Maps in P0

Faisal Jaffer
Usman Asad
Aleem Haq

04/2021

Topic

Extend the P0 compiler with (typed) maps.

Challenge

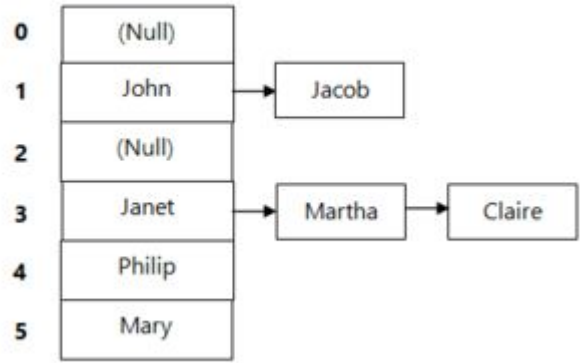
- ➔ At runtime the size of the map may increase
  - ◆ Hash table needs to be managed on WebAssembly memory
- ➔ Guarantee constant time for lookup
- ➔ Guarantee constant time for deletion
- ➔ Guarantee amortized constant time of insertion

Types of Hashing

Separate Hashing

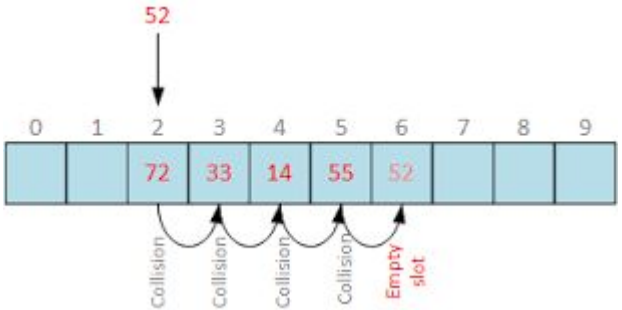
Lookup time: O(N) (worst)

Figure 2: Separate Chaining



Open Addressing

Lookup time: O(N) (worst)

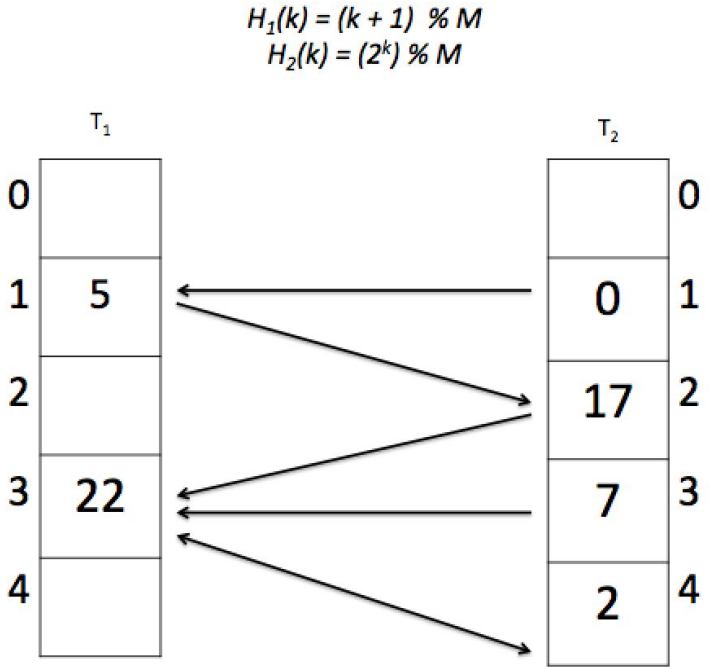


Cuckoo Hashing

Lookup time: constant (worst)

Insertion time: constant (amortized)

Deletion time: constant (worst)



Existing Implementation

- https://github.com/DeborahHier/CuckooHash/blob/master/CuckooHash.py

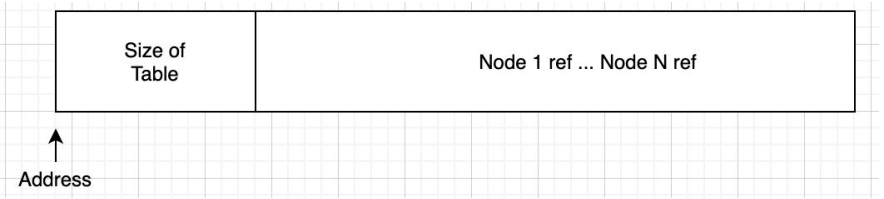
Cuckoo Hashing

1. If map starts to fill up, we need to reallocate tables with larger sizes and re-insert entries.
2. If there is a loop in the displacements of nodes, we need to reallocate tables with larger sizes and re-insert entries.

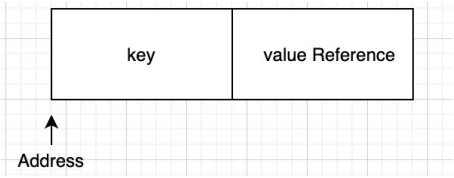
Our Implementation

Our Map (16 bytes)

Our Table (min 20 bytes)

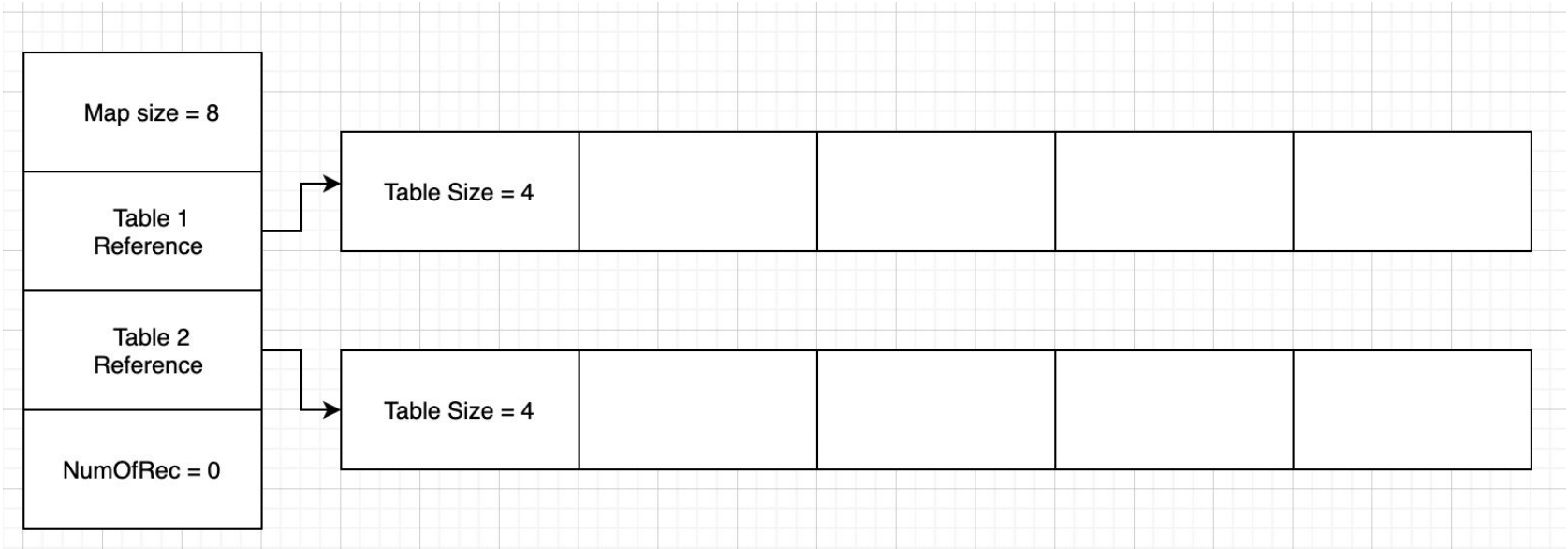


Our Node (8 bytes)



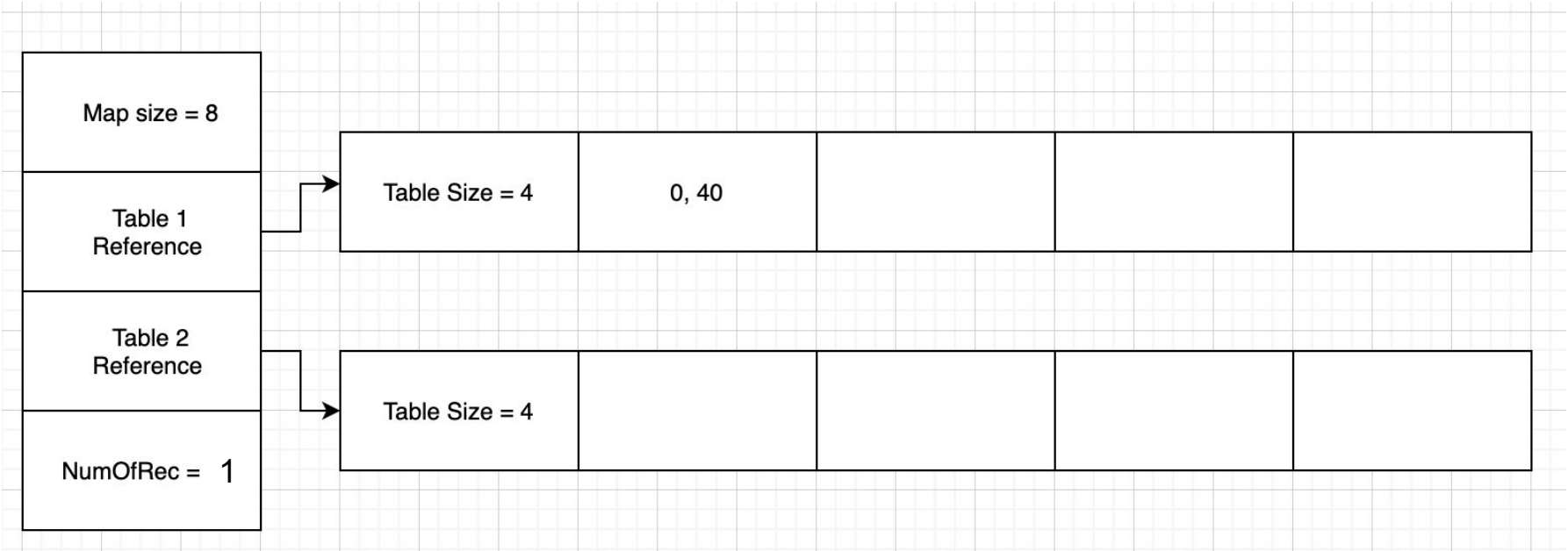
Our Implementation (Cont.)

Upon Map initialization



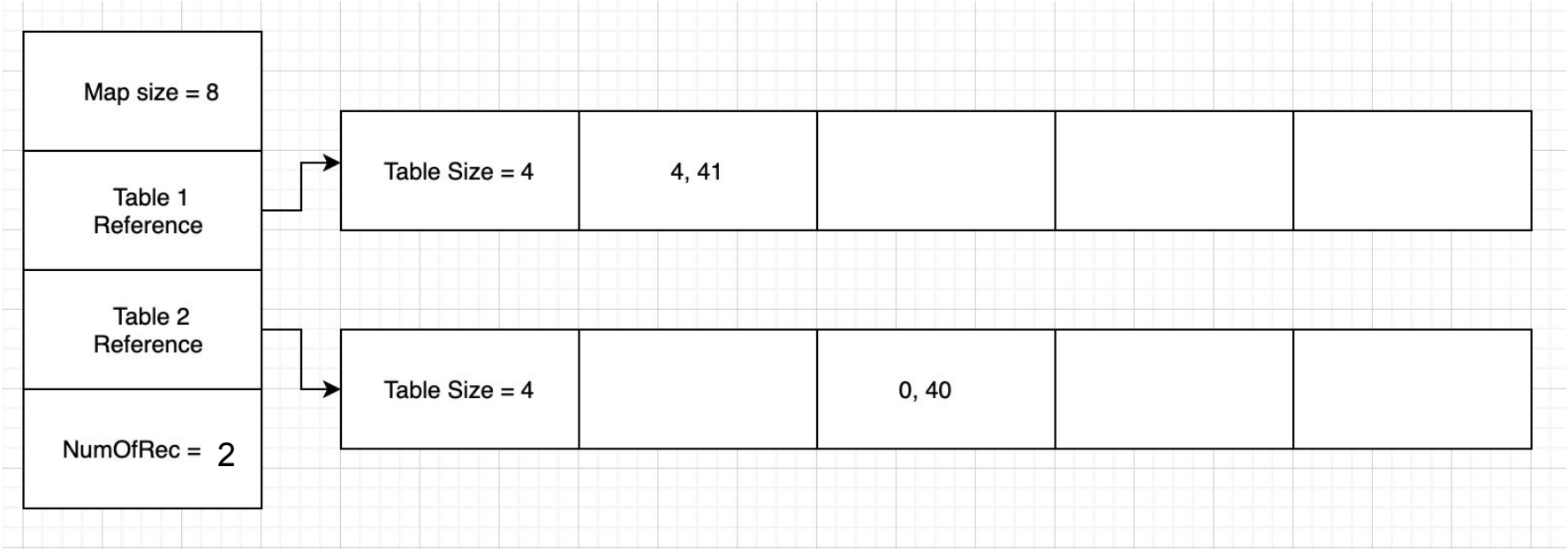
Our Implementation (Cont.)

Insert <0, 40>



Our Implementation (Cont.)

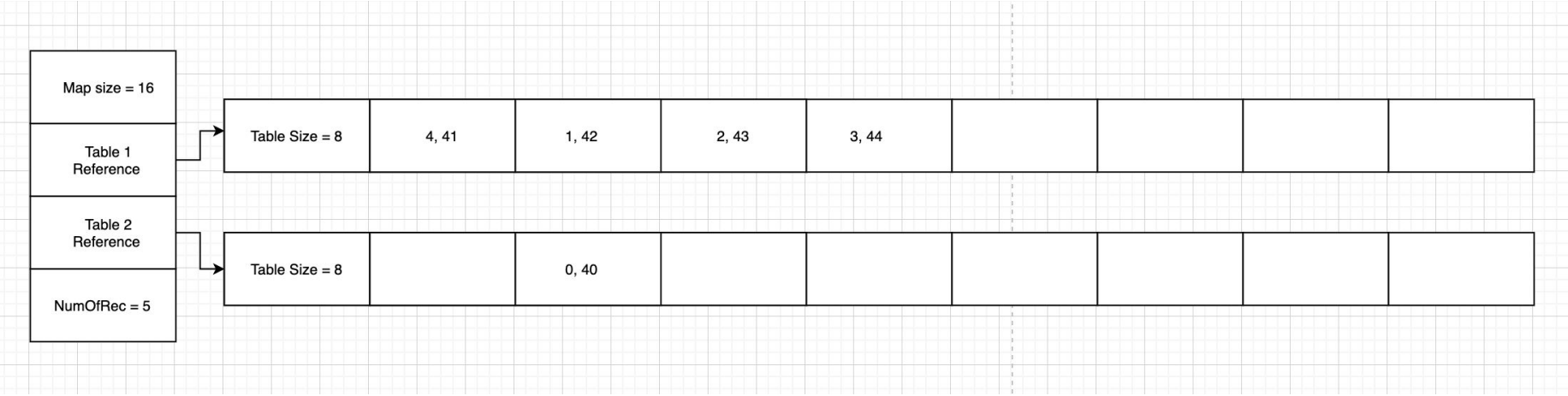
Insert <4, 41>



Our Implementation (Cont.)

Number of Records > Size of Map / 2

- Allocate new tables with 2x sizes



Our Map Syntax in P0

Map declaration:

```
var e: map <integer, integer>
```

Map initialization:

```
e.init
```

Map insert:

```
e.insert(0,10)
```

Map get:

```
write(e.get(1))
```

Map length:

```
write(e.len())
```

Key in map:

```
if 1 in e then write(1)
```

Sample Execution

```
compileString("""
program bitsets
  var a, b: set [1 .. 5]
  var z: [1 .. 5] → integer
  var y: [1 .. 5] → integer
  var c, i, k: integer
A var e: map <integer, integer>
  var d: boolean
    c := 4
B   e.init
C   e.insert(0,10)
    e.insert(1,9)
    e.insert(c, 4)
D   write(e.get(1))
    write(e.get(c))
    write(e.len())
""", 'primes.wat', target = 'wat')
!wat2wasm primes.wat
```

Output

```
9
4
3
```

Modified P0 grammar

```
selector ::= { "[" expression "]" | "." ident | ".get(" expression ")" | ".len()" }
factor ::= ident selector | integer | "(" expression ")" | "{" [expression {"," expression}] "}" | ("¬" | "#" |
    "C") factor
term ::= factor {("×" | "div" | "mod" | "∩" | "and") factor}
simpleExpression ::= ["+" | "-"] term {("+" | "-" | "∪" | "or") term}
expression ::= simpleExpression
    {("=" | "≠" | "<" | "≤" | ">" | "≥" | "∈" | "⊆" | "⊇" | "in") simpleExpression}
statementList ::= statement {";" statement}
statementBlock ::= statementList {statementList}
statementSuite ::= statementList | INDENT statementBlock DEDENT
statement ::=
    ident selector ":=" expression |
    ident {"," ident} (":=" expression {"," expression} |
        "←" ident "(" [expression {"," expression}] ")") |
    "if" expression "then" statementSuite ["else" statementSuite] |
    "while" expression "do" statementSuite |
    ident ".init" |
    ident ".insert(" expression, expression ")"
type ::=
    ident |
    "[" expression ".." expression "]" "→" type |
    "(" typedIds ")" |
    "set" "[" expression ".." expression "]" |
    "map" "<" type "," type ">"
typedIds ::= ident ":" type {"," ident ":" type}
declarations ::=
    {"const" ident "=" expression}
    {"type" ident "=" type}
    {"var" typedIds}
    {"procedure" ident "(" [typedIds] ")" [ "→" "(" typedIds ")" ] body}
body ::= INDENT declarations (statementBlock | INDENT statementBlock DEDENT) DEDENT
program ::= declarations "program" ident body
```

P0 Test

In [3]:
```
# Testing to see that an error is thrown when insertion is attempted but the map is not yet initialized
```

In [4]:
```
compileString("""
program bitsets
  var e: map <integer, integer>
  var a, b: integer
    e.insert(2,4)
    write(e.get(2))
""", 'test.wat', target = 'wat')
!wat2wasm test.wat
runpywasm("test.wasm")
```

```
<ipython-input-4-64f43042903c> in <module>
----> 1 compileString("""
      2 program bitsets
      3   var e: map <integer, integer>
      4   var a, b: integer
      5     e.insert(2,4)

~/Maps/P0.ipynb in compileString(src, dstfn, target)
     11     "#### Original Author: Emil Sekerinski, revised March 2021\n",
     12     "\n",
---> 13     "This collection of _Jupyter notebooks_ develops a compiler for P0, a programming langauge inspired by Pa
scal, a language designed for ease of compilation. The compiler currently generates WebAssebmbly and MIPS code, but i
s modularized to facilitate other targets. WebAssembly is representive of stack-based virtual machines while the MIPS
architecture is representative of Reduced Instruction Set Computing (RISC) processors.\n",
     14     "\n",
     15     "### The P0 Language\n",

Exception: line 5 pos 12 Map not initialized
```

P0 Test

In [ ]:
```
# Testing to see if error is thrown when inserting wrong type into the map
```

In [5]:
```
compileString("""
program bitsets
  var e: map <integer, integer>
  var a, b: integer
    e.init
    write(e.len())
    e.insert(1, true)
""", 'test.wat', target = 'wat')
!wat2wasm test.wat
runpywasm("test.wasm")
```

```
Exception: line 7 pos 21 map key type mismatch
```

P0 Test

In [ ]:
```
# Getting a value that does not exist
```

In [6]:
```
compileString("""
program bitsets
  var e: map <integer, integer>
  var a, b: integer
    e.init
    a ← time()
    write(e.len())
    write(e.get(13233))
""", 'test.wat', target = 'wat')
!wat2wasm test.wat
runpywasm("test.wasm")
```

```
    832             config.stack.append(e)

/usr/local/lib/python3.8/site-packages/pywasm/execution.py in call(self, function_addr, function_args)
    478             return self.exec()
    479         if isinstance(function, HostFunc):
--> 480             r = function.hostcode(self.store, *[e.val() for e in function_args])
    481             l = len(function.type.rets.data)
    482             if l == 0:

<ipython-input-2-b4d1be340ad1> in key_err(s)
      8         return time.time() * 1000
      9     def key_err(s):
---> 10         raise Exception("Key does not exist")
     11     vm = pywasm.load(wasmfile, {'P0lib': {'write': write, 'writeln': writeln, 'read': read, 'time': time_,'er
r': key_err}})
     12

Exception: Key does not exist
```

P0 Test

In [ ]:
```
# Testing to see wrong key type (Key expected to be Int, Bool, Set)
```

In [7]:
```
compileString("""
program bitsets
  var e: map <[0..5] → integer, integer>
  var a, b: integer
    e.init
    a ← time()
    write(e.len())
""", 'test.wat', target = 'wat')
!wat2wasm test.wat
runpywasm("test.wasm")
```

```
Exception: line 3 pos 31 Key expected to be Int, Bool, Set
```

P0 Test

In [ ]:
```
# Testing to see value accepts all types
```

In [8]:
```
compileString("""
program bitsets
  var e: map <integer, [0..5] → integer>
  var a, b: integer
    e.init
    a ← time()
    write(e.len())
""", 'test.wat', target = 'wat')
!wat2wasm test.wat
runpywasm("test.wasm")
```

```
1617814627815.9233
0
```

Runtime Test

Map size = 11

In [ ]:
```
compileString("""
program bitsets
  var e: map <integer, integer>
  var a, b: integer
    a ← time()
    e.init
    b := 0
    while b < 10 do
      e.insert(b,3)
      b := b + 1
    a ← time()
    write(e.get(4))
    b ← time()
    e.insert(1001,23)
    b ← time()
    write(e.len())
""", 'test1.wat', target = 'wat')
!wat2wasm test1.wat
runpywasm("test1.wasm")
```

Lookup time = 1617757775317.0266 - 1617757775404.4658 = 87
Insertion time = 1617757584101.363 - 1617757584440.0674 = 341

Runtime Test

Map size = 21

In [ ]:
```
compileString("""
program bitsets
  var e: map <integer, integer>
  var a, b: integer
    a ← time()
    e.init
    b := 0
    while b < 20 do
      e.insert(b,3)
      b := b + 1
    a ← time()
    write(e.get(4))
    b ← time()
    e.insert(1001,23)
    b ← time()
    write(e.len())
""", 'test2.wat', target = 'wat')
!wat2wasm test2.wat
runpywasm("test2.wasm")
```

Lookup time = 1617757038433.1326 - 1617757038477.5427 = 44
Insertion time = 1617757038477.5427 - 1617757038655.2637 = 140

Runtime Test

Map size = 101

In [ ]:
```
compileString("""
program bitsets
  var e: map <integer, integer>
  var a, b: integer
    a ← time()
    e.init
    b := 0
    while b < 100 do
      e.insert(b,3)
      b := b + 1
    a ← time()
    write(e.get(4))
    b ← time()
    e.insert(1001,23)
    b ← time()
    write(e.len())
""", 'test3.wat', target = 'wat')
!wat2wasm test3.wat
runpywasm("test3.wasm")
```

Lookup time = 1617757089525.4712 - 1617757089538.5867 = 13
Insertion time = 1617757089538.5867 - 1617757089610.3218 = 87

Runtime Test

Map size = 1001

In [ ]:
```
compileString("""
program bitsets
  var e: map <integer, integer>
  var a, b: integer
    a ← time()
    e.init
    b := 0
    while b < 1000 do
      e.insert(b,3)
      b := b + 1
    a ← time()
    write(e.get(4))
    b ← time()
    e.insert(1001,23)
    b ← time()
    write(e.len())
""", 'test4.wat', target = 'wat')
!wat2wasm test4.wat
runpywasm("test4.wasm")
```

Lookup time = 1617757530145.7568 - 1617757530149.9011 = 4
Insertion time = 1617757530149.9011 - 1617757530446.7769 = 296

Runtime Test Summary

Size	Lookup Time (ms)	Insertion Time (ms)
11	87	341
21	44	140
101	13	87
1001	4	296

Development Statistics

- **65** lines of code added to the P0 Compiler:
- **385** lines of code(500+ instructions) for the map implementation using Cuckoo Hashing in Web Assembly
- **4** runtime and performance tests
- **8 “**expected exceptions” tests
- **80** lines of documentation

Conclusion

- Map’s start at a finite size, as you add to them the size increases
- Managing the memory and size is complex to do in webassembly
- When choosing a key value, you can only use types that are hashable