)
Partial simplification: (in1 in1 integer_mult integer_mult 6
integer_mult 9 integer_sub in1 integer_sub 1 integer_add
integer_mult in1 integer_mult in1 integer_add in1 integer_add
in1 integer_div)
Errors: [0.0 1.0 18.0 39.0 58.0 69.0 66.0 43.0 6.0 87.0]
Total: 387.0
Mean: 38.7
Genome size: 40
Size: 41
Percent parens: 0.024
--- Population Statistics ---
Average total errors in population: 1231.18
Median total errors in population: 954.0
Error averages by case: (1.18 11.37 23.67 36.45 55.11 81.74
115.67 181.25 284.42 440.32)
Error minima by case: (0.0 0.0 0.0 0.0 2.0 3.0 22.0 1.0 0.0
2.0)
Average genome size in population (length): 34.69
Average program size in population (points): 35.69
Average percent parens in population: 0.029
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 12
Genome diversity (% unique Plush genomes):       0.43
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 12
Syntactic diversity (% unique Push programs):    0.43
Total error diversity:                           0.39
Error (vector) diversity:                        0.39
--- Run Statistics ---
Number of program evaluations used so far: 3500
Number of point (instruction) evaluations so far: 1140771
--- Timings ---
Current time: 1457558111634 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 4
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 5
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 5
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction integer_mult} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_mult} {:close 1,
:instruction 6} {:close 0, :instruction integer_mult} {:close
0, :instruction 9} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_div} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_add}
{:close 1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_div} {:close 0, :instruction
in1} {:close 0, :instruction integer_div})
Lexicase best program: (in1 integer_add integer_div in1
integer_mult integer_sub integer_add integer_div integer_mult
integer_mult integer_mult 6 integer_mult 9 integer_sub
integer_div integer_div integer_sub integer_add integer_sub
integer_sub integer_mult in1 integer_mult integer_add
```

```
integer_mult integer_sub integer_div integer_mult integer_mult
in1 integer_add integer_sub integer_div in1 integer_div)
Lexicase best partial simplification: (in1 in1 integer_mult 6
integer_mult 9 integer_sub in1 integer_mult integer_add in1
integer_add in1 integer_div)
Lexicase best errors: [0.0 0.0 18.0 40.0 60.0 72.0 70.0 48.0
0.0 80.0]
Lexicase best number of elite cases: 3
Lexicase best total error: 388.0
Lexicase best mean error: 38.8
Lexicase best size: 37
Percent parens: 0.027
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction in1} {:close
0, :instruction integer_add} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 1, :instruction
integer_div} {:close 0, :instruction integer_mult} {:close 1,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 1, :instruction 6} {:close 0,
:instruction integer_mult} {:close 0, :instruction 9} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_mult} {:close 1,
:instruction in1} {:close 0, :instruction integer_mult} {:close
0, :instruction integer_add} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_div} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_mult} {:close 0, :instruction
in1} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_div})
Zero cases best program: (in1 integer_add integer_div in1
integer_mult integer_sub integer_add integer_div integer_mult
integer_mult integer_mult 6 integer_mult 9 integer_sub
integer_div integer_div integer_sub integer_add integer_sub
integer_sub integer_mult in1 integer_mult integer_add
integer_mult integer_sub integer_div integer_mult integer_mult
in1 integer_add integer_sub integer_div in1 integer_div)
Zero cases best partial simplification: (in1 in1 integer_mult 6
integer_mult 9 integer_sub in1 integer_mult in1 integer_add in1
integer_div)
Zero cases best errors: [0.0 0.0 18.0 40.0 60.0 72.0 70.0 48.0
0.0 80.0]
Zero cases best number of elite cases: 3
Zero cases best number of zero cases: 3
Zero cases best total error: 388.0
Zero cases best mean error: 38.8
Zero cases best size: 37
Percent parens: 0.027
--- Lexicase Population Statistics ---
Count of elite individuals by case: (56 19 6 11 9 5 1 11 10 8)
Population mean number of elite cases: 1.36
Count of perfect (error zero) individuals by case: (56 19 6 11
0 0 0 0 10 0)
Population mean number of perfect (error zero) cases: 1.02
--- Best Program (based on total-error) Statistics ---
```

```
Best genome: ({:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction integer_mult} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_mult} {:close 1,
:instruction 6} {:close 0, :instruction integer_mult} {:close
0, :instruction 9} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_div} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_add}
{:close 1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_div} {:close 0, :instruction
in1} {:close 0, :instruction integer_div})
Best program: (in1 integer_add integer_div in1 integer_mult
integer_sub integer_add integer_div integer_mult integer_mult
integer_mult 6 integer_mult 9 integer_sub integer_div
integer_div integer_sub integer_add integer_sub integer_sub
integer_mult in1 integer_mult integer_add integer_mult
integer_sub integer_div integer_mult integer_mult in1
integer_add integer_sub integer_div in1 integer_div)
Partial simplification: (in1 in1 integer_mult 6 integer_mult 9
integer_sub in1 integer_mult in1 integer_add in1 integer_div)
Errors: [0.0 0.0 18.0 40.0 60.0 72.0 70.0 48.0 0.0 80.0]
Total: 388.0
Mean: 38.8
Genome size: 36
Size: 37
Percent parens: 0.027
--- Population Statistics ---
Average total errors in population: 1452.72
Median total errors in population: 1008.0
Error averages by case: (2.34 16.25 29.77 44.27 68.99 105.68
137.49 211.18 331.82 504.93)
Error minima by case: (0.0 0.0 0.0 0.0 2.0 3.0 4.0 1.0 0.0 2.0)
Average genome size in population (length): 33.55
Average program size in population (points): 34.55
Average percent parens in population: 0.031
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 10
Genome diversity (% unique Plush genomes):       0.49
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 10
Syntactic diversity (% unique Push programs):    0.49
Total error diversity:                           0.4
Error (vector) diversity:                        0.41
--- Run Statistics ---
Number of program evaluations used so far: 3600
Number of point (instruction) evaluations so far: 1174321
--- Timings ---
```

```
Current time: 1457558112235 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 5
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 6
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 6
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction integer_mult} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_mult} {:close 1,
:instruction 6} {:close 0, :instruction integer_mult} {:close
0, :instruction 9} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_div} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_add}
{:close 1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_div} {:close 0, :instruction
in1} {:close 0, :instruction integer_div})
Lexicase best program: (in1 integer_add integer_div in1
integer_mult integer_sub integer_add integer_div integer_mult
integer_mult integer_mult 6 integer_mult 9 integer_sub
integer_div integer_div integer_sub integer_add integer_sub
integer_sub integer_mult in1 integer_mult integer_add
integer_mult integer_sub integer_div integer_mult integer_mult
in1 integer_add integer_sub integer_div in1 integer_div)
Lexicase best partial simplification: (in1 in1 integer_mult 6
integer_mult 9 integer_sub in1 integer_mult in1 integer_add in1
integer_div)
Lexicase best errors: [0.0 0.0 18.0 40.0 60.0 72.0 70.0 48.0
0.0 80.0]
Lexicase best number of elite cases: 3
Lexicase best total error: 388.0
Lexicase best mean error: 38.8
Lexicase best size: 37
Percent parens: 0.027
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction in1} {:close
0, :instruction integer_add} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 1, :instruction
integer_div} {:close 0, :instruction integer_mult} {:close 1,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 1, :instruction 6} {:close 0,
:instruction integer_mult} {:close 0, :instruction 9} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_mult} {:close 1,
:instruction in1} {:close 0, :instruction integer_mult} {:close
0, :instruction integer_add} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_div} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_mult} {:close 0, :instruction
in1} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_div})
Zero cases best program: (in1 integer_add integer_div in1
integer_mult integer_sub integer_add integer_div integer_mult
integer_mult integer_mult 6 integer_mult 9 integer_sub
integer_div integer_div integer_sub integer_add integer_sub
integer_sub integer_mult in1 integer_mult integer_add
integer_mult integer_sub integer_div integer_mult integer_mult
in1 integer_add integer_sub integer_div in1 integer_div)
Zero cases best partial simplification: (in1 in1 integer_mult 6
integer_mult 9 integer_sub in1 integer_mult in1 integer_add in1
integer_div)
Zero cases best errors: [0.0 0.0 18.0 40.0 60.0 72.0 70.0 48.0
0.0 80.0]
Zero cases best number of elite cases: 3
Zero cases best number of zero cases: 3
Zero cases best total error: 388.0
Zero cases best mean error: 38.8
Zero cases best size: 37
Percent parens: 0.027
--- Lexicase Population Statistics ---
Count of elite individuals by case: (48 22 5 13 9 9 7 4 14 12)
Population mean number of elite cases: 1.43
Count of perfect (error zero) individuals by case: (48 22 5 13
0 0 0 0 14 0)
Population mean number of perfect (error zero) cases: 1.02
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction integer_mult} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_mult} {:close 1,
:instruction 6} {:close 0, :instruction integer_mult} {:close
0, :instruction 9} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_div} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_add}
{:close 1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_div} {:close 0, :instruction
```

```
in1} {:close 0, :instruction integer_div})
Best program: (in1 integer_add integer_div in1 integer_mult
integer_sub integer_add integer_div integer_mult integer_mult
integer_mult 6 integer_mult 9 integer_sub integer_div
integer_div integer_sub integer_add integer_sub integer_sub
integer_mult in1 integer_mult integer_add integer_mult
integer_sub integer_div integer_mult integer_mult in1
integer_add integer_sub integer_div in1 integer_div)
Partial simplification: (in1 in1 integer_mult 6 integer_mult 9
integer_sub in1 integer_mult in1 integer_add in1 integer_div)
Errors: [0.0 0.0 18.0 40.0 60.0 72.0 70.0 48.0 0.0 80.0]
Total: 388.0
Mean: 38.8
Genome size: 36
Size: 37
Percent parens: 0.027
--- Population Statistics ---
Average total errors in population: 1295.66
Median total errors in population: 1008.0
Error averages by case: (2.07 11.1 24.81 39.35 65.16 102.96
127.66 195.06 290.78 436.71)
Error minima by case: (0.0 0.0 0.0 0.0 2.0 3.0 4.0 1.0 0.0 2.0)
Average genome size in population (length): 33.23
Average program size in population (points): 34.23
Average percent parens in population: 0.030
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 11
Genome diversity (% unique Plush genomes):       0.49
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 11
Syntactic diversity (% unique Push programs):    0.49
Total error diversity:                           0.41
Error (vector) diversity:                        0.41
--- Run Statistics ---
Number of program evaluations used so far: 3700
Number of point (instruction) evaluations so far: 1207551
--- Timings ---
Current time: 1457558112835 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 6
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 7
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 7
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction integer_mult} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_mult} {:close 1,
:instruction 6} {:close 0, :instruction integer_mult} {:close
```

```
0, :instruction 9} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_div} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_add}
{:close 1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_div} {:close 0, :instruction
in1} {:close 0, :instruction integer_div})
Lexicase best program: (in1 integer_add integer_div in1
integer_mult integer_sub integer_add integer_div integer_mult
integer_mult integer_mult 6 integer_mult 9 integer_sub
integer_div integer_div integer_sub integer_add integer_sub
integer_sub integer_mult in1 integer_mult integer_add
integer_mult integer_sub integer_div integer_mult integer_mult
in1 integer_add integer_sub integer_div in1 integer_div)
Lexicase best partial simplification: (in1 integer_add in1
integer_mult 6 integer_mult 9 integer_sub integer_div in1
integer_mult in1 integer_add in1 integer_div)
Lexicase best errors: [0.0 0.0 18.0 40.0 60.0 72.0 70.0 48.0
0.0 80.0]
Lexicase best number of elite cases: 3
Lexicase best total error: 388.0
Lexicase best mean error: 38.8
Lexicase best size: 37
Percent parens: 0.027
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction in1} {:close
0, :instruction integer_add} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 1, :instruction
integer_div} {:close 0, :instruction integer_mult} {:close 1,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 1, :instruction 6} {:close 0,
:instruction integer_mult} {:close 0, :instruction 9} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_mult} {:close 1,
:instruction in1} {:close 0, :instruction integer_mult} {:close
0, :instruction integer_add} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_div} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_mult} {:close 0, :instruction
in1} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_div})
Zero cases best program: (in1 integer_add integer_div in1
integer_mult integer_sub integer_add integer_div integer_mult
integer_mult integer_mult 6 integer_mult 9 integer_sub
integer_div integer_div integer_sub integer_add integer_sub
integer_sub integer_mult in1 integer_mult integer_add
integer_mult integer_sub integer_div integer_mult integer_mult
```

```
in1 integer_add integer_sub integer_div in1 integer_div)
Zero cases best partial simplification: (in1 in1 integer_mult
integer_mult 6 integer_mult 9 integer_sub integer_div
integer_sub in1 integer_mult in1 integer_add in1 integer_div)
Zero cases best errors: [0.0 0.0 18.0 40.0 60.0 72.0 70.0 48.0
0.0 80.0]
Zero cases best number of elite cases: 3
Zero cases best number of zero cases: 3
Zero cases best total error: 388.0
Zero cases best mean error: 38.8
Zero cases best size: 37
Percent parens: 0.027
--- Lexicase Population Statistics ---
Count of elite individuals by case: (47 15 5 10 1 1 13 10 9 9)
Population mean number of elite cases: 1.20
Count of perfect (error zero) individuals by case: (47 15 5 10
1 0 0 0 9 0)
Population mean number of perfect (error zero) cases: 0.87
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 1, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction integer_mult} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_mult} {:close 1,
:instruction 6} {:close 0, :instruction integer_mult} {:close
0, :instruction 9} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_div} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_add}
{:close 1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_div} {:close 0, :instruction
in1} {:close 0, :instruction integer_div})
Best program: (in1 integer_sub integer_div in1 integer_mult
integer_sub integer_add integer_div integer_mult integer_mult
integer_mult 6 integer_mult 9 integer_sub integer_div
integer_div integer_sub integer_add integer_sub integer_sub
integer_mult in1 integer_mult integer_add integer_mult
integer_sub integer_div integer_mult integer_mult in1
integer_add integer_sub integer_div in1 integer_div)
Partial simplification: (in1 in1 integer_mult 6 integer_mult 9
integer_sub in1 integer_mult in1 integer_add in1 integer_div)
Errors: [0.0 0.0 18.0 40.0 60.0 72.0 70.0 48.0 0.0 80.0]
Total: 388.0
Mean: 38.8
Genome size: 36
Size: 37
Percent parens: 0.027
--- Population Statistics ---
Average total errors in population: 1074.45
Median total errors in population: 854.0
Error averages by case: (2.93 14.11 28.74 42.45 60.76 88.38
99.1 145.37 231.82 360.79)
```

```
Error minima by case: (0.0 0.0 0.0 0.0 0.0 2.0 4.0 1.0 0.0 2.0)
Average genome size in population (length): 34.79
Average program size in population (points): 35.79
Average percent parens in population: 0.029
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 11
Genome diversity (% unique Plush genomes):       0.44
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 11
Syntactic diversity (% unique Push programs):    0.44
Total error diversity:                           0.31
Error (vector) diversity:                        0.33
--- Run Statistics ---
Number of program evaluations used so far: 3800
Number of point (instruction) evaluations so far: 1242341
--- Timings ---
Current time: 1457558113493 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 7
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 8
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 8
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction integer_mult} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_mult} {:close 1,
:instruction 6} {:close 0, :instruction integer_mult} {:close
0, :instruction 9} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_div} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_add}
{:close 1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_div} {:close 0, :instruction
in1} {:close 0, :instruction integer_div})
Lexicase best program: (in1 integer_add integer_div in1
integer_mult integer_sub integer_add integer_div integer_mult
integer_mult integer_mult 6 integer_mult 9 integer_sub
integer_div integer_div integer_sub integer_add integer_sub
integer_sub integer_mult in1 integer_mult integer_add
integer_mult integer_sub integer_div integer_mult integer_mult
in1 integer_add integer_sub integer_div in1 integer_div)
```

```
Lexicase best partial simplification: (in1 in1 integer_mult
integer_mult 6 integer_mult 9 integer_sub in1 integer_mult in1
integer_add in1 integer_div)
Lexicase best errors: [0.0 0.0 18.0 40.0 60.0 72.0 70.0 48.0
0.0 80.0]
Lexicase best number of elite cases: 3
Lexicase best total error: 388.0
Lexicase best mean error: 38.8
Lexicase best size: 37
Percent parens: 0.027
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction in1} {:close
0, :instruction integer_add} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 1, :instruction
integer_div} {:close 0, :instruction integer_mult} {:close 1,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 1, :instruction 6} {:close 0,
:instruction integer_mult} {:close 0, :instruction 9} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_mult} {:close 1,
:instruction in1} {:close 0, :instruction integer_mult} {:close
0, :instruction integer_add} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_div} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_mult} {:close 0, :instruction
in1} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_div})
Zero cases best program: (in1 integer_add integer_div in1
integer_mult integer_sub integer_add integer_div integer_mult
integer_mult integer_mult 6 integer_mult 9 integer_sub
integer_div integer_div integer_sub integer_add integer_sub
integer_sub integer_mult in1 integer_mult integer_add
integer_mult integer_sub integer_div integer_mult integer_mult
in1 integer_add integer_sub integer_div in1 integer_div)
Zero cases best partial simplification: (in1 in1 integer_mult 6
integer_mult 9 integer_sub in1 integer_mult in1 integer_add in1
integer_div)
Zero cases best errors: [0.0 0.0 18.0 40.0 60.0 72.0 70.0 48.0
0.0 80.0]
Zero cases best number of elite cases: 3
Zero cases best number of zero cases: 3
Zero cases best total error: 388.0
Zero cases best mean error: 38.8
Zero cases best size: 37
Percent parens: 0.027
--- Lexicase Population Statistics ---
Count of elite individuals by case: (61 15 6 15 7 8 1 10 10 5)
Population mean number of elite cases: 1.38
Count of perfect (error zero) individuals by case: (61 15 6 15
7 0 0 0 10 0)
Population mean number of perfect (error zero) cases: 1.14
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction integer_mult} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_mult} {:close 1,
:instruction 6} {:close 0, :instruction integer_mult} {:close
0, :instruction 9} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_div} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_add}
{:close 1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_sub} {:close 0, :instruction
in1} {:close 0, :instruction integer_div})
Best program: (in1 integer_add integer_div in1 integer_mult
integer_sub integer_add integer_div integer_mult integer_mult
integer_mult 6 integer_mult 9 integer_sub integer_div
integer_div integer_sub integer_add integer_sub integer_sub
integer_mult in1 integer_mult integer_add integer_mult
integer_sub integer_div integer_mult integer_mult in1
integer_sub integer_sub integer_sub in1 integer_div)
Partial simplification: (in1 integer_div in1 integer_mult 6
integer_mult 9 integer_sub in1 integer_mult in1 integer_sub in1
integer_div)
Errors: [0.0 2.0 16.0 38.0 58.0 70.0 68.0 46.0 2.0 82.0]
Total: 382.0
Mean: 38.2
Genome size: 36
Size: 37
Percent parens: 0.027
--- Population Statistics ---
Average total errors in population: 1225.48
Median total errors in population: 813.0
Error averages by case: (1.39 11.93 25.94 39.94 60.96 90.38
116.15 175.7 276.39 426.7)
Error minima by case: (0.0 0.0 0.0 0.0 0.0 2.0 2.0 1.0 0.0 2.0)
Average genome size in population (length): 33.34
Average program size in population (points): 34.34
Average percent parens in population: 0.030
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 11
Genome diversity (% unique Plush genomes):       0.45
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 11
Syntactic diversity (% unique Push programs):    0.45
Total error diversity:                           0.34
Error (vector) diversity:                        0.35
--- Run Statistics ---
Number of program evaluations used so far: 3900
Number of point (instruction) evaluations so far: 1275681
--- Timings ---
Current time: 1457558114096 milliseconds
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 8
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 9
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 9
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
0, :instruction in1} {:close 1, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction 6} {:close 1, :instruction
in1} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_sub} {:close 2, :instruction
integer_sub} {:close 0, :instruction 1} {:close 0, :instruction
5} {:close 1, :instruction 4} {:close 0, :instruction in1}
{:close 0, :instruction integer_sub} {:close 0, :instruction
in1} {:close 1, :instruction integer_mult} {:close 0,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction in1} {:close 1, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction 6} {:close 1, :instruction
in1} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 2, :instruction
integer_sub} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_mult} {:close 1, :instruction
in1} {:close 0, :instruction integer_mult} {:close 1,
:instruction integer_mult})
Lexicase best program: (5 4 in1 integer_sub in1 in1
integer_mult integer_sub integer_div 6 in1 integer_sub
integer_sub integer_add integer_sub integer_sub 1 5 4 in1
integer_sub in1 integer_mult in1 integer_sub in1 integer_mult
integer_sub integer_div 6 in1 integer_sub integer_sub
integer_add integer_add integer_sub integer_add in1 5
integer_add integer_add integer_mult in1 integer_mult
integer_mult)
Lexicase best partial simplification: (5 4 in1 integer_sub in1
in1 integer_mult integer_sub integer_div 6 in1 integer_sub
integer_sub 1 5 4 in1 integer_sub in1 integer_mult in1
integer_sub in1 integer_mult integer_sub integer_div 6 in1
integer_sub integer_sub integer_add integer_add in1 5
integer_add integer_add integer_mult in1 integer_mult)
Lexicase best errors: [0.0 0.0 2.0 0.0 8.0 30.0 72.0 140.0
240.0 378.0]
Lexicase best number of elite cases: 3
Lexicase best total error: 870.0
Lexicase best mean error: 87.0
Lexicase best size: 46
Percent parens: 0.022
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
0, :instruction in1} {:close 1, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction 6} {:close 1, :instruction
in1} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_sub} {:close 2, :instruction
integer_sub} {:close 0, :instruction 1} {:close 0, :instruction
5} {:close 1, :instruction 4} {:close 0, :instruction in1}
{:close 0, :instruction integer_sub} {:close 0, :instruction
in1} {:close 1, :instruction integer_mult} {:close 0,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction in1} {:close 1, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction 6} {:close 1, :instruction
in1} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 2, :instruction
integer_sub} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_mult} {:close 1, :instruction
in1} {:close 0, :instruction integer_mult} {:close 1,
:instruction integer_mult})
Zero cases best program: (5 4 in1 integer_sub in1 in1
integer_mult integer_sub integer_div 6 in1 integer_sub
integer_sub integer_add integer_sub integer_sub 1 5 4 in1
integer_sub in1 integer_mult in1 integer_sub in1 integer_mult
integer_sub integer_div 6 in1 integer_sub integer_sub
integer_add integer_add integer_sub integer_add in1 5
integer_add integer_add integer_mult in1 integer_mult
integer_mult)
Zero cases best partial simplification: (5 4 in1 integer_sub
in1 in1 integer_mult integer_sub integer_div 6 in1 integer_sub
integer_sub 1 5 4 in1 integer_sub in1 integer_mult in1
integer_sub in1 integer_mult integer_sub integer_div 6 in1
integer_sub integer_sub integer_add in1 5 integer_add
integer_add in1 integer_mult)
Zero cases best errors: [0.0 0.0 2.0 0.0 8.0 30.0 72.0 140.0
240.0 378.0]
Zero cases best number of elite cases: 3
Zero cases best number of zero cases: 3
Zero cases best total error: 870.0
Zero cases best mean error: 87.0
Zero cases best size: 46
Percent parens: 0.022
--- Lexicase Population Statistics ---
Count of elite individuals by case: (65 24 10 7 6 1 1 7 11 8)
Population mean number of elite cases: 1.40
Count of perfect (error zero) individuals by case: (65 24 10 7
6 1 1 0 11 0)
Population mean number of perfect (error zero) cases: 1.25
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 0, :instruction 6} {:close
1, :instruction in1} {:close 0, :instruction integer_sub}
```

```
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_sub} {:close 2,
:instruction integer_sub} {:close 0, :instruction 1} {:close 0,
:instruction 5} {:close 1, :instruction 4} {:close 0,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction in1} {:close 1, :instruction integer_mult}
{:close 0, :instruction in1} {:close 0, :instruction
integer_sub} {:close 0, :instruction in1} {:close 1,
:instruction integer_mult} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_div} {:close 0, :instruction 6}
{:close 1, :instruction in1} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 2, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 2, :instruction in1} {:close 3,
:instruction 5} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 1, :instruction in1} {:close 0, :instruction
integer_mult} {:close 1, :instruction integer_mult})
Best program: (5 4 in1 integer_sub in1 integer_mult integer_sub
integer_div integer_mult 6 in1 integer_sub integer_sub
integer_add integer_sub integer_sub 1 5 4 in1 integer_sub in1
integer_mult in1 integer_sub in1 integer_mult integer_sub
integer_div 6 in1 integer_sub integer_sub integer_add
integer_add integer_sub integer_add in1 5 integer_add
integer_add integer_mult in1 integer_mult integer_mult)
Partial simplification: (5 4 in1 integer_sub in1 integer_mult
integer_sub 6 in1 integer_sub integer_sub 1 5 4 in1 integer_sub
in1 integer_mult in1 integer_sub in1 integer_mult integer_sub
integer_div 6 in1 integer_sub integer_sub integer_add
integer_add in1 5 integer_add integer_add in1 integer_mult)
Errors: [0.0 0.0 4.0 6.0 12.0 20.0 30.0 42.0 56.0 72.0]
Total: 242.0
Mean: 24.2
Genome size: 45
Size: 46
Percent parens: 0.022
--- Population Statistics ---
Average total errors in population: 11678.48
Median total errors in population: 870.0
Error averages by case: (1.62 14.66 32.37 46.3 115.24 308.15
722.06 1583.2 3137.9 5716.98)
Error minima by case: (0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 2.0)
Average genome size in population (length): 34.96
Average program size in population (points): 35.96
Average percent parens in population: 0.029
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 10
Genome diversity (% unique Plush genomes):       0.49
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 10
Syntactic diversity (% unique Push programs):    0.49
Total error diversity:                           0.4
Error (vector) diversity:                        0.4
--- Run Statistics ---
Number of program evaluations used so far: 4000
Number of point (instruction) evaluations so far: 1310641
--- Timings ---
```

```
Current time: 1457558115005 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 9
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 10
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 10
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 1, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction integer_mult} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_mult} {:close 1,
:instruction 6} {:close 0, :instruction integer_mult} {:close
0, :instruction 9} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_div} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_add}
{:close 1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_div} {:close 0, :instruction
in1} {:close 0, :instruction integer_div})
Lexicase best program: (in1 integer_sub integer_div in1
integer_mult integer_sub integer_add integer_div integer_mult
integer_mult integer_mult 6 integer_mult 9 integer_sub
integer_div integer_div integer_sub integer_add integer_sub
integer_sub integer_mult in1 integer_mult integer_add
integer_mult integer_sub integer_div integer_mult integer_mult
in1 integer_add integer_sub integer_div in1 integer_div)
Lexicase best partial simplification: (in1 in1 integer_mult 6
integer_mult 9 integer_sub in1 integer_mult in1 integer_add in1
integer_div)
Lexicase best errors: [0.0 0.0 18.0 40.0 60.0 72.0 70.0 48.0
0.0 80.0]
Lexicase best number of elite cases: 3
Lexicase best total error: 388.0
Lexicase best mean error: 38.8
Lexicase best size: 37
Percent parens: 0.027
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction in1} {:close
0, :instruction integer_sub} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 1, :instruction
integer_div} {:close 0, :instruction integer_mult} {:close 1,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 1, :instruction 6} {:close 0,
:instruction integer_mult} {:close 0, :instruction 9} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_mult} {:close 1,
:instruction in1} {:close 0, :instruction integer_mult} {:close
0, :instruction integer_add} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_div} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_mult} {:close 0, :instruction
in1} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_div})
Zero cases best program: (in1 integer_sub integer_div in1
integer_mult integer_sub integer_add integer_div integer_mult
integer_mult integer_mult 6 integer_mult 9 integer_sub
integer_div integer_div integer_sub integer_add integer_sub
integer_sub integer_mult in1 integer_mult integer_add
integer_mult integer_sub integer_div integer_mult integer_mult
in1 integer_add integer_sub integer_div in1 integer_div)
Zero cases best partial simplification: (in1 in1 integer_mult 6
integer_mult 9 integer_sub in1 integer_mult in1 integer_add in1
integer_div)
Zero cases best errors: [0.0 0.0 18.0 40.0 60.0 72.0 70.0 48.0
0.0 80.0]
Zero cases best number of elite cases: 3
Zero cases best number of zero cases: 3
Zero cases best total error: 388.0
Zero cases best mean error: 38.8
Zero cases best size: 37
Percent parens: 0.027
--- Lexicase Population Statistics ---
Count of elite individuals by case: (84 37 11 11 11 4 10 7 11
8)
Population mean number of elite cases: 1.94
Count of perfect (error zero) individuals by case: (84 37 11 11
11 4 10 0 11 0)
Population mean number of perfect (error zero) cases: 1.79
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 0, :instruction 6} {:close
1, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_sub} {:close 2,
:instruction integer_sub} {:close 0, :instruction 1} {:close 0,
:instruction 5} {:close 1, :instruction 4} {:close 0,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction in1} {:close 1, :instruction integer_mult}
{:close 0, :instruction in1} {:close 0, :instruction
integer_sub} {:close 0, :instruction in1} {:close 1,
:instruction integer_mult} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_div} {:close 0, :instruction 6}
{:close 1, :instruction in1} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
```

```
{:close 2, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 2, :instruction in1} {:close 3,
:instruction 5} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 1, :instruction in1} {:close 0, :instruction
integer_mult} {:close 1, :instruction integer_mult})
Best program: (5 4 in1 integer_sub in1 integer_mult integer_sub
integer_div integer_mult 6 in1 integer_sub integer_sub
integer_add integer_sub integer_sub 1 5 4 in1 integer_sub in1
integer_mult in1 integer_sub in1 integer_mult integer_sub
integer_div 6 in1 integer_sub integer_sub integer_add
integer_add integer_sub integer_add in1 5 integer_add
integer_add integer_mult in1 integer_mult integer_mult)
Partial simplification: (5 4 in1 integer_sub in1 integer_mult
integer_sub 6 in1 integer_sub integer_sub integer_sub 1 5 in1
in1 integer_mult in1 integer_sub in1 integer_mult integer_sub
integer_div 6 in1 integer_sub integer_sub integer_add
integer_sub in1 5 integer_add integer_add in1 integer_mult)
Errors: [0.0 0.0 4.0 6.0 12.0 20.0 30.0 42.0 56.0 72.0]
Total: 242.0
Mean: 24.2
Genome size: 45
Size: 46
Percent parens: 0.022
--- Population Statistics ---
Average total errors in population: 10160.92
Median total errors in population: 870.0
Error averages by case: (60.96 153.74 280.33 439.69 628.46
877.58 1169.35 1576.27 2123.95 2850.59)
Error minima by case: (0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 2.0)
Average genome size in population (length): 36.5
Average program size in population (points): 37.5
Average percent parens in population: 0.028
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 9
Genome diversity (% unique Plush genomes):       0.47
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 9
Syntactic diversity (% unique Push programs):    0.47
Total error diversity:                           0.37
Error (vector) diversity:                        0.39
--- Run Statistics ---
Number of program evaluations used so far: 4100
Number of point (instruction) evaluations so far: 1347141
--- Timings ---
Current time: 1457558115677 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 10
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 11
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 11
--- Lexicse Program with Most Elite Cases Statistics ---
```

```
Lexicase best genome: ({:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction integer_mult} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_mult} {:close 1,
:instruction 6} {:close 0, :instruction integer_mult} {:close
0, :instruction 9} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_div} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_add}
{:close 1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_div} {:close 0, :instruction
in1} {:close 0, :instruction integer_div})
Lexicase best program: (in1 integer_add integer_div in1
integer_mult integer_sub integer_add integer_div integer_mult
integer_mult integer_mult 6 integer_mult 9 integer_sub
integer_div integer_div integer_sub integer_add integer_sub
integer_sub integer_mult in1 integer_mult integer_add
integer_mult integer_sub integer_div integer_mult integer_mult
in1 integer_add integer_sub integer_div in1 integer_div)
Lexicase best partial simplification: (in1 in1 integer_mult 6
integer_mult 9 integer_sub in1 integer_mult integer_div in1
integer_add in1 integer_div)
Lexicase best errors: [0.0 0.0 18.0 40.0 60.0 72.0 70.0 48.0
0.0 80.0]
Lexicase best number of elite cases: 3
Lexicase best total error: 388.0
Lexicase best mean error: 38.8
Lexicase best size: 37
Percent parens: 0.027
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction in1} {:close
0, :instruction integer_add} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 1, :instruction
integer_div} {:close 0, :instruction integer_mult} {:close 1,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 1, :instruction 6} {:close 0,
:instruction integer_mult} {:close 0, :instruction 9} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_mult} {:close 1,
:instruction in1} {:close 0, :instruction integer_mult} {:close
0, :instruction integer_add} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_div} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_mult} {:close 0, :instruction
in1} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_div})
Zero cases best program: (in1 integer_add integer_div in1
integer_mult integer_sub integer_add integer_div integer_mult
integer_mult integer_mult 6 integer_mult 9 integer_sub
integer_div integer_div integer_sub integer_add integer_sub
integer_sub integer_mult in1 integer_mult integer_add
integer_mult integer_sub integer_div integer_mult integer_mult
in1 integer_add integer_sub integer_div in1 integer_div)
Zero cases best partial simplification: (in1 in1 integer_mult 6
integer_mult 9 integer_sub in1 integer_mult integer_sub in1
integer_add in1 integer_div)
Zero cases best errors: [0.0 0.0 18.0 40.0 60.0 72.0 70.0 48.0
0.0 80.0]
Zero cases best number of elite cases: 3
Zero cases best number of zero cases: 3
Zero cases best total error: 388.0
Zero cases best mean error: 38.8
Zero cases best size: 37
Percent parens: 0.027
--- Lexicase Population Statistics ---
Count of elite individuals by case: (85 36 20 8 11 5 12 10 7
12)
Population mean number of elite cases: 2.06
Count of perfect (error zero) individuals by case: (85 36 20 8
11 5 12 0 7 0)
Population mean number of perfect (error zero) cases: 1.84
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction in1} {:close 1,
:instruction integer_mult} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_div} {:close 0, :instruction 6}
{:close 1, :instruction in1} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 2, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 2, :instruction in1} {:close 3,
:instruction 5} {:close 0, :instruction integer_mult} {:close
0, :instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 1, :instruction integer_mult})
Best program: (in1 integer_mult integer_sub integer_div 6 in1
integer_sub integer_sub integer_add integer_add integer_sub
integer_add in1 5 integer_mult integer_mult integer_mult
integer_mult)
Partial simplification: (in1 6 in1 integer_sub integer_sub in1
5 integer_mult integer_mult)
Errors: [0.0 18.0 18.0 6.0 12.0 30.0 42.0 42.0 24.0 18.0]
Total: 210.0
Mean: 21.0
Genome size: 18
Size: 19
Percent parens: 0.053
--- Population Statistics ---
Average total errors in population: 15179.21
Median total errors in population: 870.0
Error averages by case: (11.24 24.07 57.78 143.93 339.8 718.63
1342.54 2378.78 3952.0 6210.44)
Error minima by case: (0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 2.0)
Average genome size in population (length): 35.3
Average program size in population (points): 36.29
Average percent parens in population: 0.029
--- Population Diversity Statistics ---
```

```
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 11
Genome diversity (% unique Plush genomes):       0.43
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 11
Syntactic diversity (% unique Push programs):    0.43
Total error diversity:                           0.32
Error (vector) diversity:                        0.33
--- Run Statistics ---
Number of program evaluations used so far: 4200
Number of point (instruction) evaluations so far: 1382441
--- Timings ---
Current time: 1457558116220 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 11
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 12
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 12
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction integer_mult} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_mult} {:close 1,
:instruction 6} {:close 0, :instruction integer_mult} {:close
0, :instruction 9} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_div} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_add}
{:close 1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_div} {:close 0, :instruction
in1} {:close 0, :instruction integer_div})
Lexicase best program: (in1 integer_add integer_div in1
integer_mult integer_sub integer_add integer_div integer_mult
integer_mult integer_mult 6 integer_mult 9 integer_sub
integer_div integer_div integer_sub integer_add integer_sub
integer_sub integer_mult in1 integer_mult integer_add
integer_mult integer_sub integer_div integer_mult integer_mult
in1 integer_add integer_sub integer_div in1 integer_div)
Lexicase best partial simplification: (in1 in1 integer_mult 6
integer_mult 9 integer_sub in1 integer_mult in1 integer_add in1
integer_div)
Lexicase best errors: [0.0 0.0 18.0 40.0 60.0 72.0 70.0 48.0
0.0 80.0]
```

```
Lexicase best number of elite cases: 3
Lexicase best total error: 388.0
Lexicase best mean error: 38.8
Lexicase best size: 37
Percent parens: 0.027
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction in1} {:close
0, :instruction integer_add} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 1, :instruction
integer_div} {:close 0, :instruction integer_mult} {:close 1,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 1, :instruction 6} {:close 0,
:instruction integer_mult} {:close 0, :instruction 9} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_mult} {:close 1,
:instruction in1} {:close 0, :instruction integer_mult} {:close
0, :instruction integer_add} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_div} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_mult} {:close 0, :instruction
in1} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_div})
Zero cases best program: (in1 integer_add integer_div in1
integer_mult integer_sub integer_add integer_div integer_mult
integer_mult integer_mult 6 integer_mult 9 integer_sub
integer_div integer_div integer_sub integer_add integer_sub
integer_sub integer_mult in1 integer_mult integer_add
integer_mult integer_sub integer_div integer_mult integer_mult
in1 integer_add integer_sub integer_div in1 integer_div)
Zero cases best partial simplification: (in1 in1 integer_mult 6
integer_mult 9 integer_sub integer_sub in1 integer_mult in1
integer_add in1 integer_div)
Zero cases best errors: [0.0 0.0 18.0 40.0 60.0 72.0 70.0 48.0
0.0 80.0]
Zero cases best number of elite cases: 3
Zero cases best number of zero cases: 3
Zero cases best total error: 388.0
Zero cases best mean error: 38.8
Zero cases best size: 37
Percent parens: 0.027
--- Lexicase Population Statistics ---
Count of elite individuals by case: (78 32 14 8 6 5 14 10 6 14)
Population mean number of elite cases: 1.87
Count of perfect (error zero) individuals by case: (78 32 14 8
6 5 14 0 6 0)
Population mean number of perfect (error zero) cases: 1.63
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction 6} {:close 1, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_add} {:close 2, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_add} {:close 0,
:instruction 6} {:close 1, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_add} {:close 2, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 2, :instruction in1} {:close
3, :instruction 5} {:close 0, :instruction integer_add} {:close
0, :instruction integer_add} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_mult} {:close 1,
:instruction in1} {:close 0, :instruction integer_mult} {:close
1, :instruction integer_mult})
Best program: (in1 integer_add integer_div in1 integer_sub
integer_div 6 in1 integer_sub integer_sub integer_add
integer_add integer_sub integer_add 5 4 in1 integer_sub in1
integer_mult integer_sub integer_add 6 in1 integer_sub
integer_sub integer_add integer_add integer_sub integer_add in1
5 integer_add integer_add integer_add integer_mult in1
integer_mult integer_mult)
Partial simplification: (in1 6 in1 integer_sub integer_sub
integer_add 5 4 in1 integer_sub in1 integer_mult integer_sub 6
integer_sub integer_add in1 5 integer_add integer_add in1
integer_mult)
Errors: [0.0 0.0 2.0 6.0 12.0 20.0 30.0 42.0 56.0 72.0]
Total: 240.0
Mean: 24.0
Genome size: 39
Size: 40
Percent parens: 0.025
--- Population Statistics ---
Average total errors in population: 2520.2
Median total errors in population: 870.0
Error averages by case: (1.97 14.54 34.76 54.03 91.1 139.77
155.32 294.41 591.91 1142.39)
Error minima by case: (0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 2.0)
Average genome size in population (length): 33.26
Average program size in population (points): 34.25
Average percent parens in population: 0.030
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 12
Genome diversity (% unique Plush genomes):       0.51
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 12
Syntactic diversity (% unique Push programs):    0.51
Total error diversity:                           0.39
Error (vector) diversity:                        0.41
--- Run Statistics ---
Number of program evaluations used so far: 4300
Number of point (instruction) evaluations so far: 1415701
--- Timings ---
Current time: 1457558116815 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 12
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
Producing offspring...
Installing next generation...
Processing generation: 13
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 13
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_add} {:close 1, :instruction integer_div} {:close 0,
:instruction integer_div} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_add} {:close 2, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 2, :instruction
in1} {:close 3, :instruction 5} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_mult} {:close 1, :instruction in1} {:close
0, :instruction integer_mult} {:close 1, :instruction
integer_mult})
Lexicase best program: (in1 integer_add integer_div in1
integer_add integer_div integer_div 6 in1 integer_sub
integer_sub integer_add integer_add integer_sub integer_add 6
in1 integer_sub integer_sub integer_add integer_add integer_sub
integer_add in1 5 integer_add integer_add integer_mult in1
integer_mult integer_mult)
Lexicase best partial simplification: (in1 in1 6 in1
integer_sub integer_sub integer_add 6 in1 integer_sub
integer_sub in1 5 integer_add integer_add integer_mult in1
integer_mult)
Lexicase best errors: [0.0 0.0 8.0 18.0 24.0 20.0 0.0 42.0
112.0 216.0]
Lexicase best number of elite cases: 3
Lexicase best total error: 440.0
Lexicase best mean error: 44.0
Lexicase best size: 32
Percent parens: 0.031
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction in1} {:close
0, :instruction integer_add} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction integer_div} {:close 0, :instruction 6}
{:close 1, :instruction in1} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 2, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction 6} {:close 1, :instruction
in1} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 2, :instruction
integer_sub} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_mult} {:close 1, :instruction
in1} {:close 0, :instruction integer_mult} {:close 1,
:instruction integer_mult})
Zero cases best program: (in1 integer_add integer_div in1
integer_add integer_div integer_div 6 in1 integer_sub
integer_sub integer_add integer_add integer_sub integer_add 6
in1 integer_sub integer_sub integer_add integer_add integer_sub
integer_add in1 5 integer_add integer_add integer_mult in1
integer_mult integer_mult)
Zero cases best partial simplification: (in1 in1 6 in1
integer_sub integer_sub 6 in1 integer_sub integer_sub
integer_add in1 5 integer_add integer_add in1 integer_mult)
Zero cases best errors: [0.0 0.0 8.0 18.0 24.0 20.0 0.0 42.0
112.0 216.0]
Zero cases best number of elite cases: 3
Zero cases best number of zero cases: 3
Zero cases best total error: 440.0
Zero cases best mean error: 44.0
Zero cases best size: 32
Percent parens: 0.031
--- Lexicase Population Statistics ---
Count of elite individuals by case: (92 35 16 7 5 9 9 2 14 7)
Population mean number of elite cases: 1.96
Count of perfect (error zero) individuals by case: (92 35 16 7
5 9 9 2 14 0)
Population mean number of perfect (error zero) cases: 1.89
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 1, :instruction 6} {:close
2, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 2, :instruction in1} {:close 3,
:instruction 5} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 1, :instruction in1} {:close 0, :instruction
integer_mult} {:close 1, :instruction integer_mult})
Best program: (5 4 in1 integer_sub in1 integer_mult integer_sub
integer_div integer_mult 6 integer_sub integer_add in1 5
integer_add integer_add integer_mult in1 integer_mult
integer_mult)
Partial simplification: (5 4 in1 integer_sub in1 integer_mult
integer_sub 6 integer_sub integer_add in1 5 integer_add
integer_add in1 integer_mult)
Errors: [0.0 4.0 6.0 6.0 4.0 0.0 6.0 14.0 24.0 36.0]
Total: 100.0
Mean: 10.0
Genome size: 20
Size: 21
Percent parens: 0.048
--- Population Statistics ---
Average total errors in population: 2136.46
Median total errors in population: 700.0
Error averages by case: (0.22 8.39 21.88 40.08 66.06 107.0
181.61 306.96 528.49 875.77)
Error minima by case: (0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 2.0)
Average genome size in population (length): 34.32
Average program size in population (points): 35.32
Average percent parens in population: 0.032
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 14
Genome diversity (% unique Plush genomes):       0.47
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 14
Syntactic diversity (% unique Push programs):    0.47
Total error diversity:                           0.34
Error (vector) diversity:                        0.35
--- Run Statistics ---
Number of program evaluations used so far: 4400
Number of point (instruction) evaluations so far: 1450021
--- Timings ---
Current time: 1457558117392 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 13
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 14
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 14
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction 6} {:close 1, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_add} {:close 2, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction 6} {:close 1, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_add} {:close 2, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 2, :instruction in1} {:close
3, :instruction 5} {:close 0, :instruction integer_add} {:close
0, :instruction integer_add} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_mult} {:close 1,
:instruction in1} {:close 0, :instruction integer_mult} {:close
1, :instruction integer_mult})
Lexicase best program: (in1 integer_add integer_div in1
integer_sub integer_div 6 in1 integer_sub integer_sub
integer_add integer_add integer_sub integer_add 5 4 in1
integer_sub in1 integer_mult integer_sub integer_div 6 in1
integer_sub integer_sub integer_add integer_add integer_sub
integer_add in1 5 integer_add integer_add integer_add
integer_mult in1 integer_mult integer_mult)
Lexicase best partial simplification: (in1 in1 integer_sub
integer_div 6 in1 integer_sub integer_sub 5 4 in1 integer_sub
in1 integer_mult integer_sub integer_div 6 in1 integer_sub
integer_sub integer_sub in1 5 integer_add integer_add in1
integer_mult)
Lexicase best errors: [0.0 1.0 0.0 6.0 0.0 25.0 72.0 147.0
256.0 405.0]
Lexicase best number of elite cases: 3
Lexicase best total error: 912.0
Lexicase best mean error: 91.2
Lexicase best size: 40
Percent parens: 0.025
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction in1} {:close
0, :instruction integer_add} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_add} {:close 2, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 0, :instruction 5}
{:close 1, :instruction 4} {:close 0, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction in1}
{:close 1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction 6} {:close 1, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_add} {:close 2, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 2, :instruction in1} {:close
3, :instruction 5} {:close 0, :instruction integer_add} {:close
0, :instruction integer_add} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_mult} {:close 1,
:instruction in1} {:close 0, :instruction integer_mult} {:close
1, :instruction integer_mult})
Zero cases best program: (in1 integer_add integer_div in1
integer_sub integer_div 6 in1 integer_sub integer_sub
integer_add integer_add integer_sub integer_add 5 4 in1
integer_sub in1 integer_mult integer_sub integer_div 6 in1
integer_sub integer_sub integer_add integer_add integer_sub
integer_add in1 5 integer_add integer_add integer_add
integer_mult in1 integer_mult integer_mult)
Zero cases best partial simplification: (in1 in1 integer_sub 6
in1 integer_sub integer_sub 5 4 in1 integer_sub in1
integer_mult integer_sub integer_div 6 in1 integer_sub
integer_sub integer_add in1 5 integer_add integer_add in1
integer_mult)
Zero cases best errors: [0.0 1.0 0.0 6.0 0.0 25.0 72.0 147.0
256.0 405.0]
Zero cases best number of elite cases: 3
Zero cases best number of zero cases: 3
Zero cases best total error: 912.0
Zero cases best mean error: 91.2
Zero cases best size: 40
Percent parens: 0.025
--- Lexicase Population Statistics ---
Count of elite individuals by case: (89 33 25 15 6 9 13 6 7 6)
Population mean number of elite cases: 2.09
Count of perfect (error zero) individuals by case: (89 33 25 15
6 9 13 6 7 0)
Population mean number of perfect (error zero) cases: 2.03
--- Best Program (based on total-error) Statistics ---
```

```
Best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 1, :instruction 6} {:close
2, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 2, :instruction in1} {:close 3,
:instruction 5} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 1, :instruction in1} {:close 0, :instruction
integer_mult} {:close 1, :instruction integer_mult})
Best program: (5 4 in1 integer_sub in1 integer_mult integer_sub
integer_div integer_mult 6 integer_sub integer_add in1 5
integer_add integer_add integer_mult in1 integer_mult
integer_mult)
Partial simplification: (5 4 in1 integer_sub in1 integer_mult
integer_sub 6 integer_sub in1 5 integer_add integer_add in1
integer_mult)
Errors: [0.0 4.0 6.0 6.0 4.0 0.0 6.0 14.0 24.0 36.0]
Total: 100.0
Mean: 10.0
Genome size: 20
Size: 21
Percent parens: 0.048
--- Population Statistics ---
Average total errors in population: 13963.69
Median total errors in population: 629.0
Error averages by case: (10.33 16.82 36.44 98.83 255.14 581.23
1175.24 2162.85 3693.42 5933.39)
Error minima by case: (0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 2.0)
Average genome size in population (length): 33.72
Average program size in population (points): 34.72
Average percent parens in population: 0.033
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 12
Genome diversity (% unique Plush genomes):       0.44
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 12
Syntactic diversity (% unique Push programs):    0.44
Total error diversity:                           0.34
Error (vector) diversity:                        0.37
--- Run Statistics ---
Number of program evaluations used so far: 4500
Number of point (instruction) evaluations so far: 1483741
--- Timings ---
Current time: 1457558118050 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 14
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 15
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 15
```

```
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction integer_mult} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_mult} {:close 1,
:instruction 6} {:close 0, :instruction integer_mult} {:close
0, :instruction 9} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_div} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_add}
{:close 1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_div} {:close 0, :instruction
in1} {:close 0, :instruction integer_div})
Lexicase best program: (in1 integer_add integer_div in1
integer_mult integer_sub integer_add integer_div integer_mult
integer_mult integer_mult 6 integer_mult 9 integer_sub
integer_div integer_div integer_sub integer_add integer_sub
integer_sub integer_mult in1 integer_mult integer_add
integer_mult integer_sub integer_div integer_mult integer_mult
in1 integer_add integer_sub integer_div in1 integer_div)
Lexicase best partial simplification: (in1 in1 integer_mult 6
integer_mult 9 integer_sub in1 integer_mult in1 integer_add
integer_div in1 integer_div)
Lexicase best errors: [0.0 0.0 18.0 40.0 60.0 72.0 70.0 48.0
0.0 80.0]
Lexicase best number of elite cases: 3
Lexicase best total error: 388.0
Lexicase best mean error: 38.8
Lexicase best size: 37
Percent parens: 0.027
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction in1} {:close
0, :instruction integer_add} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_mult} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 1, :instruction
integer_div} {:close 0, :instruction integer_mult} {:close 1,
:instruction integer_mult} {:close 0, :instruction
integer_mult} {:close 1, :instruction 6} {:close 0,
:instruction integer_mult} {:close 0, :instruction 9} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_mult} {:close 1,
:instruction in1} {:close 0, :instruction integer_mult} {:close
0, :instruction integer_add} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_div} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_mult} {:close 0, :instruction
in1} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_div})
Zero cases best program: (in1 integer_add integer_div in1
integer_mult integer_sub integer_add integer_div integer_mult
integer_mult integer_mult 6 integer_mult 9 integer_sub
integer_div integer_div integer_sub integer_add integer_sub
integer_sub integer_mult in1 integer_mult integer_add
integer_mult integer_sub integer_div integer_mult integer_mult
in1 integer_add integer_sub integer_div in1 integer_div)
Zero cases best partial simplification: (in1 in1 integer_mult 6
integer_mult 9 integer_sub integer_div in1 integer_mult in1
integer_add in1 integer_div)
Zero cases best errors: [0.0 0.0 18.0 40.0 60.0 72.0 70.0 48.0
0.0 80.0]
Zero cases best number of elite cases: 3
Zero cases best number of zero cases: 3
Zero cases best total error: 388.0
Zero cases best mean error: 38.8
Zero cases best size: 37
Percent parens: 0.027
--- Lexicase Population Statistics ---
Count of elite individuals by case: (90 28 15 7 10 14 11 13 10
5)
Population mean number of elite cases: 2.03
Count of perfect (error zero) individuals by case: (90 28 15 7
10 14 11 13 10 0)
Population mean number of perfect (error zero) cases: 1.98
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 1, :instruction 6} {:close
2, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 2, :instruction in1} {:close 3,
:instruction 5} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 1, :instruction in1} {:close 0, :instruction
integer_mult} {:close 1, :instruction integer_mult})
Best program: (5 4 in1 integer_sub in1 integer_mult integer_sub
integer_div integer_mult 6 integer_sub integer_add in1 5
integer_add integer_add integer_mult in1 integer_mult
integer_mult)
Partial simplification: (5 4 in1 integer_sub in1 integer_mult
integer_sub 6 integer_sub in1 5 integer_add integer_add in1
integer_mult)
Errors: [0.0 4.0 6.0 6.0 4.0 0.0 6.0 14.0 24.0 36.0]
Total: 100.0
Mean: 10.0
Genome size: 20
Size: 21
Percent parens: 0.048
--- Population Statistics ---
Average total errors in population: 2476.84
Median total errors in population: 808.0
Error averages by case: (0.46 7.41 22.44 47.18 85.04 146.11
240.07 383.25 609.51 935.37)
Error minima by case: (0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 2.0)
Average genome size in population (length): 33.44
Average program size in population (points): 34.44
Average percent parens in population: 0.031
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 13
Genome diversity (% unique Plush genomes):       0.45
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 13
Syntactic diversity (% unique Push programs):    0.45
Total error diversity:                           0.35
Error (vector) diversity:                        0.36
--- Run Statistics ---
Number of program evaluations used so far: 4600
Number of point (instruction) evaluations so far: 1517181
--- Timings ---
Current time: 1457558118614 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 15
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 16
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 16
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction 6} {:close 1, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_add} {:close 2, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction 6} {:close 1, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_add} {:close 2, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 2, :instruction in1} {:close
3, :instruction 5} {:close 0, :instruction integer_add} {:close
0, :instruction integer_add} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_mult} {:close 1,
:instruction in1} {:close 0, :instruction integer_mult} {:close
1, :instruction integer_mult})
Lexicase best program: (in1 integer_add integer_div in1
integer_sub integer_div 6 in1 integer_sub integer_sub
integer_add integer_add integer_sub integer_add 5 4 in1
integer_sub in1 integer_mult integer_sub integer_div 6 in1
integer_sub integer_sub integer_add integer_add integer_sub
integer_add in1 5 integer_add integer_add integer_add
integer_mult in1 integer_mult integer_mult)
Lexicase best partial simplification: (in1 in1 integer_sub
integer_div 6 in1 integer_sub integer_sub 5 4 in1 integer_sub
in1 integer_mult integer_sub integer_div 6 in1 integer_sub
integer_sub integer_add integer_add in1 5 integer_add
integer_add in1 integer_mult)
Lexicase best errors: [0.0 1.0 0.0 6.0 0.0 25.0 72.0 147.0
256.0 405.0]
Lexicase best number of elite cases: 3
Lexicase best total error: 912.0
Lexicase best mean error: 91.2
Lexicase best size: 40
Percent parens: 0.025
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction in1} {:close
0, :instruction integer_add} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_add} {:close 2, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 0, :instruction 5}
{:close 1, :instruction 4} {:close 0, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction in1}
{:close 1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction 6} {:close 1, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_add} {:close 2, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 2, :instruction in1} {:close
3, :instruction 5} {:close 0, :instruction integer_add} {:close
0, :instruction integer_add} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_mult} {:close 1,
:instruction in1} {:close 0, :instruction integer_mult} {:close
1, :instruction integer_mult})
Zero cases best program: (in1 integer_add integer_div in1
integer_sub integer_div 6 in1 integer_sub integer_sub
integer_add integer_add integer_sub integer_add 5 4 in1
integer_sub in1 integer_mult integer_sub integer_div 6 in1
integer_sub integer_sub integer_add integer_add integer_sub
integer_add in1 5 integer_add integer_add integer_add
integer_mult in1 integer_mult integer_mult)
Zero cases best partial simplification: (in1 in1 integer_sub 6
in1 integer_sub integer_sub 5 4 in1 integer_sub in1
integer_mult integer_sub integer_div 6 in1 integer_sub
integer_sub in1 5 integer_add integer_add integer_mult in1
integer_mult)
Zero cases best errors: [0.0 1.0 0.0 6.0 0.0 25.0 72.0 147.0
256.0 405.0]
Zero cases best number of elite cases: 3
Zero cases best number of zero cases: 3
Zero cases best total error: 912.0
Zero cases best mean error: 91.2
Zero cases best size: 40
Percent parens: 0.025
--- Lexicase Population Statistics ---
Count of elite individuals by case: (93 29 33 13 20 10 6 5 10
1)
Population mean number of elite cases: 2.20
Count of perfect (error zero) individuals by case: (93 29 33 13
20 10 6 5 10 1)
```

```
Population mean number of perfect (error zero) cases: 2.20
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 1, :instruction 6} {:close
2, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 2, :instruction in1} {:close 3,
:instruction 5} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 1, :instruction in1} {:close 0, :instruction
integer_mult} {:close 1, :instruction integer_mult})
Best program: (5 4 in1 integer_sub in1 integer_mult integer_sub
integer_div integer_mult 6 integer_sub integer_add in1 5
integer_add integer_add integer_mult in1 integer_mult
integer_mult)
Partial simplification: (5 4 in1 integer_sub in1 integer_mult
integer_sub 6 integer_sub in1 5 integer_add integer_add in1
integer_mult)
Errors: [0.0 4.0 6.0 6.0 4.0 0.0 6.0 14.0 24.0 36.0]
Total: 100.0
Mean: 10.0
Genome size: 20
Size: 21
Percent parens: 0.048
--- Population Statistics ---
Average total errors in population: 12880.94
Median total errors in population: 700.0
Error averages by case: (0.31 43.56 126.7 265.97 482.12 820.33
1329.48 2072.91 3132.78 4606.78)
Error minima by case: (0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0)
Average genome size in population (length): 35.52
Average program size in population (points): 36.52
Average percent parens in population: 0.029
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 16
Genome diversity (% unique Plush genomes):       0.46
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 16
Syntactic diversity (% unique Push programs):    0.46
Total error diversity:                           0.33
Error (vector) diversity:                        0.34
--- Run Statistics ---
Number of program evaluations used so far: 4700
Number of point (instruction) evaluations so far: 1552701
--- Timings ---
Current time: 1457558119302 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 16
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 17
Computing errors... Done computing errors.
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 17
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction in1} {:close 0, :instruction
integer_add} {:close 1, :instruction integer_div} {:close 0,
:instruction integer_div} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction in1} {:close 0, :instruction
integer_div} {:close 0, :instruction in1} {:close 1,
:instruction integer_mult} {:close 0, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction in1}
{:close 1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction 6} {:close 1, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_add} {:close 2, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 2, :instruction in1} {:close
3, :instruction 5} {:close 0, :instruction integer_add} {:close
0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult})
Lexicase best program: (in1 integer_add integer_div in1
integer_add integer_div integer_div 6 in1 integer_sub
integer_sub integer_add integer_add integer_sub integer_add in1
integer_div in1 integer_mult in1 integer_sub in1 integer_mult
integer_sub integer_div 6 in1 integer_sub integer_sub
integer_add integer_add integer_sub integer_add in1 5
integer_add integer_add integer_mult in1 integer_mult
integer_mult)
Lexicase best partial simplification: (in1 in1 6 in1
integer_sub integer_sub integer_add in1 integer_div in1
integer_mult in1 integer_sub in1 integer_mult 6 in1 integer_sub
integer_sub in1 5 integer_add integer_add in1 integer_mult)
Lexicase best errors: [0.0 1.0 0.0 9.0 0.0 25.0 144.0 196.0
256.0 324.0]
Lexicase best number of elite cases: 3
Lexicase best total error: 955.0
Lexicase best mean error: 95.5
Lexicase best size: 42
Percent parens: 0.024
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction in1} {:close
0, :instruction integer_add} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 1, :instruction integer_div}
{:close 0, :instruction integer_div} {:close 0, :instruction 6}
{:close 1, :instruction in1} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 2, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction in1} {:close 0,
:instruction integer_div} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 0, :instruction in1}
{:close 0, :instruction integer_sub} {:close 0, :instruction
in1} {:close 1, :instruction integer_mult} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_add} {:close 2, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 2, :instruction
in1} {:close 3, :instruction 5} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_mult} {:close 1, :instruction in1} {:close
0, :instruction integer_mult} {:close 1, :instruction
integer_mult})
Zero cases best program: (in1 integer_add integer_div in1
integer_add integer_div integer_div 6 in1 integer_sub
integer_sub integer_add integer_add integer_sub integer_add in1
integer_div in1 integer_mult in1 integer_sub in1 integer_mult
integer_sub integer_div 6 in1 integer_sub integer_sub
integer_add integer_add integer_sub integer_add in1 5
integer_add integer_add integer_mult in1 integer_mult
integer_mult)
Zero cases best partial simplification: (in1 integer_add in1
integer_add 6 in1 integer_sub integer_sub integer_add in1
integer_div in1 integer_mult in1 integer_sub in1 integer_mult 6
in1 integer_sub integer_sub in1 5 integer_add integer_add
integer_mult in1 integer_mult)
Zero cases best errors: [0.0 1.0 0.0 9.0 0.0 25.0 144.0 196.0
256.0 324.0]
Zero cases best number of elite cases: 3
Zero cases best number of zero cases: 3
Zero cases best total error: 955.0
Zero cases best mean error: 95.5
Zero cases best size: 42
Percent parens: 0.024
--- Lexicase Population Statistics ---
Count of elite individuals by case: (95 28 25 18 11 6 11 11 8
13)
Population mean number of elite cases: 2.26
Count of perfect (error zero) individuals by case: (95 28 25 18
11 6 11 11 8 13)
Population mean number of perfect (error zero) cases: 2.26
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 1, :instruction 6} {:close
2, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 2, :instruction in1} {:close 3,
:instruction 5} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 1, :instruction in1} {:close 0, :instruction
integer_mult} {:close 1, :instruction integer_mult})
Best program: (5 4 in1 integer_sub in1 integer_mult integer_sub
integer_div integer_mult 6 integer_sub integer_add in1 5
integer_add integer_add integer_mult in1 integer_mult
integer_mult)
Partial simplification: (5 4 in1 integer_sub in1 integer_mult
integer_sub 6 integer_sub in1 5 integer_add integer_add in1
integer_mult)
Errors: [0.0 4.0 6.0 6.0 4.0 0.0 6.0 14.0 24.0 36.0]
```

```
Total: 100.0
Mean: 10.0
Genome size: 20
Size: 21
Percent parens: 0.048
--- Population Statistics ---
Average total errors in population: 5010.81
Median total errors in population: 559.0
Error averages by case: (0.23 8.59 22.76 38.54 44.35 148.61
223.13 632.2 1352.49 2539.91)
Error minima by case: (0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0)
Average genome size in population (length): 35.36
Average program size in population (points): 36.36
Average percent parens in population: 0.031
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 13
Genome diversity (% unique Plush genomes):       0.48
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 13
Syntactic diversity (% unique Push programs):    0.48
Total error diversity:                           0.36
Error (vector) diversity:                        0.37
--- Run Statistics ---
Number of program evaluations used so far: 4800
Number of point (instruction) evaluations so far: 1588061
--- Timings ---
Current time: 1457558119975 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 17
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 18
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 18
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult})
Lexicase best program: (5 4 in1 integer_sub in1 integer_mult
integer_div in1 integer_sub integer_div 6 in1 integer_sub
integer_sub integer_sub integer_add integer_mult integer_sub
integer_div integer_add integer_sub integer_add 5 4 in1
integer_sub in1 integer_mult integer_sub integer_add 6 in1
integer_sub integer_sub integer_add integer_add in1 5
integer_add integer_add integer_add integer_mult in1
integer_mult integer_mult)
Lexicase best partial simplification: (5 4 in1 integer_sub in1
integer_mult integer_div in1 integer_sub integer_div 6 in1
integer_sub integer_sub integer_div 5 4 in1 integer_sub in1
integer_mult integer_sub 6 in1 integer_sub integer_sub
integer_add in1 5 integer_add integer_add in1 integer_mult)
Lexicase best errors: [0.0 0.0 0.0 0.0 8.0 10.0 6.0 7.0 8.0
9.0]
Lexicase best number of elite cases: 4
Lexicase best total error: 48.0
Lexicase best mean error: 4.8
Lexicase best size: 46
Percent parens: 0.022
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult})
Zero cases best program: (5 4 in1 integer_sub in1 integer_mult
integer_div in1 integer_sub integer_div 6 in1 integer_sub
integer_sub integer_sub integer_add integer_mult integer_sub
integer_div integer_add integer_sub integer_add 5 4 in1
integer_sub in1 integer_mult integer_sub integer_add 6 in1
integer_sub integer_sub integer_add integer_add in1 5
integer_add integer_add integer_add integer_mult in1
integer_mult integer_mult)
Zero cases best partial simplification: (5 4 in1 integer_sub
in1 integer_mult integer_div in1 integer_sub integer_div 6 in1
integer_sub integer_sub integer_add integer_add 5 4 in1
integer_sub in1 integer_mult integer_sub integer_add 6 in1
integer_sub integer_sub in1 5 integer_add integer_add
integer_mult in1 integer_mult)
Zero cases best errors: [0.0 0.0 0.0 0.0 8.0 10.0 6.0 7.0 8.0
9.0]
Zero cases best number of elite cases: 4
Zero cases best number of zero cases: 4
Zero cases best total error: 48.0
Zero cases best mean error: 4.8
Zero cases best size: 46
Percent parens: 0.022
--- Lexicase Population Statistics ---
Count of elite individuals by case: (95 21 24 15 13 10 11 14 2
8)
Population mean number of elite cases: 2.13
Count of perfect (error zero) individuals by case: (95 21 24 15
13 10 11 14 2 8)
Population mean number of perfect (error zero) cases: 2.13
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult})
Best program: (5 4 in1 integer_sub in1 integer_mult integer_div
in1 integer_sub integer_div 6 in1 integer_sub integer_sub
integer_sub integer_add integer_mult integer_sub integer_div
integer_add integer_sub integer_add 5 4 in1 integer_sub in1
integer_mult integer_sub integer_add 6 in1 integer_sub
integer_sub integer_add integer_add in1 5 integer_add
integer_add integer_add integer_mult in1 integer_mult
integer_mult)
Partial simplification: (5 4 in1 integer_sub in1 integer_mult
integer_div in1 integer_sub integer_div 6 in1 integer_sub
integer_sub integer_add 5 4 in1 integer_sub in1 integer_mult
integer_sub 6 in1 integer_sub integer_sub integer_add in1 5
integer_add integer_add in1 integer_mult)
Errors: [0.0 0.0 0.0 0.0 8.0 10.0 6.0 7.0 8.0 9.0]
Total: 48.0
Mean: 4.8
Genome size: 45
Size: 46
Percent parens: 0.022
--- Population Statistics ---
Average total errors in population: 2144.2
Median total errors in population: 440.0
Error averages by case: (0.4 8.42 13.69 22.79 43.9 91.73 179.26
319.49 555.83 908.69)
Error minima by case: (0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0)
Average genome size in population (length): 33.51
Average program size in population (points): 34.51
Average percent parens in population: 0.034
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 9
Genome diversity (% unique Plush genomes):       0.52
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 9
Syntactic diversity (% unique Push programs):    0.52
Total error diversity:                           0.39
Error (vector) diversity:                        0.4
--- Run Statistics ---
Number of program evaluations used so far: 4900
Number of point (instruction) evaluations so far: 1621571
--- Timings ---
Current time: 1457558120763 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 18
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 19
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 19
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult})
Lexicase best program: (5 4 in1 integer_sub in1 integer_mult
integer_div in1 integer_sub integer_div 6 in1 integer_sub
integer_sub integer_sub integer_add integer_mult integer_sub
integer_div integer_add integer_sub integer_add 5 4 in1
integer_sub in1 integer_mult integer_sub integer_add 6 in1
integer_sub integer_sub integer_add integer_add in1 5
integer_add integer_add integer_add integer_mult in1
integer_mult integer_mult)
Lexicase best partial simplification: (5 4 in1 integer_sub in1
integer_mult integer_div in1 integer_sub integer_div 6 in1
integer_sub integer_sub integer_add 5 4 in1 integer_sub in1
integer_mult integer_sub 6 in1 integer_sub integer_sub
integer_add in1 5 integer_add integer_add in1 integer_mult)
Lexicase best errors: [0.0 0.0 0.0 0.0 8.0 10.0 6.0 7.0 8.0
9.0]
Lexicase best number of elite cases: 4
Lexicase best total error: 48.0
Lexicase best mean error: 4.8
Lexicase best size: 46
Percent parens: 0.022
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult})
```

```
Zero cases best program: (5 4 in1 integer_sub in1 integer_mult
integer_div in1 integer_sub integer_div 6 in1 integer_sub
integer_sub integer_sub integer_add integer_mult integer_sub
integer_div integer_add integer_sub integer_add 5 4 in1
integer_sub in1 integer_mult integer_sub integer_add 6 in1
integer_sub integer_sub integer_add integer_add in1 5
integer_add integer_add integer_add integer_mult in1
integer_mult integer_mult)
Zero cases best partial simplification: (5 4 in1 integer_sub
in1 integer_mult integer_div in1 integer_sub integer_div 6 in1
integer_sub integer_sub integer_sub integer_mult integer_add 5
4 in1 integer_sub in1 integer_mult integer_sub integer_add 6
in1 integer_sub integer_sub in1 5 integer_add integer_add
integer_mult in1 integer_mult)
Zero cases best errors: [0.0 0.0 0.0 0.0 8.0 10.0 6.0 7.0 8.0
9.0]
Zero cases best number of elite cases: 4
Zero cases best number of zero cases: 4
Zero cases best total error: 48.0
Zero cases best mean error: 4.8
Zero cases best size: 46
Percent parens: 0.022
--- Lexicase Population Statistics ---
Count of elite individuals by case: (94 40 27 18 16 17 12 4 8
5)
Population mean number of elite cases: 2.41
Count of perfect (error zero) individuals by case: (94 40 27 18
16 17 12 4 8 5)
Population mean number of perfect (error zero) cases: 2.41
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult})
Best program: (5 4 in1 integer_sub in1 integer_mult integer_div
in1 integer_sub integer_div 6 in1 integer_sub integer_sub
integer_sub integer_add integer_mult integer_sub integer_div
integer_add integer_sub integer_add 5 4 in1 integer_sub in1
```

```
integer_mult integer_sub integer_add 6 in1 integer_sub
integer_sub integer_add integer_add in1 5 integer_add
integer_add integer_add integer_mult in1 integer_mult
integer_mult)
Partial simplification: (5 4 in1 integer_sub in1 integer_mult
integer_div in1 integer_sub integer_div 6 in1 integer_sub
integer_sub 5 4 in1 integer_sub in1 integer_mult integer_sub 6
in1 integer_sub integer_sub integer_add in1 5 integer_add
integer_add integer_mult in1 integer_mult)
Errors: [0.0 0.0 0.0 0.0 8.0 10.0 6.0 7.0 8.0 9.0]
Total: 48.0
Mean: 4.8
Genome size: 45
Size: 46
Percent parens: 0.022
--- Population Statistics ---
Average total errors in population: 2570.15
Median total errors in population: 440.0
Error averages by case: (0.36 17.31 20.81 27.06 50.96 109.35
214.7 381.37 655.48 1092.75)
Error minima by case: (0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0)
Average genome size in population (length): 36.13
Average program size in population (points): 37.13
Average percent parens in population: 0.030
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 15
Genome diversity (% unique Plush genomes):       0.48
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 15
Syntactic diversity (% unique Push programs):    0.48
Total error diversity:                           0.4
Error (vector) diversity:                        0.4
--- Run Statistics ---
Number of program evaluations used so far: 5000
Number of point (instruction) evaluations so far: 1657701
--- Timings ---
Current time: 1457558121547 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 19
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 20
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 20
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult})
Lexicase best program: (5 4 in1 integer_sub in1 integer_mult
integer_div in1 integer_sub integer_div 6 in1 integer_sub
integer_sub integer_sub integer_add integer_mult integer_sub
integer_div integer_add integer_sub integer_add 5 4 in1
integer_sub in1 integer_mult integer_sub integer_add 6 in1
integer_sub integer_sub integer_add integer_add in1 5
integer_add integer_add integer_add integer_mult in1
integer_mult integer_mult)
Lexicase best partial simplification: (5 4 in1 integer_sub in1
integer_mult integer_div in1 integer_sub integer_div 6 in1
integer_sub integer_sub integer_add 5 4 in1 integer_sub in1
integer_mult integer_sub 6 in1 integer_sub integer_sub
integer_add in1 5 integer_add integer_add in1 integer_mult)
Lexicase best errors: [0.0 0.0 0.0 0.0 8.0 10.0 6.0 7.0 8.0
9.0]
Lexicase best number of elite cases: 4
Lexicase best total error: 48.0
Lexicase best mean error: 4.8
Lexicase best size: 46
Percent parens: 0.022
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult})
Zero cases best program: (5 4 in1 integer_sub in1 integer_mult
integer_div in1 integer_sub integer_div 6 in1 integer_sub
integer_sub integer_sub integer_add integer_mult integer_sub
integer_div integer_add integer_sub integer_add 5 4 in1
integer_sub in1 integer_mult integer_sub integer_add 6 in1
integer_sub integer_sub integer_add integer_add in1 5
integer_add integer_add integer_add integer_mult in1
integer_mult integer_mult)
Zero cases best partial simplification: (5 4 in1 integer_sub
in1 integer_mult integer_div in1 integer_sub integer_div 6 in1
integer_sub integer_sub integer_add integer_mult integer_sub 5
4 in1 integer_sub in1 integer_mult integer_sub 6 in1
integer_sub integer_sub integer_add in1 5 integer_add
integer_add in1 integer_mult)
Zero cases best errors: [0.0 0.0 0.0 0.0 8.0 10.0 6.0 7.0 8.0
9.0]
Zero cases best number of elite cases: 4
Zero cases best number of zero cases: 4
Zero cases best total error: 48.0
Zero cases best mean error: 4.8
Zero cases best size: 46
Percent parens: 0.022
--- Lexicase Population Statistics ---
Count of elite individuals by case: (93 38 19 12 10 16 10 5 10
6)
Population mean number of elite cases: 2.19
Count of perfect (error zero) individuals by case: (93 38 19 12
10 16 10 5 10 6)
Population mean number of perfect (error zero) cases: 2.19
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
```

```
integer_mult})
Best program: (5 4 in1 integer_sub in1 integer_mult integer_div
in1 integer_sub integer_div 6 in1 integer_sub integer_sub
integer_sub integer_add integer_mult integer_sub integer_div
integer_add integer_sub integer_add 5 4 in1 integer_sub in1
integer_mult integer_sub integer_add 6 in1 integer_sub
integer_sub integer_add integer_add in1 5 integer_add
integer_add integer_add integer_mult in1 integer_mult
integer_mult)
Partial simplification: (5 4 in1 integer_sub in1 integer_mult
integer_div in1 integer_sub integer_div 6 in1 integer_sub
integer_sub integer_add 5 4 in1 integer_sub in1 integer_mult
integer_sub 6 in1 integer_sub integer_sub integer_add in1 5
integer_add integer_add in1 integer_mult)
Errors: [0.0 0.0 0.0 0.0 8.0 10.0 6.0 7.0 8.0 9.0]
Total: 48.0
Mean: 4.8
Genome size: 45
Size: 46
Percent parens: 0.022
--- Population Statistics ---
Average total errors in population: 847.99
Median total errors in population: 422.0
Error averages by case: (0.57 8.24 14.33 18.66 24.74 40.84
75.09 121.15 202.43 341.94)
Error minima by case: (0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0)
Average genome size in population (length): 34.31
Average program size in population (points): 35.3
Average percent parens in population: 0.032
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 10
Genome diversity (% unique Plush genomes):       0.53
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 10
Syntactic diversity (% unique Push programs):    0.53
Total error diversity:                           0.47
Error (vector) diversity:                        0.47
--- Run Statistics ---
Number of program evaluations used so far: 5100
Number of point (instruction) evaluations so far: 1692011
--- Timings ---
Current time: 1457558122365 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 20
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 21
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 21
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 1, :instruction
```

```
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult})
Lexicase best program: (5 4 in1 integer_sub in1 integer_mult
integer_div in1 integer_sub integer_div 6 in1 integer_sub
integer_sub integer_sub integer_add integer_mult integer_sub
integer_div integer_add integer_sub integer_add 5 4 in1
integer_sub in1 integer_mult integer_sub integer_add 6 in1
integer_sub integer_sub integer_add integer_add in1 5
integer_add integer_add integer_add integer_mult in1
integer_mult integer_mult)
Lexicase best partial simplification: (5 4 in1 integer_sub in1
integer_mult integer_div in1 integer_sub integer_div 6 in1
integer_sub integer_sub integer_div 5 4 in1 integer_sub in1
integer_mult integer_sub 6 in1 integer_sub integer_sub
integer_add in1 5 integer_add integer_add in1 integer_mult)
Lexicase best errors: [0.0 0.0 0.0 0.0 8.0 10.0 6.0 7.0 8.0
9.0]
Lexicase best number of elite cases: 4
Lexicase best total error: 48.0
Lexicase best mean error: 4.8
Lexicase best size: 46
Percent parens: 0.022
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult})
Zero cases best program: (5 4 in1 integer_sub in1 integer_mult
integer_div in1 integer_sub integer_div 6 in1 integer_sub
integer_sub integer_sub integer_add integer_mult integer_sub
integer_div integer_add integer_sub integer_add 5 4 in1
integer_sub in1 integer_mult integer_sub integer_add 6 in1
integer_sub integer_sub integer_add integer_add in1 5
integer_add integer_add integer_add integer_mult in1
integer_mult integer_mult)
Zero cases best partial simplification: (5 4 in1 integer_sub
in1 integer_mult integer_div in1 integer_sub integer_div 6 in1
integer_sub integer_sub integer_add integer_mult integer_add 5
4 in1 integer_sub in1 integer_mult integer_sub integer_add 6
in1 integer_sub integer_sub integer_add in1 5 integer_add
integer_add in1 integer_mult)
Zero cases best errors: [0.0 0.0 0.0 0.0 8.0 10.0 6.0 7.0 8.0
9.0]
Zero cases best number of elite cases: 4
Zero cases best number of zero cases: 4
Zero cases best total error: 48.0
Zero cases best mean error: 4.8
Zero cases best size: 46
Percent parens: 0.022
--- Lexicase Population Statistics ---
Count of elite individuals by case: (94 34 22 15 12 17 4 11 12
8)
Population mean number of elite cases: 2.29
Count of perfect (error zero) individuals by case: (94 34 22 15
12 17 4 11 12 8)
Population mean number of perfect (error zero) cases: 2.29
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult})
Best program: (5 4 in1 integer_sub in1 integer_mult integer_div
in1 integer_sub integer_div 6 in1 integer_sub integer_sub
integer_sub integer_add integer_mult integer_sub integer_div
integer_add integer_sub integer_add 5 4 in1 integer_sub in1
integer_mult integer_sub integer_add 6 in1 integer_sub
integer_sub integer_add integer_add in1 5 integer_add
integer_add integer_add integer_mult in1 integer_mult
integer_mult)
Partial simplification: (5 4 in1 integer_sub in1 integer_mult
integer_div in1 integer_sub integer_div 6 in1 integer_sub
integer_sub 5 4 in1 integer_sub in1 integer_mult integer_sub 6
in1 integer_sub integer_sub integer_add integer_add in1 5
integer_add integer_add in1 integer_mult)
Errors: [0.0 0.0 0.0 0.0 8.0 10.0 6.0 7.0 8.0 9.0]
Total: 48.0
Mean: 4.8
Genome size: 45
Size: 46
Percent parens: 0.022
--- Population Statistics ---
Average total errors in population: 1255.51
Median total errors in population: 422.0
Error averages by case: (0.35 15.84 21.31 25.66 29.7 58.74
98.34 168.31 301.08 536.18)
Error minima by case: (0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0)
Average genome size in population (length): 36.32
Average program size in population (points): 37.32
Average percent parens in population: 0.029
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 12
Genome diversity (% unique Plush genomes):       0.48
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 12
Syntactic diversity (% unique Push programs):    0.48
Total error diversity:                           0.42
Error (vector) diversity:                        0.42
--- Run Statistics ---
Number of program evaluations used so far: 5200
Number of point (instruction) evaluations so far: 1728331
--- Timings ---
Current time: 1457558123141 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 21
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 22
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 22
```

```
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 1, :instruction 6} {:close
2, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 2, :instruction in1} {:close 3,
:instruction 5} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_add} {:close 2, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 0, :instruction 5}
{:close 1, :instruction 4} {:close 0, :instruction in1} {:close
0, :instruction in1} {:close 1, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction 6} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 1, :instruction in1} {:close 0, :instruction
integer_mult} {:close 1, :instruction integer_mult})
Lexicase best program: (5 4 in1 integer_sub in1 integer_mult
integer_sub integer_div integer_mult 6 integer_sub integer_add
in1 5 in1 integer_sub integer_div 6 in1 integer_sub integer_sub
integer_add integer_add integer_sub integer_add 5 4 in1 in1
integer_mult integer_sub integer_div 6 integer_add integer_add
integer_add integer_mult in1 integer_mult integer_mult)
Lexicase best partial simplification: (5 4 in1 integer_sub in1
integer_mult integer_sub 6 integer_sub in1 5 in1 integer_sub
integer_div 6 in1 integer_sub integer_sub integer_add 5 4 in1
in1 integer_mult integer_sub integer_div 6 integer_add
integer_add integer_add in1 integer_mult)
Lexicase best errors: [0.0 0.0 6.0 9.0 0.0 0.0 72.0 70.0 80.0
99.0]
Lexicase best number of elite cases: 4
Lexicase best total error: 336.0
Lexicase best mean error: 33.6
Lexicase best size: 41
Percent parens: 0.024
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 1, :instruction 6} {:close
2, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 2, :instruction in1} {:close 3,
:instruction 5} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_add} {:close 2, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 0, :instruction 5}
{:close 1, :instruction 4} {:close 0, :instruction in1} {:close
0, :instruction in1} {:close 1, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction 6} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 1, :instruction in1} {:close 0, :instruction
integer_mult} {:close 1, :instruction integer_mult})
Zero cases best program: (5 4 in1 integer_sub in1 integer_mult
integer_sub integer_div integer_mult 6 integer_sub integer_add
in1 5 in1 integer_sub integer_div 6 in1 integer_sub integer_sub
integer_add integer_add integer_sub integer_add 5 4 in1 in1
integer_mult integer_sub integer_div 6 integer_add integer_add
integer_add integer_mult in1 integer_mult integer_mult)
Zero cases best partial simplification: (5 4 in1 integer_sub
in1 integer_mult integer_sub in1 5 in1 integer_sub integer_div
6 in1 integer_sub integer_sub integer_add integer_add 5 4 in1
in1 integer_mult integer_sub integer_div integer_add
integer_add in1 integer_mult)
Zero cases best errors: [0.0 0.0 6.0 9.0 0.0 0.0 72.0 70.0 80.0
99.0]
Zero cases best number of elite cases: 4
Zero cases best number of zero cases: 4
Zero cases best total error: 336.0
Zero cases best mean error: 33.6
Zero cases best size: 41
Percent parens: 0.024
--- Lexicase Population Statistics ---
Count of elite individuals by case: (96 48 25 20 11 12 16 5 10
10)
Population mean number of elite cases: 2.53
Count of perfect (error zero) individuals by case: (96 48 25 20
11 12 16 5 10 10)
Population mean number of perfect (error zero) cases: 2.53
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult})
Best program: (5 4 in1 integer_sub in1 integer_mult integer_div
in1 integer_sub integer_div 6 in1 integer_sub integer_sub
```

```
integer_sub integer_add integer_mult integer_sub integer_div
integer_add integer_sub integer_add 5 4 in1 integer_sub in1
integer_mult integer_sub integer_add 6 in1 integer_sub
integer_sub integer_add integer_add in1 5 integer_add
integer_add integer_add integer_mult in1 integer_mult
integer_mult)
Partial simplification: (5 4 in1 integer_sub in1 integer_mult
integer_div in1 integer_sub integer_div 6 in1 integer_sub
integer_sub integer_sub integer_add 5 4 in1 integer_sub in1
integer_mult integer_sub integer_add 6 in1 integer_sub
integer_sub in1 5 integer_add integer_add in1 integer_mult)
Errors: [0.0 0.0 0.0 0.0 8.0 10.0 6.0 7.0 8.0 9.0]
Total: 48.0
Mean: 4.8
Genome size: 45
Size: 46
Percent parens: 0.022
--- Population Statistics ---
Average total errors in population: 2312.36
Median total errors in population: 422.0
Error averages by case: (0.3 8.06 21.16 30.53 37.9 84.19 144.08
304.03 596.79 1085.32)
Error minima by case: (0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0)
Average genome size in population (length): 37.44
Average program size in population (points): 38.44
Average percent parens in population: 0.030
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 12
Genome diversity (% unique Plush genomes):       0.43
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 12
Syntactic diversity (% unique Push programs):    0.42
Total error diversity:                           0.35
Error (vector) diversity:                        0.36
--- Run Statistics ---
Number of program evaluations used so far: 5300
Number of point (instruction) evaluations so far: 1765771
--- Timings ---
Current time: 1457558123940 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 22
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 23
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 23
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
```

```
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult})
Lexicase best program: (5 4 in1 integer_sub in1 integer_mult
integer_div in1 integer_sub integer_div 6 in1 integer_sub
integer_sub integer_sub integer_add integer_mult integer_sub
integer_div integer_add integer_sub integer_add 5 4 in1
integer_sub in1 integer_mult integer_sub integer_add 6 in1
integer_sub integer_sub integer_add integer_add in1 5
integer_add integer_add integer_add integer_mult in1
integer_mult integer_mult)
Lexicase best partial simplification: (5 4 in1 integer_sub in1
integer_mult integer_div in1 integer_sub integer_div 6 in1
integer_sub integer_sub integer_mult integer_sub 5 4 in1
integer_sub in1 integer_mult integer_sub 6 in1 integer_sub
integer_sub integer_add in1 5 integer_add integer_add in1
integer_mult)
Lexicase best errors: [0.0 0.0 0.0 0.0 8.0 10.0 6.0 7.0 8.0
9.0]
Lexicase best number of elite cases: 4
Lexicase best total error: 48.0
Lexicase best mean error: 4.8
Lexicase best size: 46
Percent parens: 0.022
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
```

```
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult})
Zero cases best program: (5 4 in1 integer_sub in1 integer_mult
integer_div in1 integer_sub integer_div 6 in1 integer_sub
integer_sub integer_sub integer_add integer_mult integer_sub
integer_div integer_add integer_sub integer_add 5 4 in1
integer_sub in1 integer_mult integer_sub integer_add 6 in1
integer_sub integer_sub integer_add integer_add in1 5
integer_add integer_add integer_add integer_mult in1
integer_mult integer_mult)
Zero cases best partial simplification: (5 4 in1 integer_sub
in1 integer_mult integer_div in1 integer_sub integer_div 6 in1
integer_sub integer_sub integer_mult integer_sub integer_div
integer_add 5 4 in1 integer_sub in1 integer_mult integer_sub 6
in1 integer_sub integer_sub in1 5 integer_add integer_add
integer_add integer_mult in1 integer_mult)
Zero cases best errors: [0.0 0.0 0.0 0.0 8.0 10.0 6.0 7.0 8.0
9.0]
Zero cases best number of elite cases: 4
Zero cases best number of zero cases: 4
Zero cases best total error: 48.0
Zero cases best mean error: 4.8
Zero cases best size: 46
Percent parens: 0.022
--- Lexicase Population Statistics ---
Count of elite individuals by case: (95 33 19 13 16 19 8 11 4
7)
Population mean number of elite cases: 2.25
Count of perfect (error zero) individuals by case: (95 33 19 13
16 19 8 11 4 7)
Population mean number of perfect (error zero) cases: 2.25
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
```

```
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult})
Best program: (5 4 in1 integer_sub in1 integer_mult integer_div
in1 integer_sub integer_div 6 in1 integer_sub integer_sub
integer_sub integer_add integer_mult integer_sub integer_div
integer_add integer_sub integer_add 5 4 in1 integer_sub in1
integer_mult integer_sub integer_add 6 in1 integer_sub
integer_sub integer_add integer_add in1 5 integer_add
integer_add integer_add integer_mult in1 integer_mult
integer_mult)
Partial simplification: (5 4 in1 integer_sub in1 integer_mult
integer_div in1 integer_sub integer_div 6 in1 integer_sub
integer_sub 5 4 in1 integer_sub in1 integer_mult integer_sub 6
in1 integer_sub integer_sub integer_add in1 5 integer_add
integer_add in1 integer_mult)
Errors: [0.0 0.0 0.0 0.0 8.0 10.0 6.0 7.0 8.0 9.0]
Total: 48.0
Mean: 4.8
Genome size: 45
Size: 46
Percent parens: 0.022
--- Population Statistics ---
Average total errors in population: 908.21
Median total errors in population: 422.0
Error averages by case: (10.33 14.06 20.29 25.87 29.78 46.73
86.9 154.3 204.78 315.17)
Error minima by case: (0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0)
Average genome size in population (length): 36.21
Average program size in population (points): 37.2
Average percent parens in population: 0.031
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 10
Genome diversity (% unique Plush genomes):       0.56
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 10
Syntactic diversity (% unique Push programs):    0.56
Total error diversity:                           0.44
Error (vector) diversity:                        0.45
--- Run Statistics ---
Number of program evaluations used so far: 5400
Number of point (instruction) evaluations so far: 1801981
--- Timings ---
Current time: 1457558124749 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 23
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 24
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 24
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 1, :instruction 6} {:close
2, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 2, :instruction in1} {:close 3,
:instruction 5} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_add} {:close 2, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 0, :instruction 5}
{:close 1, :instruction 4} {:close 0, :instruction in1} {:close
0, :instruction in1} {:close 1, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction 6} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 1, :instruction in1} {:close 0, :instruction
integer_mult} {:close 1, :instruction integer_mult})
Lexicase best program: (5 4 in1 integer_sub in1 integer_mult
integer_sub integer_div integer_mult 6 integer_sub integer_add
in1 5 in1 integer_sub integer_div 6 in1 integer_sub integer_sub
integer_add integer_add integer_sub integer_add 5 4 in1 in1
integer_mult integer_sub integer_div 6 integer_add integer_add
integer_add integer_mult in1 integer_mult integer_mult)
Lexicase best partial simplification: (5 4 in1 integer_sub in1
integer_mult integer_sub 6 integer_sub in1 5 in1 integer_sub
integer_div 6 in1 integer_sub integer_sub integer_add 5 4 in1
in1 integer_mult integer_sub integer_div 6 integer_add
integer_add integer_add integer_mult in1 integer_mult)
Lexicase best errors: [0.0 0.0 6.0 9.0 0.0 0.0 72.0 70.0 80.0
99.0]
Lexicase best number of elite cases: 4
Lexicase best total error: 336.0
Lexicase best mean error: 33.6
Lexicase best size: 41
Percent parens: 0.024
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 1, :instruction 6} {:close
2, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 2, :instruction in1} {:close 3,
:instruction 5} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_add} {:close 2, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 0, :instruction 5}
{:close 1, :instruction 4} {:close 0, :instruction in1} {:close
0, :instruction in1} {:close 1, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction 6} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 1, :instruction in1} {:close 0, :instruction
integer_mult} {:close 1, :instruction integer_mult})
Zero cases best program: (5 4 in1 integer_sub in1 integer_mult
integer_sub integer_div integer_mult 6 integer_sub integer_add
in1 5 in1 integer_sub integer_div 6 in1 integer_sub integer_sub
integer_add integer_add integer_sub integer_add 5 4 in1 in1
integer_mult integer_sub integer_div 6 integer_add integer_add
integer_add integer_mult in1 integer_mult integer_mult)
Zero cases best partial simplification: (5 4 in1 integer_sub
in1 integer_mult integer_sub integer_div 6 integer_sub in1 5
in1 integer_sub integer_div 6 in1 integer_sub integer_sub
integer_add 5 4 in1 in1 integer_mult integer_sub integer_div 6
integer_add integer_add integer_add in1 integer_mult)
Zero cases best errors: [0.0 0.0 6.0 9.0 0.0 0.0 72.0 70.0 80.0
99.0]
Zero cases best number of elite cases: 4
Zero cases best number of zero cases: 4
Zero cases best total error: 336.0
Zero cases best mean error: 33.6
Zero cases best size: 41
Percent parens: 0.024
--- Lexicase Population Statistics ---
Count of elite individuals by case: (90 48 24 15 17 18 13 3 9
10)
Population mean number of elite cases: 2.47
Count of perfect (error zero) individuals by case: (90 48 24 15
17 18 13 3 9 10)
Population mean number of perfect (error zero) cases: 2.47
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult})
Best program: (5 4 in1 integer_sub in1 integer_mult integer_div
in1 integer_sub integer_div 6 in1 integer_sub integer_sub
integer_sub integer_add integer_mult integer_sub integer_div
integer_add integer_sub integer_add 5 4 in1 integer_sub in1
```

```
integer_mult integer_sub integer_add 6 in1 integer_sub
integer_sub integer_add integer_add in1 5 integer_add
integer_add integer_add integer_mult in1 integer_mult
integer_mult)
Partial simplification: (5 4 in1 integer_sub in1 integer_mult
integer_div in1 integer_sub integer_div 6 in1 integer_sub
integer_sub 5 4 in1 integer_sub in1 integer_mult integer_sub 6
in1 integer_sub integer_sub integer_add integer_add in1 5
integer_add integer_add in1 integer_mult)
Errors: [0.0 0.0 0.0 0.0 8.0 10.0 6.0 7.0 8.0 9.0]
Total: 48.0
Mean: 4.8
Genome size: 45
Size: 46
Percent parens: 0.022
--- Population Statistics ---
Average total errors in population: 1198.28
Median total errors in population: 351.0
Error averages by case: (0.59 6.52 16.14 25.34 32.96 78.14
128.16 191.11 289.07 430.25)
Error minima by case: (0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0)
Average genome size in population (length): 37.95
Average program size in population (points): 38.95
Average percent parens in population: 0.028
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 12
Genome diversity (% unique Plush genomes):       0.52
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 12
Syntactic diversity (% unique Push programs):    0.52
Total error diversity:                           0.42
Error (vector) diversity:                        0.42
--- Run Statistics ---
Number of program evaluations used so far: 5500
Number of point (instruction) evaluations so far: 1839931
--- Timings ---
Current time: 1457558125523 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 24
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 25
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 25
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 1, :instruction 6} {:close
2, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 2, :instruction in1} {:close 3,
:instruction 5} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_add} {:close 2, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 0, :instruction 5}
{:close 1, :instruction 4} {:close 0, :instruction in1} {:close
0, :instruction in1} {:close 1, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction 6} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 1, :instruction in1} {:close 0, :instruction
integer_mult} {:close 1, :instruction integer_mult})
Lexicase best program: (5 4 in1 integer_sub in1 integer_mult
integer_sub integer_div integer_mult 6 integer_sub integer_add
in1 5 in1 integer_sub integer_div 6 in1 integer_sub integer_sub
integer_add integer_add integer_sub integer_add 5 4 in1 in1
integer_mult integer_sub integer_div 6 integer_add integer_add
integer_add integer_mult in1 integer_mult integer_mult)
Lexicase best partial simplification: (5 4 in1 integer_sub in1
integer_mult integer_sub integer_div 6 integer_sub in1 5 in1
integer_sub integer_div 6 in1 integer_sub integer_sub
integer_add integer_add integer_sub 5 4 in1 in1 integer_mult
integer_sub integer_div 6 integer_add integer_add integer_add
in1 integer_mult)
Lexicase best errors: [0.0 0.0 6.0 9.0 0.0 0.0 72.0 70.0 80.0
99.0]
Lexicase best number of elite cases: 4
Lexicase best total error: 336.0
Lexicase best mean error: 33.6
Lexicase best size: 41
Percent parens: 0.024
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 1, :instruction 6} {:close
2, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 2, :instruction in1} {:close 3,
:instruction 5} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_add} {:close 2, :instruction integer_sub}
{:close 0, :instruction integer_add} {:close 0, :instruction 5}
{:close 1, :instruction 4} {:close 0, :instruction in1} {:close
0, :instruction in1} {:close 1, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction 6} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 1, :instruction in1} {:close 0, :instruction
integer_mult} {:close 1, :instruction integer_mult})
Zero cases best program: (5 4 in1 integer_sub in1 integer_mult
integer_sub integer_div integer_mult 6 integer_sub integer_add
in1 5 in1 integer_sub integer_div 6 in1 integer_sub integer_sub
integer_add integer_add integer_sub integer_add 5 4 in1 in1
```

```
integer_mult integer_sub integer_div 6 integer_add integer_add
integer_add integer_mult in1 integer_mult integer_mult)
Zero cases best partial simplification: (5 4 in1 integer_sub
in1 integer_mult integer_sub integer_mult 6 integer_sub in1 5
in1 integer_sub integer_div 6 in1 integer_sub integer_sub
integer_add 5 4 in1 in1 integer_mult integer_sub integer_div 6
integer_add integer_add integer_add in1 integer_mult)
Zero cases best errors: [0.0 0.0 6.0 9.0 0.0 0.0 72.0 70.0 80.0
99.0]
Zero cases best number of elite cases: 4
Zero cases best number of zero cases: 4
Zero cases best total error: 336.0
Zero cases best mean error: 33.6
Zero cases best size: 41
Percent parens: 0.024
--- Lexicase Population Statistics ---
Count of elite individuals by case: (90 40 26 17 12 20 12 4 7
10)
Population mean number of elite cases: 2.38
Count of perfect (error zero) individuals by case: (90 40 26 17
12 20 12 4 7 10)
Population mean number of perfect (error zero) cases: 2.38
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 1, :instruction 6} {:close
2, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction 6} {:close 1, :instruction
in1} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 2, :instruction
integer_sub} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_mult} {:close 1, :instruction
in1} {:close 0, :instruction integer_mult})
Best program: (5 4 in1 integer_sub in1 integer_mult integer_sub
integer_div integer_mult 6 integer_sub integer_add 6 in1
integer_sub integer_sub integer_add integer_add integer_sub
integer_add in1 5 integer_add integer_add integer_mult in1
integer_mult)
Partial simplification: (5 4 in1 integer_sub in1 integer_mult
integer_sub integer_div 6 integer_sub 6 in1 integer_sub
integer_sub integer_add in1 5 integer_add integer_add
integer_mult in1 integer_mult)
Errors: [0.0 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0]
Total: 45.0
Mean: 4.5
Genome size: 27
Size: 28
Percent parens: 0.036
--- Population Statistics ---
Average total errors in population: 787.73
Median total errors in population: 336.0
Error averages by case: (0.59 3.67 7.88 13.61 20.4 36.26 75.25
121.22 199.01 309.84)
Error minima by case: (0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0)
Average genome size in population (length): 36.89
Average program size in population (points): 37.89
Average percent parens in population: 0.029
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 13
Genome diversity (% unique Plush genomes):       0.48
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 13
Syntactic diversity (% unique Push programs):    0.48
Total error diversity:                           0.41
Error (vector) diversity:                        0.41
--- Run Statistics ---
Number of program evaluations used so far: 5600
Number of point (instruction) evaluations so far: 1876821
--- Timings ---
Current time: 1457558126234 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 25
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 26
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 26
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult})
Lexicase best program: (5 4 in1 integer_sub in1 integer_mult
integer_div in1 integer_sub integer_div 6 in1 integer_sub
integer_sub integer_sub integer_add integer_mult integer_sub
integer_div integer_add integer_sub integer_add 5 4 in1
integer_sub in1 integer_mult integer_sub integer_add 6 in1
integer_sub integer_sub integer_add integer_add in1 5
integer_add integer_add integer_add integer_mult in1
integer_mult integer_mult)
Lexicase best partial simplification: (5 4 in1 integer_sub in1
integer_mult integer_div in1 integer_sub integer_div 6 in1
integer_sub integer_sub 5 4 in1 integer_sub in1 integer_mult
integer_sub integer_add 6 in1 integer_sub integer_sub in1 5
integer_add integer_add in1 integer_mult)
Lexicase best errors: [0.0 0.0 0.0 0.0 8.0 10.0 6.0 7.0 8.0
9.0]
Lexicase best number of elite cases: 4
Lexicase best total error: 48.0
Lexicase best mean error: 4.8
Lexicase best size: 46
Percent parens: 0.022
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult})
Zero cases best program: (5 4 in1 integer_sub in1 integer_mult
integer_div in1 integer_sub integer_div 6 in1 integer_sub
integer_sub integer_sub integer_add integer_mult integer_sub
integer_div integer_add integer_sub integer_add 5 4 in1
integer_sub in1 integer_mult integer_sub integer_add 6 in1
integer_sub integer_sub integer_add integer_add in1 5
integer_add integer_add integer_add integer_mult in1
integer_mult integer_mult)
Zero cases best partial simplification: (5 4 in1 integer_sub
in1 integer_mult integer_div in1 integer_sub integer_div 6 in1
integer_sub integer_sub integer_add integer_add integer_add 5 4
in1 integer_sub in1 integer_mult integer_sub 6 in1 integer_sub
integer_sub integer_add integer_add in1 5 integer_add
integer_add in1 integer_mult)
Zero cases best errors: [0.0 0.0 0.0 0.0 8.0 10.0 6.0 7.0 8.0
9.0]
Zero cases best number of elite cases: 4
Zero cases best number of zero cases: 4
Zero cases best total error: 48.0
Zero cases best mean error: 4.8
Zero cases best size: 46
Percent parens: 0.022
--- Lexicase Population Statistics ---
Count of elite individuals by case: (90 30 22 14 12 14 12 8 6
11)
Population mean number of elite cases: 2.19
Count of perfect (error zero) individuals by case: (90 30 22 14
12 14 12 8 6 11)
Population mean number of perfect (error zero) cases: 2.19
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult})
Best program: (5 4 in1 integer_add in1 integer_mult integer_div
in1 integer_sub integer_div 6 in1 integer_sub integer_sub
integer_sub integer_add integer_mult integer_sub integer_div
integer_add integer_sub integer_add 5 4 in1 integer_sub in1
integer_mult integer_sub integer_add 6 in1 integer_sub
integer_sub integer_add integer_add in1 5 integer_add
integer_add integer_add integer_mult in1 integer_mult
integer_mult)
Partial simplification: (5 4 in1 integer_add in1 integer_mult
in1 integer_sub integer_div 6 in1 integer_sub integer_sub 5 4
in1 integer_sub in1 integer_mult integer_sub 6 in1 integer_sub
integer_sub integer_add 5 integer_add integer_add integer_mult
in1 integer_mult)
Errors: [0.0 0.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0]
Total: 44.0
Mean: 4.4
Genome size: 45
Size: 46
Percent parens: 0.022
--- Population Statistics ---
Average total errors in population: 780.93
```

```
Median total errors in population: 422.0
Error averages by case: (0.77 4.43 10.53 17.76 26.03 50.46
67.14 109.23 188.89 305.69)
Error minima by case: (0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0)
Average genome size in population (length): 37.99
Average program size in population (points): 38.99
Average percent parens in population: 0.028
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 8
Genome diversity (% unique Plush genomes):       0.53
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 8
Syntactic diversity (% unique Push programs):    0.53
Total error diversity:                           0.42
Error (vector) diversity:                        0.42
--- Run Statistics ---
Number of program evaluations used so far: 5700
Number of point (instruction) evaluations so far: 1914811
--- Timings ---
Current time: 1457558126998 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 26
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 27
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 27
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult})
Lexicase best program: (5 4 in1 integer_sub in1 integer_mult
integer_div in1 integer_sub integer_div 6 in1 integer_sub
integer_sub integer_sub integer_add integer_mult integer_sub
integer_div integer_add integer_sub integer_add 5 4 in1
integer_sub in1 integer_mult integer_sub integer_add 6 in1
integer_sub integer_sub integer_add integer_add in1 5
integer_add integer_add integer_add integer_mult in1
integer_mult integer_mult)
Lexicase best partial simplification: (5 4 in1 integer_sub in1
integer_mult integer_div in1 integer_sub integer_div 6 in1
integer_sub integer_sub integer_mult integer_div integer_sub 5
4 in1 integer_sub in1 integer_mult integer_sub 6 in1
integer_sub integer_sub integer_add in1 5 integer_add
integer_add integer_mult in1 integer_mult)
Lexicase best errors: [0.0 0.0 0.0 0.0 8.0 10.0 6.0 7.0 8.0
9.0]
Lexicase best number of elite cases: 4
Lexicase best total error: 48.0
Lexicase best mean error: 4.8
Lexicase best size: 46
Percent parens: 0.022
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult})
Zero cases best program: (5 4 in1 integer_sub in1 integer_mult
integer_div in1 integer_sub integer_div 6 in1 integer_sub
integer_sub integer_sub integer_add integer_mult integer_sub
integer_div integer_add integer_sub integer_add 5 4 in1
integer_sub in1 integer_mult integer_sub integer_add 6 in1
integer_sub integer_sub integer_add integer_add in1 5
integer_add integer_add integer_add integer_mult in1
integer_mult integer_mult)
Zero cases best partial simplification: (5 4 in1 integer_sub
in1 integer_mult integer_div in1 integer_sub integer_div 6 in1
integer_sub integer_sub integer_add 5 4 in1 integer_sub in1
```

```
integer_mult integer_sub integer_add 6 in1 integer_sub
integer_sub in1 5 integer_add integer_add in1 integer_mult)
Zero cases best errors: [0.0 0.0 0.0 0.0 8.0 10.0 6.0 7.0 8.0
9.0]
Zero cases best number of elite cases: 4
Zero cases best number of zero cases: 4
Zero cases best total error: 48.0
Zero cases best mean error: 4.8
Zero cases best size: 46
Percent parens: 0.022
--- Lexicase Population Statistics ---
Count of elite individuals by case: (91 28 28 17 15 8 6 10 5
13)
Population mean number of elite cases: 2.21
Count of perfect (error zero) individuals by case: (91 28 28 17
15 8 6 10 5 13)
Population mean number of perfect (error zero) cases: 2.21
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_add} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult})
Best program: (5 4 in1 integer_add in1 integer_mult integer_div
in1 integer_sub integer_div 6 in1 integer_sub integer_sub
integer_sub integer_add integer_mult integer_sub integer_div
integer_add integer_sub integer_add 5 4 in1 integer_sub in1
integer_mult integer_sub integer_add 6 in1 integer_sub
integer_sub integer_add integer_add in1 5 integer_add
integer_add integer_add integer_mult in1 integer_mult
integer_mult)
Partial simplification: (5 4 in1 integer_add in1 integer_mult
in1 integer_sub integer_div 6 integer_sub integer_add
integer_sub 5 4 in1 integer_sub in1 integer_mult integer_sub
integer_add 6 in1 integer_sub integer_sub in1 5 integer_add
integer_add in1 integer_mult)
Errors: [0.0 0.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0]
Total: 44.0
Mean: 4.4
```

```
Genome size: 45
Size: 46
Percent parens: 0.022
--- Population Statistics ---
Average total errors in population: 592.04
Median total errors in population: 422.0
Error averages by case: (0.57 3.82 6.88 12.25 20.11 32.57 66.91
89.38 141.67 217.88)
Error minima by case: (0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0)
Average genome size in population (length): 38.25
Average program size in population (points): 39.25
Average percent parens in population: 0.029
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 10
Genome diversity (% unique Plush genomes):       0.51
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 10
Syntactic diversity (% unique Push programs):    0.51
Total error diversity:                           0.45
Error (vector) diversity:                        0.45
--- Run Statistics ---
Number of program evaluations used so far: 5800
Number of point (instruction) evaluations so far: 1953061
--- Timings ---
Current time: 1457558127782 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 27
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 28
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 28
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult})
Lexicase best program: (5 4 in1 integer_sub in1 integer_mult
integer_div in1 integer_sub integer_div 6 in1 integer_sub
integer_sub integer_sub integer_add integer_mult integer_sub
integer_div integer_add integer_sub integer_add 5 4 in1
integer_sub in1 integer_mult integer_sub integer_add 6 in1
integer_sub integer_sub integer_add integer_add in1 5
integer_add integer_add integer_add integer_mult in1
integer_mult integer_mult)
Lexicase best partial simplification: (5 4 in1 integer_sub in1
integer_mult integer_div in1 integer_sub integer_div 6 in1
integer_sub integer_sub integer_mult integer_sub integer_add 5
4 in1 integer_sub in1 integer_mult integer_sub integer_add 6
in1 integer_sub integer_sub in1 5 integer_add integer_add
integer_mult in1 integer_mult)
Lexicase best errors: [0.0 0.0 0.0 0.0 8.0 10.0 6.0 7.0 8.0
9.0]
Lexicase best number of elite cases: 4
Lexicase best total error: 48.0
Lexicase best mean error: 4.8
Lexicase best size: 46
Percent parens: 0.022
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult})
Zero cases best program: (5 4 in1 integer_sub in1 integer_mult
integer_div in1 integer_sub integer_div 6 in1 integer_sub
integer_sub integer_sub integer_add integer_mult integer_sub
integer_div integer_add integer_sub integer_add 5 4 in1
integer_sub in1 integer_mult integer_sub integer_add 6 in1
integer_sub integer_sub integer_add integer_add in1 5
```

```
integer_add integer_add integer_add integer_mult in1
integer_mult integer_mult)
Zero cases best partial simplification: (5 4 in1 integer_sub
in1 integer_mult integer_div in1 integer_sub integer_div 6 in1
integer_sub integer_sub integer_mult 5 4 in1 integer_sub in1
integer_mult integer_sub 6 in1 integer_sub integer_sub
integer_add in1 5 integer_add integer_add in1 integer_mult)
Zero cases best errors: [0.0 0.0 0.0 0.0 8.0 10.0 6.0 7.0 8.0
9.0]
Zero cases best number of elite cases: 4
Zero cases best number of zero cases: 4
Zero cases best total error: 48.0
Zero cases best mean error: 4.8
Zero cases best size: 46
Percent parens: 0.022
--- Lexicase Population Statistics ---
Count of elite individuals by case: (94 31 31 18 12 11 13 5 5
15)
Population mean number of elite cases: 2.35
Count of perfect (error zero) individuals by case: (94 31 31 18
12 11 13 5 5 15)
Population mean number of perfect (error zero) cases: 2.35
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 1, :instruction
integer_div} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction integer_div}
{:close 0, :instruction 6} {:close 1, :instruction in1} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_sub} {:close 1,
:instruction integer_add} {:close 0, :instruction integer_mult}
{:close 0, :instruction integer_sub} {:close 0, :instruction
integer_div} {:close 0, :instruction integer_add} {:close 2,
:instruction integer_sub} {:close 0, :instruction integer_add}
{:close 0, :instruction 5} {:close 1, :instruction 4} {:close
0, :instruction in1} {:close 0, :instruction integer_sub}
{:close 0, :instruction in1} {:close 1, :instruction
integer_mult} {:close 0, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 0, :instruction 6} {:close 1,
:instruction in1} {:close 0, :instruction integer_sub} {:close
0, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 0, :instruction integer_add} {:close 2,
:instruction in1} {:close 3, :instruction 5} {:close 0,
:instruction integer_add} {:close 0, :instruction integer_add}
{:close 0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult} {:close 1, :instruction
integer_mult})
Best program: (5 4 in1 integer_sub in1 integer_mult integer_div
in1 integer_sub integer_div 6 in1 integer_sub integer_sub
integer_sub integer_add integer_mult integer_sub integer_div
integer_add integer_sub integer_add 5 4 in1 integer_sub in1
integer_mult integer_sub integer_add 6 in1 integer_sub
integer_sub integer_add integer_add in1 5 integer_add
integer_add integer_add integer_mult in1 integer_mult
integer_mult)
Partial simplification: (5 4 in1 integer_sub in1 integer_mult
integer_div in1 integer_sub integer_div 6 in1 integer_sub
integer_sub 5 4 in1 integer_sub in1 integer_mult integer_sub
```

```
integer_add 6 in1 integer_sub integer_sub in1 5 integer_add
integer_add in1 integer_mult)
Errors: [0.0 0.0 0.0 0.0 8.0 10.0 6.0 7.0 8.0 9.0]
Total: 48.0
Mean: 4.8
Genome size: 45
Size: 46
Percent parens: 0.022
--- Population Statistics ---
Average total errors in population: 613.98
Median total errors in population: 422.0
Error averages by case: (1.09 4.91 9.1 14.39 19.4 32.27 59.74
92.09 149.57 231.42)
Error minima by case: (0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0)
Average genome size in population (length): 37.44
Average program size in population (points): 38.44
Average percent parens in population: 0.029
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 10
Genome diversity (% unique Plush genomes):       0.48
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 10
Syntactic diversity (% unique Push programs):    0.48
Total error diversity:                           0.42
Error (vector) diversity:                        0.42
--- Run Statistics ---
Number of program evaluations used so far: 5900
Number of point (instruction) evaluations so far: 1990501
--- Timings ---
Current time: 1457558128561 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 28
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Producing offspring...
Installing next generation...
Processing generation: 29
Computing errors... Done computing errors.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- Report at generation 29
--- Lexicse Program with Most Elite Cases Statistics ---
Lexicase best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 1, :instruction 6} {:close
2, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 2, :instruction in1} {:close 3,
:instruction 5} {:close 2, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 2, :instruction in1} {:close
3, :instruction 5} {:close 0, :instruction integer_add} {:close
0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult})
Lexicase best program: (5 4 in1 integer_sub in1 integer_mult
integer_sub integer_div integer_mult 6 integer_sub integer_add
```

```
in1 5 integer_sub integer_add in1 5 integer_add integer_add
integer_mult in1 integer_mult)
Lexicase best partial simplification: (5 4 in1 integer_sub in1
integer_mult integer_sub 6 integer_sub in1 5 integer_sub
integer_add in1 5 integer_add integer_add in1 integer_mult)
Lexicase best errors: [0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
Lexicase best number of elite cases: 10
Lexicase best total error: 0.0
Lexicase best mean error: 0.0
Lexicase best size: 24
Percent parens: 0.042
--- Lexicse Program with Most Zero Cases Statistics ---
Zero cases best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 1, :instruction 6} {:close
2, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 2, :instruction in1} {:close 3,
:instruction 5} {:close 2, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 2, :instruction in1} {:close
3, :instruction 5} {:close 0, :instruction integer_add} {:close
0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult})
Zero cases best program: (5 4 in1 integer_sub in1 integer_mult
integer_sub integer_div integer_mult 6 integer_sub integer_add
in1 5 integer_sub integer_add in1 5 integer_add integer_add
integer_mult in1 integer_mult)
Zero cases best partial simplification: (5 4 in1 integer_sub
in1 integer_mult integer_sub integer_div 6 integer_sub in1 5
integer_sub integer_add in1 5 integer_add integer_add in1
integer_mult)
Zero cases best errors: [0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0]
Zero cases best number of elite cases: 10
Zero cases best number of zero cases: 10
Zero cases best total error: 0.0
Zero cases best mean error: 0.0
Zero cases best size: 24
Percent parens: 0.042
--- Lexicase Population Statistics ---
Count of elite individuals by case: (94 37 31 22 11 13 12 9 9
9)
Population mean number of elite cases: 2.47
Count of perfect (error zero) individuals by case: (94 37 31 22
11 13 12 9 9 9)
Population mean number of perfect (error zero) cases: 2.47
--- Best Program (based on total-error) Statistics ---
Best genome: ({:close 0, :instruction 5} {:close 1,
:instruction 4} {:close 0, :instruction in1} {:close 0,
:instruction integer_sub} {:close 0, :instruction in1} {:close
1, :instruction integer_mult} {:close 0, :instruction
integer_sub} {:close 0, :instruction integer_div} {:close 0,
:instruction integer_mult} {:close 1, :instruction 6} {:close
2, :instruction integer_sub} {:close 0, :instruction
integer_add} {:close 2, :instruction in1} {:close 3,
:instruction 5} {:close 2, :instruction integer_sub} {:close 0,
:instruction integer_add} {:close 2, :instruction in1} {:close
3, :instruction 5} {:close 0, :instruction integer_add} {:close
0, :instruction integer_add} {:close 0, :instruction
integer_mult} {:close 1, :instruction in1} {:close 0,
:instruction integer_mult})
Best program: (5 4 in1 integer_sub in1 integer_mult integer_sub
integer_div integer_mult 6 integer_sub integer_add in1 5
integer_sub integer_add in1 5 integer_add integer_add
integer_mult in1 integer_mult)
Partial simplification: (5 4 in1 integer_sub in1 integer_mult
integer_sub integer_div integer_mult 6 integer_sub in1 5
integer_sub integer_add in1 5 integer_add integer_add in1
integer_mult)
Errors: [0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
Total: 0.0
Mean: 0.0
Genome size: 23
Size: 24
Percent parens: 0.042
--- Population Statistics ---
Average total errors in population: 497.77
Median total errors in population: 336.0
Error averages by case: (0.42 3.51 7.54 13.0 19.3 26.93 48.51
73.48 119.41 185.67)
Error minima by case: (0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0)
Average genome size in population (length): 39.64
Average program size in population (points): 40.64
Average percent parens in population: 0.027
--- Population Diversity Statistics ---
Min copy number of one Plush genome: 1
Median copy number of one Plush genome: 1
Max copy number of one Plush genome: 11
Genome diversity (% unique Plush genomes):       0.52
Min copy number of one Push program: 1
Median copy number of one Push program: 1
Max copy number of one Push program: 11
Syntactic diversity (% unique Push programs):    0.52
Total error diversity:                           0.47
Error (vector) diversity:                        0.47
--- Run Statistics ---
Number of program evaluations used so far: 6000
Number of point (instruction) evaluations so far: 2030141
--- Timings ---
Current time: 1457558129110 milliseconds
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -*- End of report for generation 29
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


SUCCESS at generation 29
Successful program: (5 4 in1 integer_sub in1 integer_mult
integer_sub integer_div integer_mult 6 integer_sub integer_add
in1 5 integer_sub integer_add in1 5 integer_add integer_add
integer_mult in1 integer_mult)
Errors: [0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
Total error: 0.0
History: null
Size: 24


Auto-simplifying with starting size: 24
step: 0
program: (5 4 in1 integer_sub in1 integer_mult integer_sub
integer_div integer_mult 6 integer_sub integer_add in1 5
integer_sub integer_add in1 5 integer_add integer_add
integer_mult in1 integer_mult)
errors: [0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
total: 0.0
size: 24

step: 500
program: (5 4 in1 integer_sub in1 integer_mult integer_sub 6
integer_sub in1 5 integer_sub integer_add in1 5 integer_add
integer_add in1 integer_mult)
errors: [0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
total: 0.0
size: 20

step: 1000
program: (5 4 in1 integer_sub in1 integer_mult integer_sub 6
integer_sub in1 5 integer_sub integer_add in1 5 integer_add
integer_add in1 integer_mult)
errors: [0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
total: 0.0
size: 20

;;******************************
;; Problem-Specific Report of Simplified Solution
```

```
:no-problem-specific-report-function-defined
```