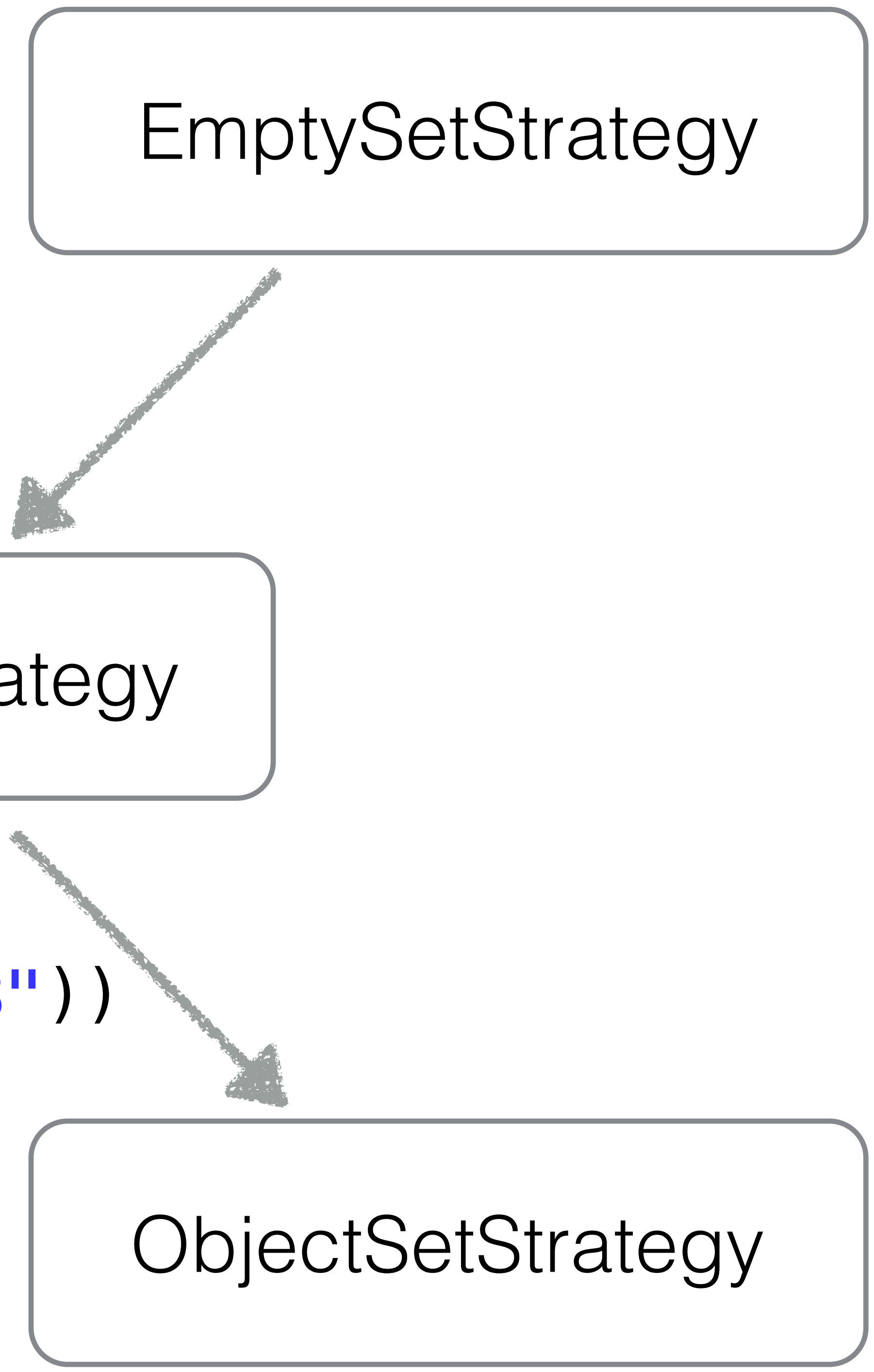
Heterogeneous Data Structures for the Masses

Michael Steindorfer

Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
Johannes Kepler University, Linz, Austria

Optimizing Data Structures in
Dynamically Typed Languages

```
set = new HashSet()


set.add(32)


set.add(2)
set.add(4098)
set.add(34)


set.add(new BigInteger("1099511627778"))


set.add(898)
```



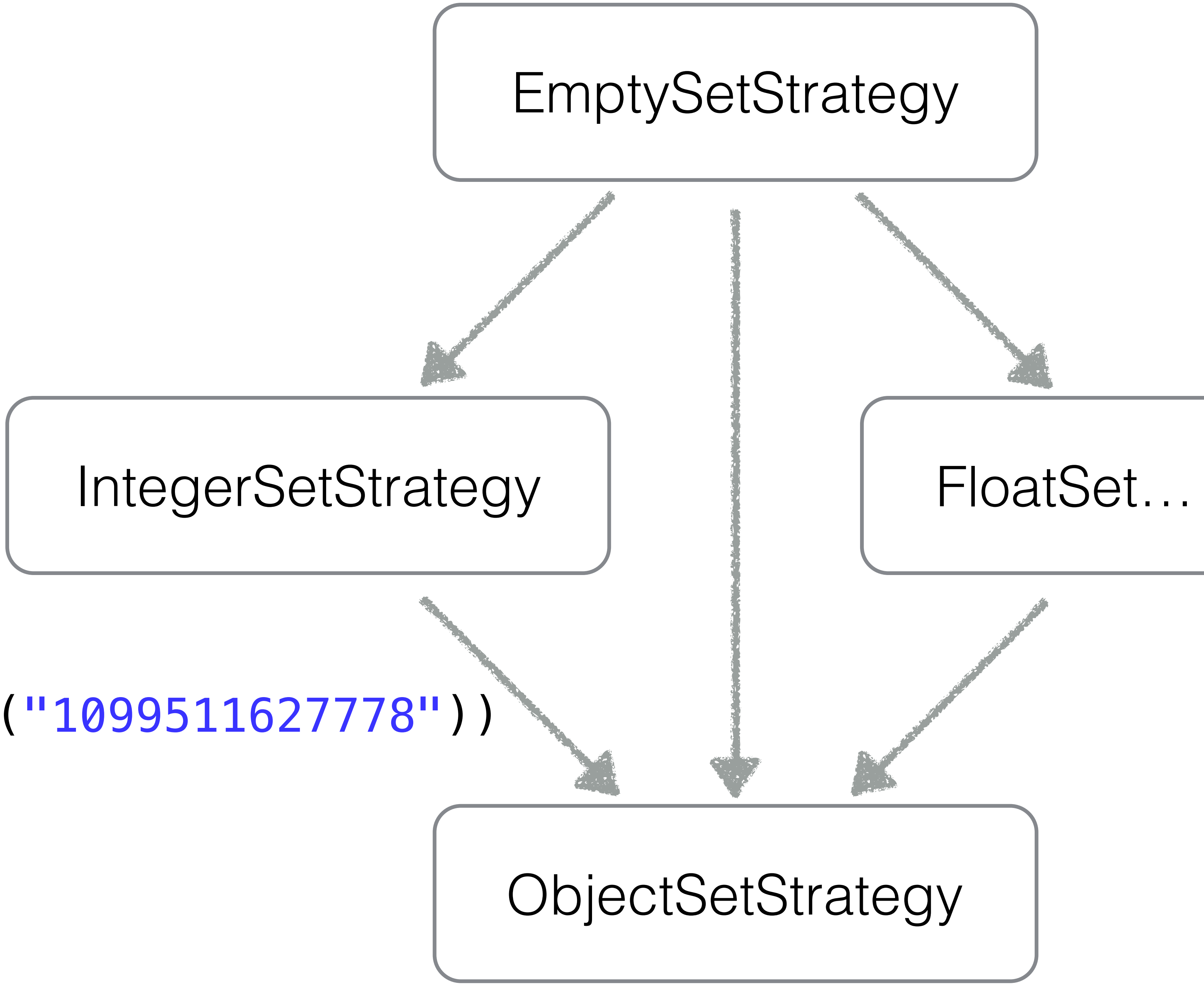
EmptySetStrategy

IntegerSetStrategy

ObjectSetStrategy

Bolz, C. F., Diekmann, L., & Tratt, L. (2013). Storage strategies
for collections in dynamically typed languages. OOPSLA’13

```
set = new HashSet()


set.add(32)


set.add(2)
set.add(4098)
set.add(34)


set.add(new BigInteger("1099511627778"))


set.add(898)
```

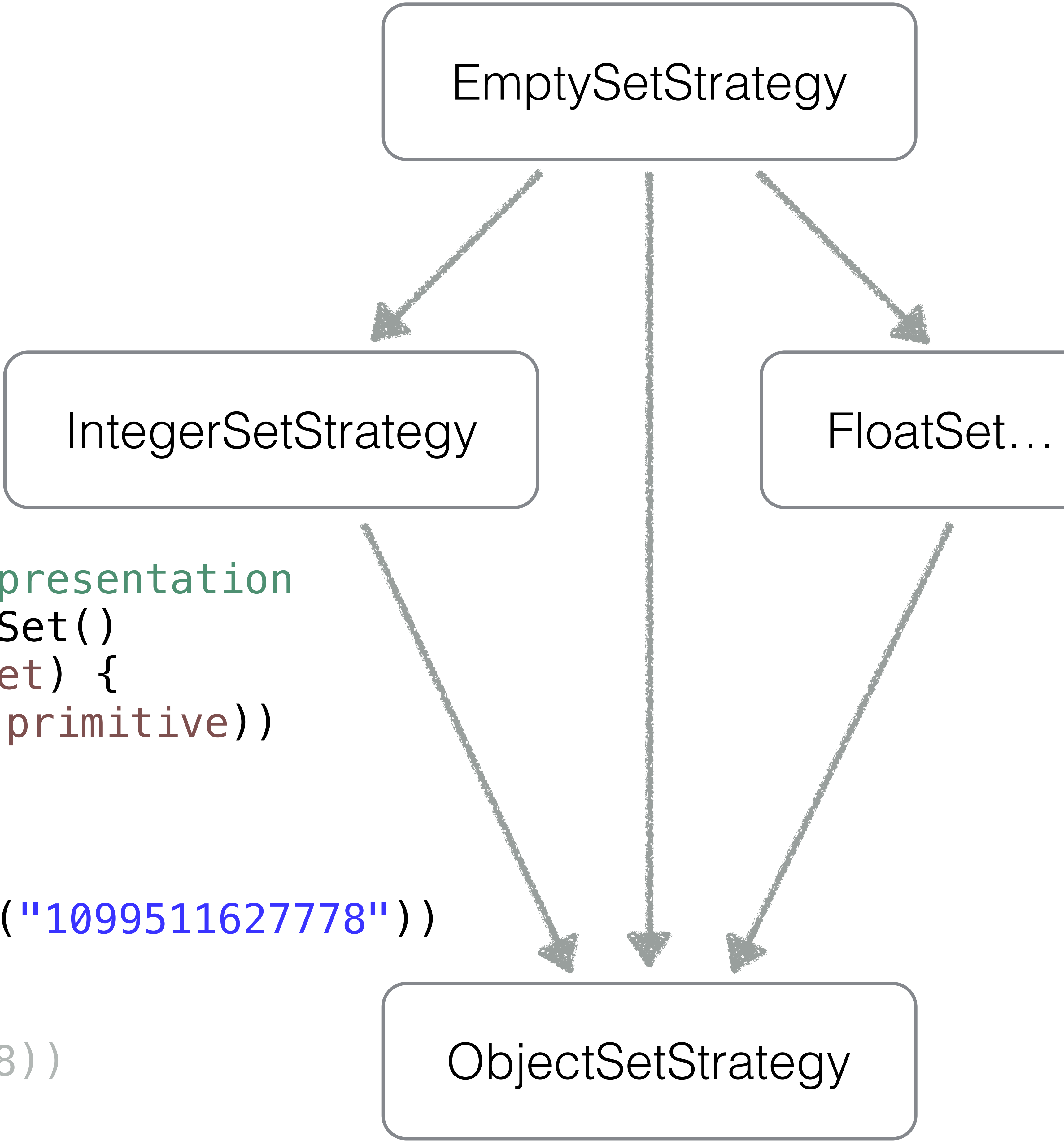


Bolz, C. F., Diekmann, L., & Tratt, L. (2013). Storage strategies
for collections in dynamically typed languages. OOPSLA'13

Issue #1

Cost of Conversion

```
set = new HashSet()


set.add(32)


set.add(2)
set.add(4098)
set.add(34)

// convert to boxed representation
HashSet tmp = new HashSet()
for (int primitive : set) {
   tmp.add(new Integer(primitive))
}
set = tmp

set.add(new BigInteger("1099511627778"))


set.add(new Integer(898))
```



Issue #2

Impedance Mismatch

Language Level:

- **`ArbitraryPrecisionInteger`**

Run-time Level:

- **`Either[int, BigInteger]`**

Wanted:

```
HashSet[Either[int, BigInteger]]
```

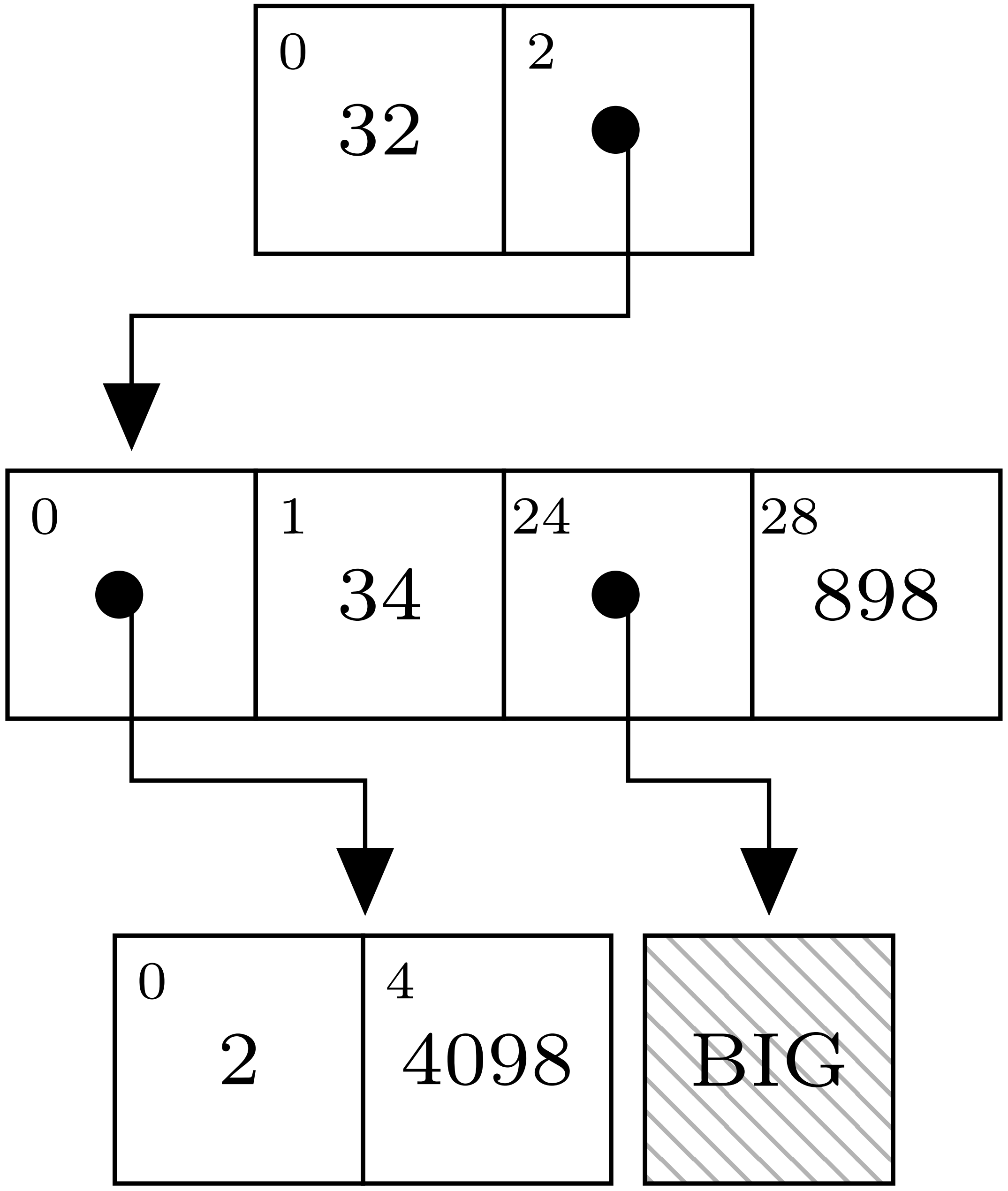
Wanted:

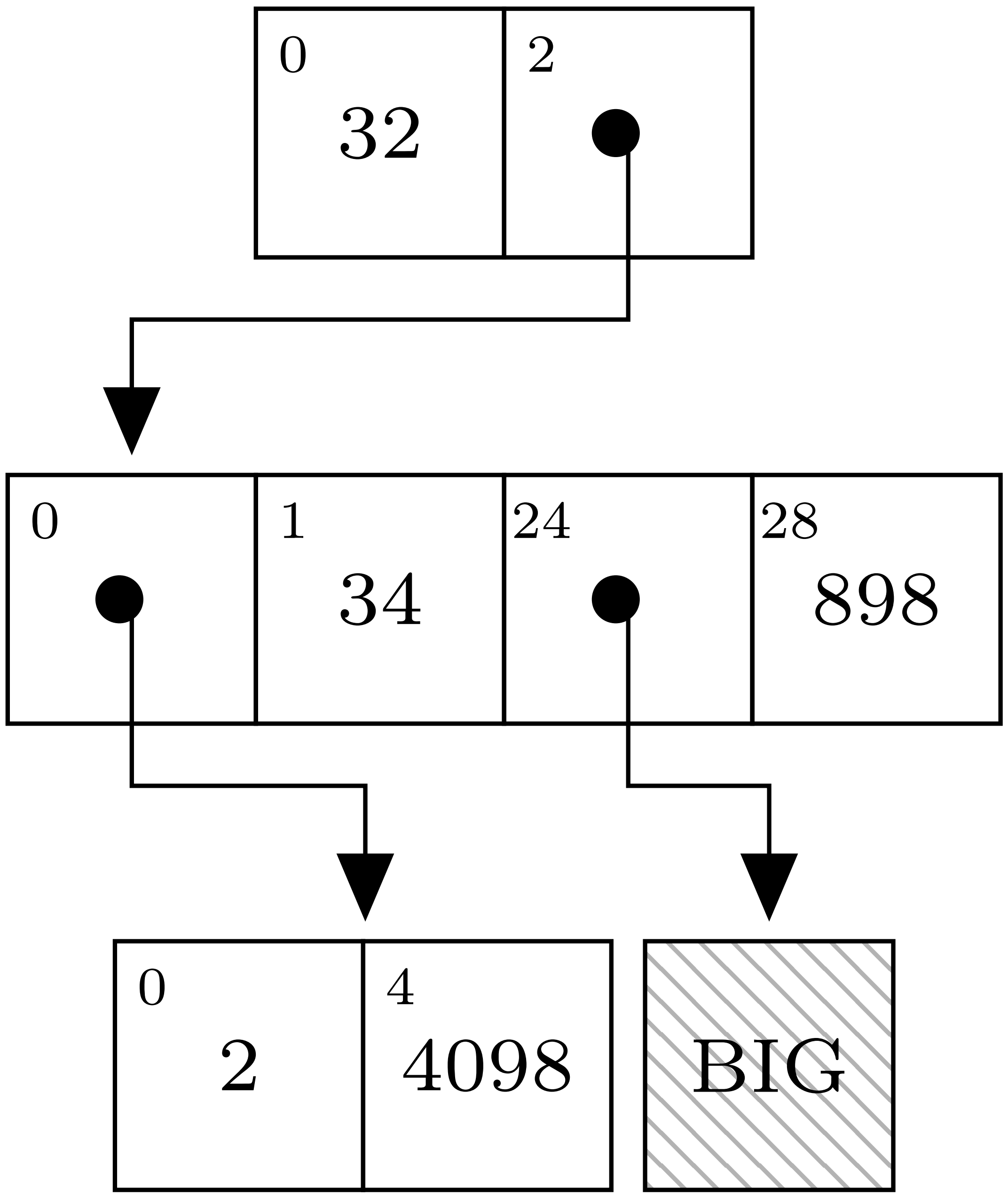
```
HashSet[Either[int, BigInteger]]
```

Possible with arrays:

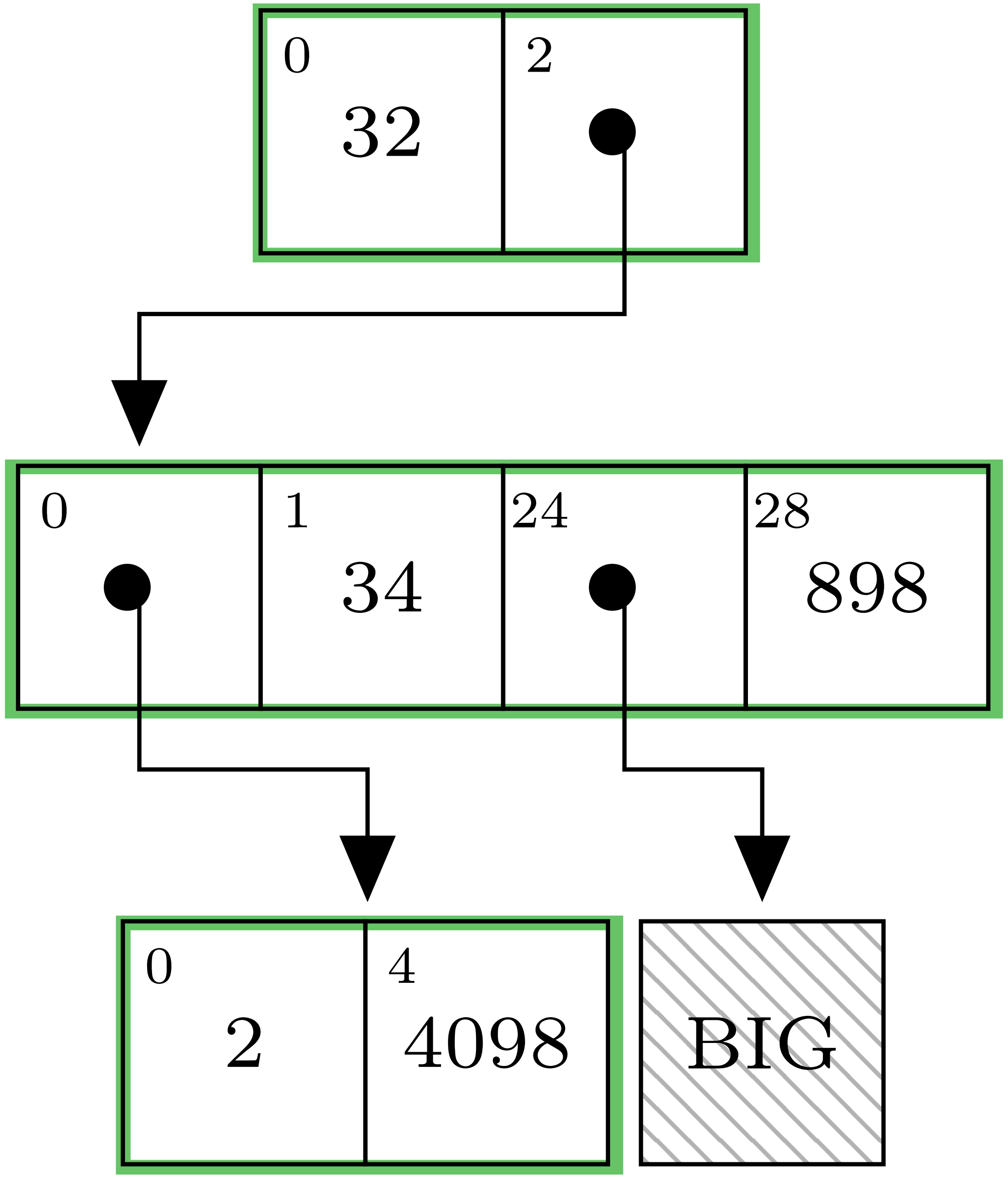
```
Either[
   HashSet[int],
   HashSet[Number]
]
```

Fine Granularity?

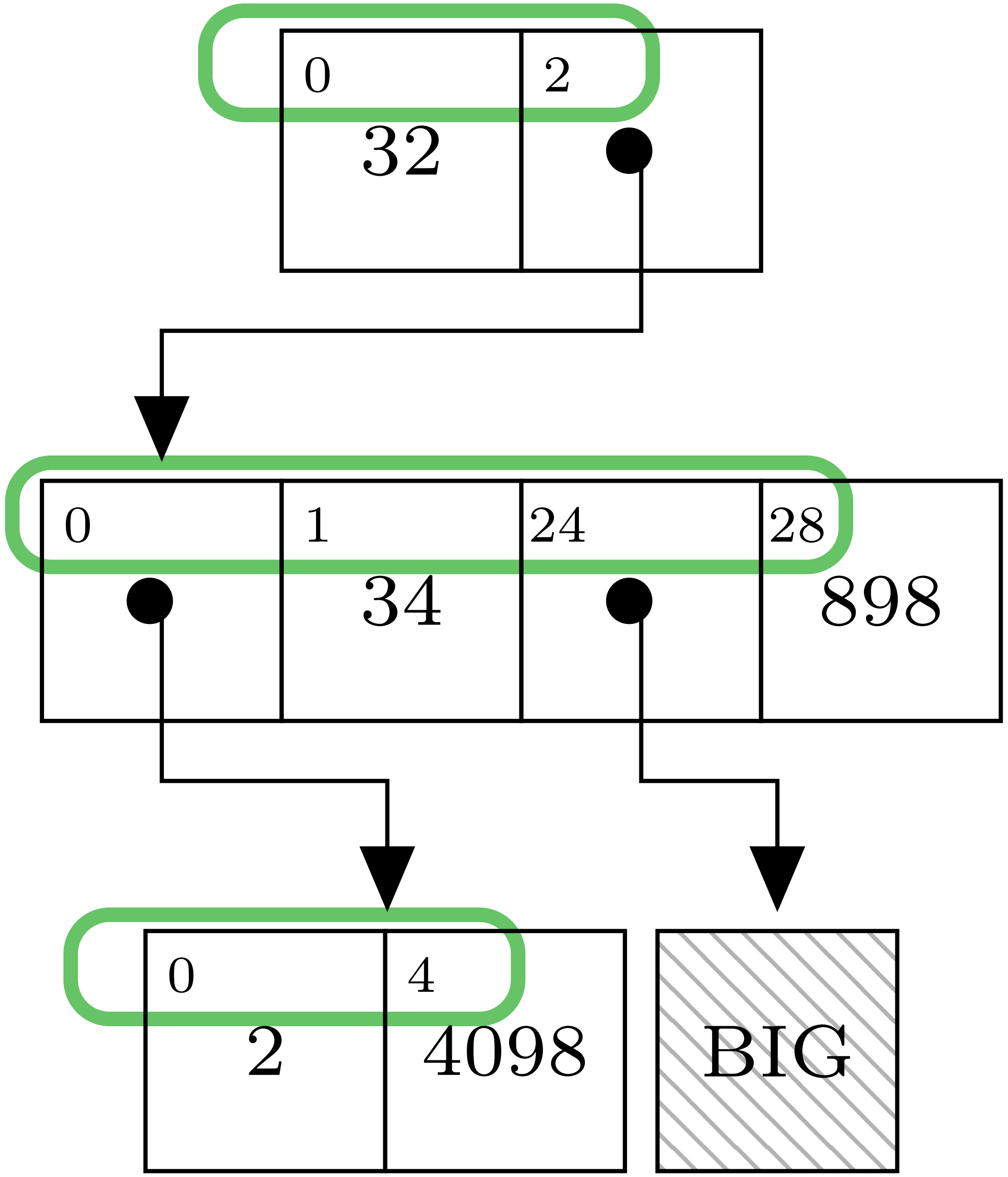

0
32
2
0
1
34
24
28
898
0
2
4
4098
BIG



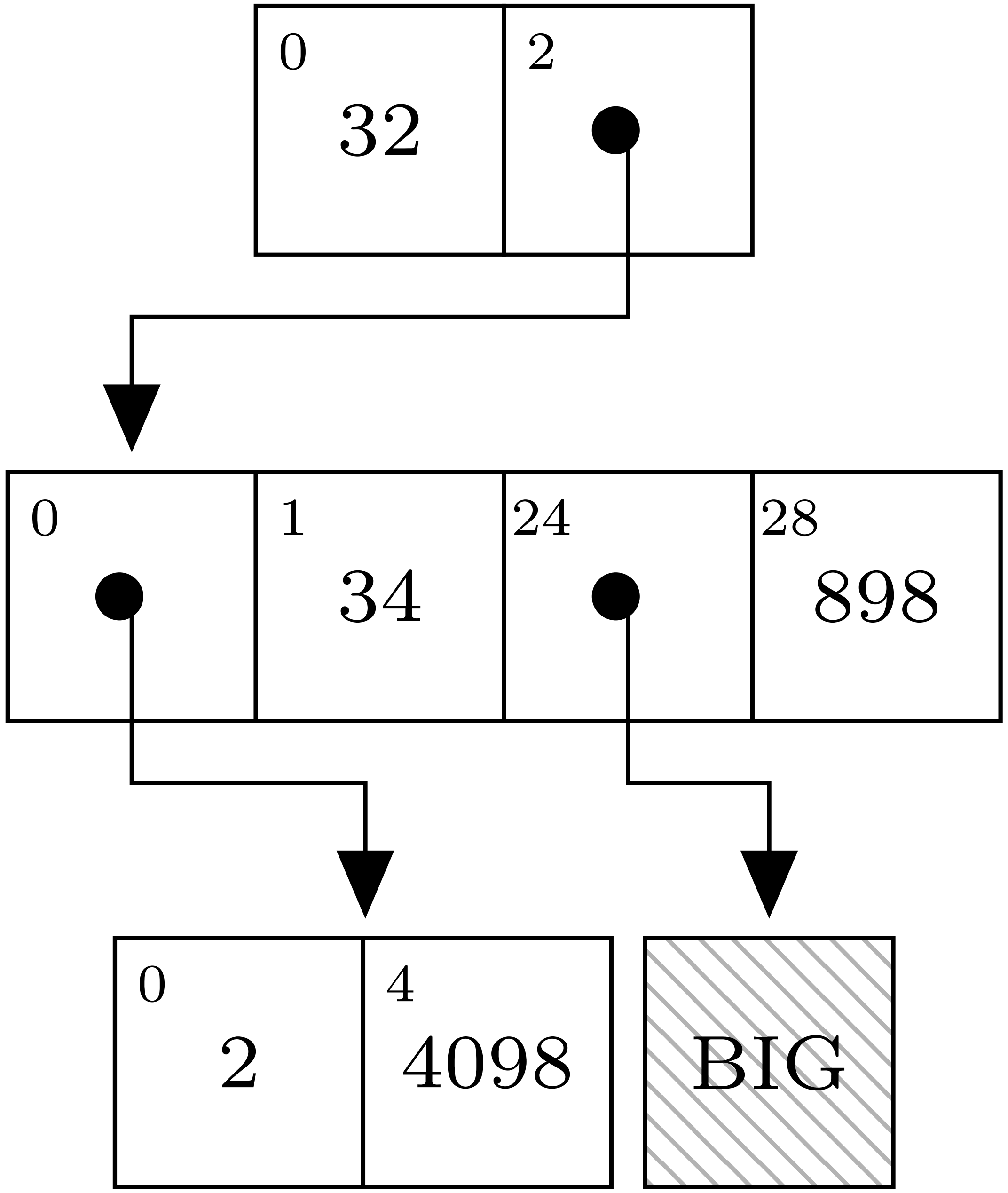
```
class TrieNode {
  int bitmap;
  Object[] mixedContent;
}
```



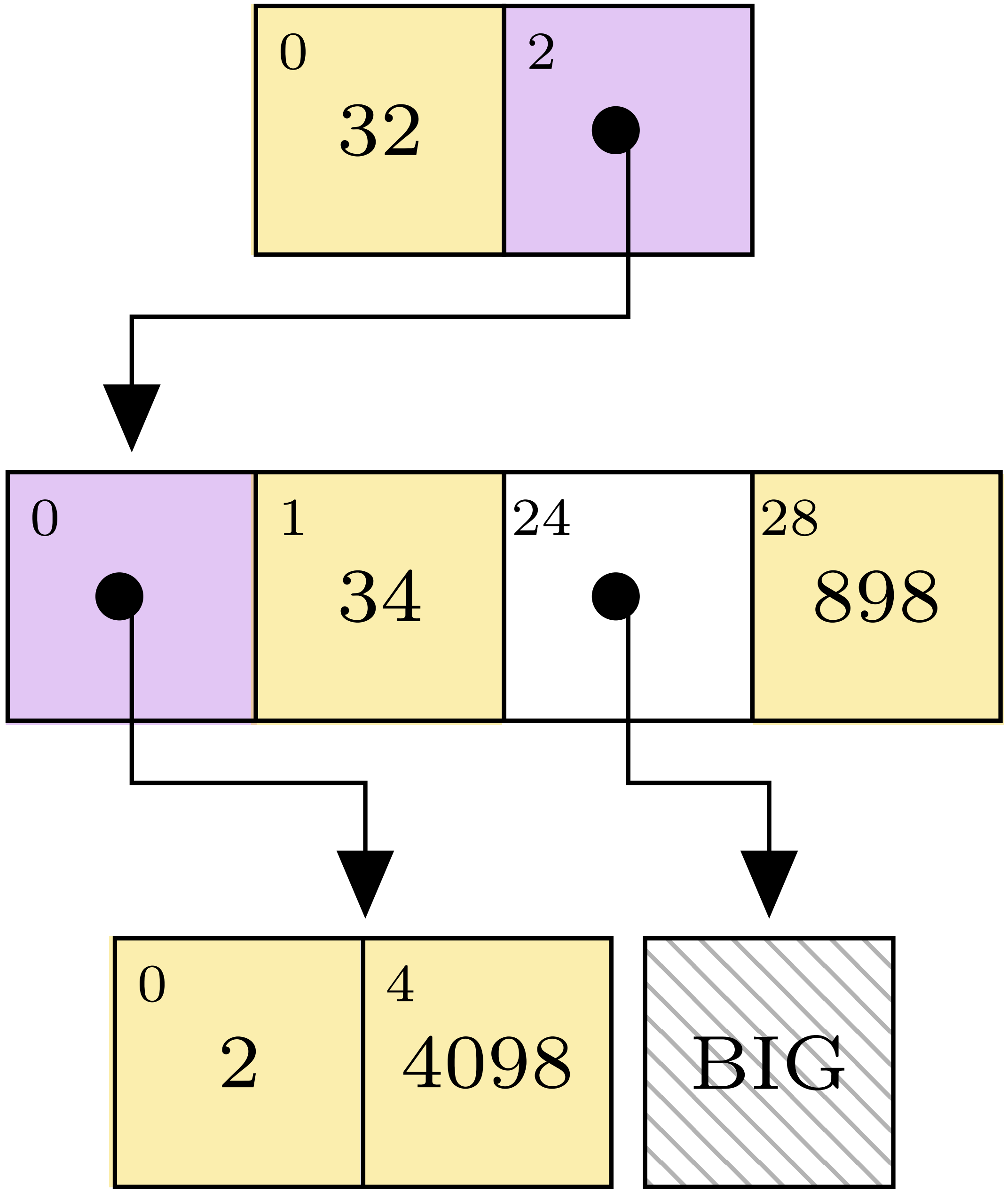
```
class TrieNode {
  int bitmap;
  Object[] mixedContent;
}
```



```
class TrieNode {
  int bitmap;
  Object[] mixedContent;
}
```

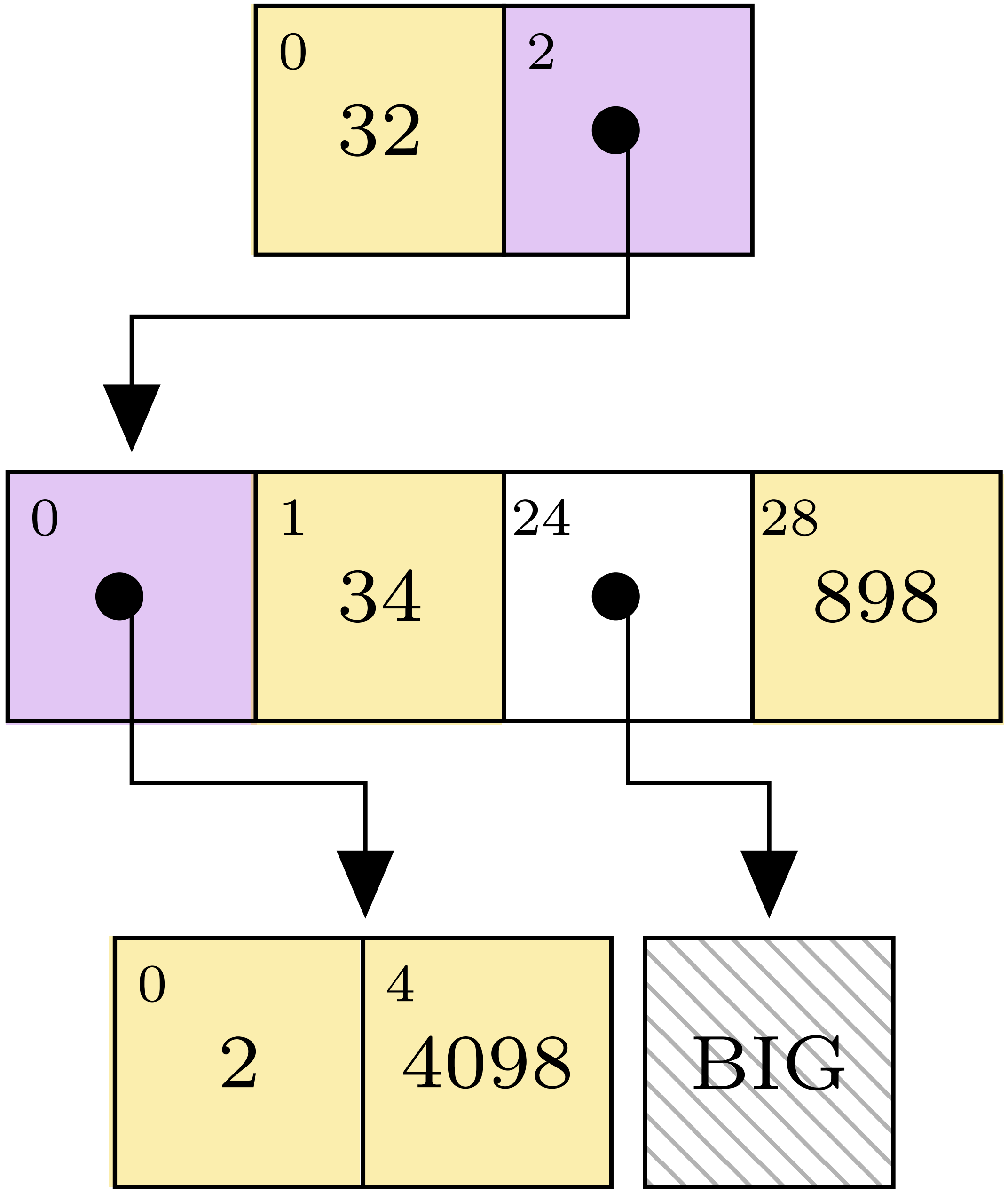


```
class TrieNode {
  int bitmap;
  Object[] mixedContent;
}
```

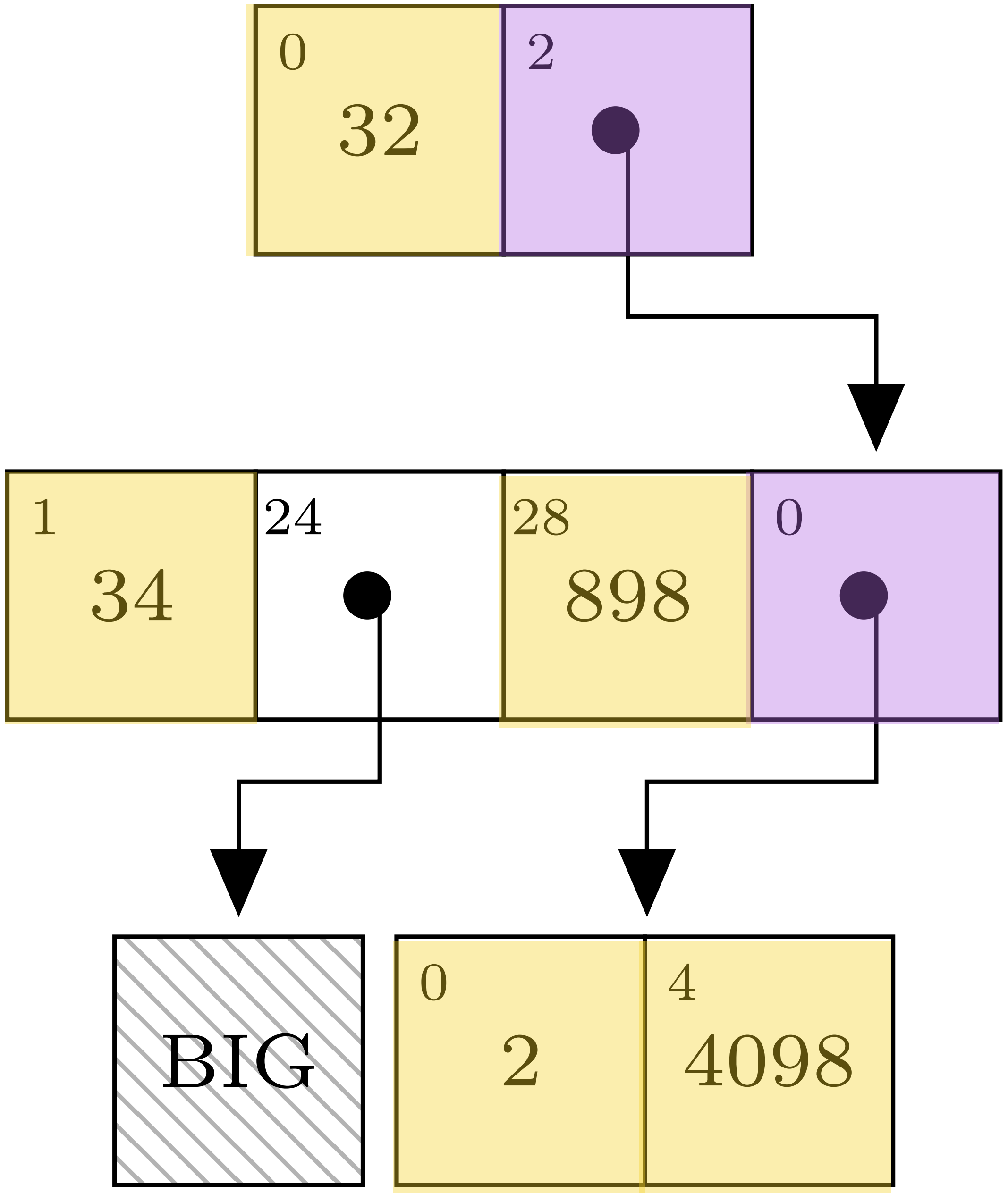


```
class TrieNode {
  int bitmap;
  Object[] mixedContent;
}
```

	1	0
bitmap	is present	-



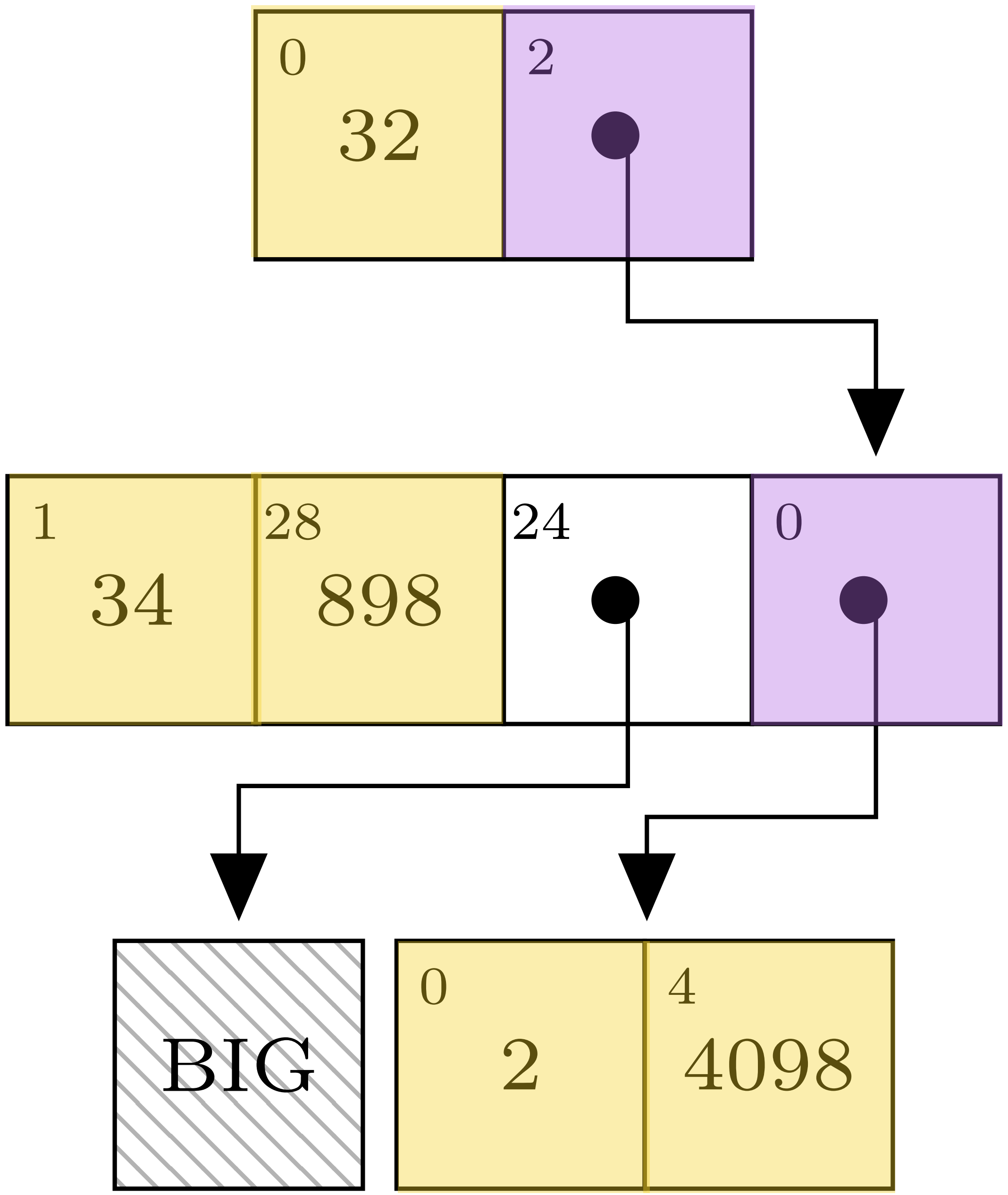
```
class TrieNode {
  int bitmap;
  Object[] mixedContent;
}
```



	1	0
datamap	number	-
nodemap	node	-

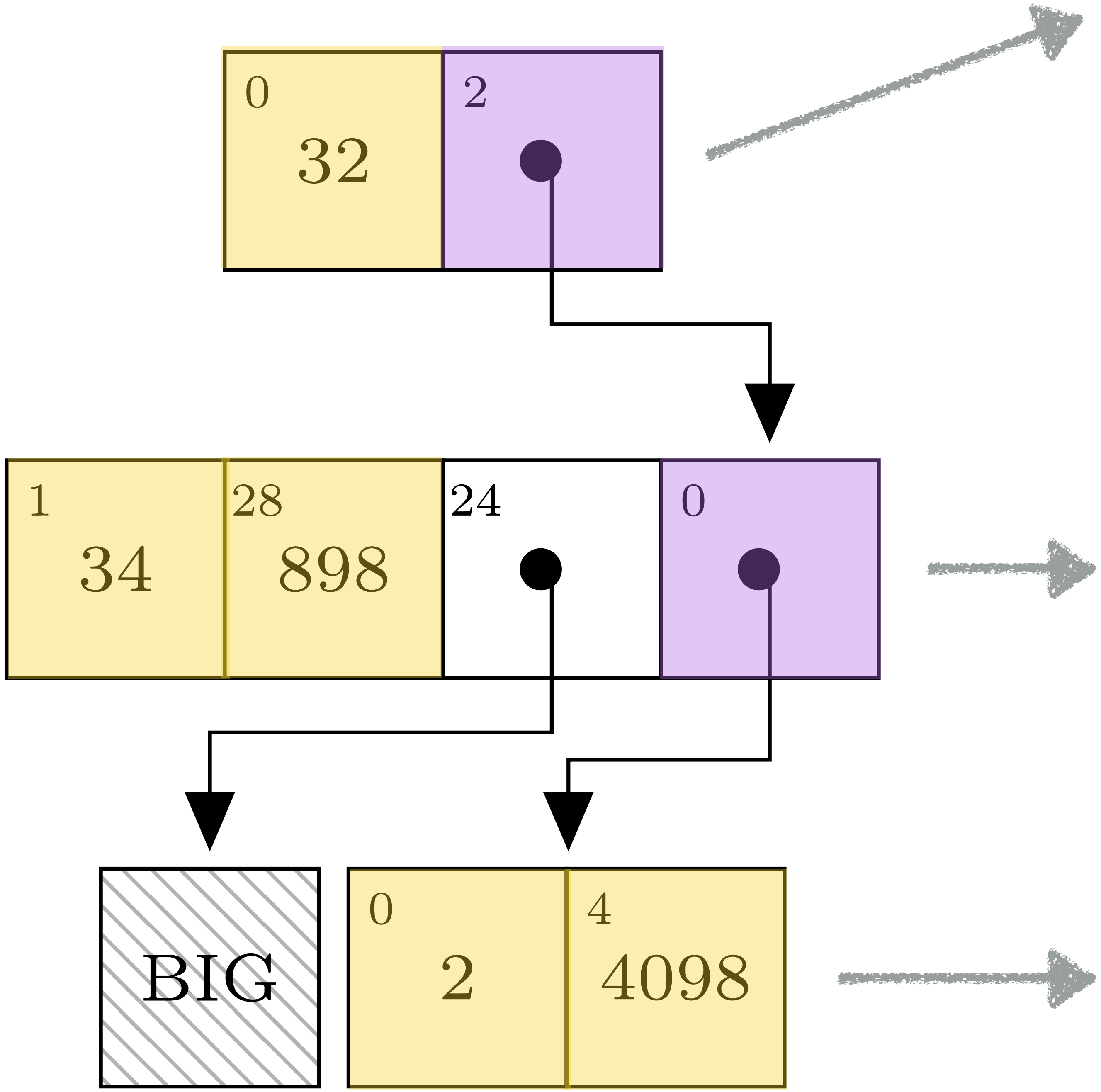
```
class TrieNode {
  int datamap;
  int nodemap;
  Object[] mixedContent;
}
```

Steindorfer, M. J., & Vinju, J. J. (2015). Optimizing Hash-Array Mapped Tries
for Fast and Lean Immutable JVM Collections. Under Submission.



	1	0
datamap	primitive	-
ref. map	node	-
both maps	big integer	-

```
class TrieNode {
  int datamap;
  int referencemap;
  int[] primitives;
  Object[] references;
}
```



```
class Node1x1
     extends TrieNode {
   int datamap;
   int nodemap;
   int key0;
   TrieNode node0;
}

class Node2x2
     extends TrieNode {
   int datamap;
   int referencemap;
   int key0;
   int key1;
   Object slot0;
   Object slot1;
}

class Node2x0
     extends TrieNode {
   int valmap;
   int key0;
   int key1;
}
```

Steindorfer, M. J., & Vinju, J. J. (2014). Code specialization for memory efficient hash tries. GPCE'14

Fine Granularity!

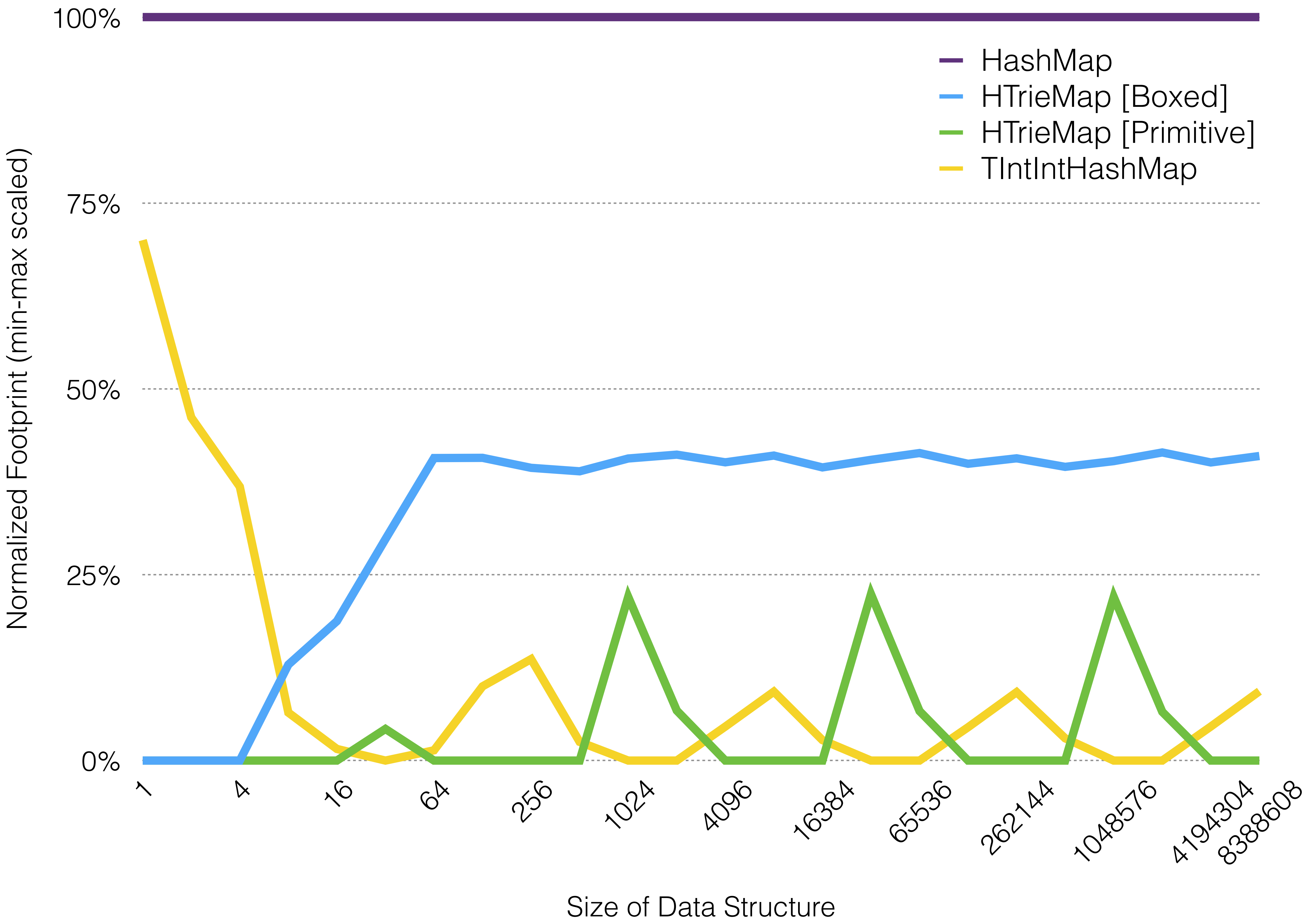
Comparison of Map[Integer, Integer]:

- `HTrieMap vs java.util.HashMap`

Comparison of Map[`int, int`]:

- `HTrieMap vs gnu.trove.map.hash.TIntIntHashMap`

Footprints of different HashMaps [2^x, 1 <= x <= 23]



Summary & Expectations

Summary & Expectations

- fine grained heterogeneous encoding

Summary & Expectations

- fine grained heterogeneous encoding
- uninitialized primitive properties (“null”)

Summary & Expectations

- fine grained heterogeneous encoding
- uninitialized primitive properties (“null”)
- storage strategy substitution/alternative

Summary & Expectations

- fine grained heterogeneous encoding
- uninitialized primitive properties (“null”)
- storage strategy substitution/alternative
- lowering worst case performance