Sketching data and other magic tricks

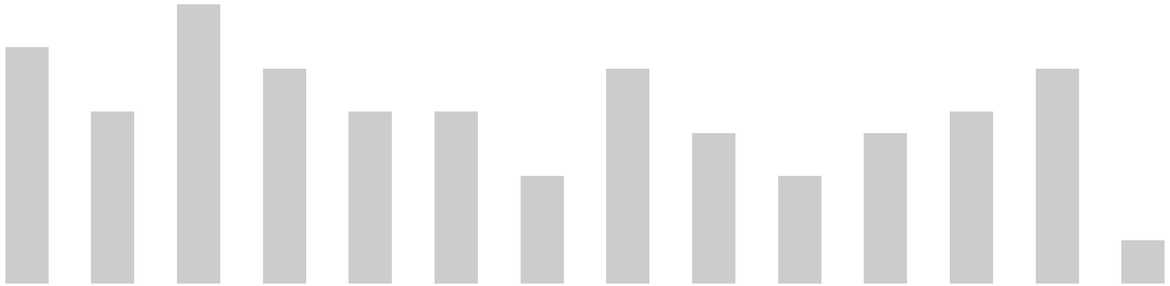
Sophie Watson and William Benton

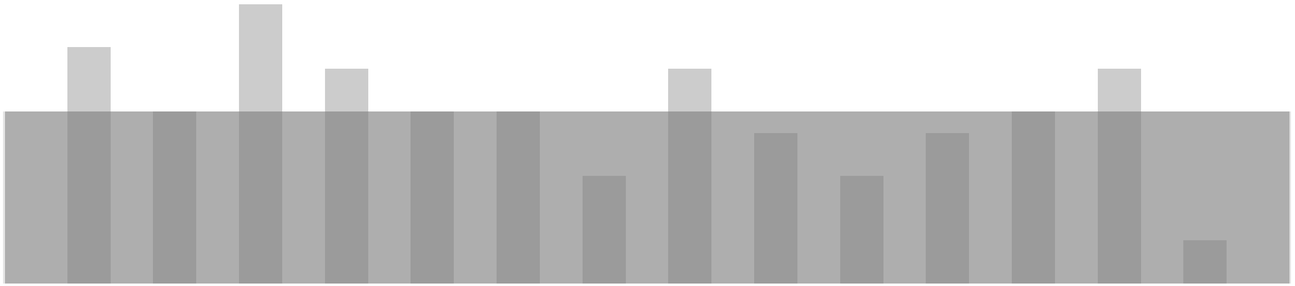
@sophwats and @willb

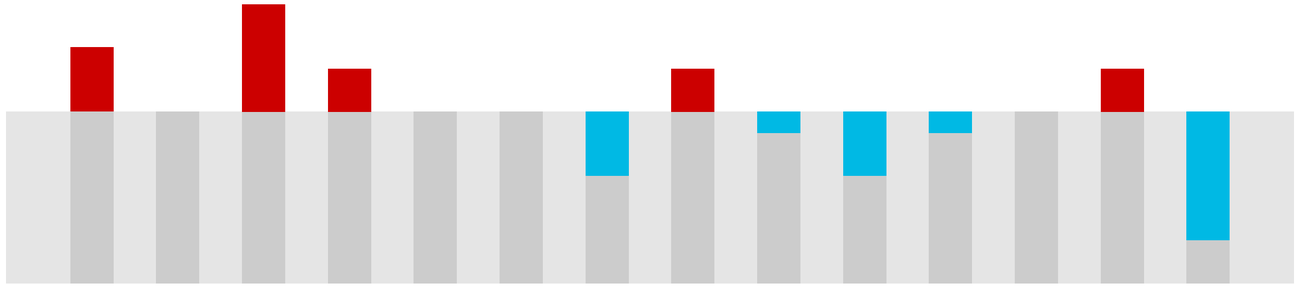
{sophie, willb}@redhat.com

Mean and variance

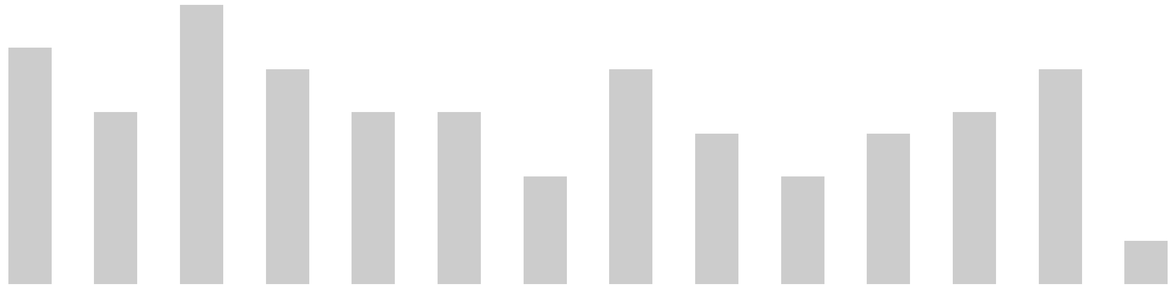
TEXTBOOK METHOD

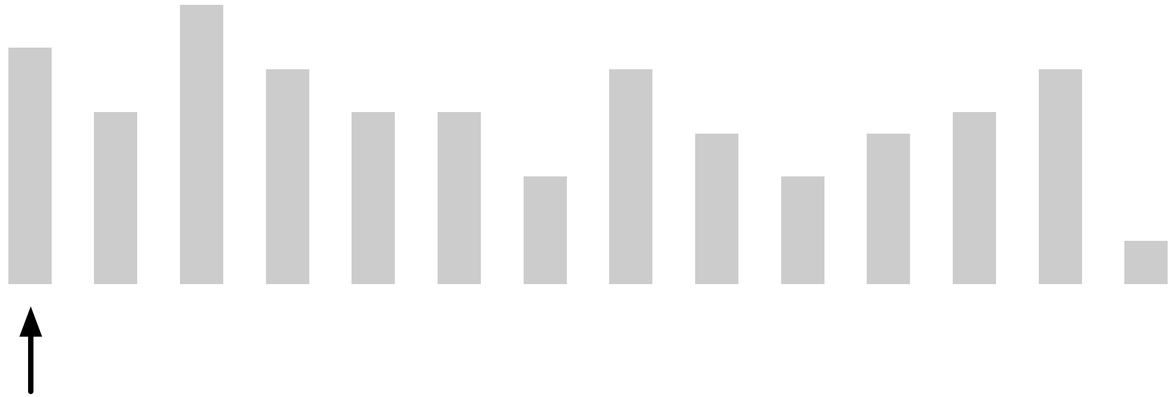


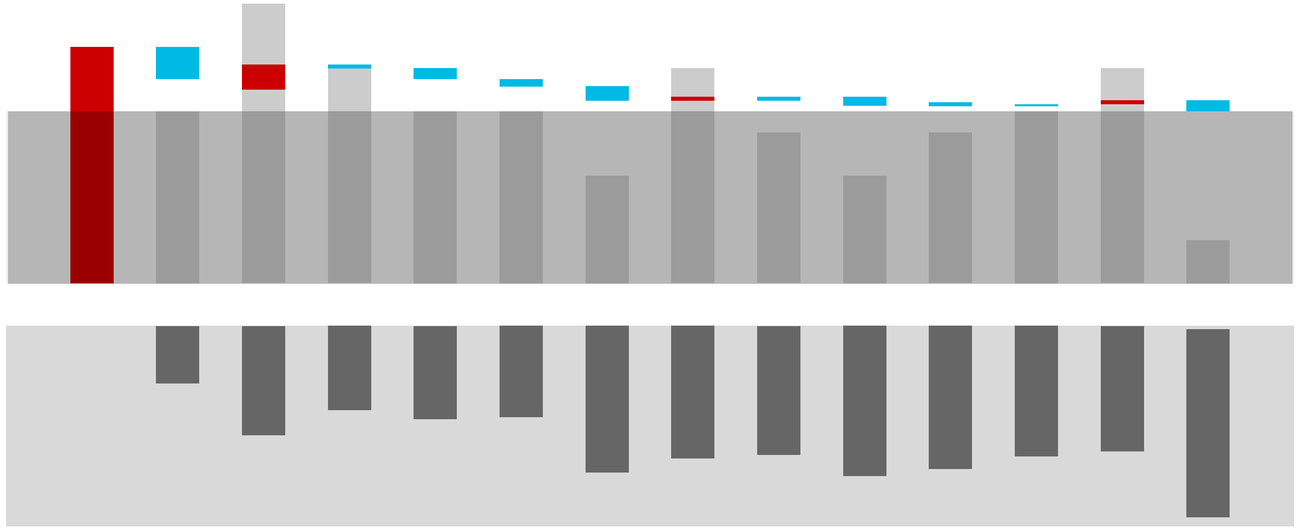


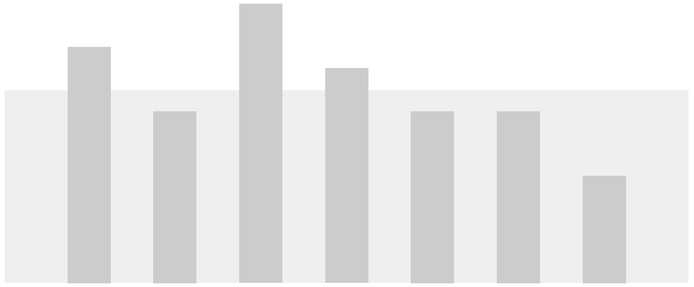


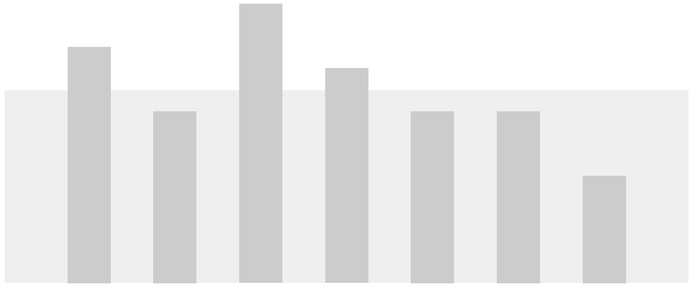
ON-LINE ESTIMATES

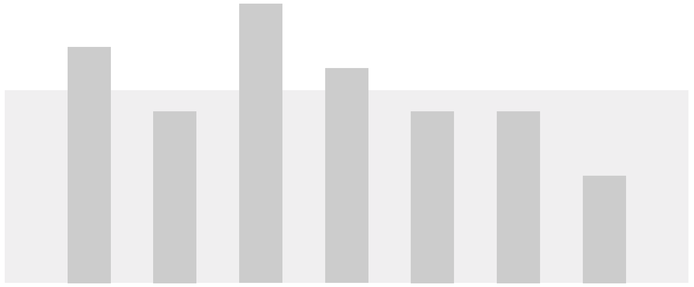


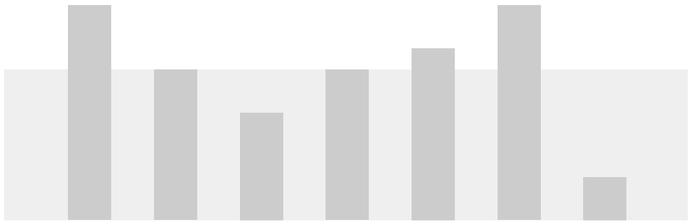


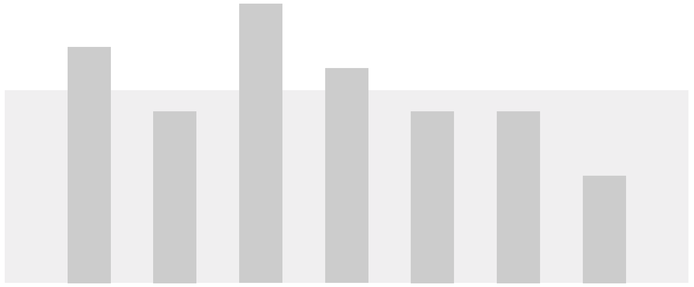


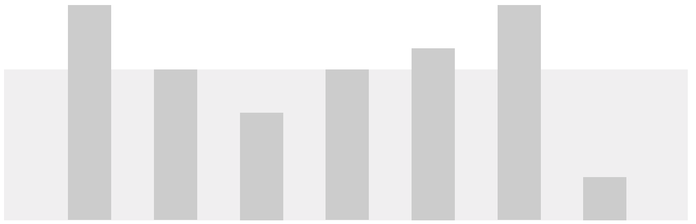


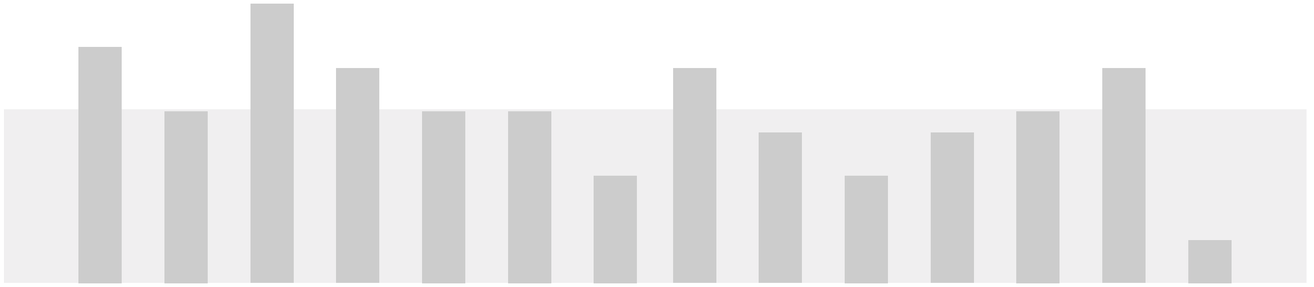


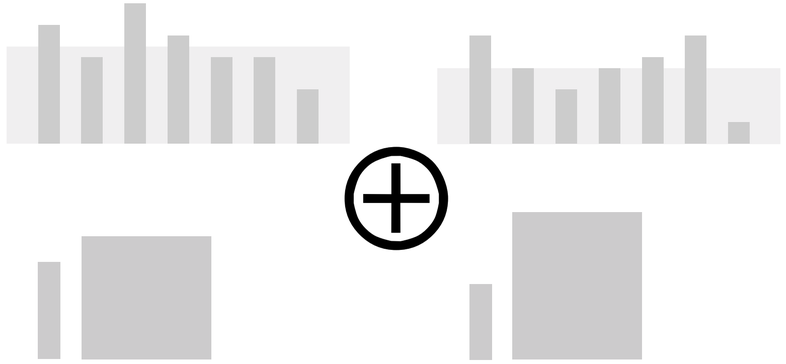












Forecast

Set membership: the Bloom filter

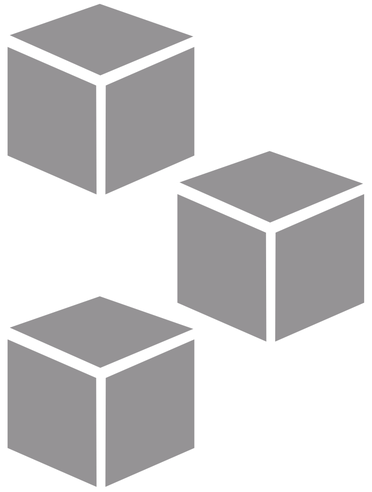
Event counts: the count-min sketch

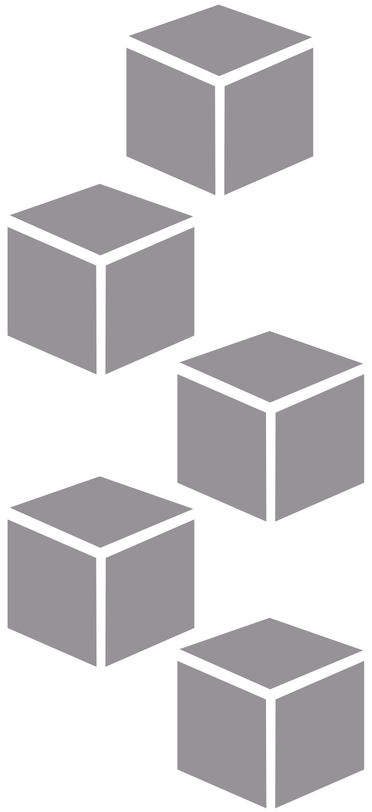
Hashing: the real magic trick behind everything

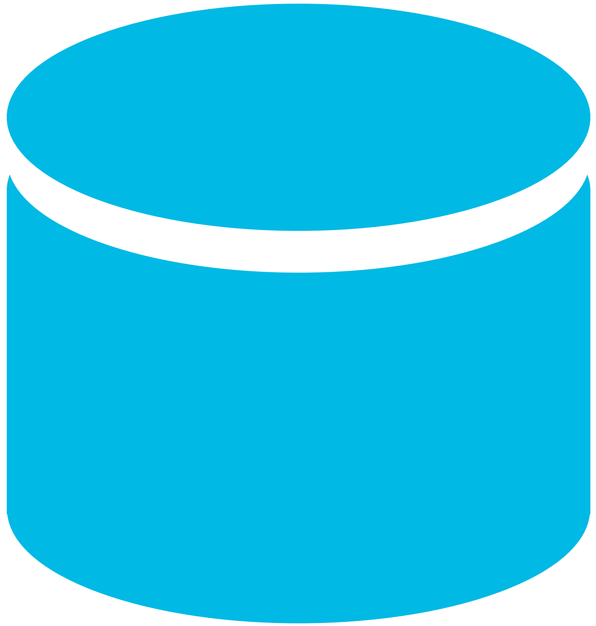
Cardinality: HyperLogLog

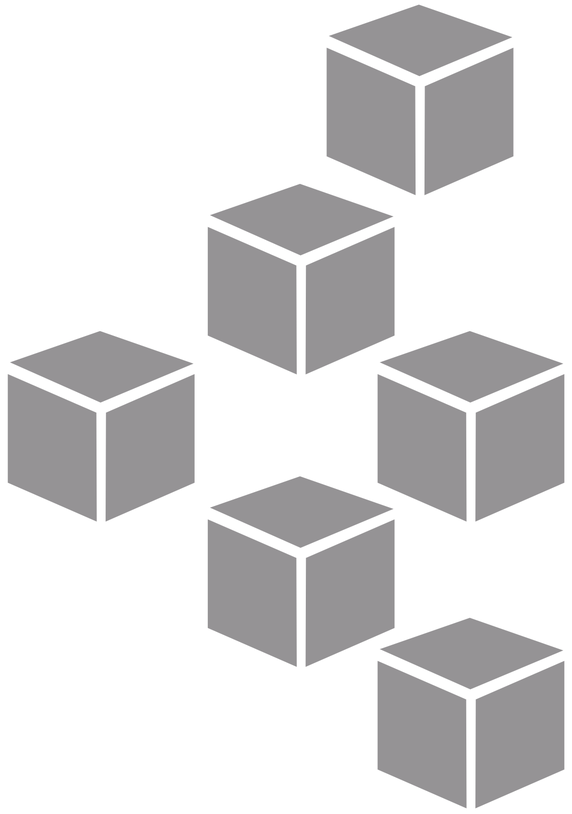
Set similarity: Minhash

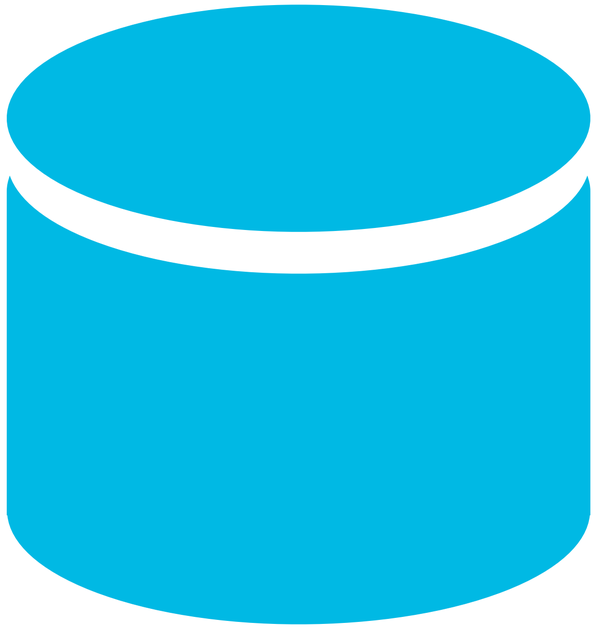
Set membership

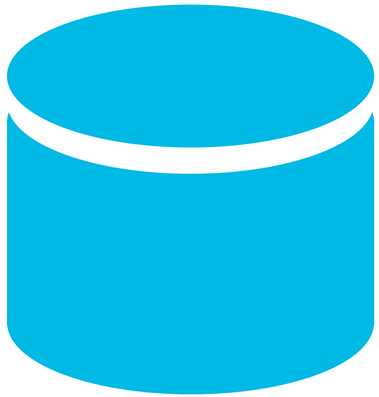


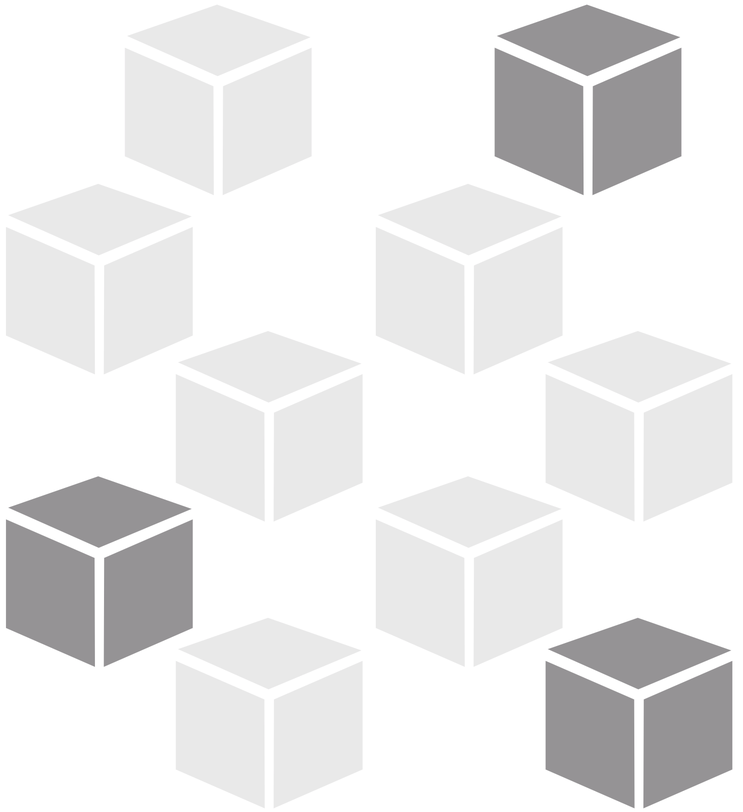




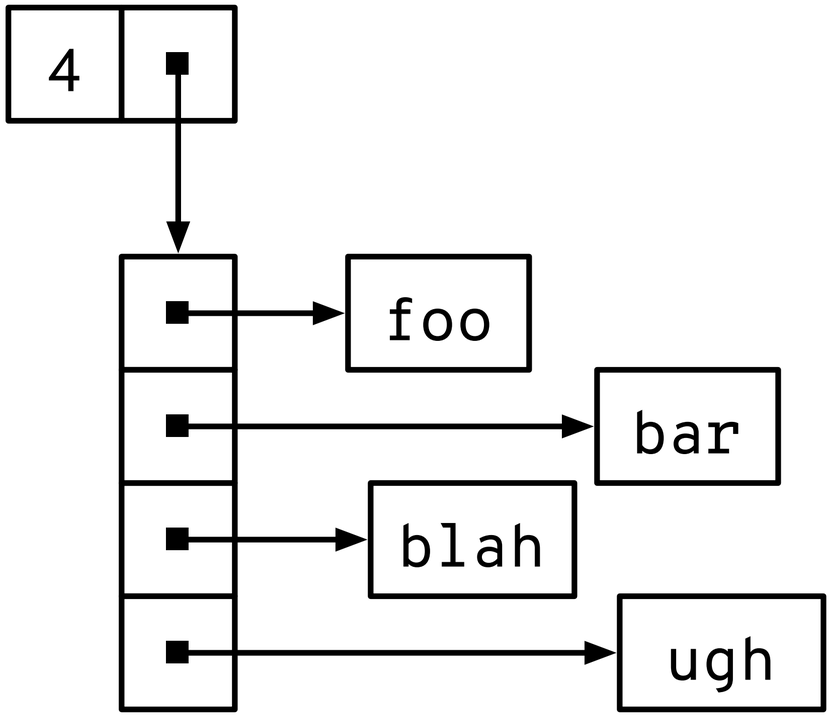


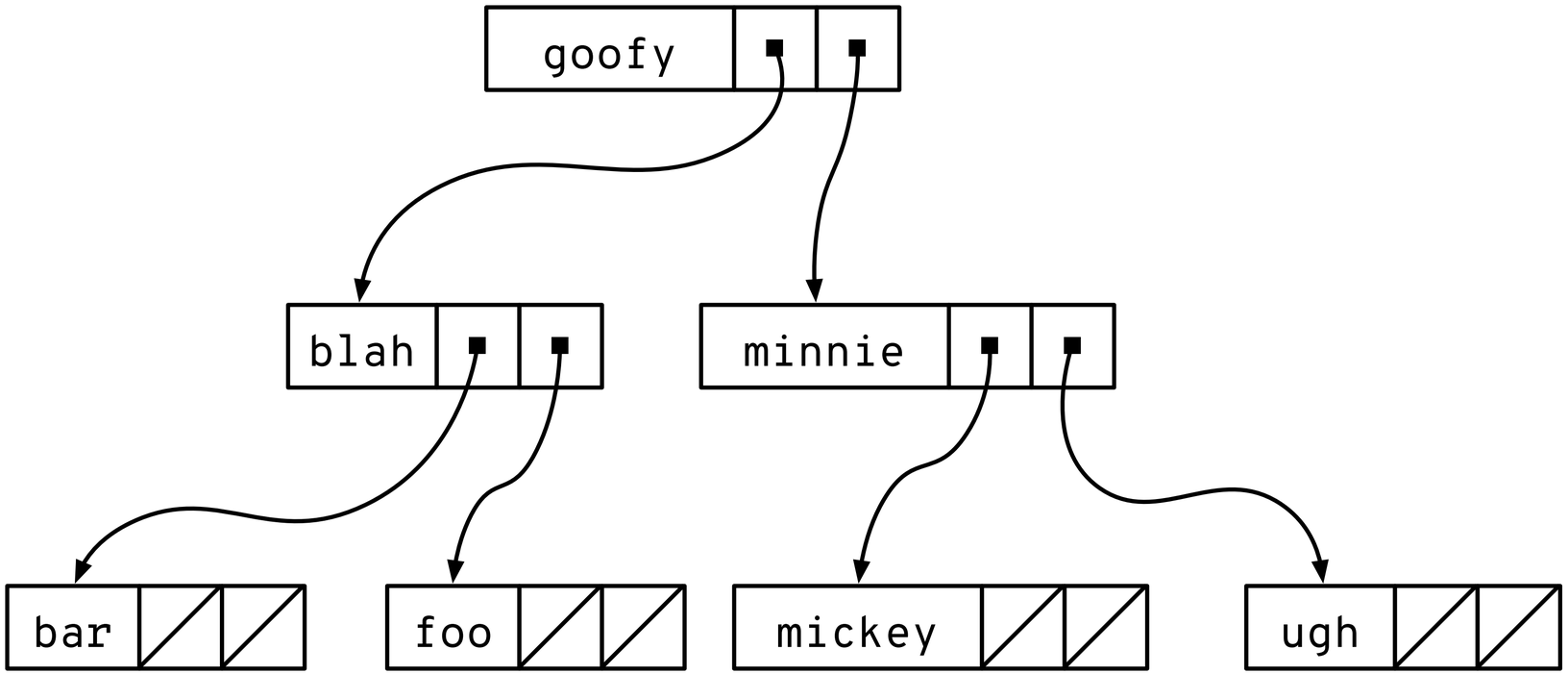


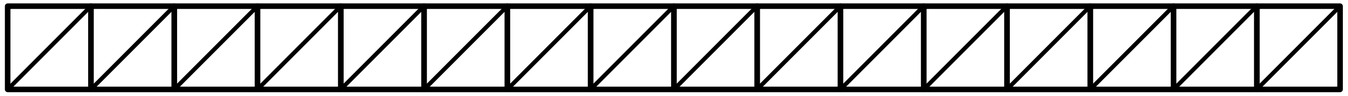


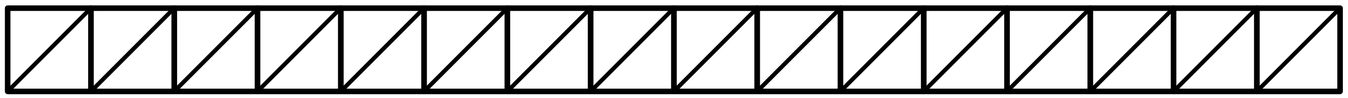


PRECISE STRUCTURES

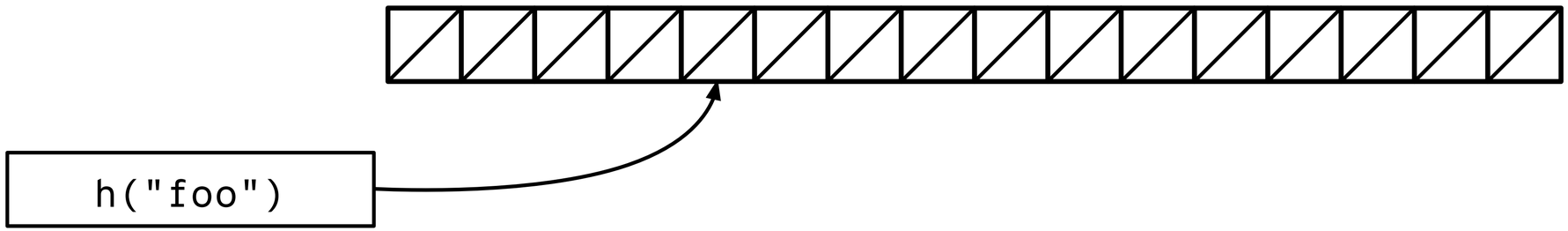

4
foo
bar
blah
ugh


goofy
blah
minnie
bar
foo
mickey
ugh

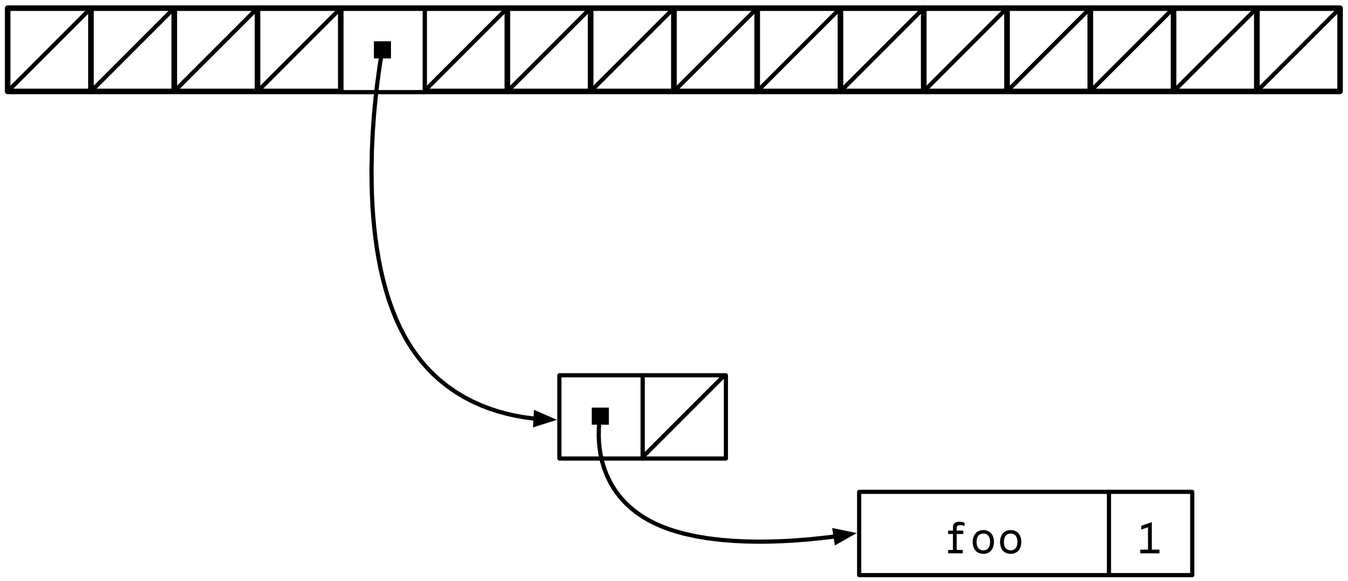




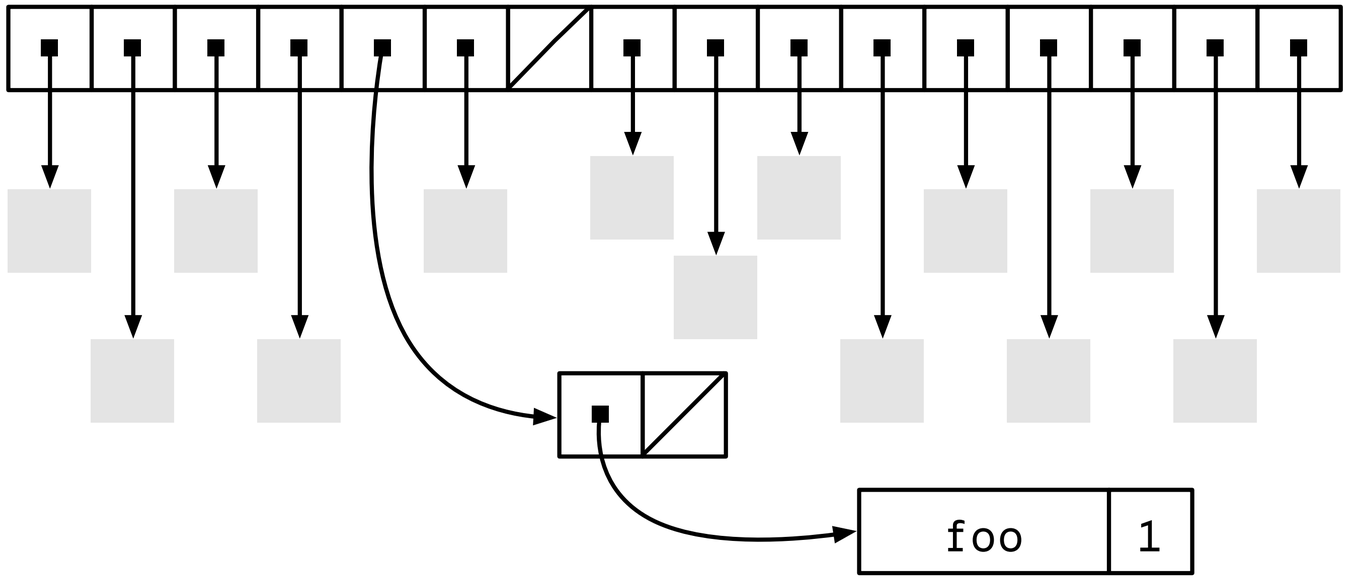
```
h.put("foo", 1)
```

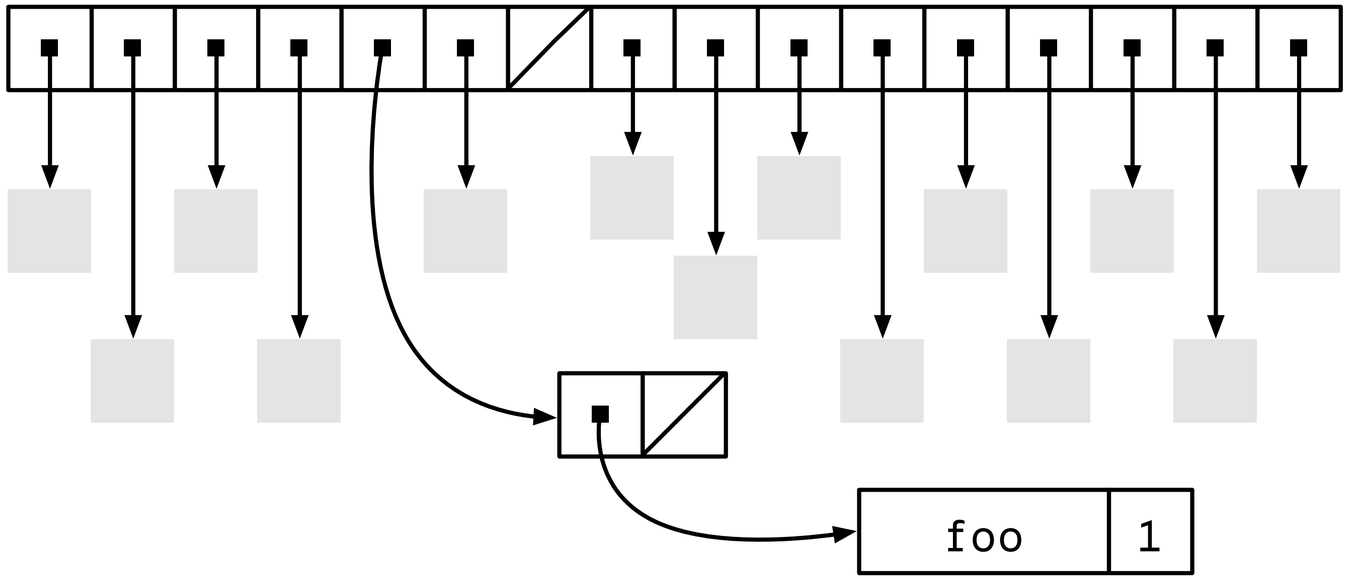



```
h.put("foo", 1)
```

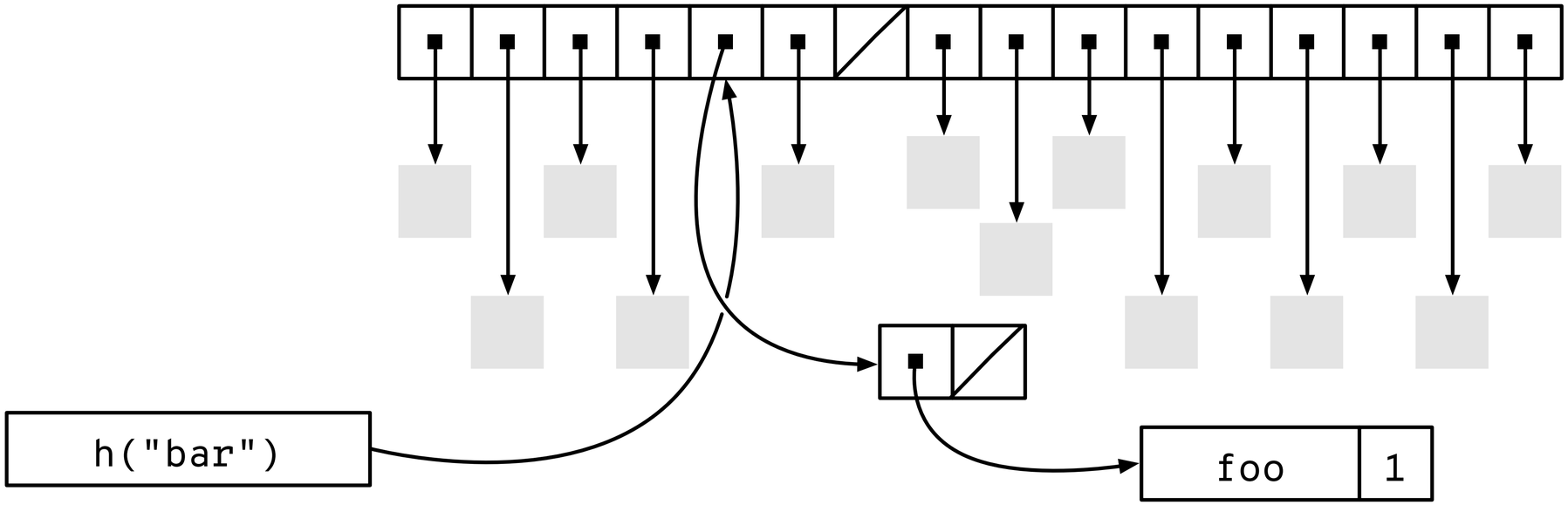



```
h.put("foo", 1)
```

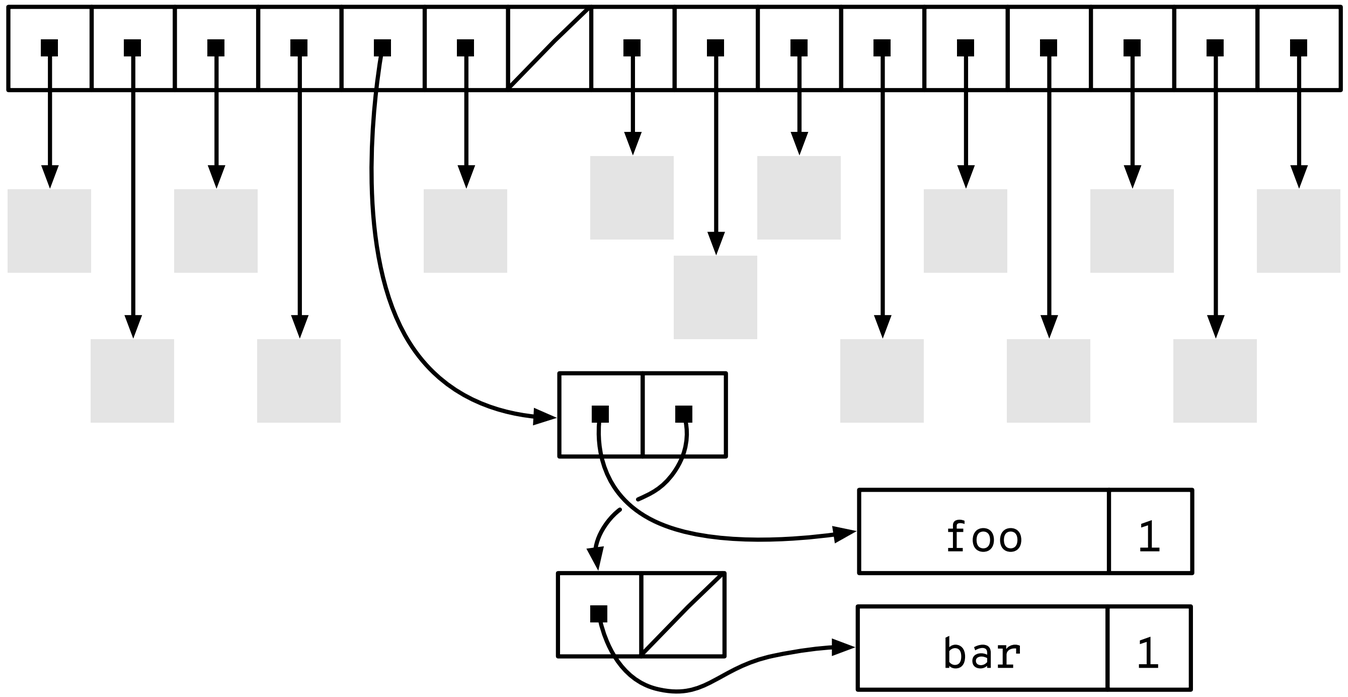

foo
1




```
h.put("bar", 1)
```

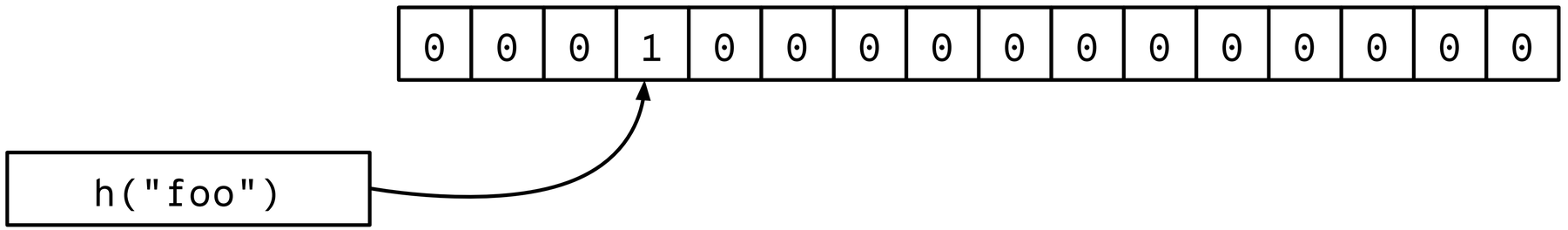


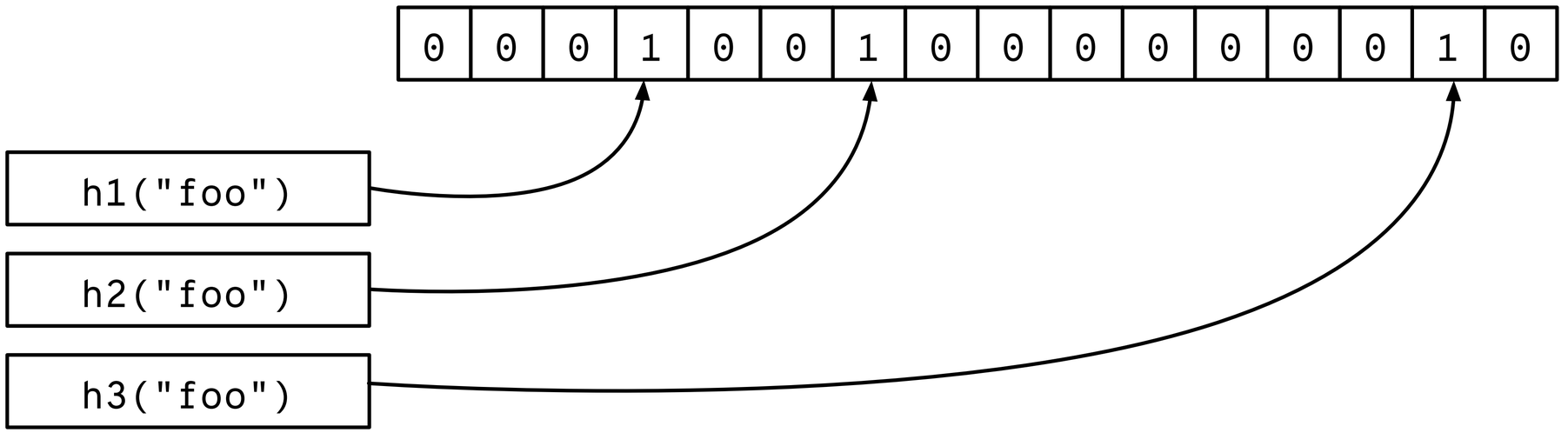
h.put("bar", 1)

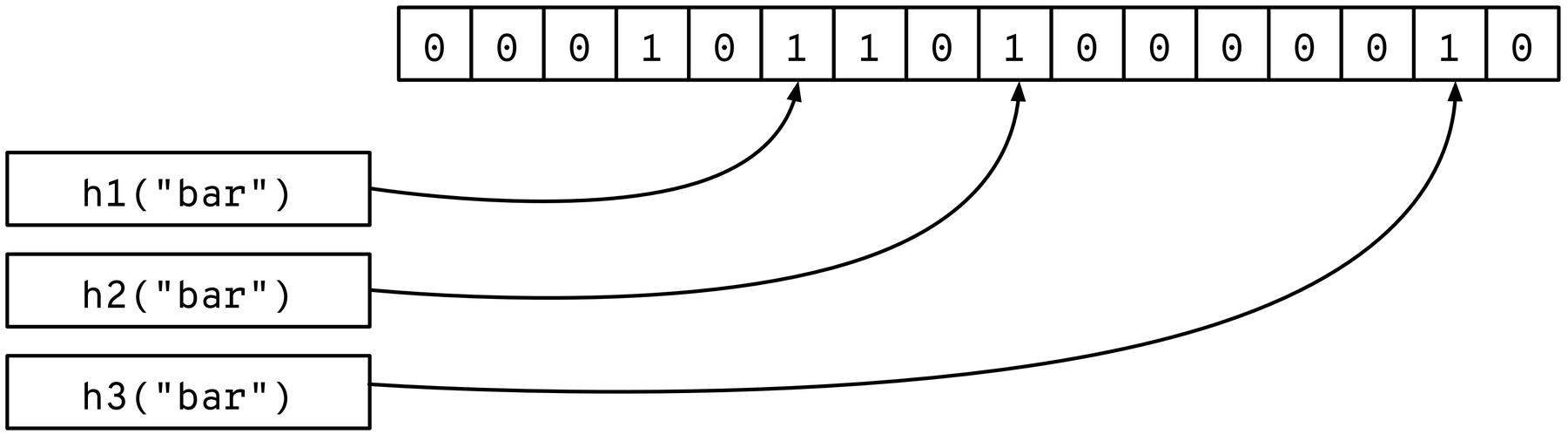


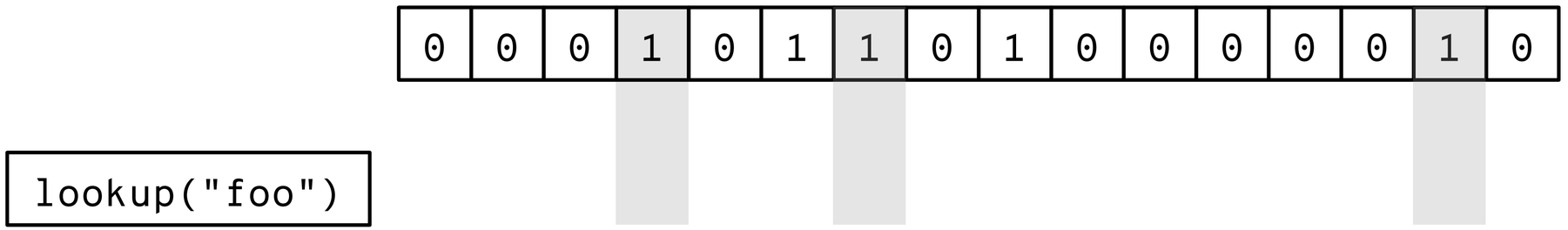

```
h.put("bar", 1)
```

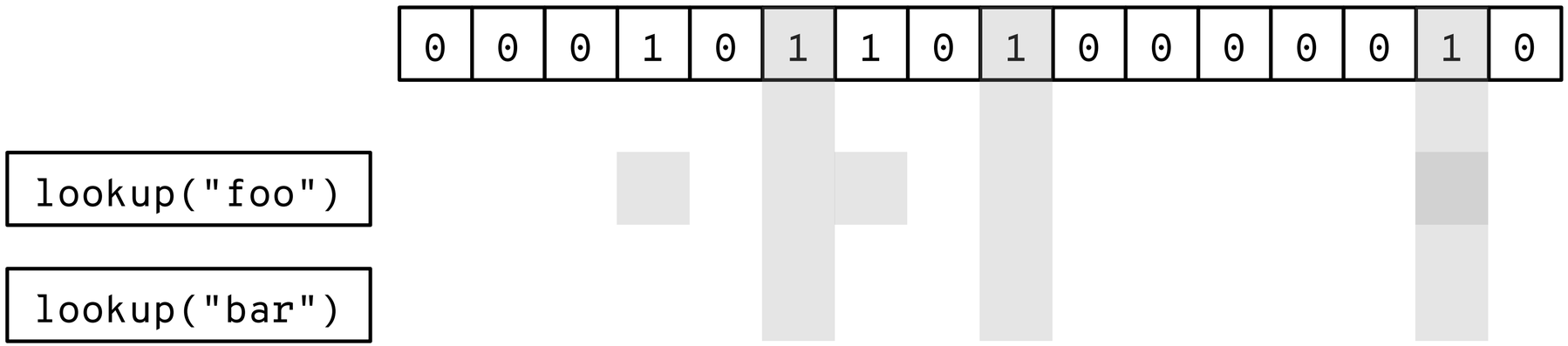
BLOOM FILTERS

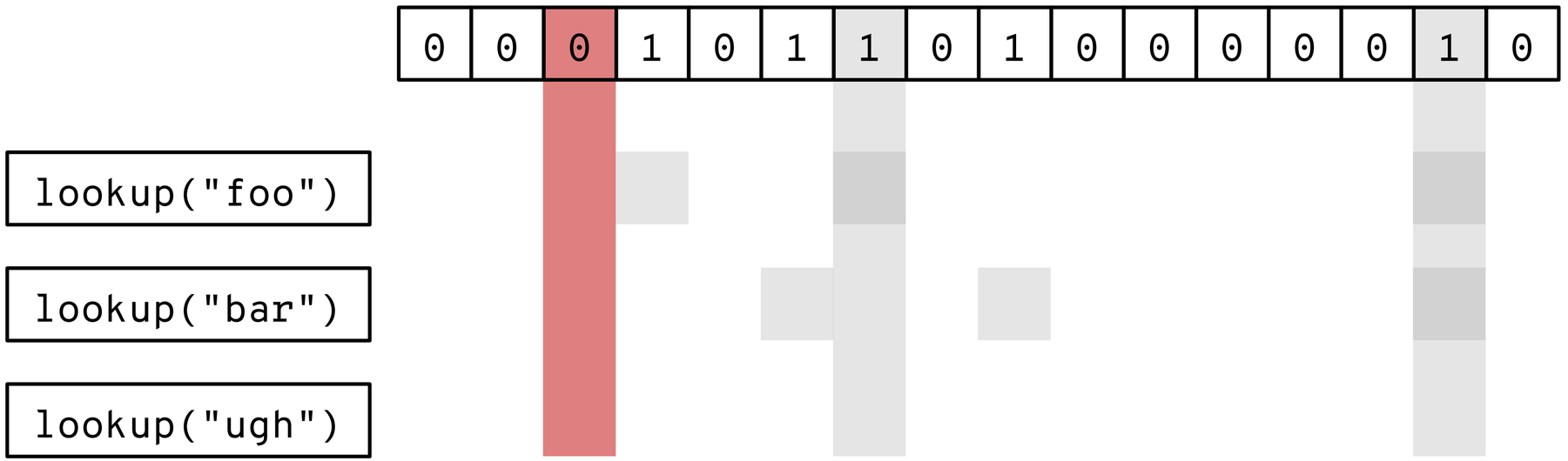

0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
h("foo")

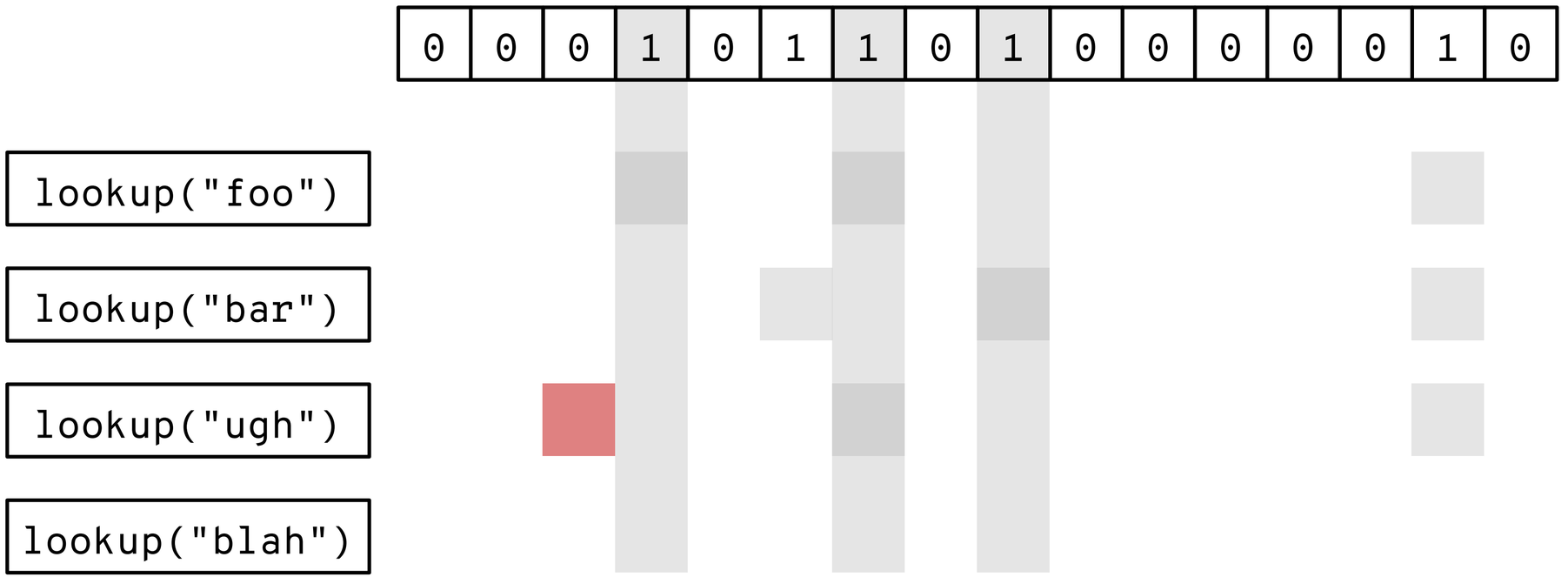

0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0
h1("foo")
h2("foo")
h3("foo")

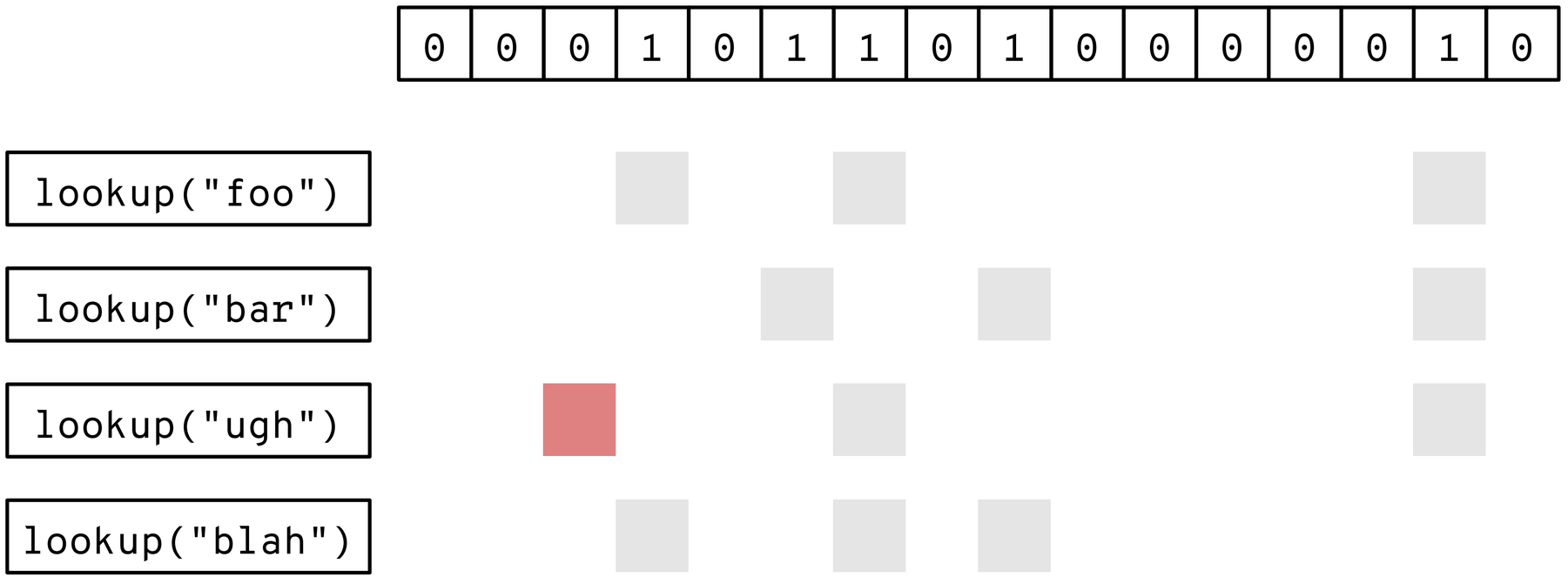

0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0
h1("bar")
h2("bar")
h3("bar")

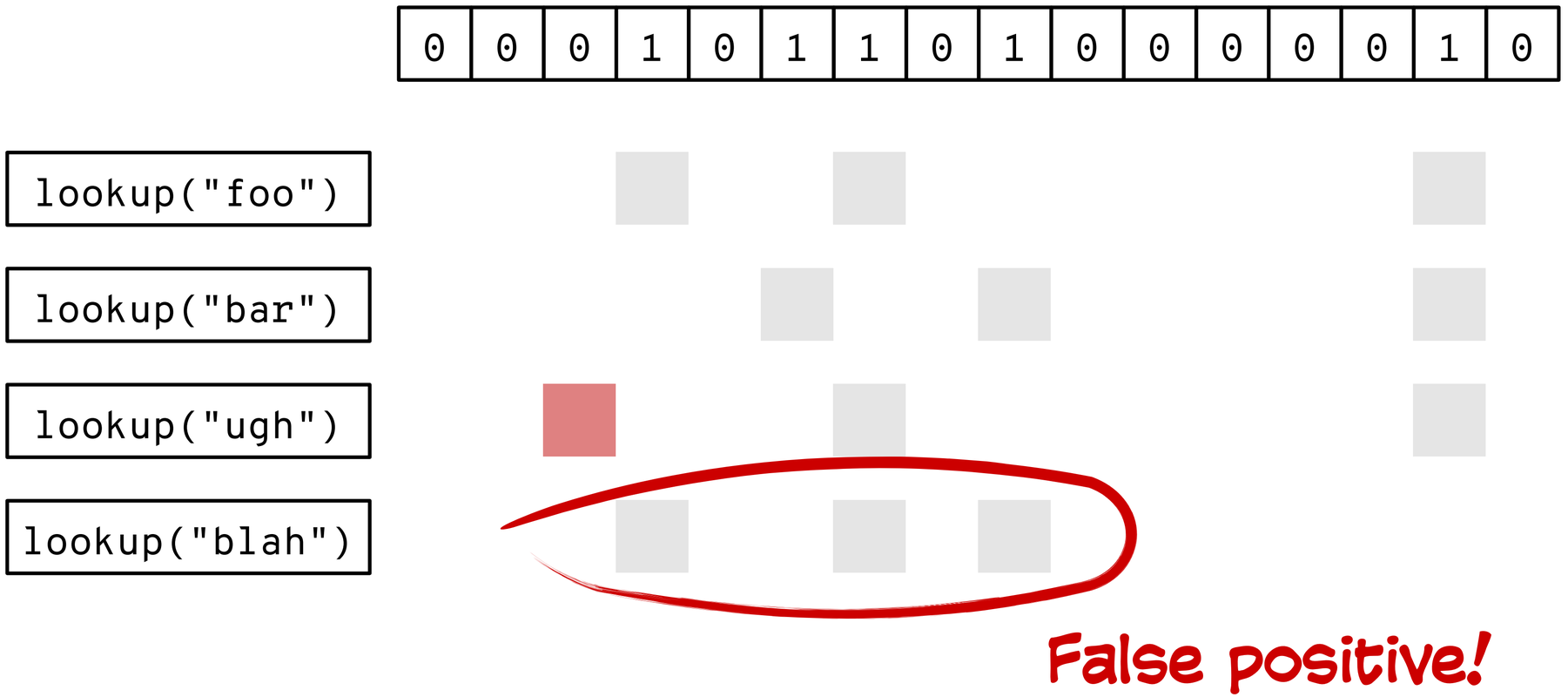

0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0
lookup("foo")


0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0
lookup("foo")
lookup("bar")

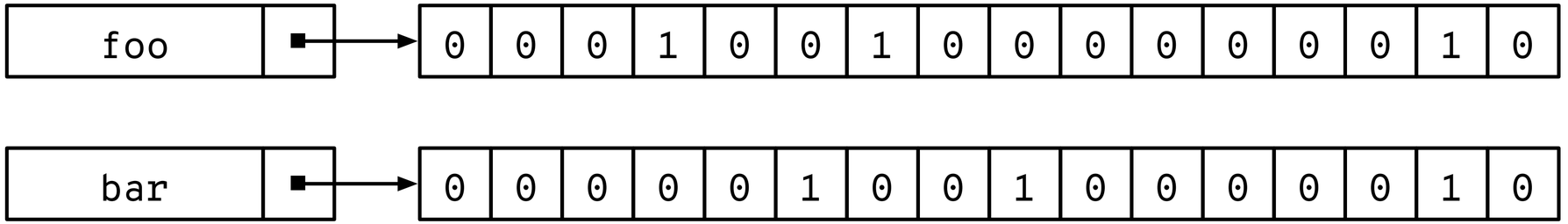

0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0
lookup("foo")
lookup("bar")
lookup("ugh")

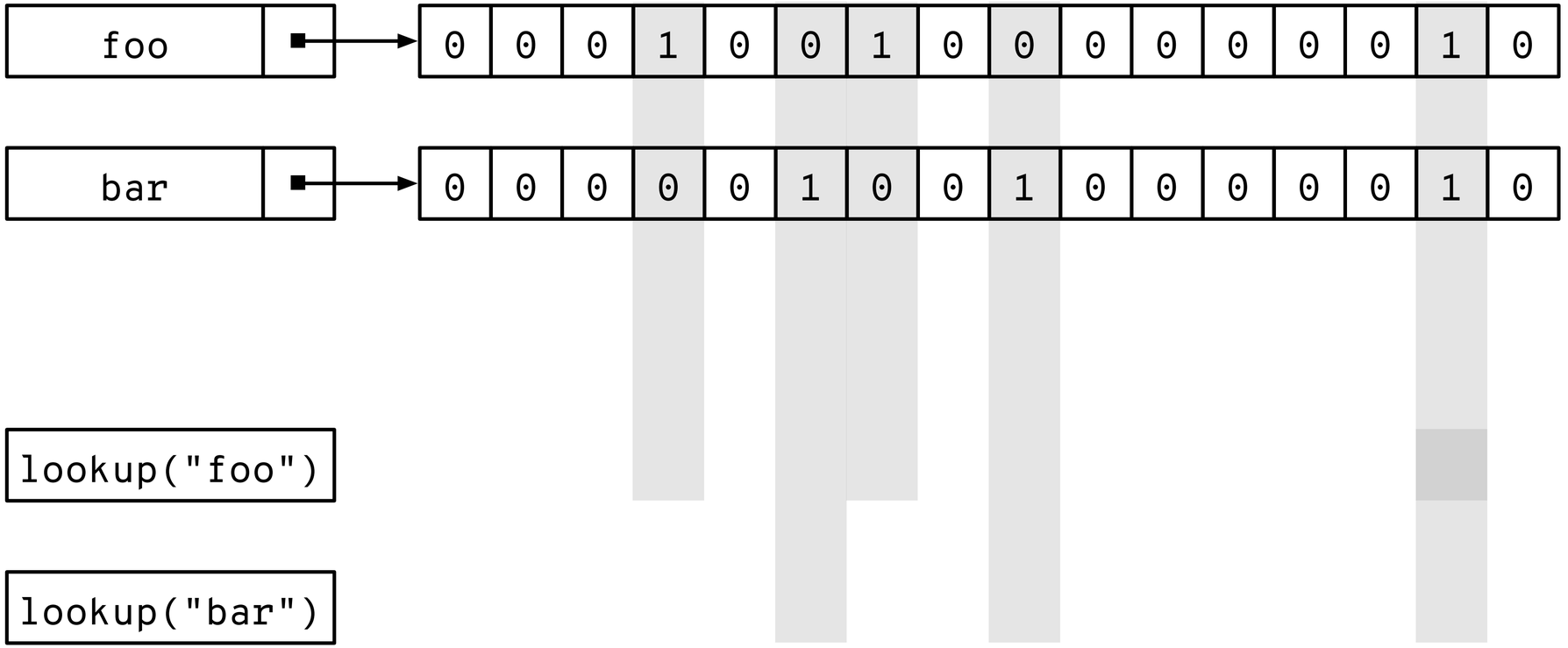

0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0
lookup("foo")
lookup("bar")
lookup("ugh")
lookup("blah")

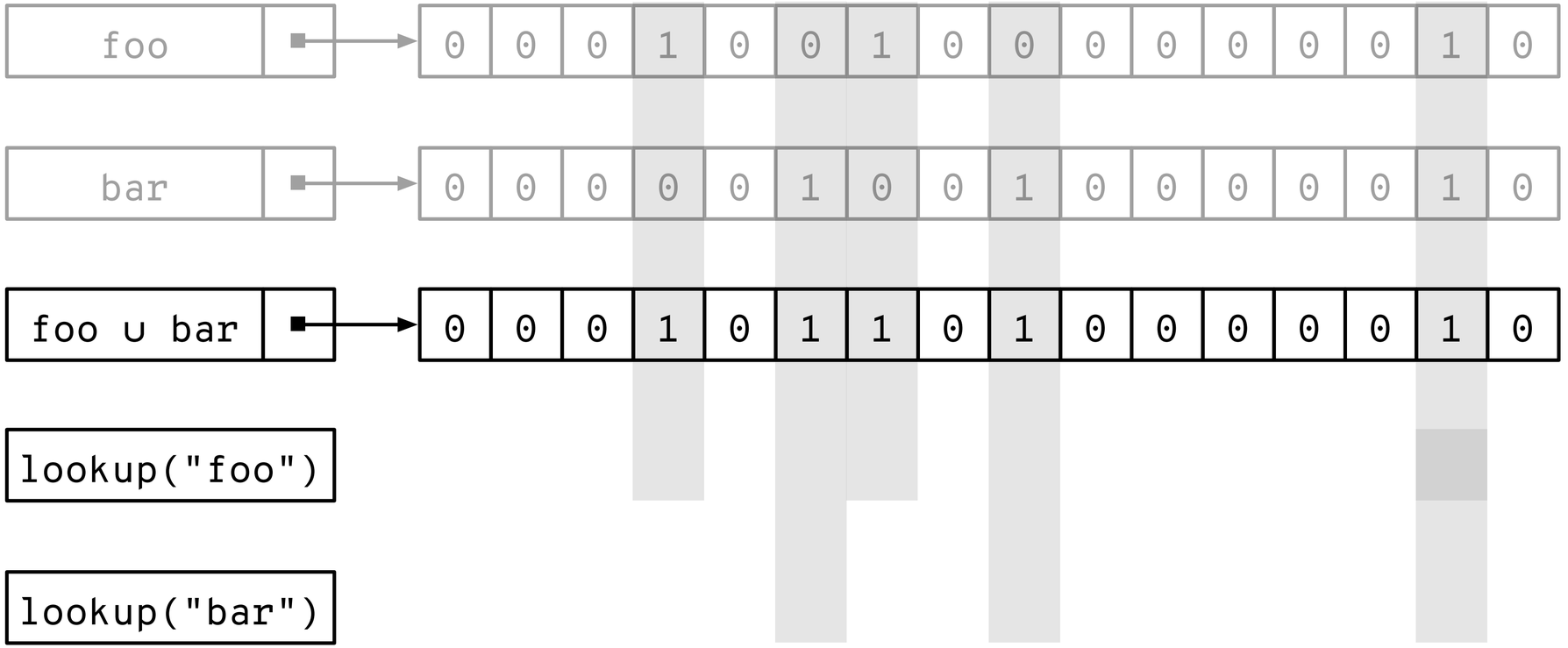

0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0
lookup("foo")
lookup("bar")
lookup("ugh")
lookup("blah")

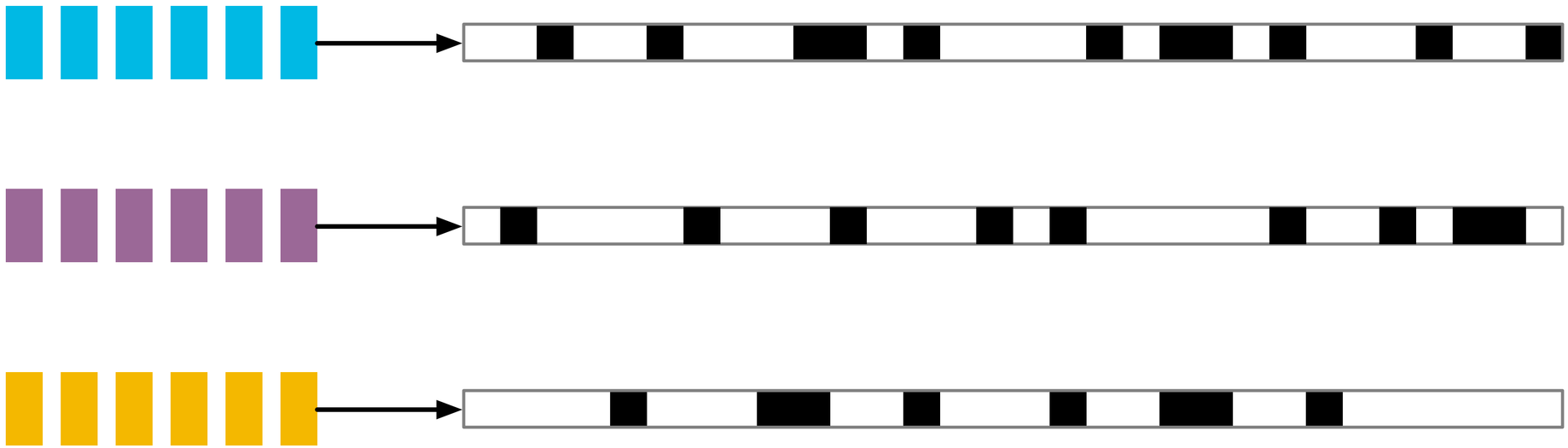

0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0
lookup("foo")
lookup("bar")
lookup("ugh")
lookup("blah")
False positive!

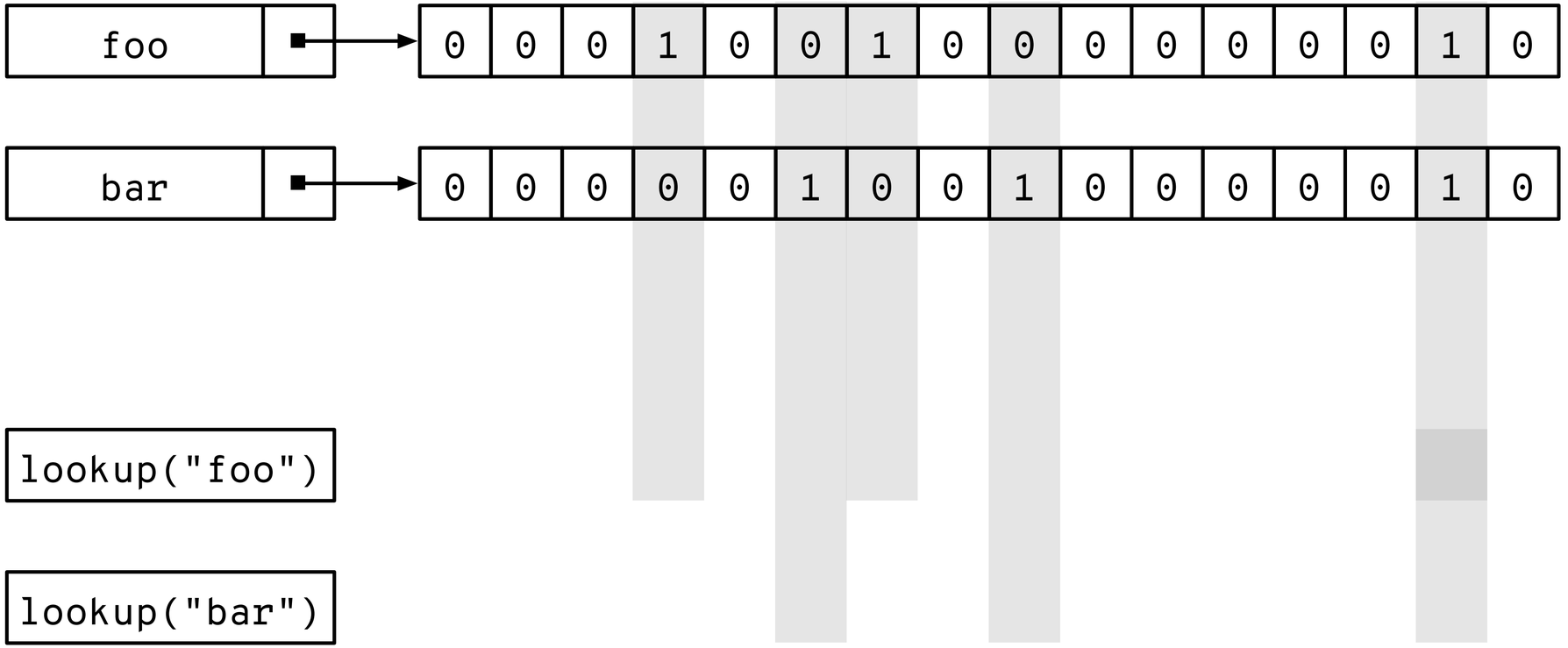
```
class Bloom(object):
    def __init__(self, size, hashes):
        self.__buckets = BitVector(size)
        self.__size = len(self.__buckets)
        self.__hashes = lambda v: [f(v) for f in hashes[:]]

    def insert(self, value):
       for h in self.__hashes(value):
            self.__buckets[h % self.__size] = True

    def lookup(self, value):
        for h in self.__hashes(value):
            if self.__buckets[h % self.__size] == False:
                return False
        return True
```

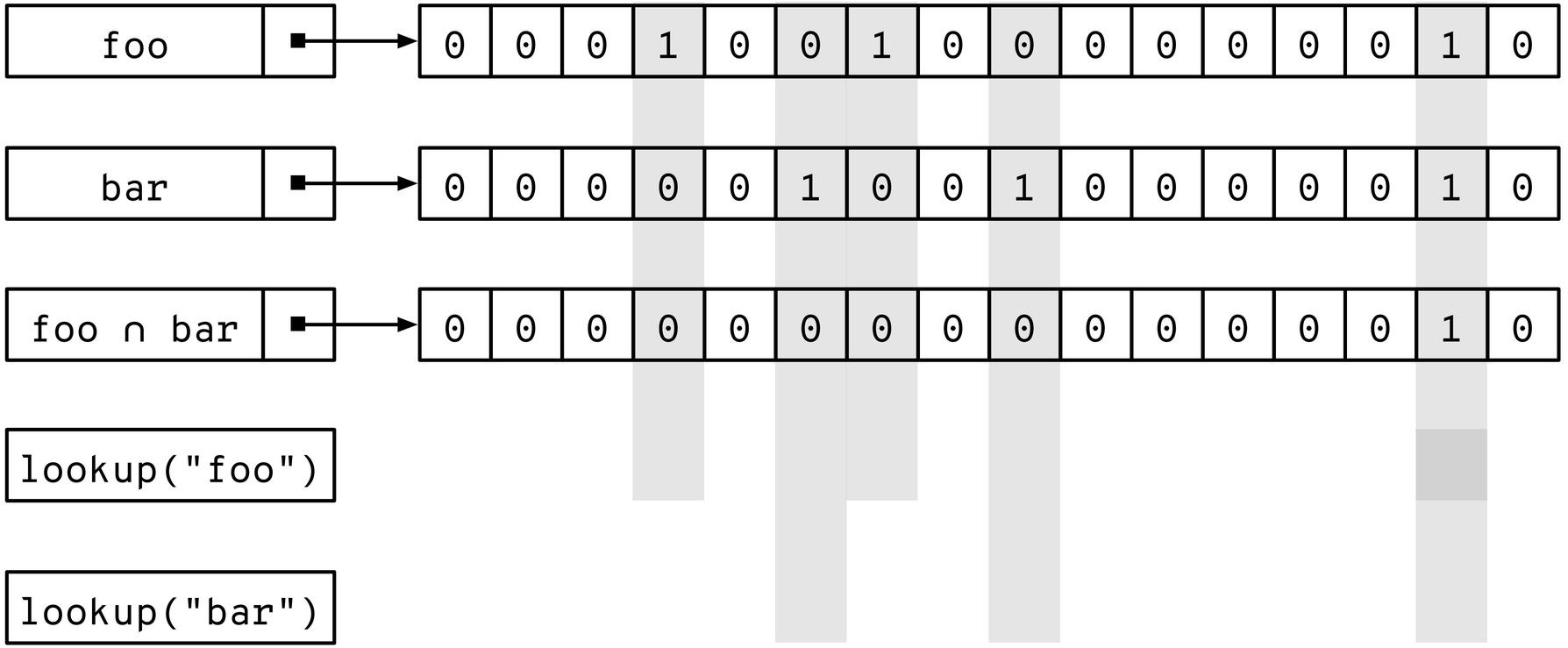

foo
0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0
bar
0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0

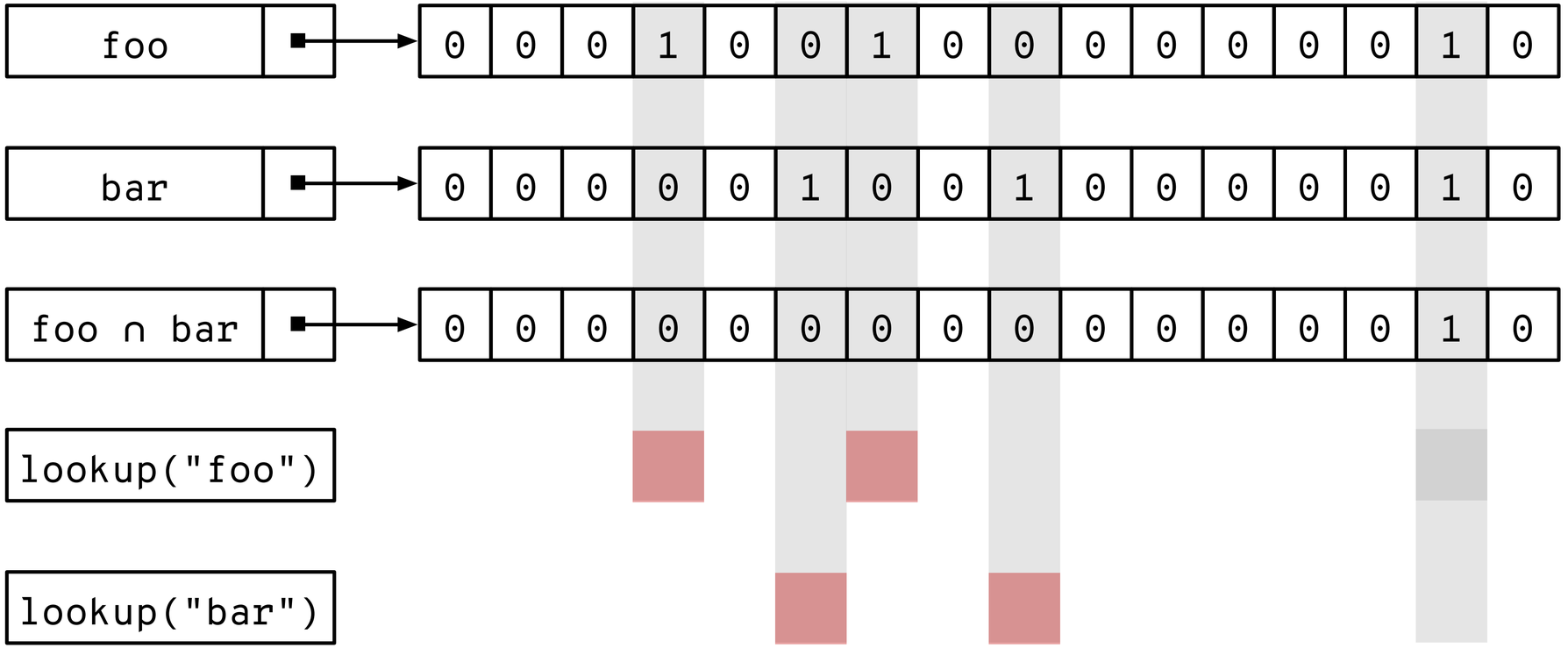

foo
0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0
bar
0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0
lookup("foo")
lookup("bar")


foo
0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0
bar
0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0
foo ∪ bar
0 0 0 1 0 1 1 0 1 0 0 0 0 0 0 1 0
lookup("foo")
lookup("bar")

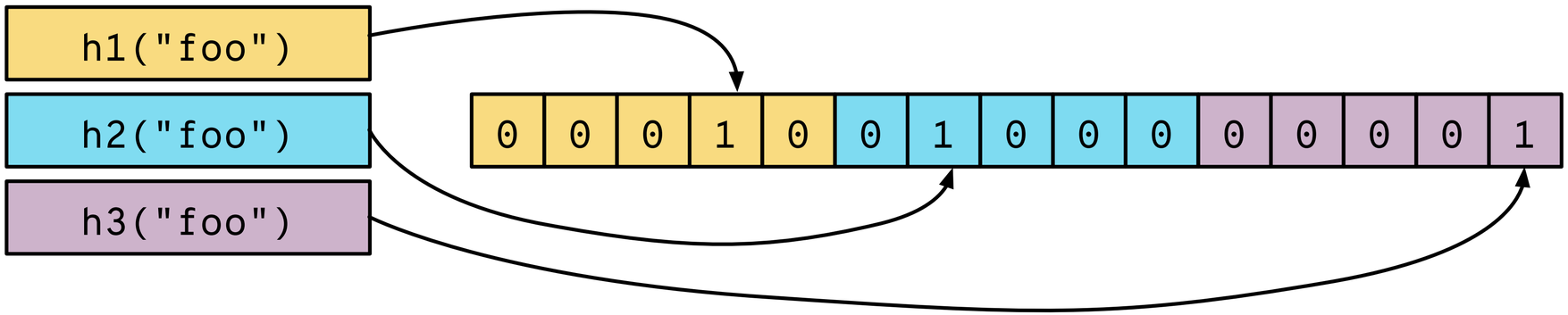


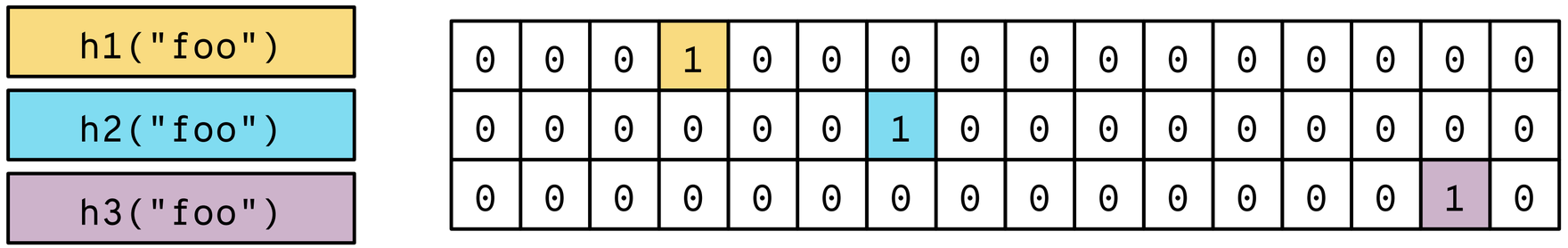

foo
0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0
bar
0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0
lookup("foo")
lookup("bar")


foo
0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0
bar
0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0
foo ∩ bar
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
lookup("foo")
lookup("bar")


foo
0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0
bar
0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0
foo ∩ bar
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
lookup("foo")
lookup("bar")

PARTITIONED FILTERS

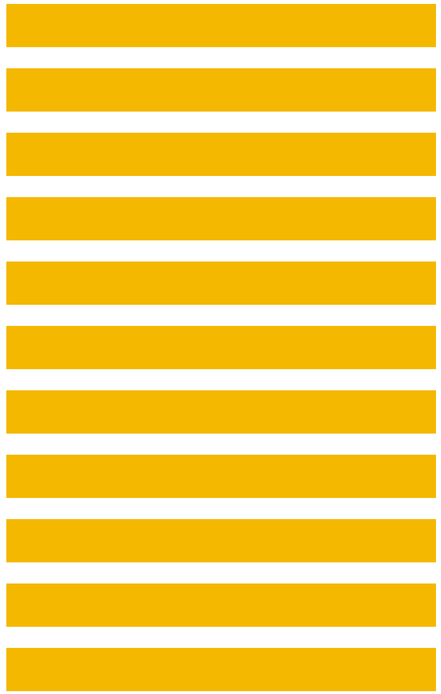

h1("foo")
h2("foo")
h3("foo")
0 0 0 1 0 0 1 0 0 0 0 0 0 0 1

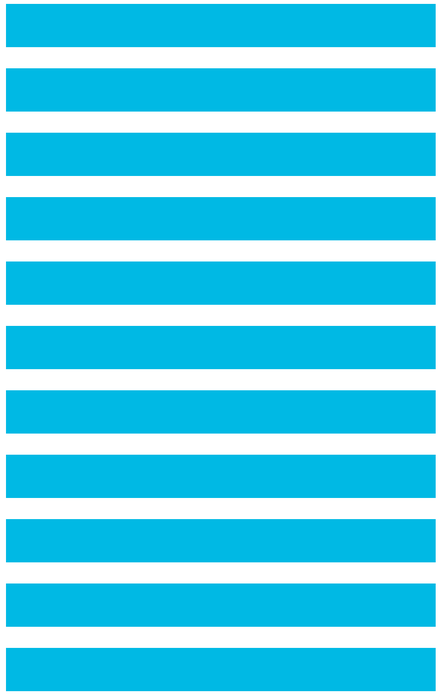

h1("foo")
h2("foo")
h3("foo")
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0

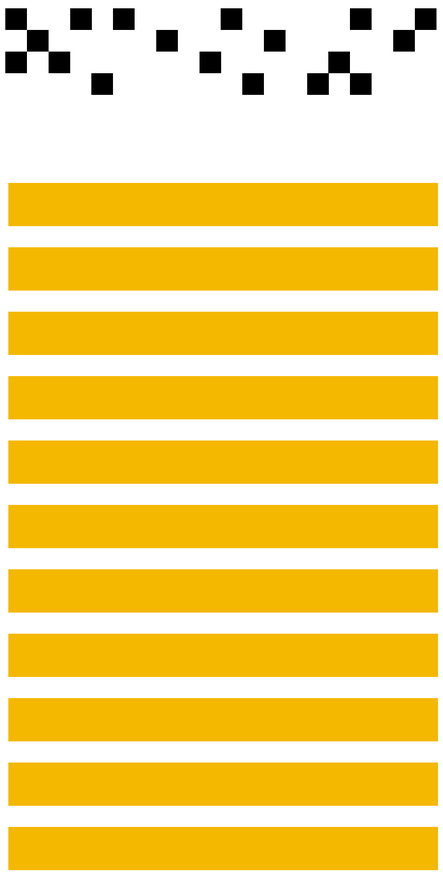
APPLICATIONS

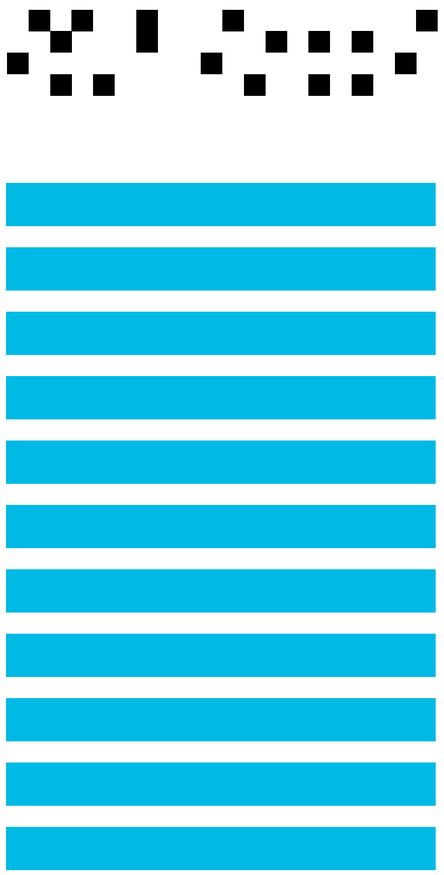
It’s prob•a•bly dif•fi•cult to imag•ine a time when a nat•u•ral lan•guage dic•tio•nary didn’t fit in main mem•ory.

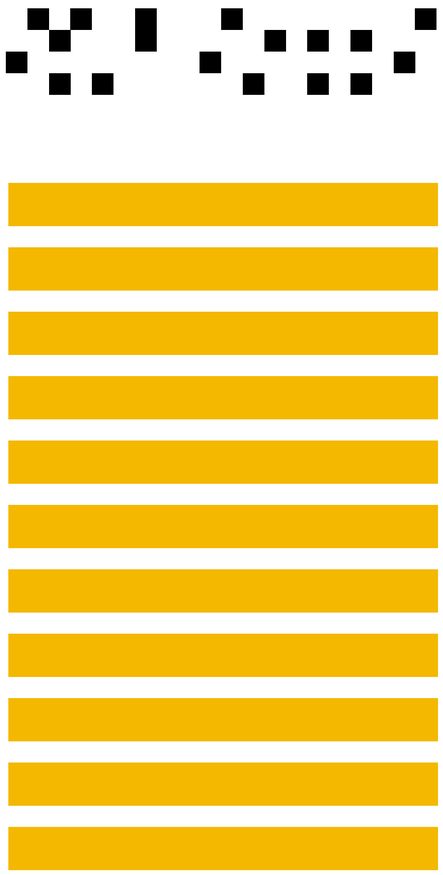
```
SELECT * FROM A,B
WHERE A.X = B.X
```

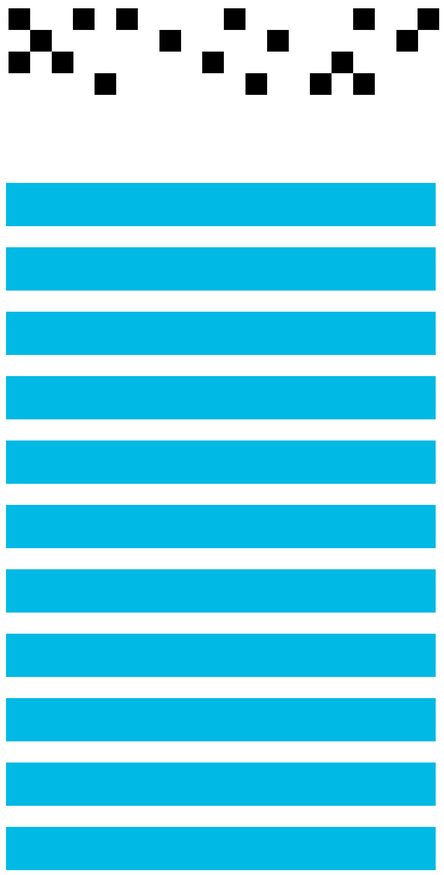


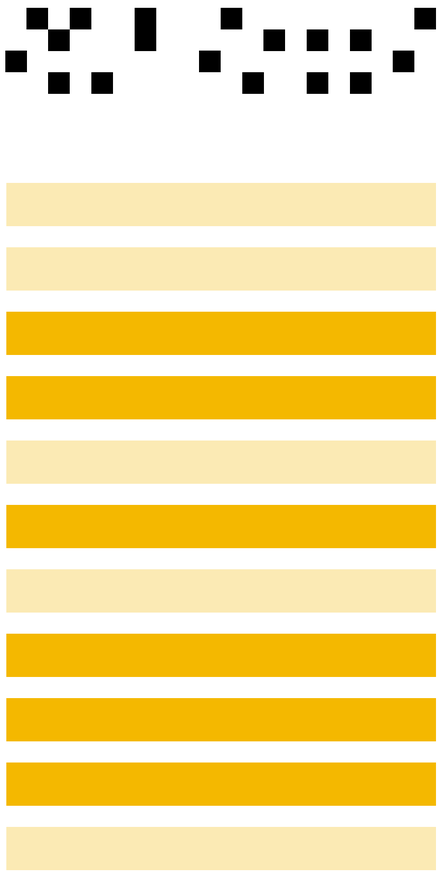


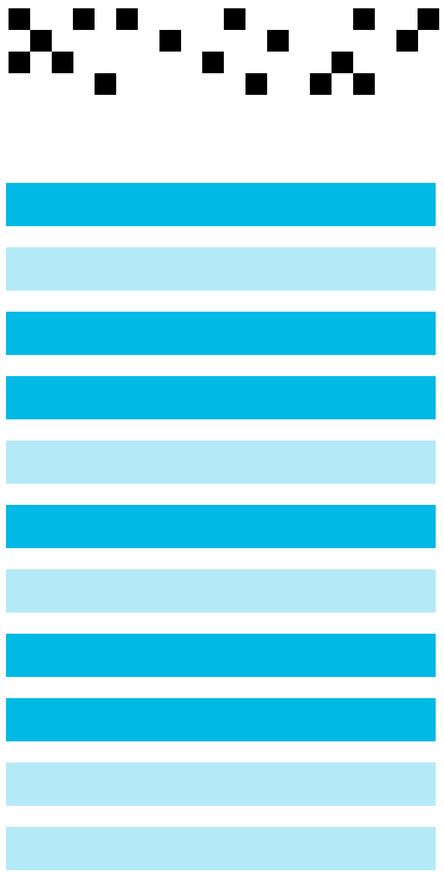


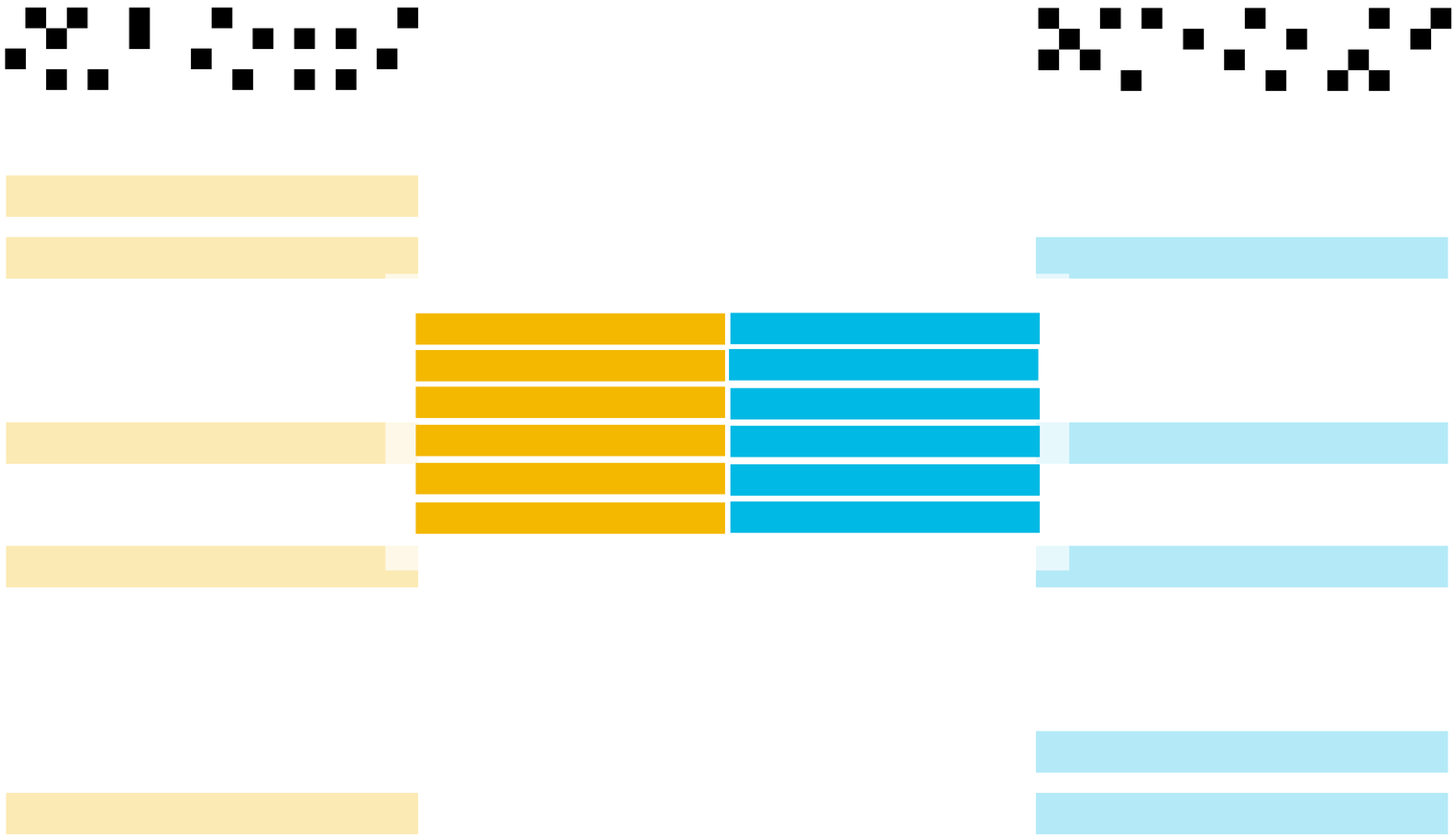


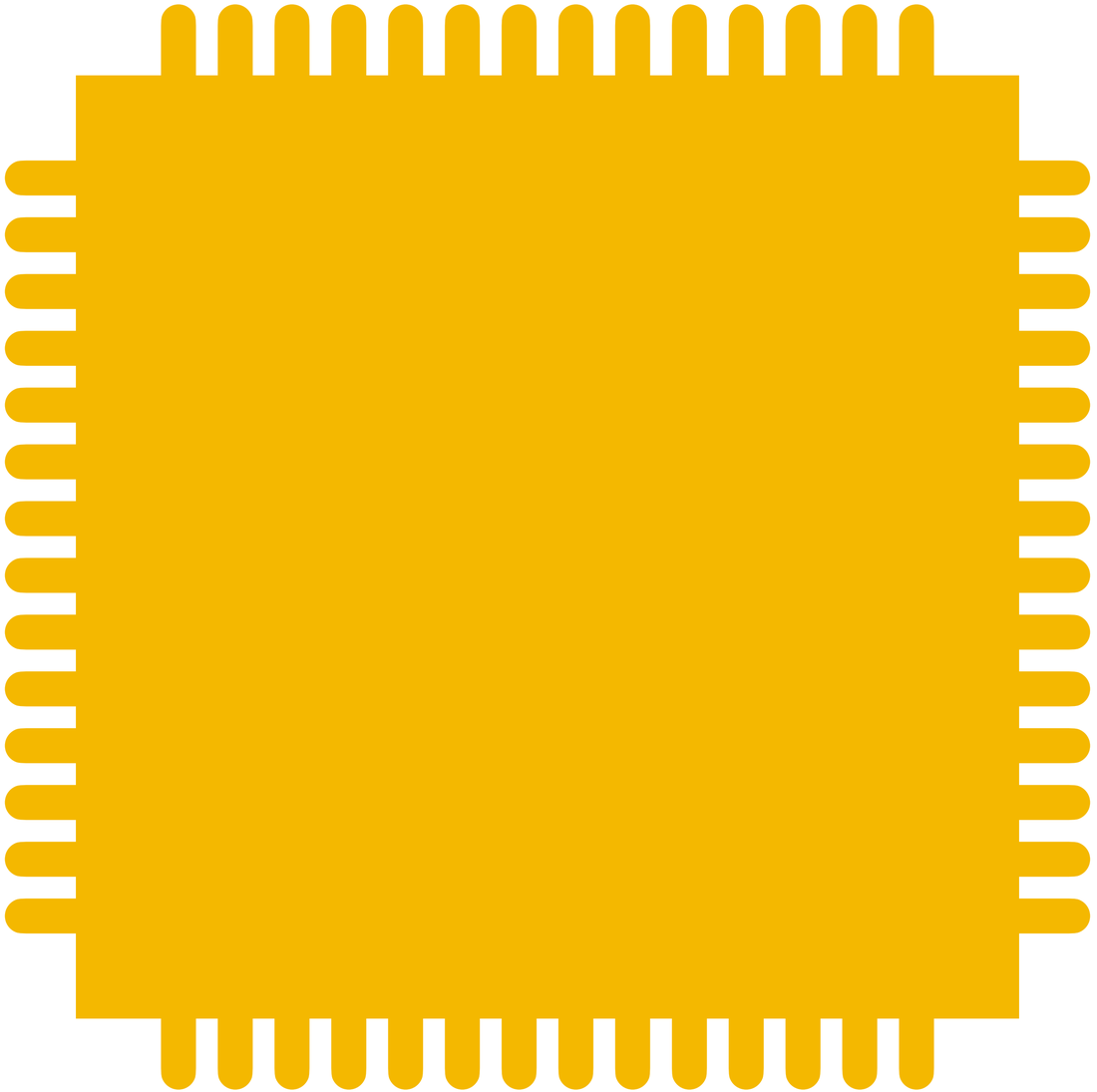












```
void a_inc(int *v, int c) {
    int i = 0;
    while (i < c) {
        v[i] = v[i] + 1;
        i++;
    }
}
```

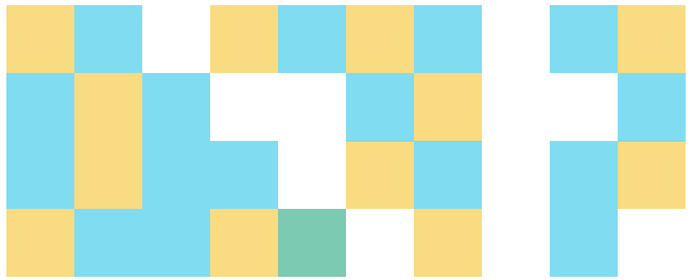
```
void a_inc(int *v, int c) {
  int i = 0;
  while (i < c) {
     v[i] = v[i] + 1;
     i++;
  }
}

void test(int *v1, int c1, int *v2, int c2) {
  a_inc(v1, c1);
  a_inc(v2, c2);
}
```

```
void a_inc(int *v, int c) {
  int i = 0;
  while (i < c) {
      v[i] = v[i] + 1;
      i++;
  }
}

void test(int *v1, int c1, int *v2, int c2) {
  a_inc(v1, c1);
  a_inc(v2, c2);
}
```

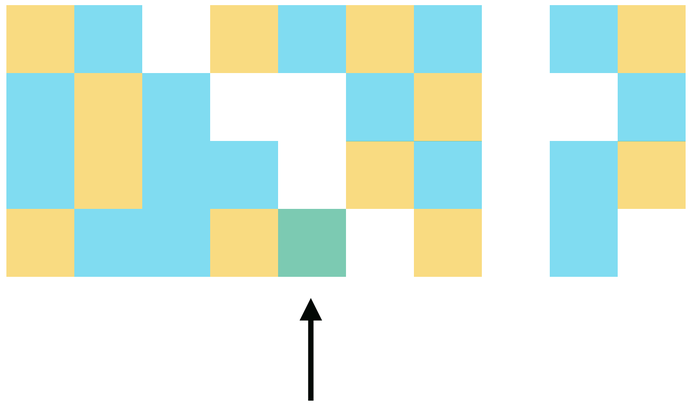
```
void test(int *v1, int c1, int *v2, int c2) {
  a_inc(v1, c1);
  a_inc(v2, c2);
}
```

```
void test(int *v1, int c1, int *v2, int c2) {
  a_inc(v1, c1);
  a_inc(v2, c2);
}
```

```
void test(int *v1, int c1, int *v2, int c2) {
  a_inc(v1, c1);
  a_inc(v2, c2);
}
```



```
void test(int *v1, int c1, int *v2, int c2) {
  a_inc(v1, c1);
  a_inc(v2, c2);
}
```

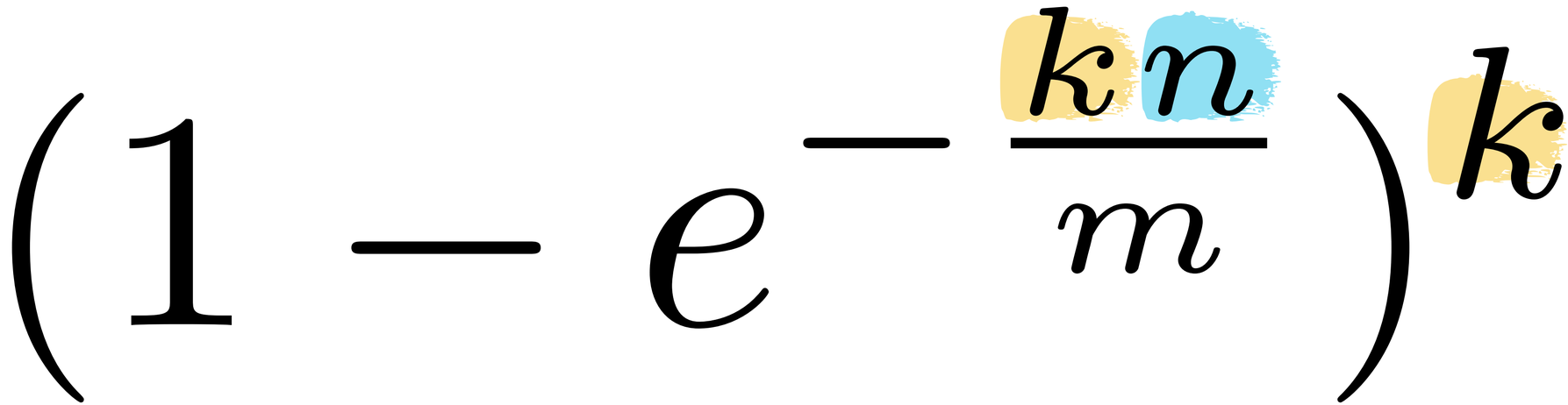


ANALYTICALLY DETERMINING FALSE POSITIVE RATES

$$\left(1 - e^{-\frac{kn}{m}}\right)^k$$

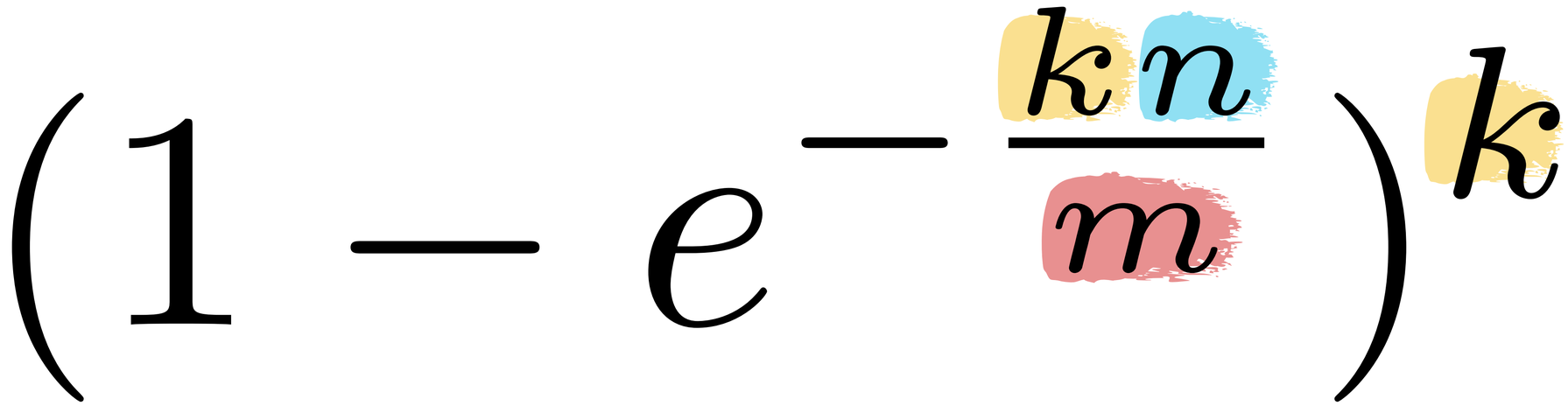
$$\left(1 - e^{-\frac{kn}{m}}\right)^{k}$$

number of hashes



number of hashes

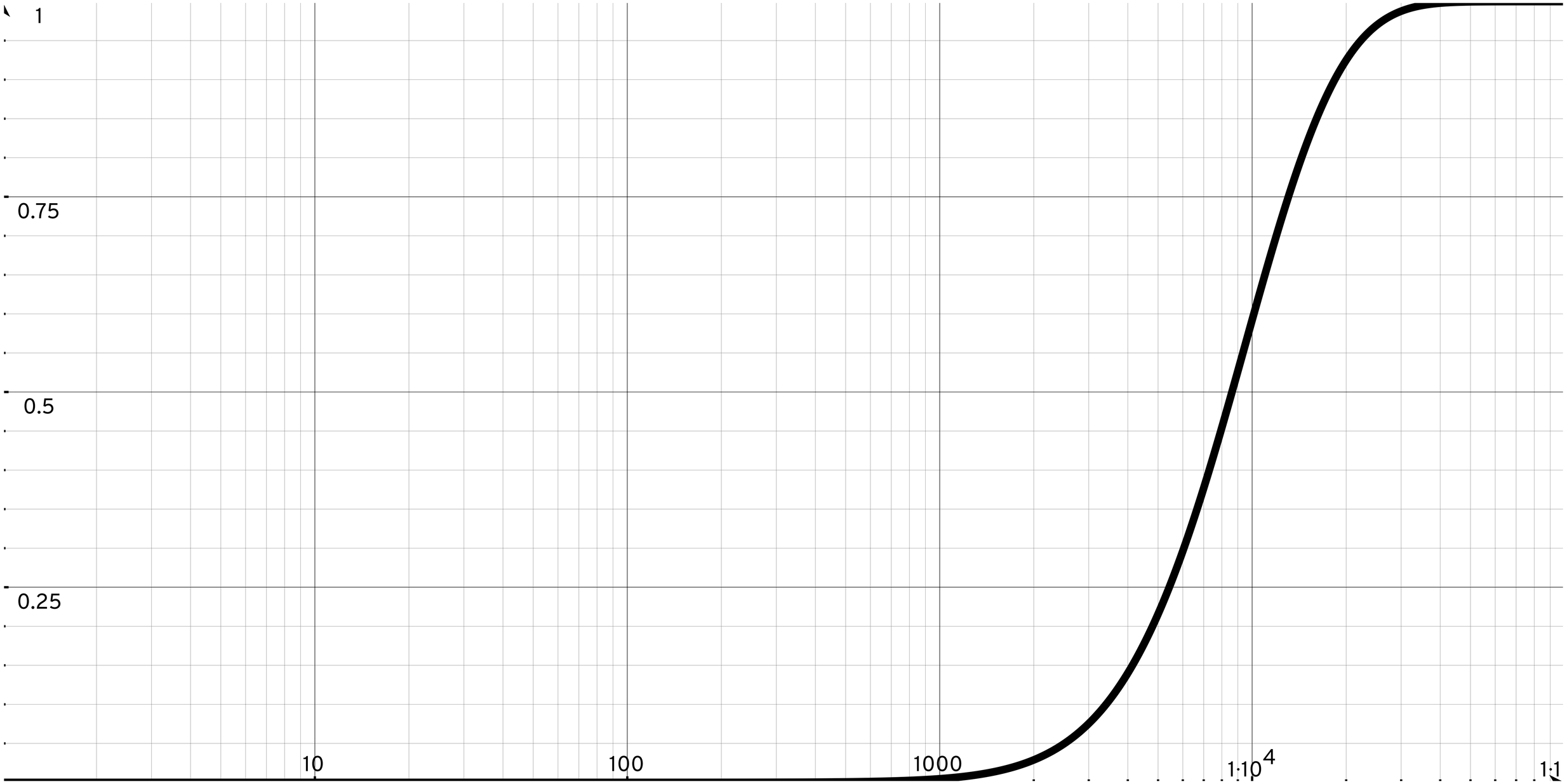
filter size in bits

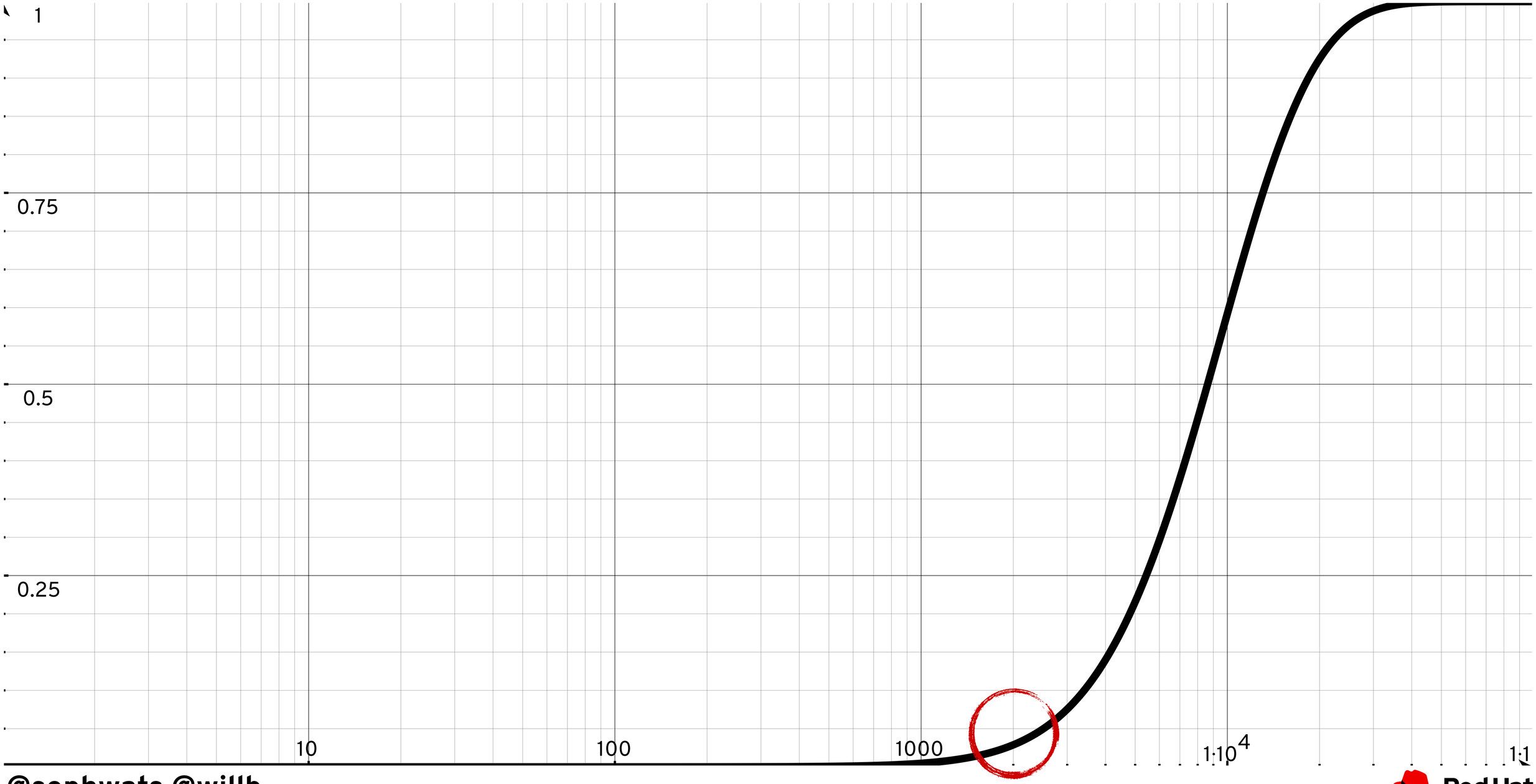


number of hashes

filter size in bits

actual set cardinality


1
0.75
0.5
0.25
10
100
1000
1·10^4
1·1

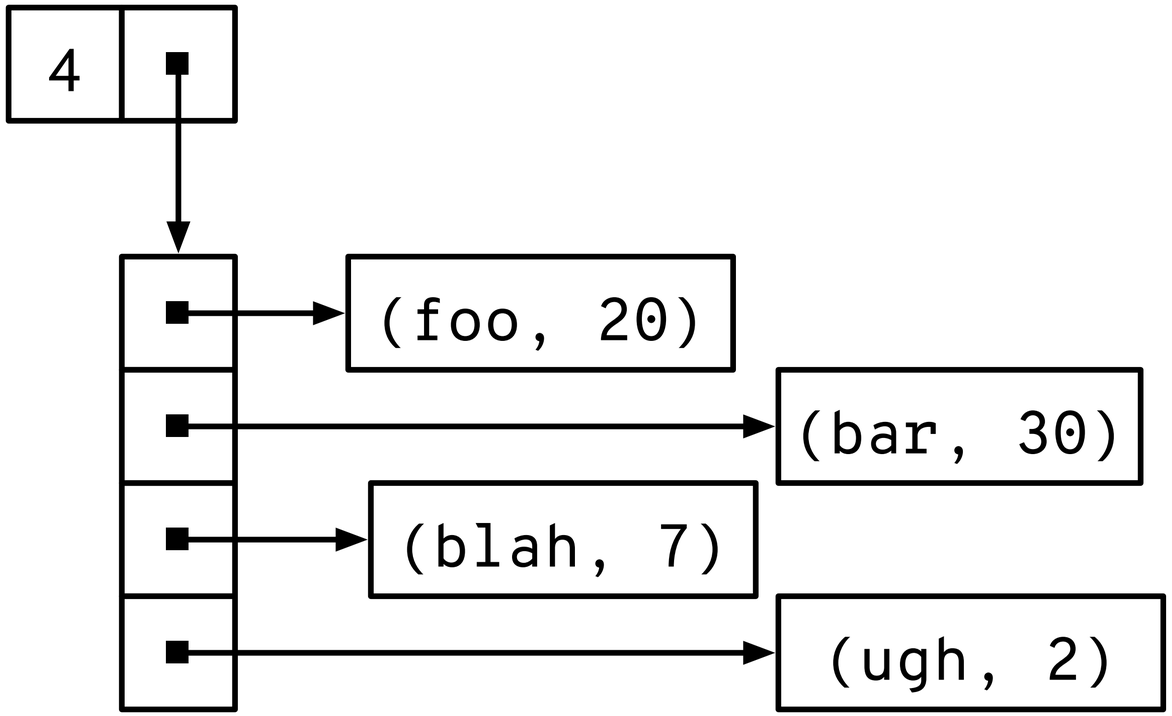

1
0.75
0.5
0.25
10
100
1000
1·10^4
1·1

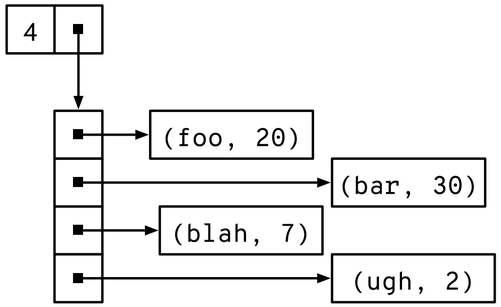
LET'S GO TO THE NOTEBOOK

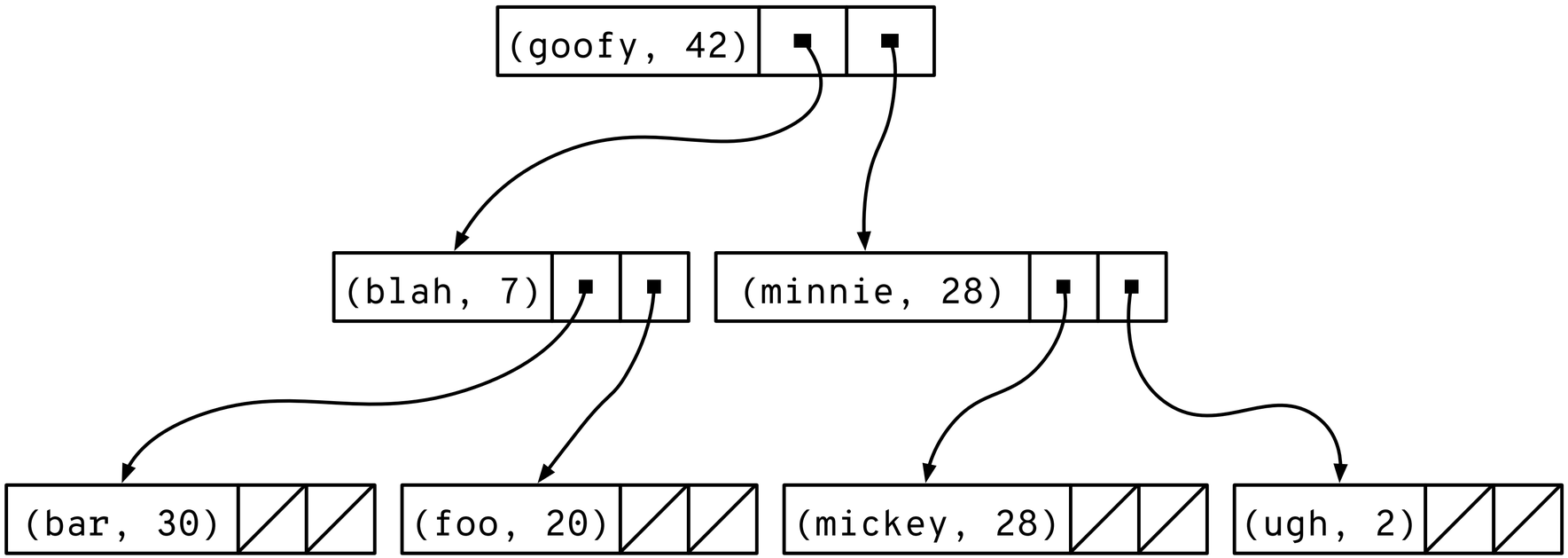
http://bit.ly/data-sketching-binder

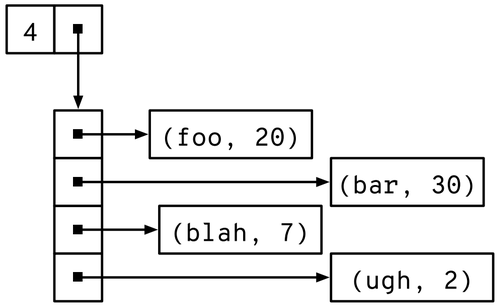
Event frequencies

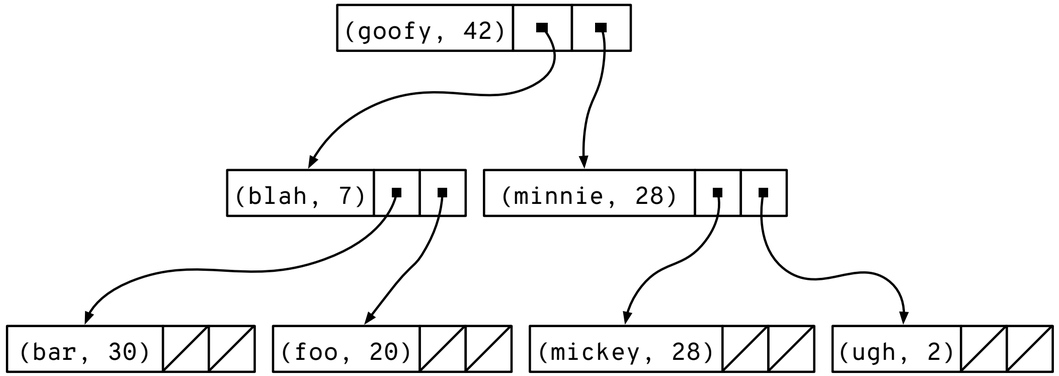
PRECISE STRUCTURES

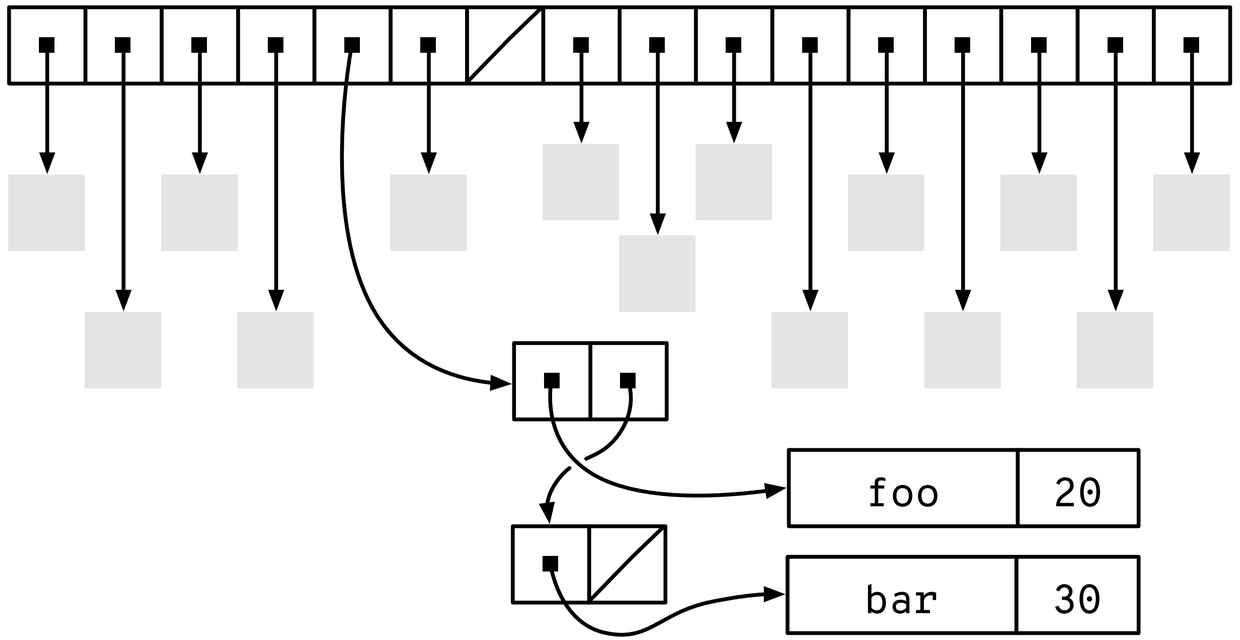

4
(foo, 20)
(bar, 30)
(blah, 7)
(ugh, 2)


4
(foo, 20)
(bar, 30)
(blah, 7)
(ugh, 2)


(goofy, 42)
(blah, 7)
(minnie, 28)
(bar, 30)
(foo, 20)
(mickey, 28)
(ugh, 2)


4
(foo, 20)
(bar, 30)
(blah, 7)
(ugh, 2)

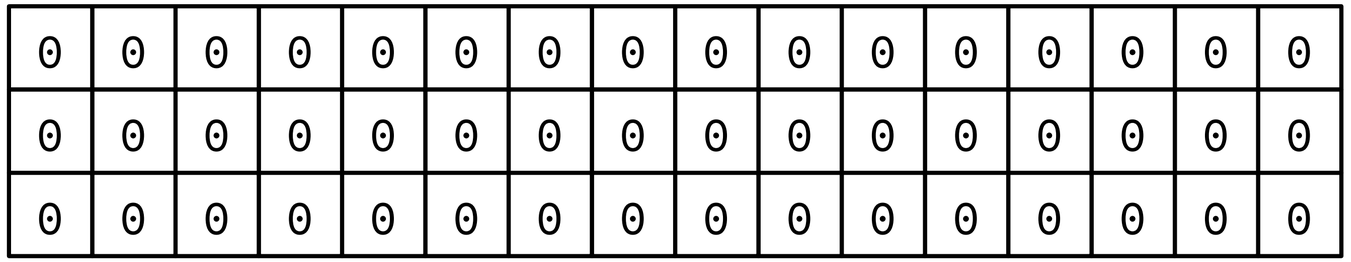

(goofy, 42)
(blah, 7)
(minnie, 28)
(bar, 30)
(foo, 20)
(mickey, 28)
(ugh, 2)

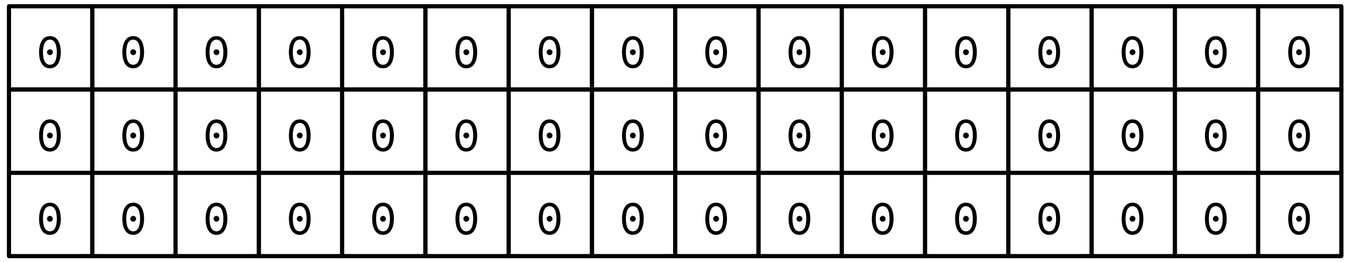

foo
20
bar
30

GENERALIZING THE BLOOM FILTER: COUNT-MIN SKETCH

@sophwats @willb

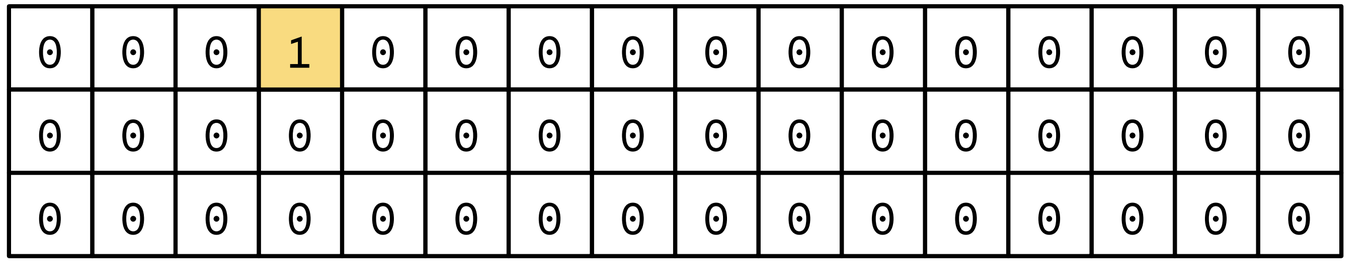
Red Hat


0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0




```
put("foo")
```

h1("foo")

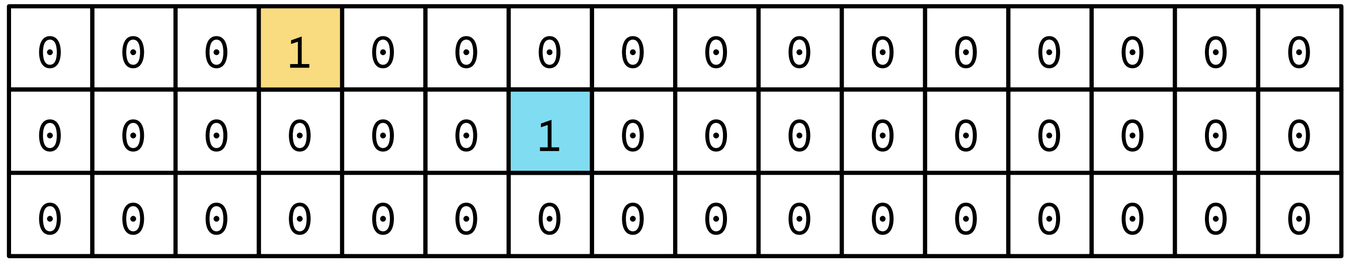


0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

put("foo")

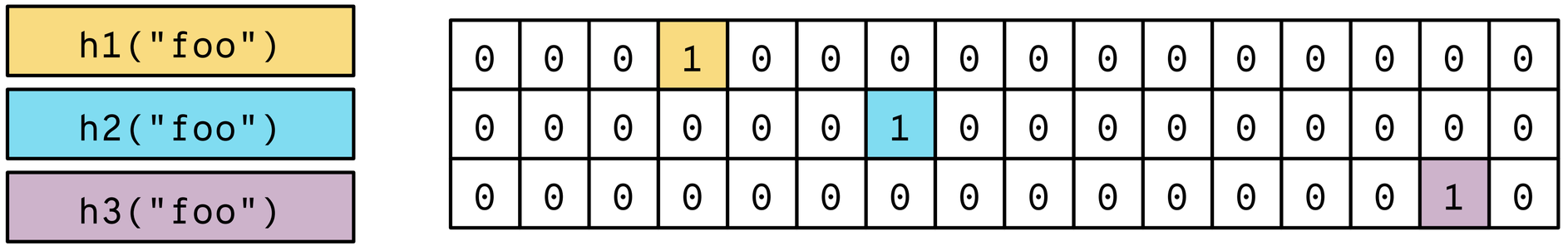
h1("foo")

h2("foo")



0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

put("foo")


h1("foo")
h2("foo")
h3("foo")
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0

put("foo")

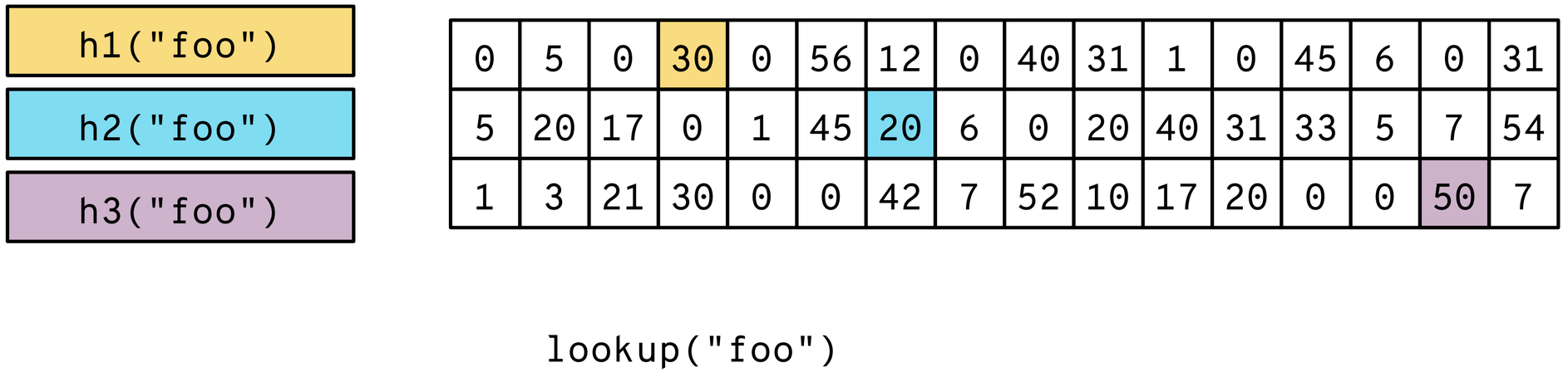
0	5	0	30	0	56	12	0	40	31	1	0	45	6	0	31
5	20	17	0	1	45	20	6	0	20	40	31	33	5	7	54
1	3	21	30	0	0	42	7	52	10	17	20	0	0	50	7

```
lookup("foo")
```

h1("foo")

0	5	0	30	0	56	12	0	40	31	1	0	45	6	0	31
5	20	17	0	1	45	20	6	0	20	40	31	33	5	7	54
1	3	21	30	0	0	42	7	52	10	17	20	0	0	50	7

lookup("foo")


h1("foo")
h2("foo")
h3("foo")
0 5 0 30 0 56 12 0 40 31 1 0 45 6 0 31
5 20 17 0 1 45 20 6 0 20 40 31 33 5 7 54
1 3 21 30 0 0 42 7 52 10 17 20 0 0 50 7
lookup("foo")

h1("foo")

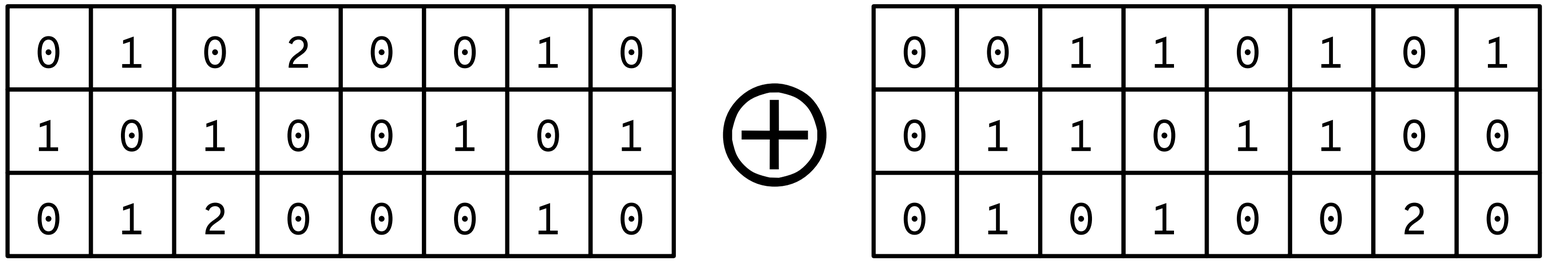
h2("foo")

h3("foo")

0	5	0	30	0	56	12	0	40	31	1	0	45	6	0	31
5	20	17	0	1	45	20	6	0	20	40	31	33	5	7	54
1	3	21	30	0	0	42	7	52	10	17	20	0	0	50	7

```
lookup("foo") = min(30, 20, 50)
```

```
class CMS(object):
    import numpy as np
    def __init__(self, w, hashes):
        self._width = w
        self._hashes = lambda v: [int(f(v)) for f in hashes[:]]
        self._buckets = np.zeros((int(w), len(hashes)), np.uint64)

    def insert(self, value):
        """ Inserts a value into this sketch """
        for (row, col) in enumerate(self._hashes(value)):
            self._buckets[col % self._width][row] += 1

    def lookup(self, value):
        """ Returns a biased estimate of number of times
        value has been inserted in this sketch"""
        return min([self._buckets[col % self._width][row] for
           (row, col) in enumerate(self._hashes(value))])
```


0 1 0 2 0 0 1 0
1 0 1 0 0 1 0 1
0 1 2 0 0 0 1 0
⊕
0 0 1 1 0 1 0 1
0 1 1 0 1 1 0 0
0 1 0 1 0 0 2 0

0	1	0	2	0	0	1	0
1	0	1	0	0	1	0	1
0	1	2	0	0	0	1	0

0	0	1	1	0	1	0	1
0	1	1	0	1	1	0	0
0	1	0	1	0	0	2	0

0	1	1	3	0	0	2	1
1	1	2	0	1	2	0	1
0	2	2	1	0	0	2	0

TOP-K ELEMENTS

0	5	0	30	0	56	12	0	40	31	1	0	45	6	0	31
5	20	17	0	1	45	20	6	0	20	40	31	33	5	7	54
1	3	21	30	0	0	42	7	52	10	17	20	0	0	50	7

```
put("foo")
```

CMS

0	5	0	30	0	56	12	0	40	31	1	0	45	6	0	31
5	20	17	0	1	45	20	6	0	20	40	31	33	5	7	54
1	3	21	30	0	0	42	7	52	10	17	20	0	0	50	7

```
put("foo")
```

Priority queue

CMS

0	5	0	30	0	56	12	0	40	31	1	0	45	6	0	31
5	20	17	0	1	45	20	6	0	20	40	31	33	5	7	54
1	3	21	30	0	0	42	7	52	10	17	20	0	0	50	7

```
put("foo")
```

Priority queue

0	5	0	30	0	56	12	0	40	31	1	0	45	6	0	31
5	20	17	0	1	45	20	6	0	20	40	31	33	5	7	54
1	3	21	30	0	0	42	7	52	10	17	20	0	0	50	7

```
put("ugh")
```

0	5	0	30	0	56	12	0	40	31	2	0	45	6	0	31
5	20	17	0	2	45	20	6	0	20	40	31	33	5	7	54
1	4	21	30	0	0	42	7	52	10	17	20	0	0	50	7

```
put("ugh")
```

0	5	0	30	0	56	12	0	40	31	2	0	45	7	0	31
5	20	17	0	2	45	20	6	0	20	41	31	33	5	7	54
1	4	21	30	0	0	42	7	52	10	17	20	0	0	50	8

```
put("blah")
```

(20, foo)	(2, ugh)			

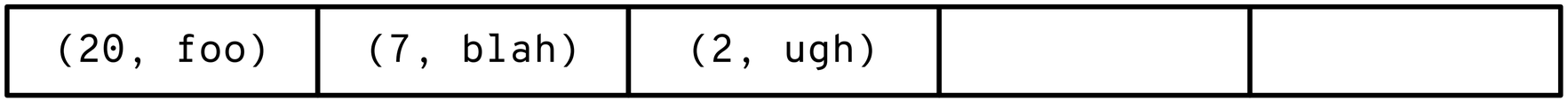
0	5	0	30	0	56	12	0	40	31	2	0	45	7	0	31
5	20	17	0	2	45	20	6	0	20	41	31	33	5	7	54
1	4	21	30	0	0	42	7	52	10	17	20	0	0	50	8

```
put("blah")
```

(20, foo)	(7, blah)	(2, ugh)		

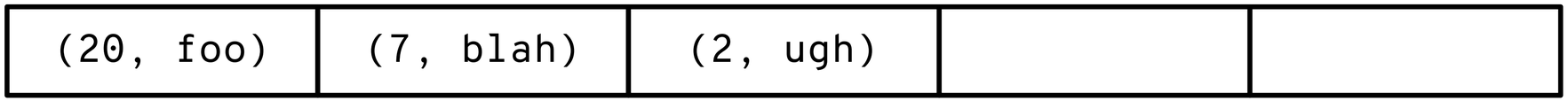
0	5	0	30	0	56	12	0	40	31	2	0	45	7	0	31
5	20	17	0	2	45	20	6	0	20	41	31	33	5	7	54
1	4	21	30	0	0	42	7	52	10	17	20	0	0	50	8

```
put("goofy")
```



0	5	0	30	0	57	12	0	40	31	2	0	45	7	0	31
5	20	17	0	2	45	20	6	0	20	42	31	33	5	7	54
1	4	21	30	0	0	42	7	52	10	17	20	0	0	51	8

```
put("goofy")
```



0	5	0	30	0	57	12	0	40	31	2	0	45	7	0	31
5	20	17	0	2	45	20	6	0	20	42	31	33	5	7	54
1	4	21	30	0	0	42	7	52	10	17	20	0	0	51	8

```
put("goofy")
```

0	5	0	30	0	57	12	0	40	31	2	0	45	7	0	31
5	20	17	0	2	45	20	6	0	20	42	31	33	5	7	54
1	4	21	30	0	0	42	7	52	10	17	20	0	0	51	8

```
put("bar")
```

0	5	0	31	0	57	12	0	40	31	2	0	45	7	0	31
5	20	17	0	2	46	20	6	0	20	42	31	33	5	7	54
1	4	21	30	0	0	43	7	52	10	17	20	0	0	51	8

```
put("bar")
```

(42, goofy)	(20, foo)	(7, blah)	(2, ugh)	

0	5	0	31	0	57	12	0	40	31	2	0	45	7	0	31
5	20	17	0	2	46	20	6	0	20	42	31	33	5	7	54
1	4	21	30	0	0	43	7	52	10	17	20	0	0	51	8

```
put("bar")
```

(42, goofy)	(31, bar)	(20, foo)	(7, blah)	(2, ugh)

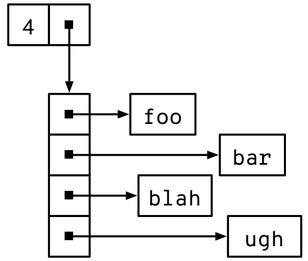
#bbuzz	18.4 million
#vbuzz	17.8 million
#hashing	17.5 million
#theBARNberlin	17.4 million
#AIML	17.2 million

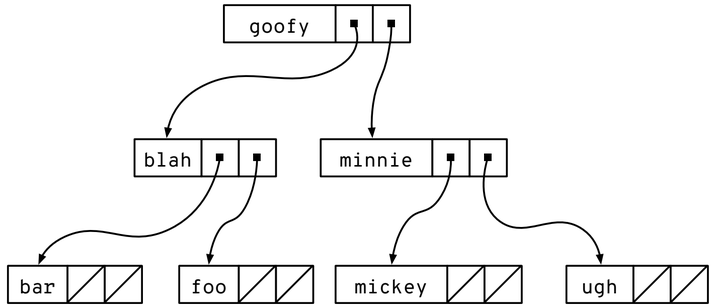
LET'S GO TO THE NOTEBOOK

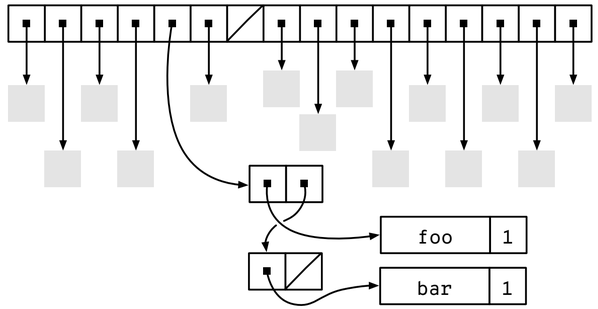
http://bit.ly/data-sketching-binder

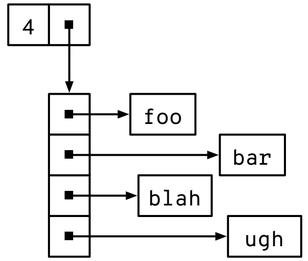
Counting distinct items

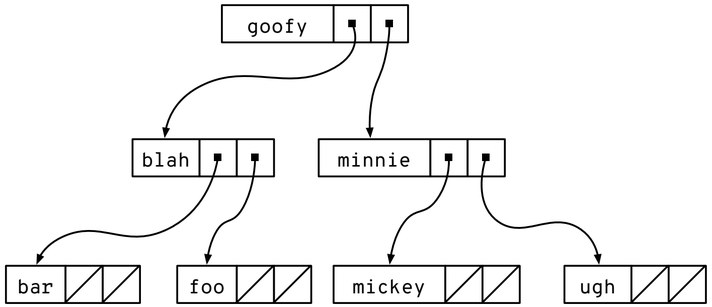
PRECISE APPROACHES

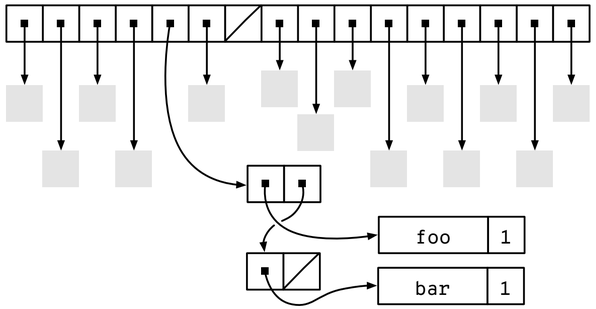

4
foo
bar
blah
ugh


goofy
blah
minnie
bar
foo
mickey
ugh


foo
1
bar
1







|s| = len(s)

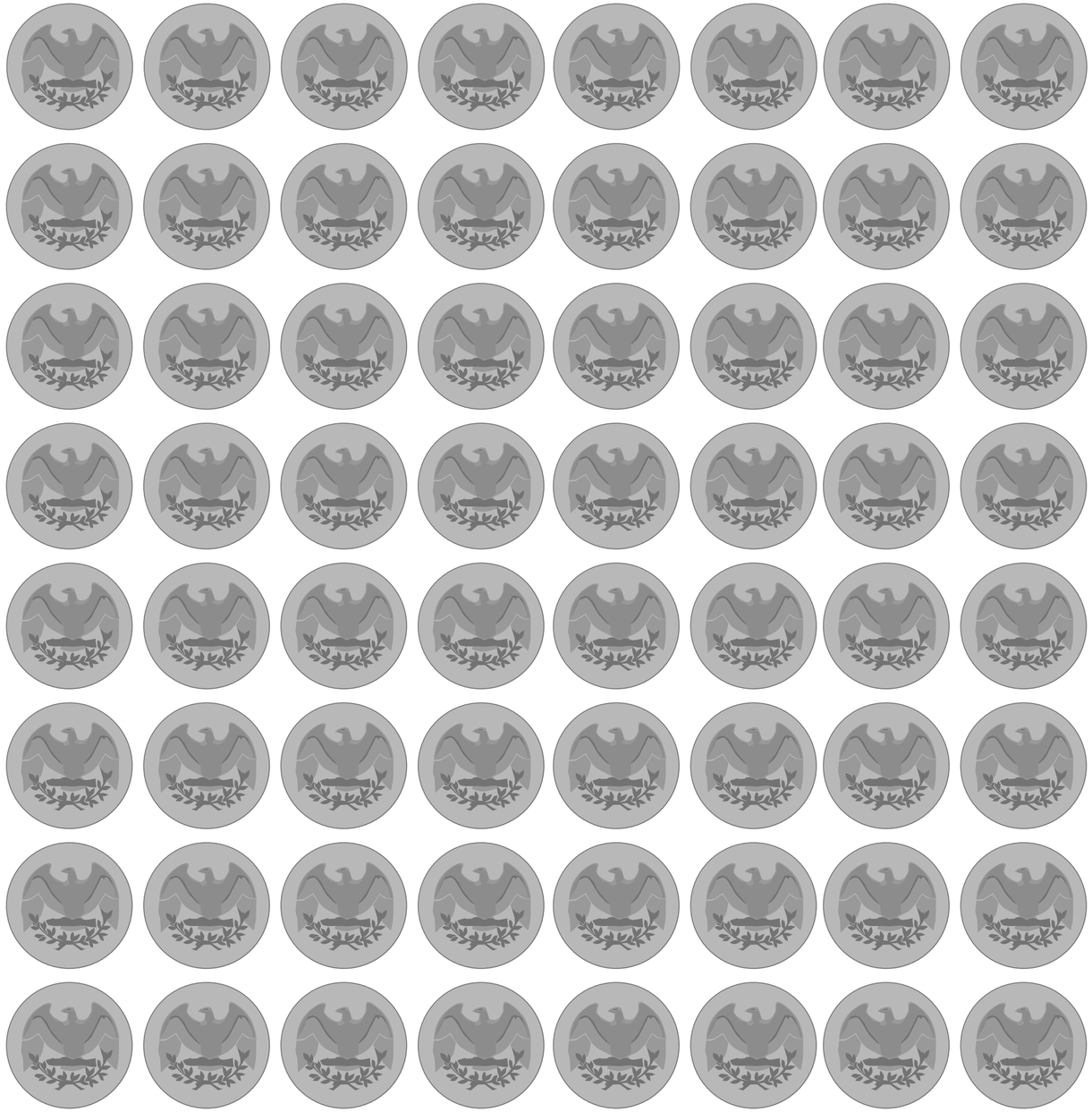
ESTIMATING CARDINALITY WITH A BLOOM FILTER

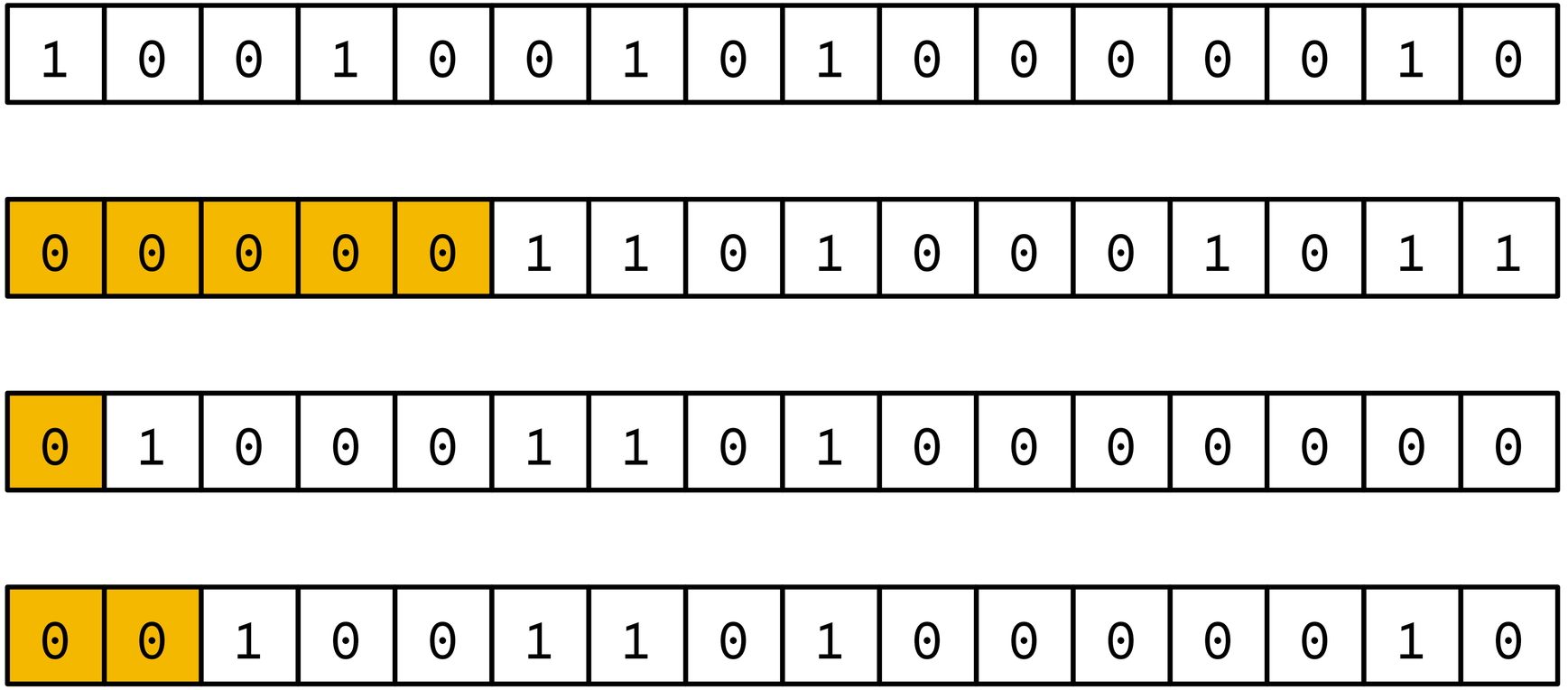
@sophwats @willb

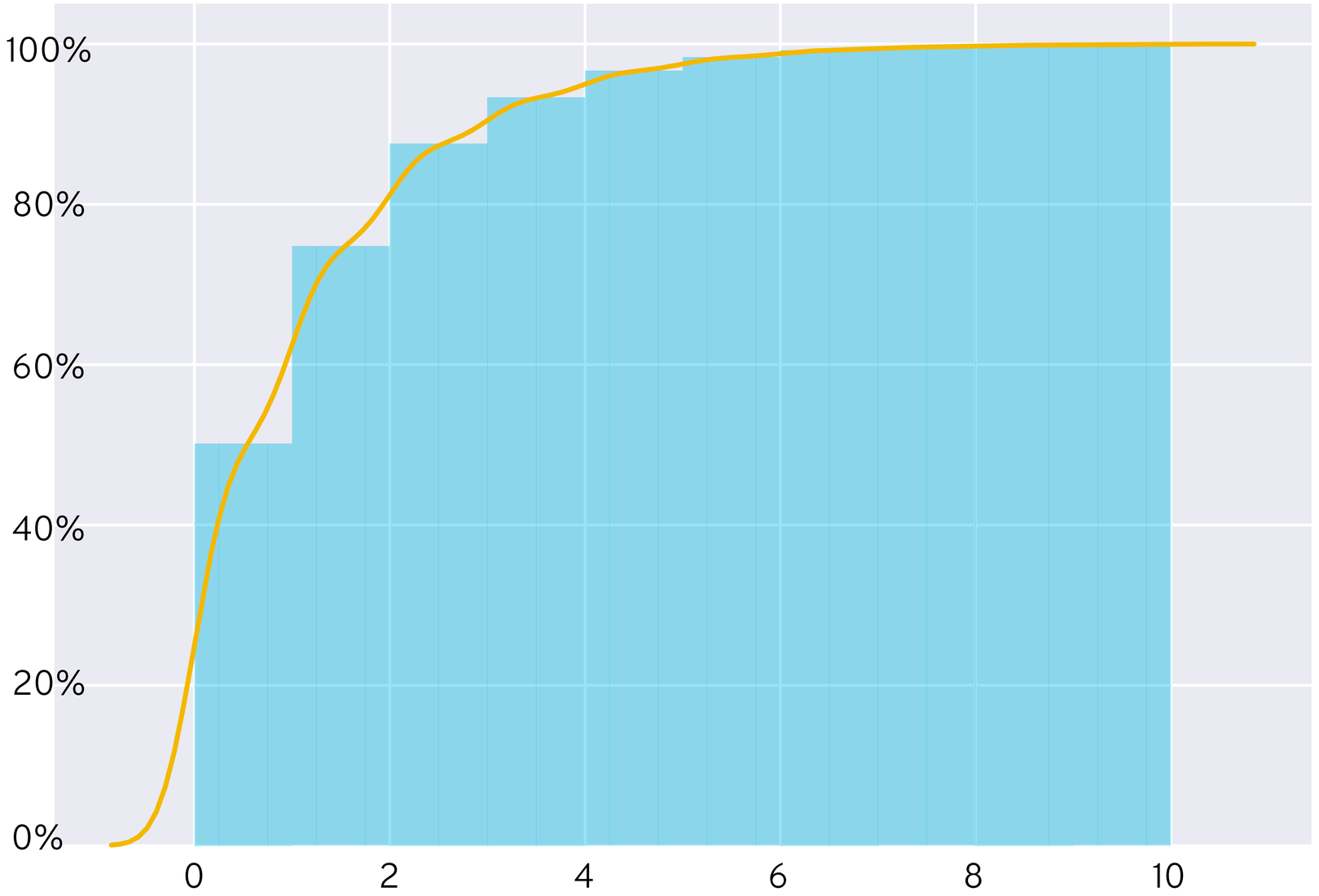
Red Hat

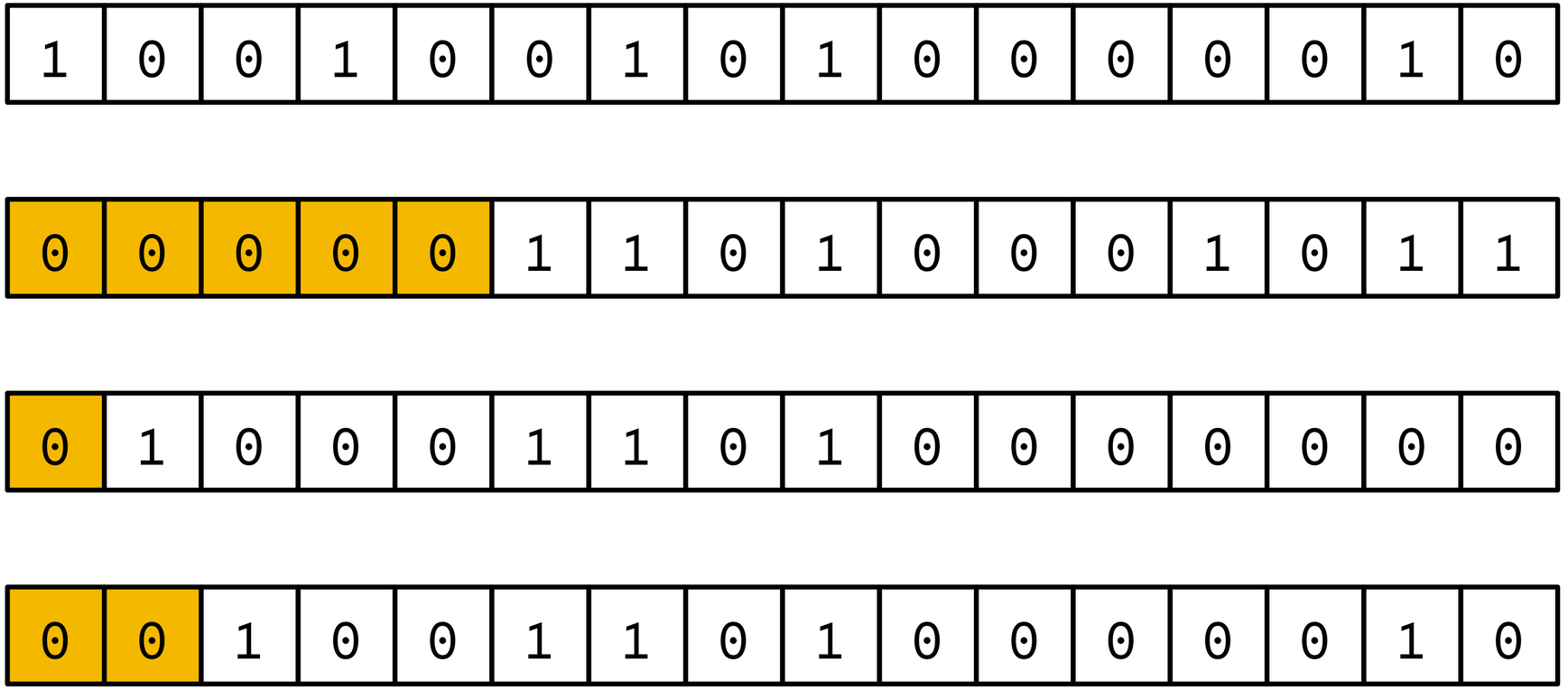
```
def approx_cardinality(self):
    """ Estimates the cardinality of the
        set modeled by this filter.
        Uses a technique from Swamidass
        and Baldi (2007). """
    from math import log
    m = self.size() * self.partitions()
    k = self.partitions()
    X = self.__buckets.count_set_bits()
    return -(m / k) * log(1 - (X / m))
```

HYPERLOGLOG




1 0 0 1 0 0 1 0 1 0 0 0 0 0 1 0
0 0 0 0 0 1 1 0 1 0 0 0 1 0 1 1
0 1 0 0 0 1 1 0 1 0 0 0 0 0 0 0
0 0 1 0 0 1 1 0 1 0 0 0 0 0 1 0


100%
80%
60%
40%
20%
0%
0
2
4
6
8
10

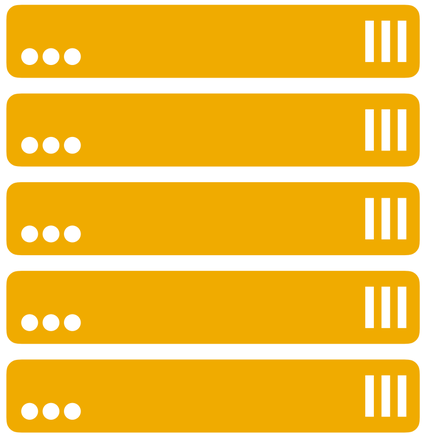

1 0 0 1 0 0 1 0 1 0 0 0 0 0 1 0
0 0 0 0 0 1 1 0 1 0 0 0 1 0 1 1
0 1 0 0 0 1 1 0 1 0 0 0 0 0 0 0
0 0 1 0 0 1 1 0 1 0 0 0 0 0 1 0

1	0	0	1	0	0	1	0	1	0	0	0	0	0	1	0

0	0	0	0	0	1	1	0	1	0	0	0	1	0	1	1

0	1	0	0	0	1	1	0	1	0	0	0	0	0	0	0

0	0	1	0	0	1	1	0	1	0	0	0	0	0	1	0

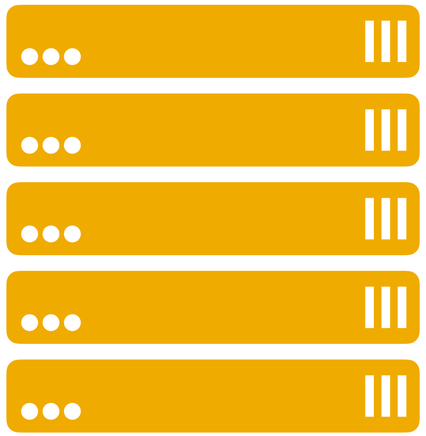


1	0	0	1	0	0	1	0	1	0	0	0	0	0	1	0

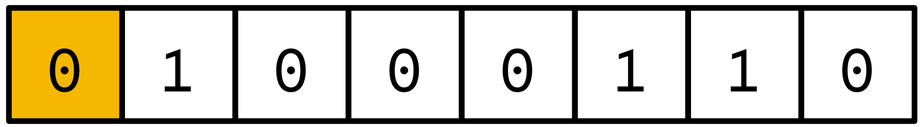
0	0	0	0	0	1	1	0	1	0	0	0	1	0	1	1

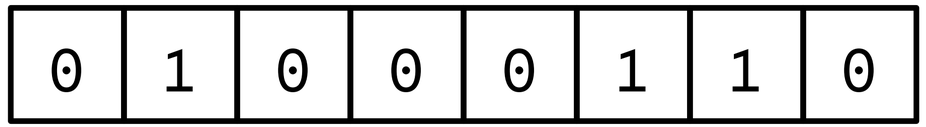
0	1	0	0	0	1	1	0	1	0	0	0	0	0	0	0

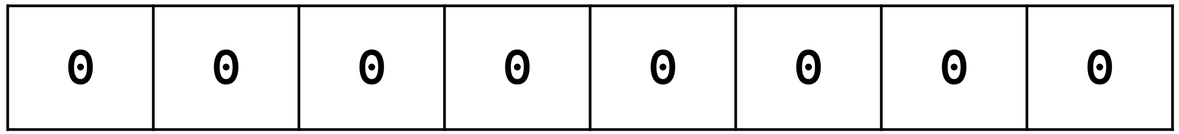
0	0	1	0	0	1	1	0	1	0	0	0	0	0	1	0



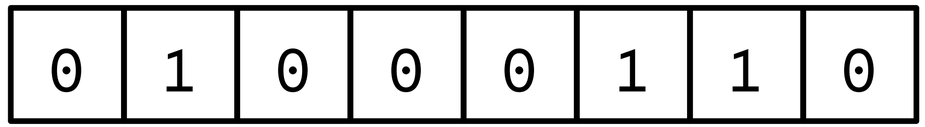
{ ... }

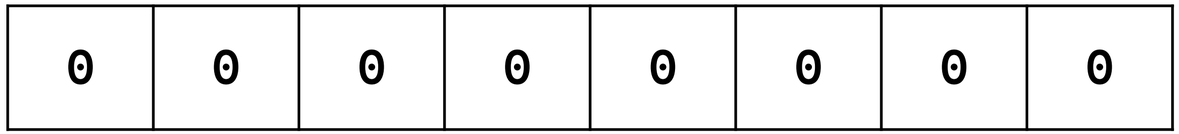
h({ ... }) = 

h({ ... }) = 

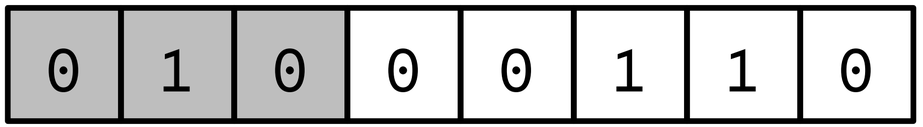


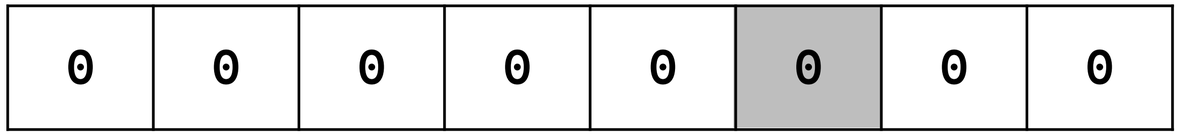
h({ ... }) =



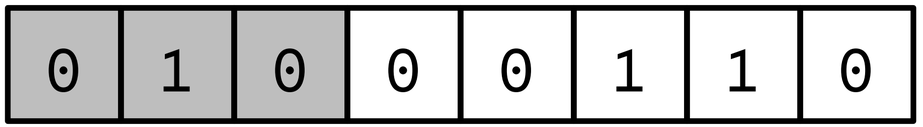


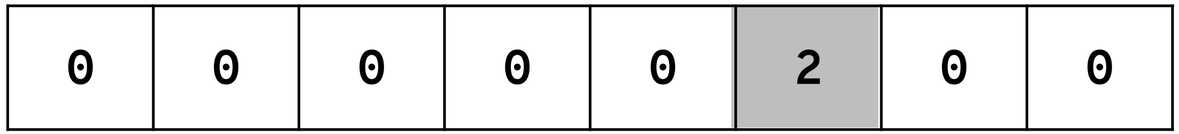
h({ ... }) =



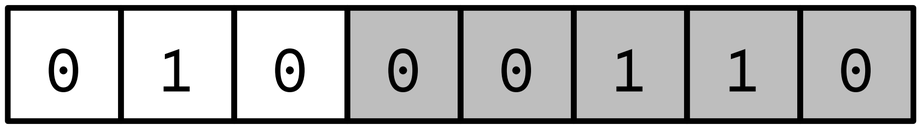


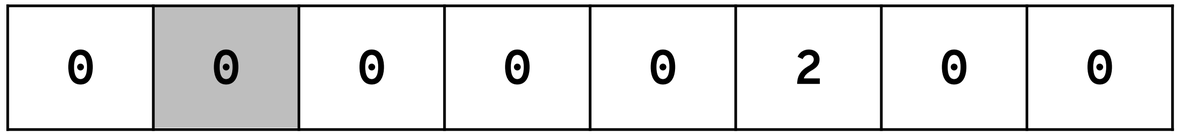
h({ ... }) =



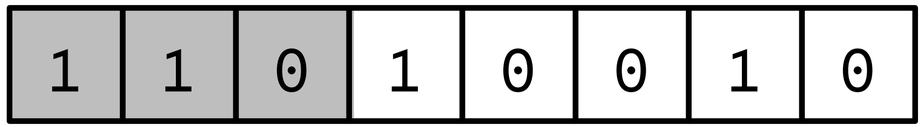


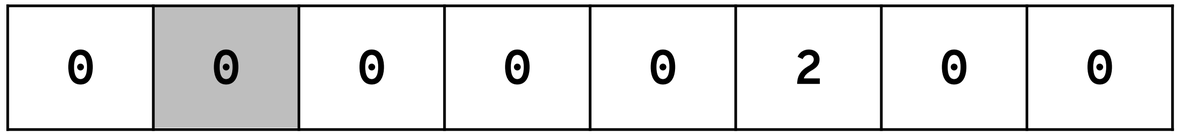
h({ ... }) =



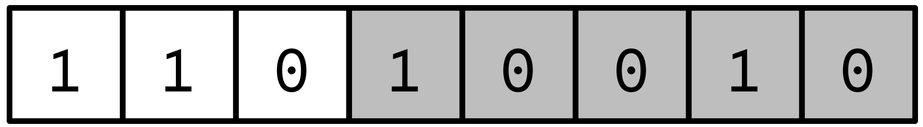


h({ ... }) =

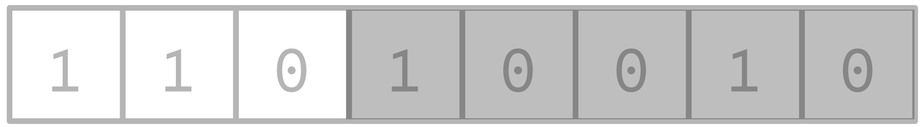




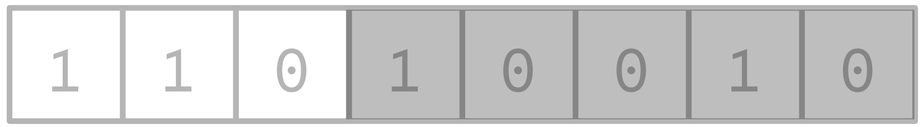
h({ ... }) =



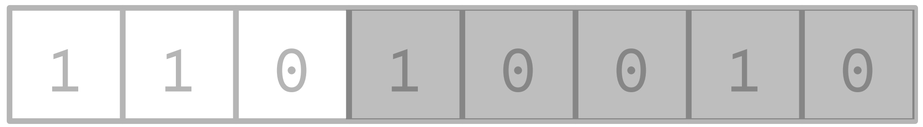
0	0	0	0	0	2	0	0

h({ ... }) = 

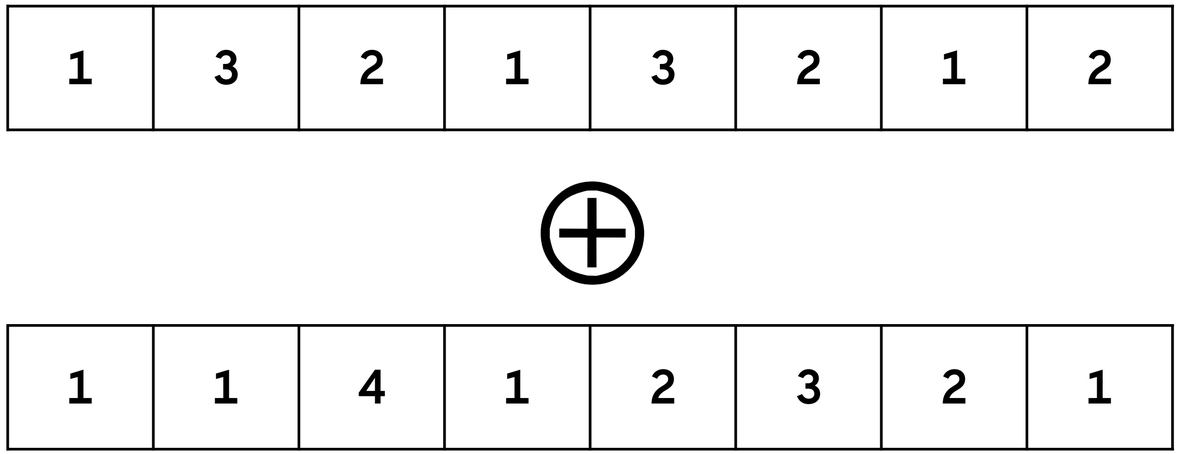
1	3	2	1	3	2	1	2

h({ ... }) = 

1	3	2	1	3	2	1	2

h({ ... }) = 

|set| ≈ 16


1 3 2 1 3 2 1 2
⊕
1 1 4 1 2 3 2 1

1	3	2	1	3	2	1	2

1	1	4	1	2	3	2	1

1	3	2	1	3	2	1	2

1	1	4	1	2	3	2	1

1	3	4	1	3	3	2	2

```
import numpy as np
from scipy.stats import hmean

class HLL(object):
    def __init__(self, p=4):
        self.p = min(max(p, 4), 12)
        self.alpha = get_alpha(self.p)
        self._registers = np.zeros(int(2 ** self.p), np.uint8)

    def add(self, v):
        h = h64(v)
        idx = h & (len(self._registers) - 1)
        h >>= self.p
        fsb = first_set_bit(h, 64 - self.p)
        self._registers[idx] = max(self._registers[idx], fsb)

    def approx_count(self):
        m = len(self._registers)
        if self._registers.count_nonzero() < len(self._registers):
            return m * math.log(float(m) / self._zeros)
        else:
            return self.alpha * m * hmean(np.power(2.0, self._registers))
```

IMPROVING PERFORMANCE

Heule, Nunkesser, and Hall. “HyperLogLog in Practice: Algorithmic Engineering of a State of the Art Cardinality Estimation Algorithm.” In *Proceedings of the EBDT 2013 Conference*. Genoa, Italy. © ACM.

Set similarity

It is a universally acknowledged truth that a single man in possession of a good fortune must want a wife.

However little known the thoughts of such a man may be on his first entering a neighborhood, this truth is so fixed in the minds of the surrounding families that they consider him to be the rightful property of one of their daughters.

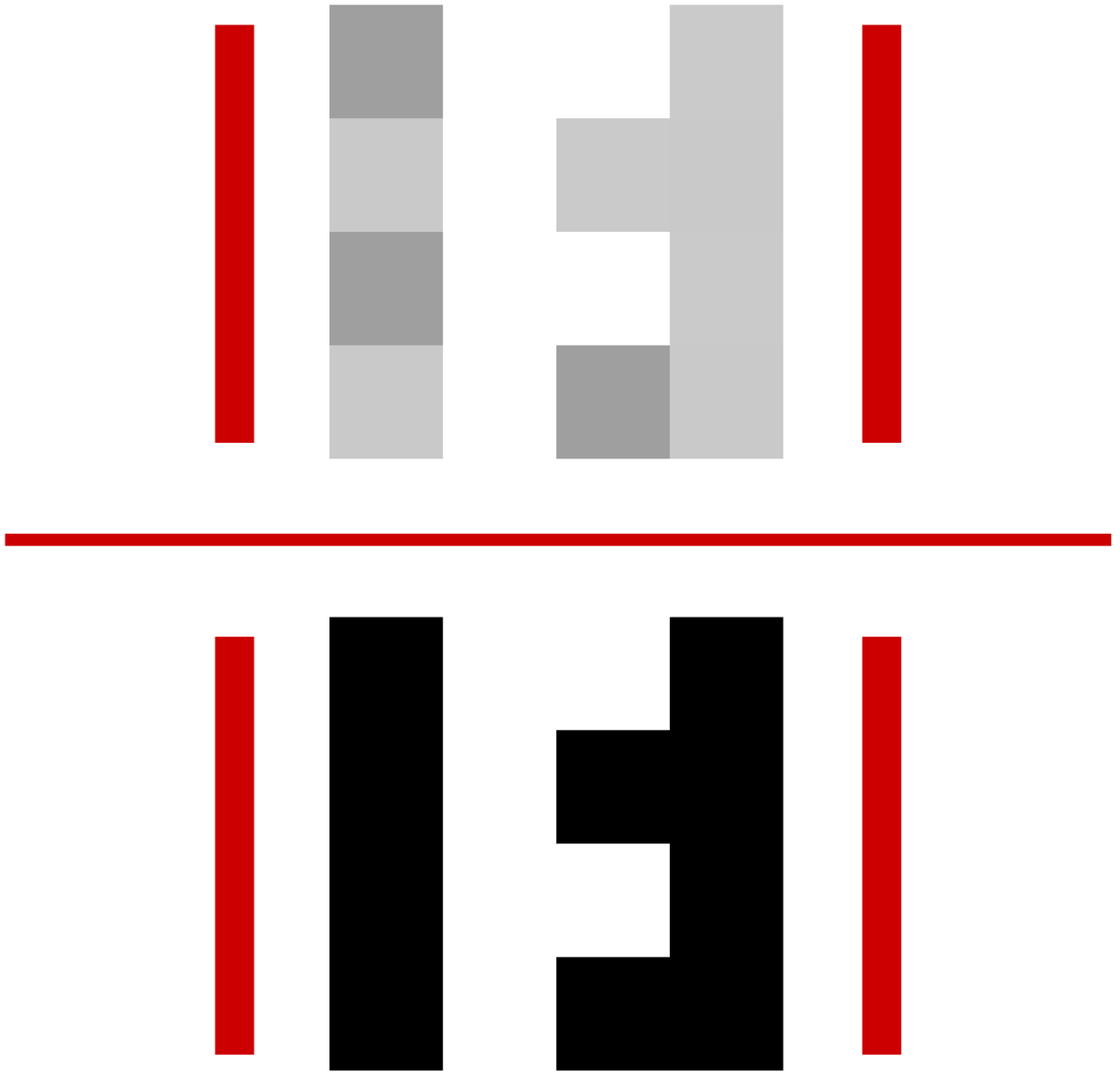
It is a truth universally acknowledged, that a single man in possession of a good fortune must be in want of a wife.

However little known the feelings or views of such a man may be on his first entering a neighbourhood, this truth is so well fixed in the minds of the surrounding families, that he is considered as the rightful property of some one or other of their daughters.

It is a truth universally acknowledged, that a single man in possession of a good fortune must be in want of a wife.

However little known the feelings or views of such a man may be on his first entering a neighbourhood, this truth is so well fixed in the minds of the surrounding families, that he is considered as the rightful property of some one or other of their daughters.

```
{'However', 'man', 'that', 'well', 'considered', 'daughters.',
'fixed', 'of', 'their', 'universally', 'possession', 'so',
'fortune', 'feelings', 'single', 'one', 'a', 'as', 'acknowledged,',
'entering', 'views', 'this', 'he', 'such', 'neighbourhood,', 'good',
'rightful', 'other', 'property', 'some', 'families,', 'on', 'minds',
'in', 'little', 'surrounding', 'want', 'first', 'known', 'or',
'may', 'is', 'the', 'his', 'be', 'wife.', 'must', 'truth', 'It'}
```



{However, It, a, acknowledged, as, be, consider, considered, daughters, entering, families, feelings, first, fixed, fortune, good, he, him, his, in, is, known, little, man, may, minds, must, neighborhood, neighbourhood, of, on, one, or, other, possession, property, rightful, single, so, some, such, surrounding, that, the, their, they, this, thoughts, to, truth, universally, views, want, well, wife}

$$|s_1 \cup s_2| = 55$$

{However, It, a, acknowledged, as, be, consider, considered, daughters, entering, families, feelings, first, fixed, fortune, good, he, him, his, in, is, known, little, man, may, minds, must, neighborhood, neighbourhood, of, on, one, or, other, possession, property, rightful, single, so, some, such, surrounding, that, the, their, they, this, thoughts, to, truth, universally, views, want, well, wife}

$$|s_1 \cap s_2| = 39 \qquad |s_1 \cup s_2| = 55$$

{However, It, a, acknowledged, as, be, consider, considered, daughters, entering, families, feelings, first, fixed, fortune, good, he, him, his, in, is, known, little, man, may, minds, must, neighborhood, neighbourhood, of, on, one, or, other, possession, property, rightful, single, so, some, such, surrounding, that, the, their, they, this, thoughts, to, truth, universally, views, want, well, wife}

$$\frac{|s_1 \cap s_2| = 39}{|s_1 \cup s_2| = 55}$$

{However, It, a, acknowledged, as, be, consider, considered, daughters, entering, families, feelings, first, fixed, fortune, good, he, him, his, in, is, known, little, man, may, minds, must, neighborhood, neighbourhood, of, on, one, or, other, possession, property, rightful, single, so, some, such, surrounding, that, the, their, they, this, thoughts, to, truth, universally, views, want, well, wife}

{However, It, a, acknowledged, as, be, consider, considered, daughters, entering, families, feelings, first, fixed, fortune, good, he, him, his, in, is, known, little, man, may, minds, must, neighborhood, neighbourhood, of, on, one, or, other, possession, property, rightful, single, so, some, such, surrounding, that, the, their, they, this, thoughts, to, truth, universally, views, want, well, wife}

$$\frac{|s_1 \cap s_2| = 39}{|s_1 \cup s_2| = 55} \approx 71\%$$

{However, It, a, acknowledged, as, be, consider, considered, daughters, entering, families, feelings, first, fixed, fortune, good, he, him, his, in, is, known, little, man, may, minds, must, neighborhood, neighbourhood, of, on, one, or, other, possession, property, rightful, single, so, some, such, surrounding, that, the, their, they, this, thoughts, to, truth, universally, views, want, well, wife}

{However, It, a, acknowledged, as, be, consider, considered, daughters, entering, families, feelings, first, fixed, fortune, good, he, him, his, in, is, known, little, man, may, minds, must, neighborhood, neighbourhood, of, on, one, or, other, possession, property, rightful, single, so, some, such, surrounding, that, the, their, they, this, thoughts, to, truth, universally, views, want, well, wife}

$$|s_1 \cap s_2| = 39$$

{However, It, a, acknowledged, as, be, consider, considered, daughters, entering, families, feelings, first, fixed, fortune, good, he, him, his, in, is, known, little, man, may, minds, must, neighborhood, neighbourhood, of, on, one, or, other, possession, property, rightful, single, so, some, such, surrounding, that, the, their, they, this, thoughts, to, truth, universally, views, want, well, wife}

$$|s_1 \cup s_2| = 55$$

```
the quick brown fox jumps over the lazy dog
```

```
{'the ', 'he q', 'e qu', ' qui', 'quic',
'uick', 'ick ', 'ck b', 'k br', ' bro',
'brow', 'rown', 'own ', 'wn f', 'n fo',
' fox', 'fox ', 'ox j', 'x ju', ' jum',
'jump', 'umps', 'mps ', 'ps o', 's ov',
' ove', 'over', 'ver ', 'er t', 'r th',
' the’, 'he l', 'e la', ' laz', 'lazy',
'azy ', 'zy d', 'y do', ' dog'}
```

```
{'in p', 'f su', ' one', 'ty o', 'ackn', 'sion', 'thou', ' lit', 'a ma', 'so f',
'to b', 'own ', 'ds o', 'o be', 'wled', 'at a', 'une ', 'surr', 'ente', 'ight',
'edge', ' the', 'heir', 'sess', ' is ', 'ay b', 'ood,', 'his ', 'neig', 'they',
' pro', ' wan', 'ts o', 'nter', 'nown', ' a n', 'ssio', 'er h', 'ne m', 'g fa',
'y of', 'h th', 'an m', 'rrou', 'st e', 'ver ', ' uni', ' nei', 'augh', 'n po',
'g a ', 'nds ', ' fix', 'hood', 'him ', ' fam', ' a g', 'o fi', 'cons', 'trut',
'nsid', 'of a', 'd tr', ' pos', 'nowl', 'weve', 'a un', 'fe. ', ' a s', ' goo',
'gle ', 'he t', 'undi', 'on h', 'ledg', 'man ', 'lly ', 'ortu', 'ndin', 'e kn',
'righ', 'ghbo', 'ring', 'may ', 'e on', 'a go', ' so ', 'ged ', 'houg', 'is f',
'vers', ' mus', 'im t', 'erty', 'in t', 'd, t', 'a ne', 'ruth', 'on o', 'd fo',
'fort', ' fir', 'at t', 'n ma', 'is s', 'tune', 'der ', 'hey ', 'f a ', ' suc',
'tle ', 'ngle', 'such', 'ally', 'of s', 'ing ', 'hat ', ' ent', 'ant ', 'y co',
'wife', 'e ri', 'ul p', 'oper', ' tru', ' his', 'uth ', 'It i', 'n in', 'e ma',
'e mu', ' a w', ' on ', 'essi', 'amil', 'ng f', ' con', 's a ', 'h is', 'ir d',
'he s', ' sin', 's of', 's tr', 'ust ', ', th', 'ey c', 'od f', 'st w', 'le k',
'rhoo', 'of t', 'good', 'ed t', 'd in', 'urro', ' him', 'a si', 'ood ', 'osse',
'r li', 'of o', 'one ', ' tha', 'ighb', 'is a', 'fami', 'litt', ' dau', 'th i',
'od, ', 'th t', 'htfu', 'hbor', 't is', 'eigh', 'ghts', 'sing', 'this', ' min',
'l pr', 'ers.', 'be o', 'wn t', 'nive', 'ng a', 'y be', 'erin', 'want', 's fi',
'teri', 'mili', 't en', 'ly a', ' rig', 'rsal', 'thei', 'sses', 'r hi', 'Howe',
'le m', 'rtun', ' How', ' man', 'ixed', 'eir ', 'r da', 'ters', 'an i', 'e th',
'ion ', 'rst ', 'dged', 'ed i', 'orho', 'e su', 'roun', 'iver', ' of ', 'm to',
'rope', 'hter', 's so', 'he m', 'e of', 'know', 'ersa', ' for', 'lies', 'tful',
't wa', 'univ', 'e. H', 'n th', ' ack', 'ingl', 'ife.', 'ough', 'ch a', 'side',
'firs', 'y ac', 'that', ' a m', 'a wi', 'uch ', 'ding', 'ilie', 'e mi', 's th',
'n hi', 'ckno', 'ittl', 'ghtf', 'mind', 'es t', 'must', 'ies ', 'sall', ' wif',
'xed ', 't a ', 'irst', 'ttle', ' thi', 'owle', 'poss', 'ught', 'fixe', 'prop',
'ound', ' kno', 'ful ', 'be t', 'he r', 'ever', 'the ', 'hts ', 'f on', 'n of',
'ne o', 'nt a', 't th', ' tho', 'ider', 'h a ', 'f th', '. Ho', 'is t', ' sur',
'inds', 'er l', 'ghte', ' in ', 'pert', 'daug', 'onsi', ' to ', ' may', 'rty ',
'borh', 'owev', ' be ', ' a u'}
```

```
{'sion', 'me o', 'ood,', 'nown', 'ssio', 'g fa', 'an m', 'n wa', ' fix', 'nsid',
'trut', 'her ', 'gle ', 'e on', 'ws o', 'in t', 'd, t', 'ruth', 'well', 'ourh',
'f a ', 'er o', 'tle ', 'ngle', ' ent', ' tru', ' on ', 'amil', 't be', 'he s',
'good', 'gs o', 'osse', 's or', 'sing', 'l pr', 'wn t', 'ng a', 'erin', 'ixed',
' wel', 'eir ', 'dged', 'ed i', 'hter', 'univ', 'e. H', 'ings', 'side', 'y ac',
'that', 'ding', 'e mi', 'must', ' thi', 's, t', 'othe', 'ound', 'ne o', 'f th',
'he i', 'pert', 'view', 'in p', 'f su', 'hbou', 'ty o', 'ackn', 'wled', 'at a',
'surr', ' is ', 'rrou', ' or ', ' a g', 'so w', 'r ot', ' a s', ' goo', 'ring',
'may ', 'ews ', 'ell ', 'a go', 'is f', 'vers', 'erty', 'l fi', 'fort', 'be i',
'tune', 'ed a', 't of', 'such', 'ally', 'e ri', 'wife', 'oper', ' his', 'It i',
' a w', 'essi', 'ng f', ' sin', 'a tr', 'd in', 'iews', ' dau', 'htfu', 'this',
'e or', 'some', 'thei', 'rtun', ' How', 'nt o', ' vie', 'an i', 'ion ', ' som',
' of ', 'ersa', 'tful', 'ife.', 'ch a', 'ilie', 'n hi', 'ittl', ' wif', 'fixe',
'ught', 'poss', 'prop', 'll f', 'ful ', 'he r', 'ever', 'n of', 'e in', 'h a ',
'is t', ' sur', 'ther', ' in ', ' one', 'a ma', 'bour', 'own ', 'ds o', 'une ',
'ight', 'heir', 'ered', 'as t', 's co', ' pro', ' wan', 'nter', ' a n', ' uni',
'n po', 'undi', 'ledg', 'lly ', 'ortu', 'righ', 'ghbo', ' so ', 'a ne', 'on o',
'd fo', ' fir', 'n ma', 'or v', 'hat ', 'ant ', 'ies,', 'e ma', 'e mu', 'ir d',
'st b', 's of', 's tr', 'he f', 'od f', 'es, ', 'le k', 'rhoo', 'a si', 'ood ',
'r li', 'f so', 'one ', ' tha', 'litt', 'th i', 'o we', 'od, ', 'ngs ', 't is',
'nive', 'teri', ' rig', 'ly a', 'sses', 'or o', 'ters', ' man', 'roun', 'rst ',
'at h', 'e su', 'he m', 'know', ' for', 'n th', ' ack', 'firs', 'uch ', 's th',
'ghtf', 'xed ', 'red ', 't a ', 'ttle', 'owle', 'inds', 'er l', 'ghte', 'daug',
'd as', 'owev', 'th u', ' lit', 'e is', 'ed, ', 'ente', 'edge', ' the', 'sess',
'ay b', 'his ', 'neig', 'ne m', 'ling', 'y of', 't he', 'st e', 'ver ', ' nei',
'augh', 'nds ', 'g a ', 'is c', 'hood', 'rty ', ' fam', 'cons', 'of a', ' pos',
'weve', 'nowl', 'in w', 'fe. ', 'ndin', 'on h', 'man ', 'elin', 'e kn', ' he ',
' mus', 'is s', ' suc', 'of s', 'ing ', 'r of', 'eeli', 'ul p', ' as ', 'uth ',
' con', 'n in', 'h un', 's a ', 'h is', ' a t', ', th', 'ust ', 'of t', 'urro',
'ighb', 'is a', 'fami', 'ome ', ' min', 'eigh', 'ers.', 'e fe', 'be o', 'y be',
'want', 's fi', 'mili', 't en', 'rsal', 'ged,', 'Howe', 'le m', 'r da', 'urho',
'rope', 'r vi', 'dere', 'iver', 's so', 'lies', 'ingl', 'feel', ' a m', 'a wi',
'ckno', 'mind', 'sall', 'irst', ' oth', ' fee', ' kno', 'the ', 'ider', '. Ho',
'onsi', ' may', ' be '}
```

the quick brown fox

the quick brown fox

```
{'the ', 'he q', 'e qu', ' qui', 'quic', 'uick', 'ick ', 'ck b',
 'k br', ' bro', 'brow', 'rown', 'own ', 'wn f', 'n fo', ' fox'}
```

the quick brown fox

```
{'the ', 'he q', 'e qu', ' qui', 'quic', 'uick', 'ick ', 'ck b',
 'k br', ' bro', 'brow', 'rown', 'own ', 'wn f', 'n fo', ' fox'}
```

```
{' fox', 'rown', 'k br', ...}
{' qui', 'uick', 'own ', ...}
{'brow', ' fox', 'n fo', ...}
{'ck b', ' bro', 'uick', ...}
{'e qu', 'rown', 'he q', ...}
```

```
{'he q', ' qui', 'ck b', ...}
{'n fo', ' qui', ' fox', ...}
{'rown', 'uick', ' bro', ...}
{'the ', 'brow', 'e qu', ...}
{'uick', 'own ', ' bro', ...}
```

the quick brown fox

```
{'the ', 'he q', 'e qu', ' qui', 'quic', 'uick', 'ick ', 'ck b',
 'k br', ' bro', 'brow', 'rown', 'own ', 'wn f', 'n fo', ' fox'}
```

```
{' fox', 'rown', 'k br', ...}
{' qui', 'uick', 'own ', ...}
{'brow', ' fox', 'n fo', ...}
{'ck b', ' bro', 'uick', ...}
{'e qu', 'rown', 'he q', ...}
```

```
{'he q', ' qui', 'ck b', ...}
{'n fo', ' qui', ' fox', ...}
{'rown', 'uick', ' bro', ...}
{'the ', 'brow', 'e qu', ...}
{'uick', 'own ', ' bro', ...}
```

the quick brown fox

```
{'the ', 'he q', 'e qu', ' qui', 'quic', 'uick', 'ick ', 'ck b',
 'k br', ' bro', 'brow', 'rown', 'own ', 'wn f', 'n fo', ' fox'}
```

```
{' fox', 'rown', 'k br', ...}
{' qui', 'uick', 'own ', ...}
{'brow', ' fox', 'n fo', ...}
{'ck b', ' bro', 'uick', ...}
{'e qu', 'rown', 'he q', ...}
```

```
{'he q', ' qui', 'ck b', ...}
{'n fo', ' qui', ' fox', ...}
{'rown', 'uick', ' bro', ...}
{'the ', 'brow', 'e qu', ...}
{'uick', 'own ', ' bro', ...}
```

```
'the ' ⟶ [277, 16, 414, 201]
'he q' ⟶ [433, 99, 47, 45]
'e qu' ⟶ [240, 247, 101, 472]
' qui' ⟶ [285, 4, 493, 460]
'quic' ⟶ [168, 282, 191, 98]
'uick' ⟶ [69, 495, 184, 316]
'ick ' ⟶ [459, 158, 240, 44]
'ck b' ⟶ [167, 335, 327, 97]
'k br' ⟶ [37, 161, 234, 438]
' bro' ⟶ [242, 61, 478, 50]
'brow' ⟶ [20, 427, 278, 251]
'rown' ⟶ [117, 366, 469, 218]
'own ' ⟶ [183, 359, 308, 301]
'wn f' ⟶ [462, 98, 465, 120]
'n fo' ⟶ [261, 349, 103, 331]
' fox' ⟶ [169, 249, 151, 75]
```

'brow'

```
'brow' ⟶ [20, 427, 278, 251]
'k br' ⟶ [37, 161, 234, 438]
'uick' ⟶ [69, 495, 184, 316]
'rown' ⟶ [117, 366, 469, 218]
'ck b' ⟶ [167, 335, 327, 97]
'quic' ⟶ [168, 282, 191, 98]
' fox' ⟶ [169, 249, 151, 75]
'own ' ⟶ [183, 359, 308, 301]
'e qu' ⟶ [240, 247, 101, 472]
' bro' ⟶ [242, 61, 478, 50]
'n fo' ⟶ [261, 349, 103, 331]
'the ' ⟶ [277, 16, 414, 201]
' qui' ⟶ [285, 4, 493, 460]
'he q' ⟶ [433, 99, 47, 45]
'ick ' ⟶ [459, 158, 240, 44]
'wn f' ⟶ [462, 98, 465, 120]
```

'brow'

```
'brow' ⟶ [20, 427, 278, 251]
'k br' ⟶ [37, 161, 234, 438]
'uick' ⟶ [69, 495, 184, 316]
'rown' ⟶ [117, 366, 469, 218]
'ck b' ⟶ [167, 335, 327, 97]
'quic' ⟶ [168, 282, 191, 98]
' fox' ⟶ [169, 249, 151, 75]
'own ' ⟶ [183, 359, 308, 301]
'e qu' ⟶ [240, 247, 101, 472]
' bro' ⟶ [242, 61, 478, 50]
'n fo' ⟶ [261, 349, 103, 331]
'the ' ⟶ [277, 16, 414, 201]
' qui' ⟶ [285, 4, 493, 460]
'he q' ⟶ [433, 99, 47, 45]
'ick ' ⟶ [459, 158, 240, 44]
'wn f' ⟶ [462, 98, 465, 120]
```

'brow'

' qui'

```
' qui' ⟶ [285, 4, 493, 460]
'the ' ⟶ [277, 16, 414, 201]
' bro' ⟶ [242, 61, 478, 50]
'wn f' ⟶ [462, 98, 465, 120]
'he q' ⟶ [433, 99, 47, 45]
'ick ' ⟶ [459, 158, 240, 44]
'k br' ⟶ [37, 161, 234, 438]
'e qu' ⟶ [240, 247, 101, 472]
' fox' ⟶ [169, 249, 151, 75]
'quic' ⟶ [168, 282, 191, 98]
'ck b' ⟶ [167, 335, 327, 97]
'n fo' ⟶ [261, 349, 103, 331]
'own ' ⟶ [183, 359, 308, 301]
'rown' ⟶ [117, 366, 469, 218]
'brow' ⟶ [20, 427, 278, 251]
'uick' ⟶ [69, 495, 184, 316]
```

'brow'

' qui'

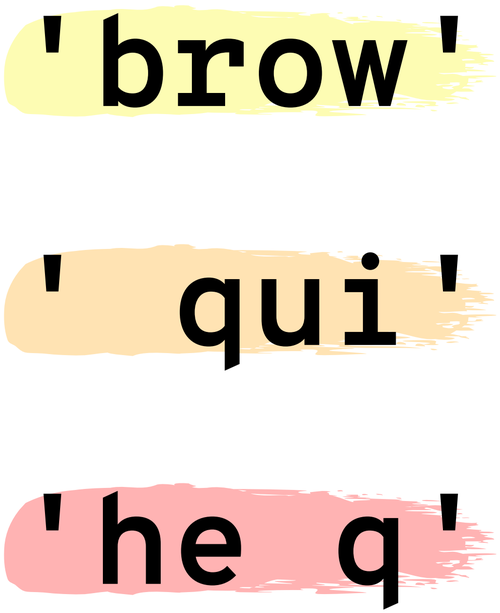
```
' qui' ⟶ [285, 4, 493, 460]
'the ' ⟶ [277, 16, 414, 201]
' bro' ⟶ [242, 61, 478, 50]
'wn f' ⟶ [462, 98, 465, 120]
'he q' ⟶ [433, 99, 47, 45]
'ick ' ⟶ [459, 158, 240, 44]
'k br' ⟶ [37, 161, 234, 438]
'e qu' ⟶ [240, 247, 101, 472]
' fox' ⟶ [169, 249, 151, 75]
'quic' ⟶ [168, 282, 191, 98]
'ck b' ⟶ [167, 335, 327, 97]
'n fo' ⟶ [261, 349, 103, 331]
'own ' ⟶ [183, 359, 308, 301]
'rown' ⟶ [117, 366, 469, 218]
'brow' ⟶ [20, 427, 278, 251]
'uick' ⟶ [69, 495, 184, 316]
```

'brow'

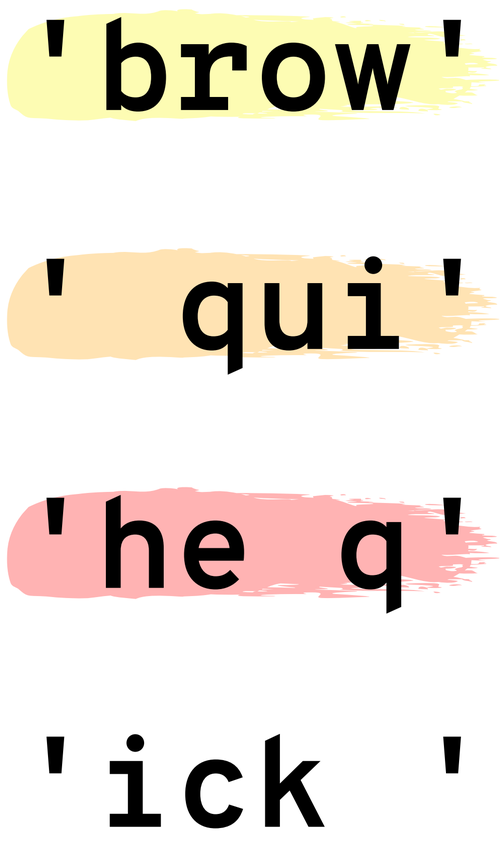
' qui'

'he q'

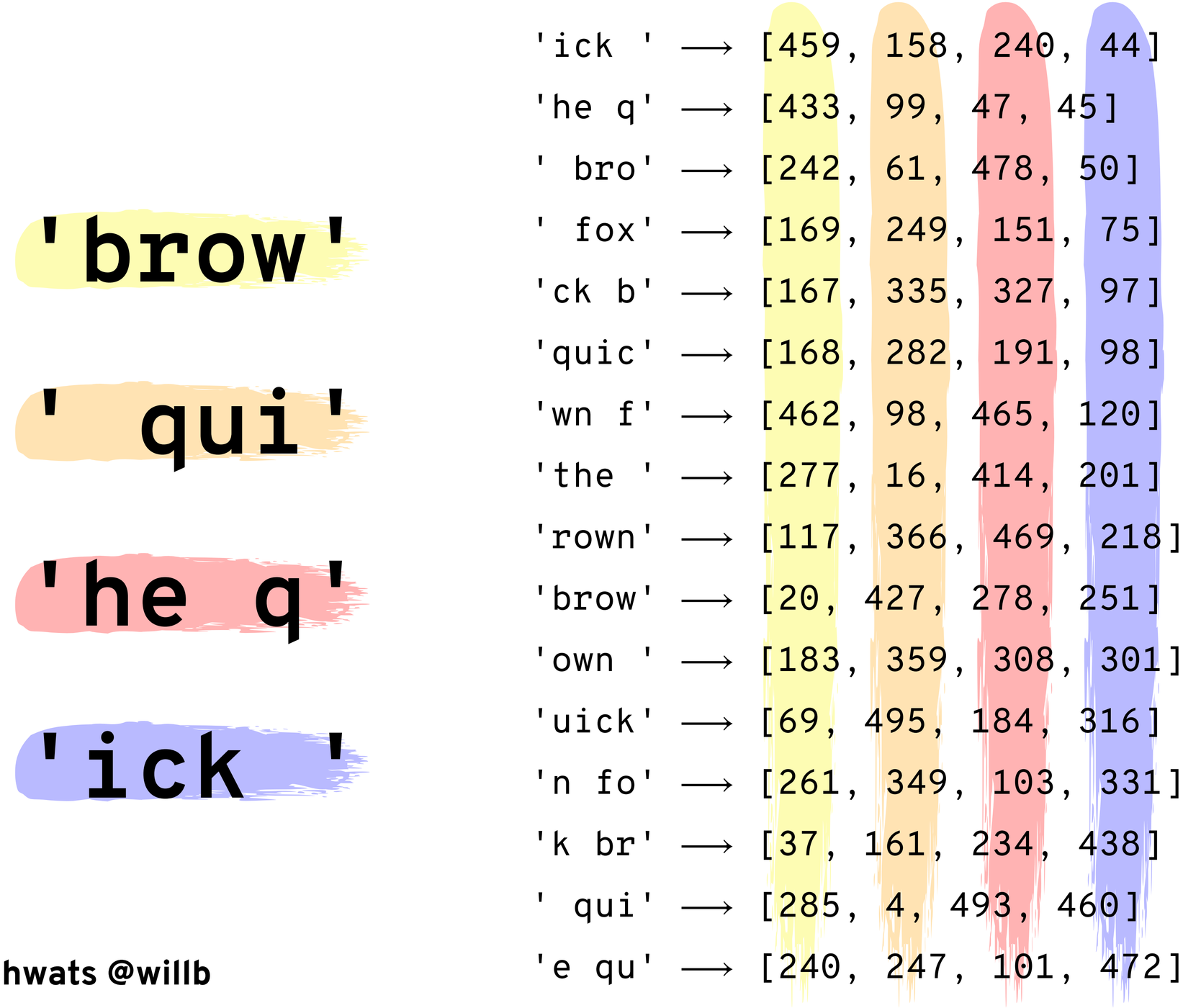
```
'he q' ⟶ [433, 99, 47, 45]
'e qu' ⟶ [240, 247, 101, 472]
'n fo' ⟶ [261, 349, 103, 331]
' fox' ⟶ [169, 249, 151, 75]
'uick' ⟶ [69, 495, 184, 316]
'quic' ⟶ [168, 282, 191, 98]
'k br' ⟶ [37, 161, 234, 438]
'ick ' ⟶ [459, 158, 240, 44]
'brow' ⟶ [20, 427, 278, 251]
'own ' ⟶ [183, 359, 308, 301]
'ck b' ⟶ [167, 335, 327, 97]
'the ' ⟶ [277, 16, 414, 201]
'wn f' ⟶ [462, 98, 465, 120]
'rown' ⟶ [117, 366, 469, 218]
' bro' ⟶ [242, 61, 478, 50]
' qui' ⟶ [285, 4, 493, 460]
```




```
'he q' ⟶ [433, 99, 47, 45]
'e qu' ⟶ [240, 247, 101, 472]
'n fo' ⟶ [261, 349, 103, 331]
' fox' ⟶ [169, 249, 151, 75]
'uick' ⟶ [69, 495, 184, 316]
'quic' ⟶ [168, 282, 191, 98]
'k br' ⟶ [37, 161, 234, 438]
'ick ' ⟶ [459, 158, 240, 44]
'brow' ⟶ [20, 427, 278, 251]
'own ' ⟶ [183, 359, 308, 301]
'ck b' ⟶ [167, 335, 327, 97]
'the ' ⟶ [277, 16, 414, 201]
'wn f' ⟶ [462, 98, 465, 120]
'rown' ⟶ [117, 366, 469, 218]
' bro' ⟶ [242, 61, 478, 50]
' qui' ⟶ [285, 4, 493, 460]
```




```
'ick ' ⟶ [459, 158, 240, 44]
'he q' ⟶ [433, 99, 47, 45]
' bro' ⟶ [242, 61, 478, 50]
' fox' ⟶ [169, 249, 151, 75]
'ck b' ⟶ [167, 335, 327, 97]
'quic' ⟶ [168, 282, 191, 98]
'wn f' ⟶ [462, 98, 465, 120]
'the ' ⟶ [277, 16, 414, 201]
'rown' ⟶ [117, 366, 469, 218]
'brow' ⟶ [20, 427, 278, 251]
'own ' ⟶ [183, 359, 308, 301]
'uick' ⟶ [69, 495, 184, 316]
'n fo' ⟶ [261, 349, 103, 331]
'k br' ⟶ [37, 161, 234, 438]
' qui' ⟶ [285, 4, 493, 460]
'e qu' ⟶ [240, 247, 101, 472]
```


'brow'
' qui'
'he q'
'ick '
'ick ' ⟶ [459, 158, 240, 44]
'he q' ⟶ [433, 99, 47, 45]
' bro' ⟶ [242, 61, 478, 50]
' fox' ⟶ [169, 249, 151, 75]
'ck b' ⟶ [167, 335, 327, 97]
'quic' ⟶ [168, 282, 191, 98]
'wn f' ⟶ [462, 98, 465, 120]
'the ' ⟶ [277, 16, 414, 201]
'rown' ⟶ [117, 366, 469, 218]
'brow' ⟶ [20, 427, 278, 251]
'own ' ⟶ [183, 359, 308, 301]
'uick' ⟶ [69, 495, 184, 316]
'n fo' ⟶ [261, 349, 103, 331]
'k br' ⟶ [37, 161, 234, 438]
' qui' ⟶ [285, 4, 493, 460]
'e qu' ⟶ [240, 247, 101, 472]

```
[20, 4, 47, 44]
```

```
[20, 4, 47, 44]
```

```
18
[20, 4, 47, 44]
```

[18, 4, 47, 44]

[20, 4, 32, 44]

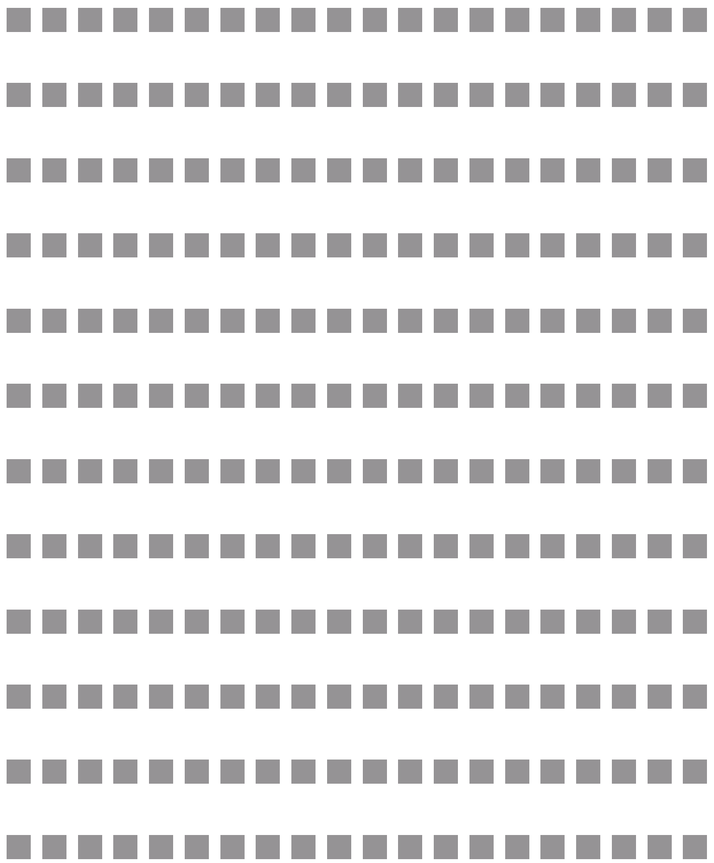
```
[18, 4, 32, 44]
```

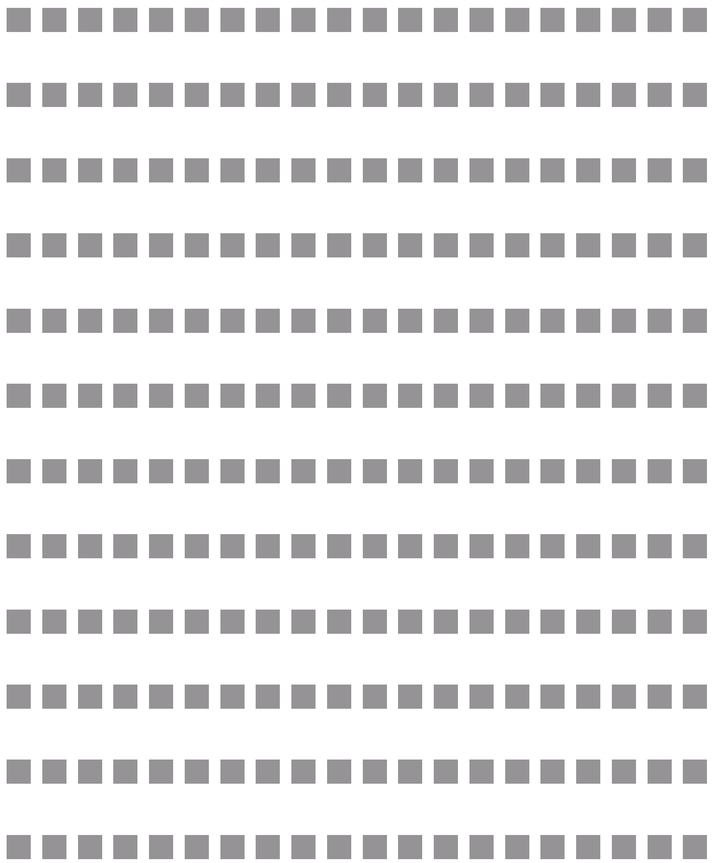
```
[20, 4, 47, 44]
```

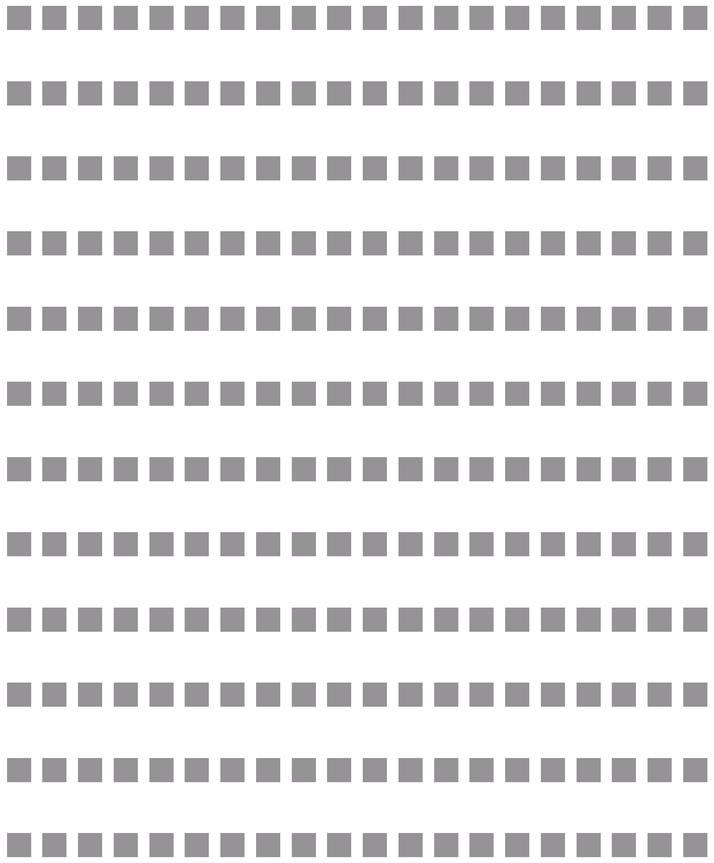
```
[20, 4, 32, 44]
```

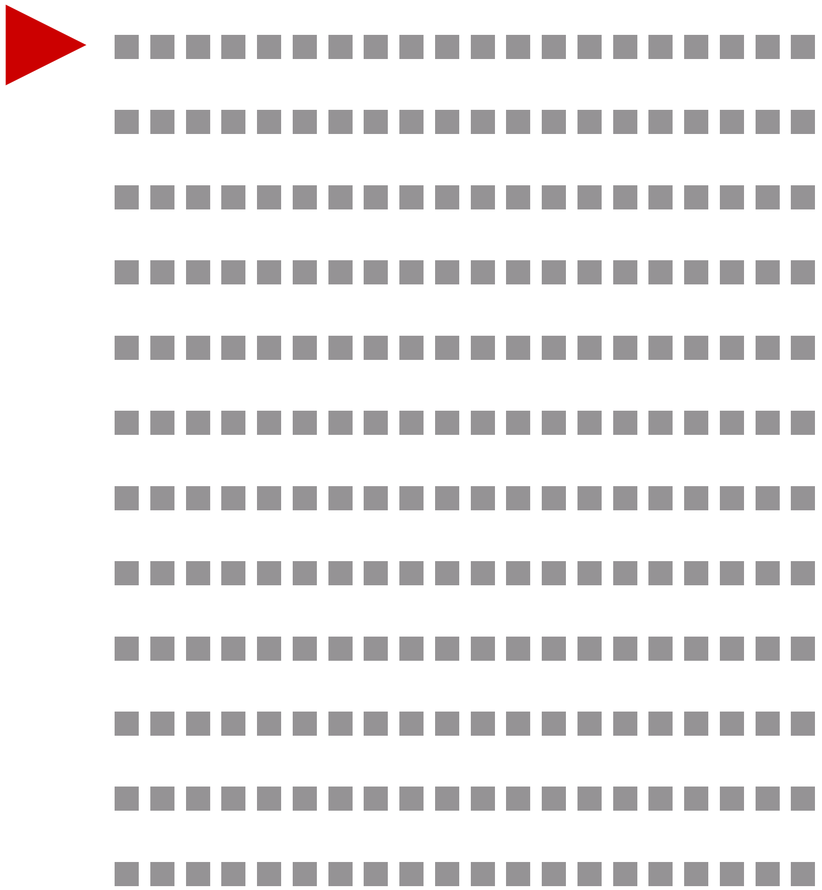
```
[20, 4, 47, 44]
```

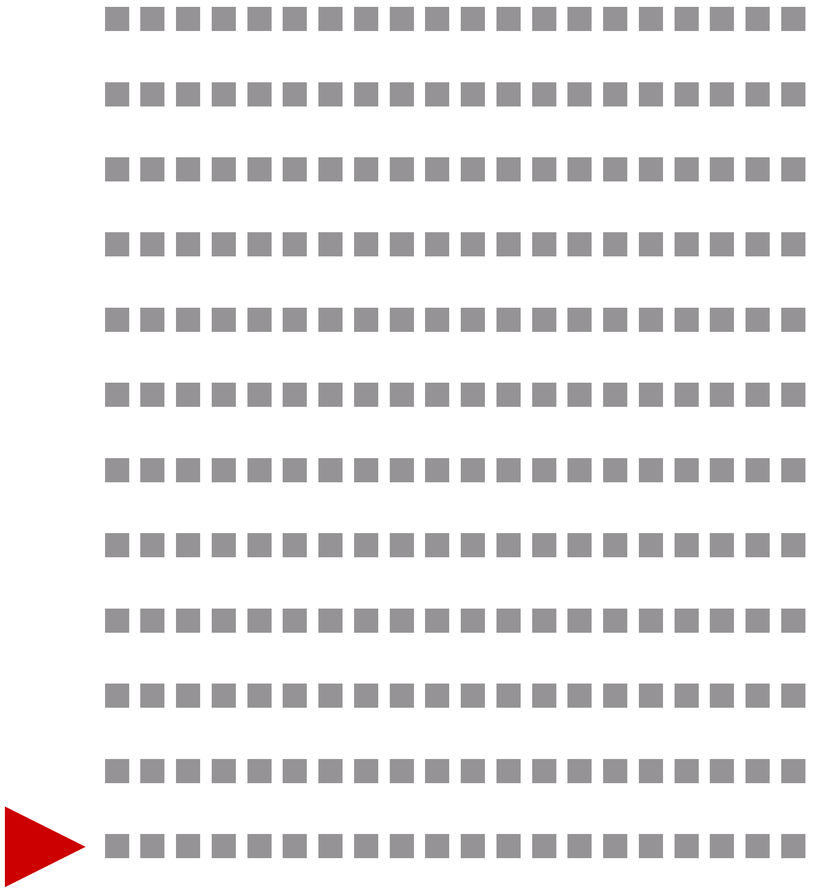
```
[20, 4, 32, 44]
```

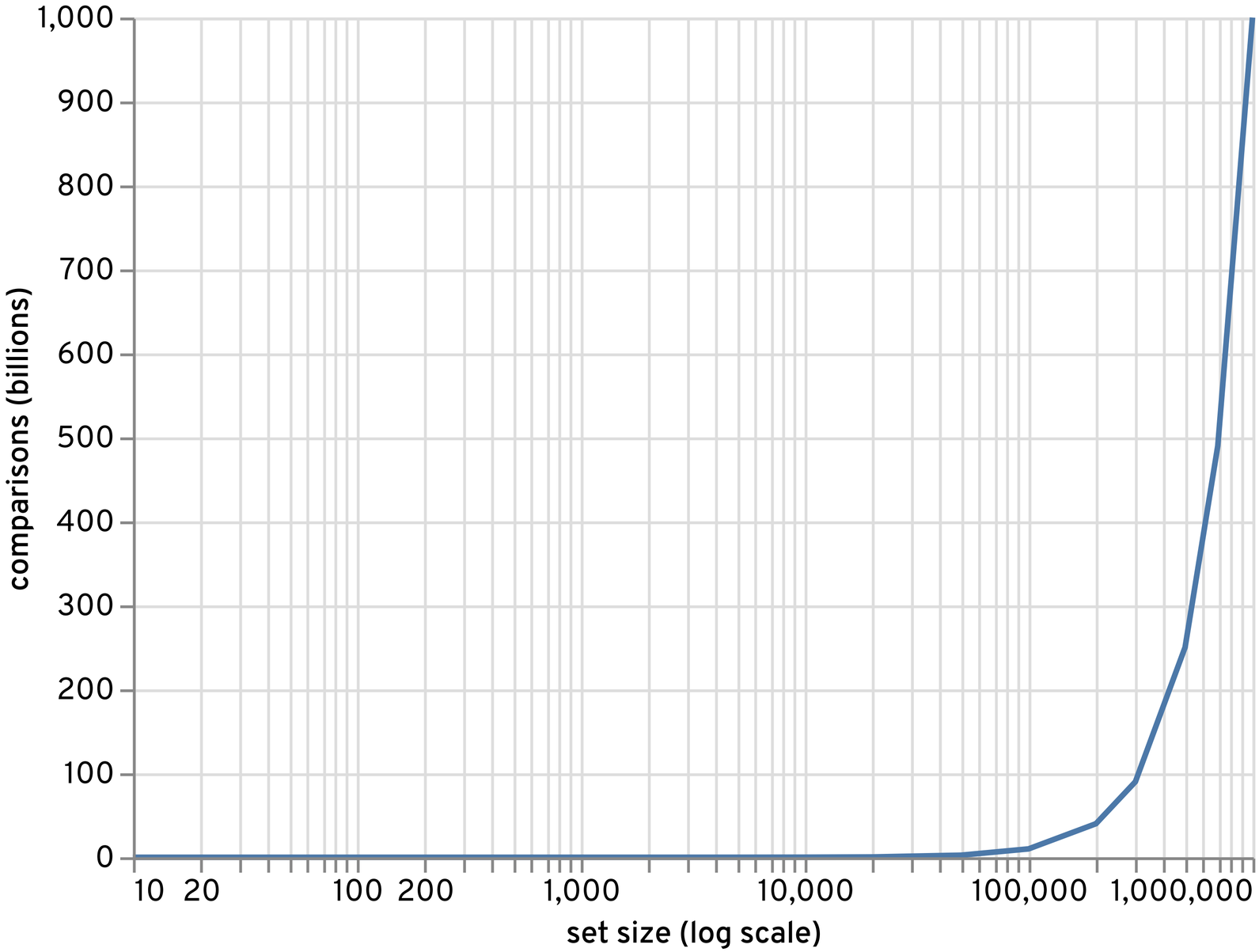


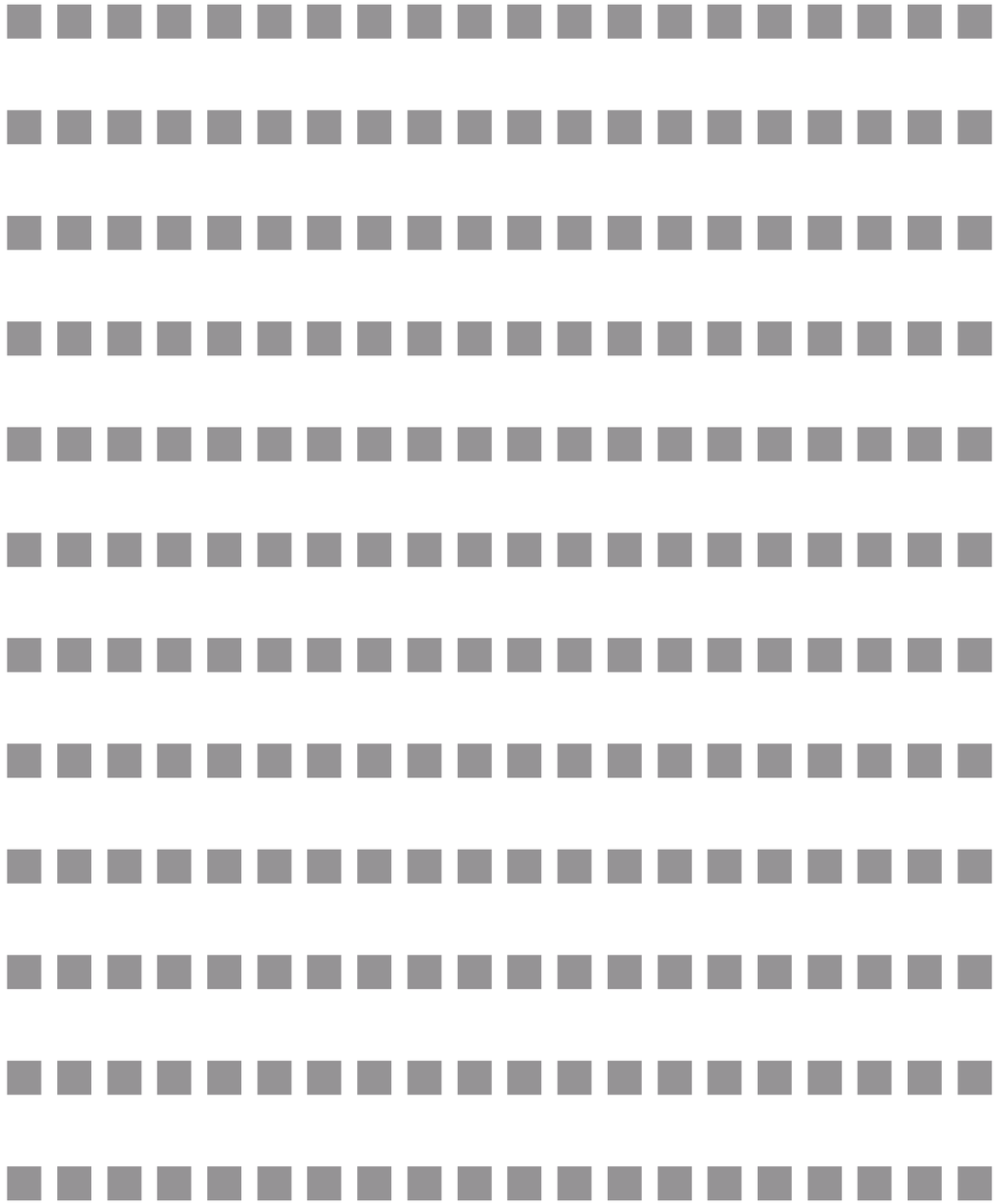


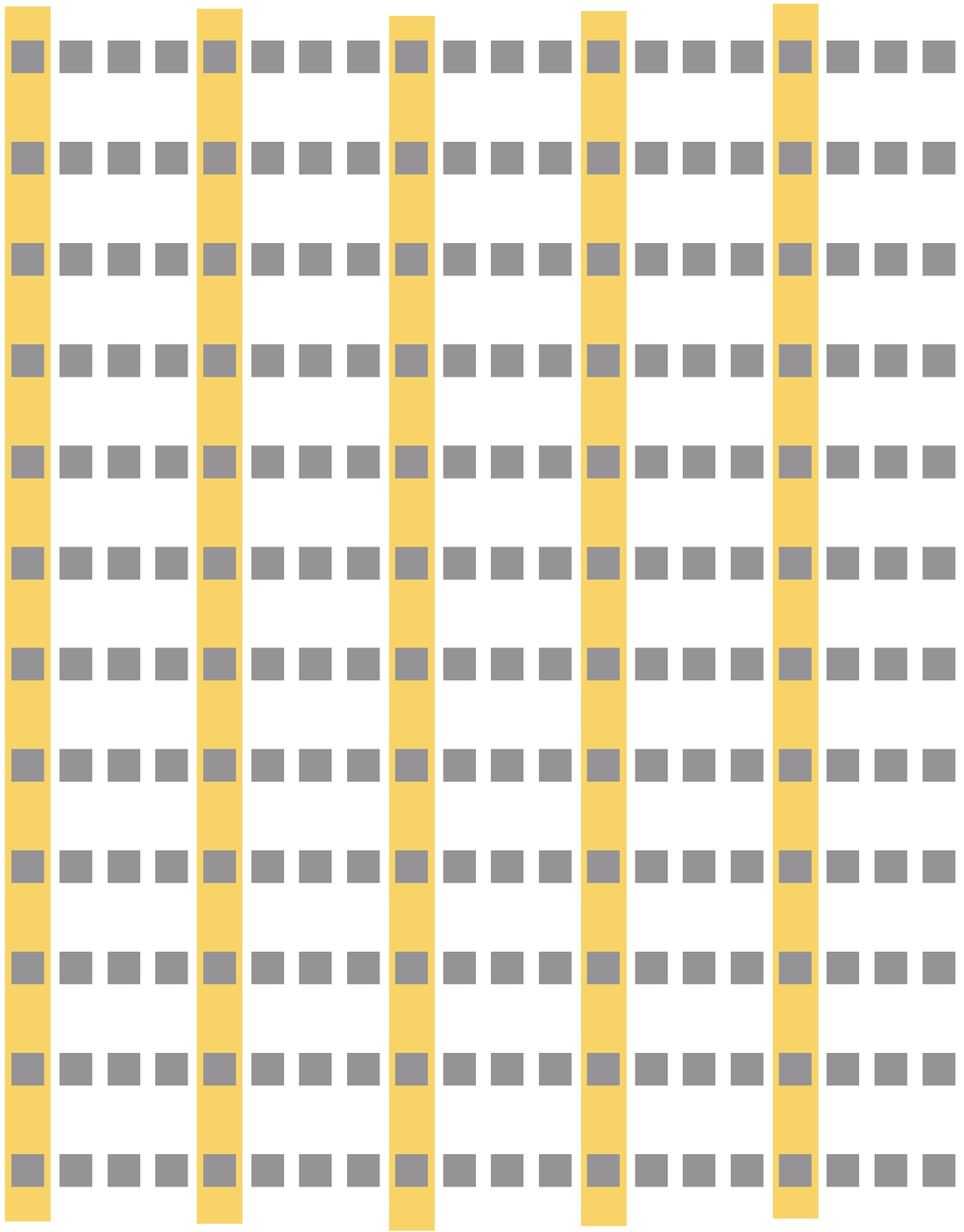


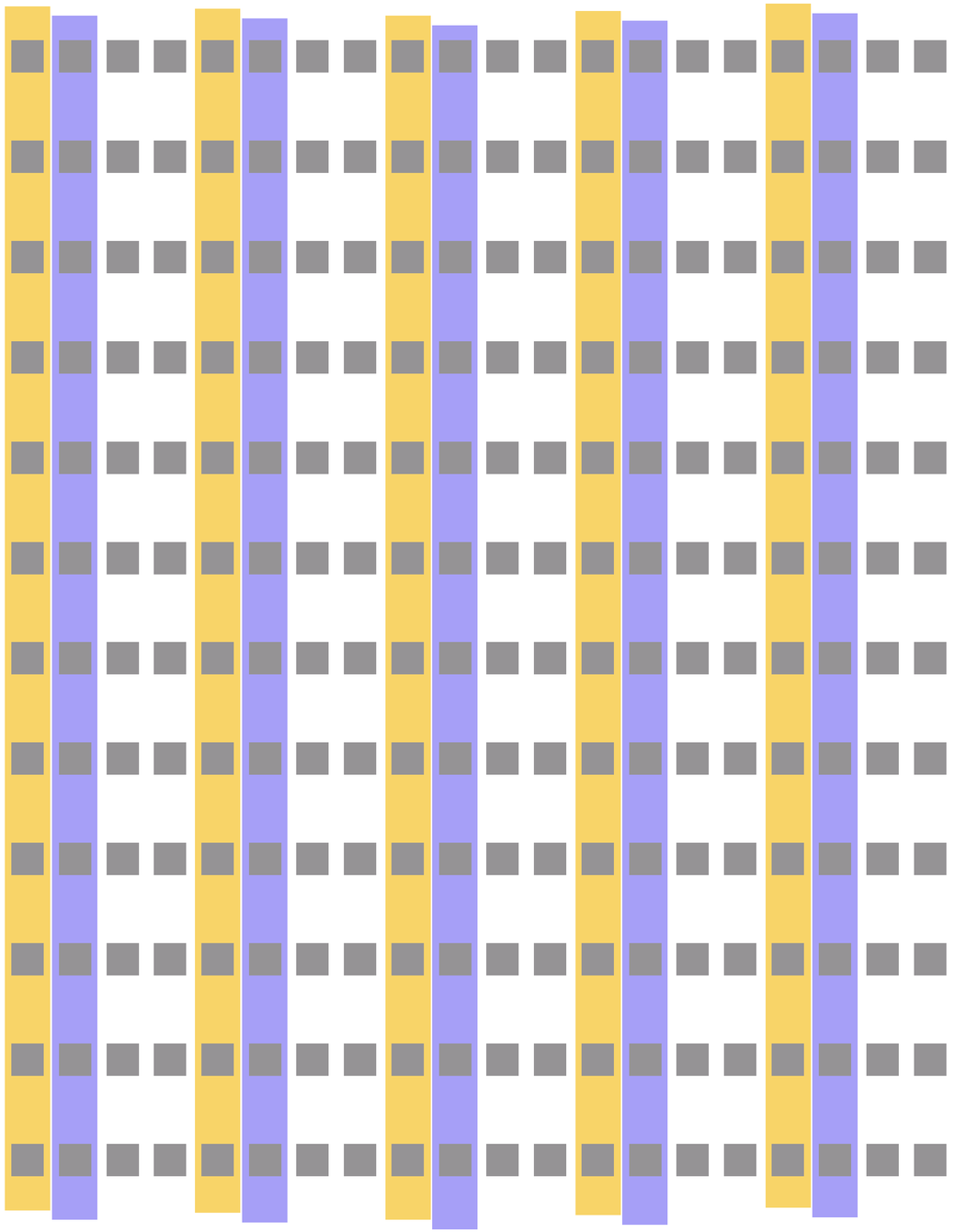


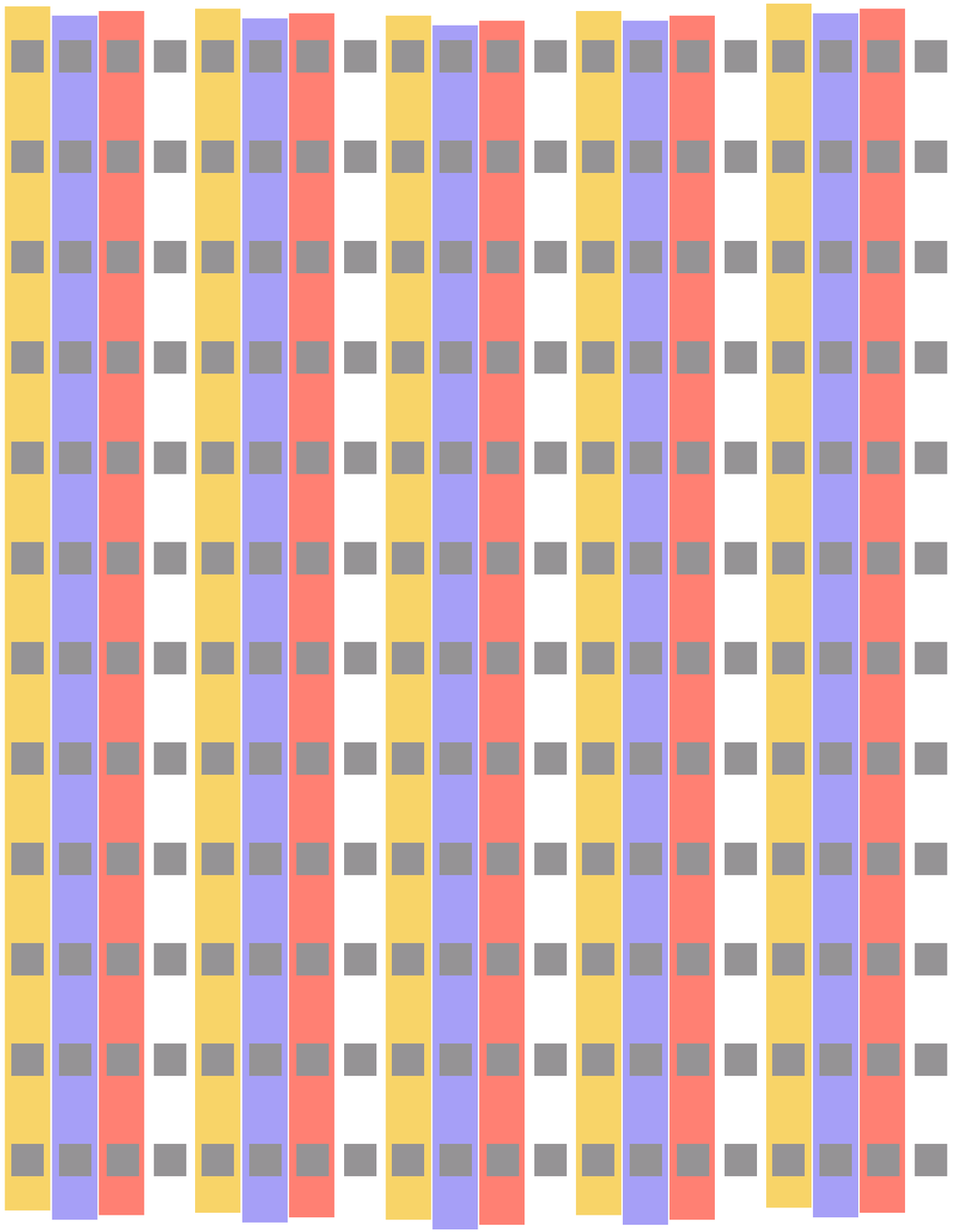


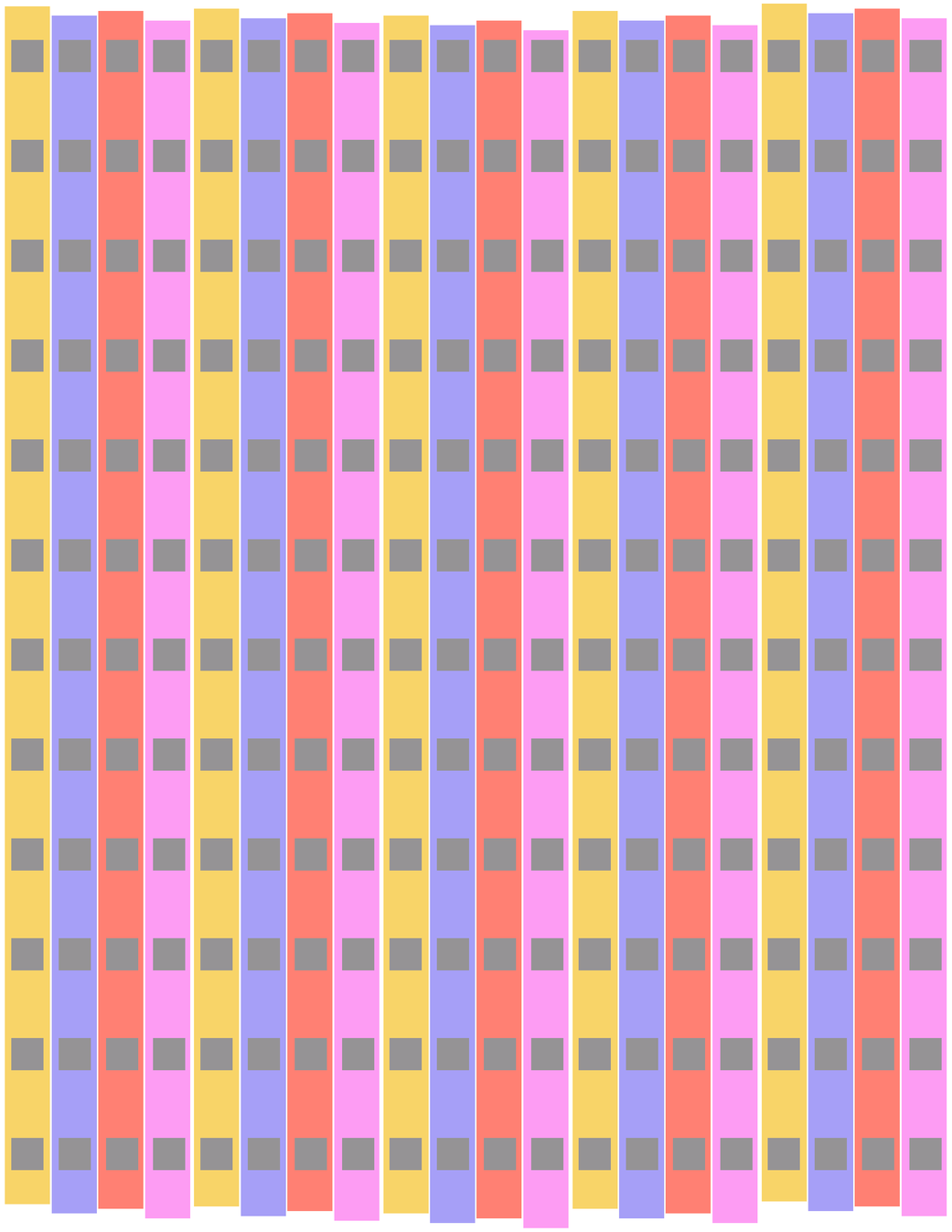

1,000
900
800
700
600
500
400
300
200
100
0
comparisons (billions)
10
20
100
200
1,000
10,000
100,000
1,000,000
set size (log scale)

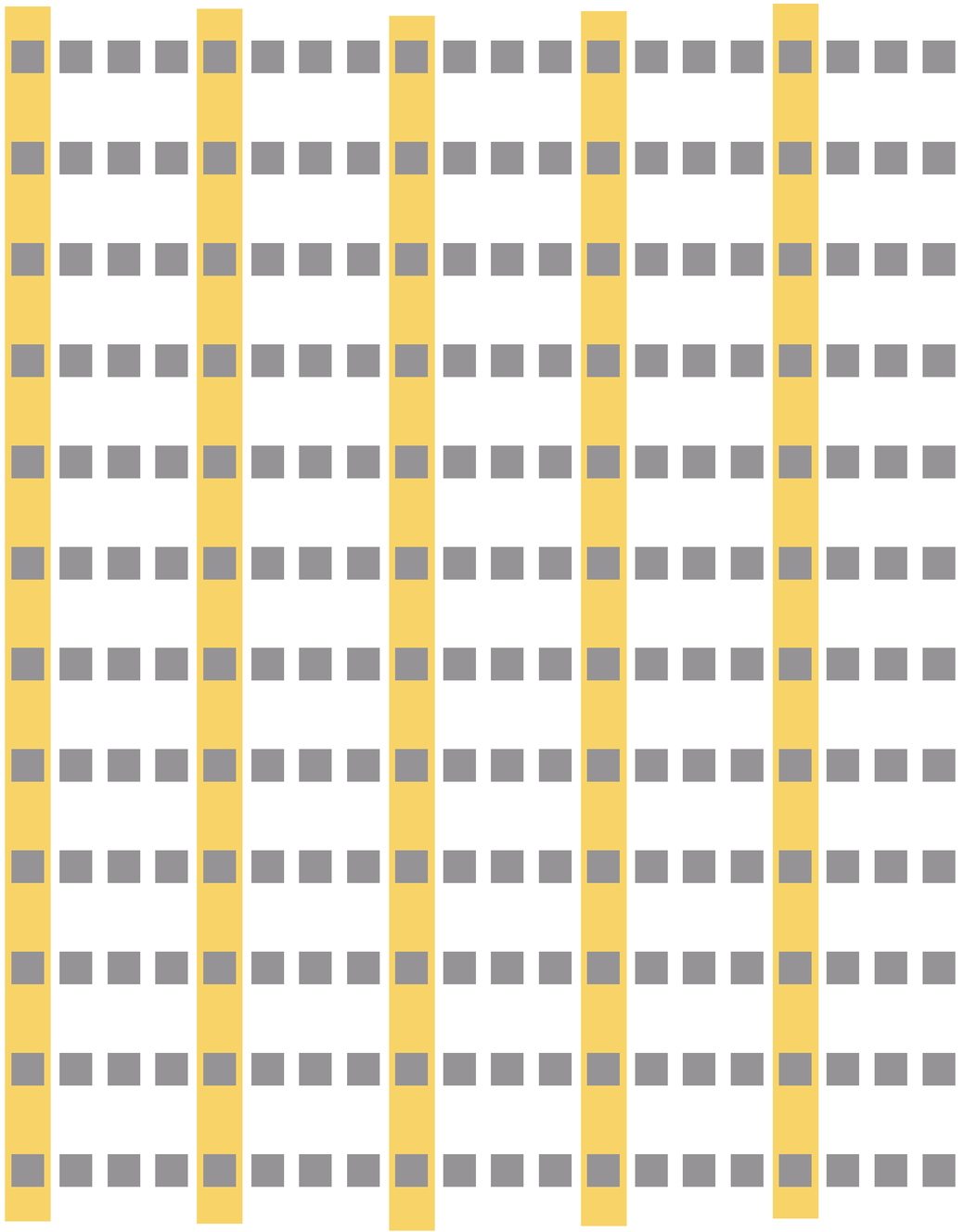








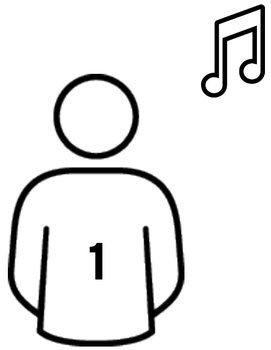




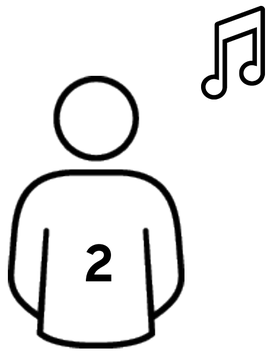
Scalable recommendations with MinHash

@sophwats @willb

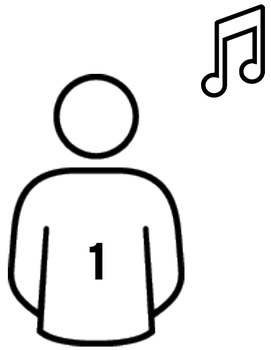
Red Hat



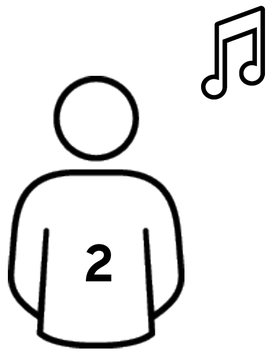
[Taylor Swift, Pulp, The Smiths,
Joy Division, Billy Bragg]



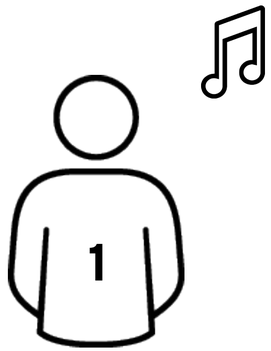
[Taylor Swift, Pulp, The Smiths,
Joy Division, New Order]



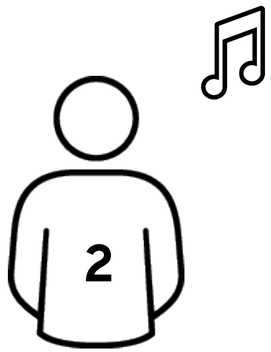
[Taylor Swift, Pulp, The Smiths,
Joy Division, Billy Bragg]



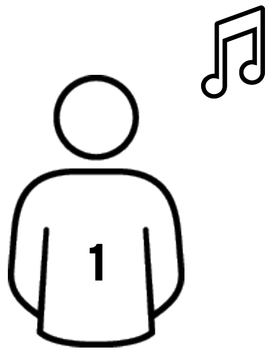
[Taylor Swift, Pulp, The Smiths,
Joy Division, New Order]



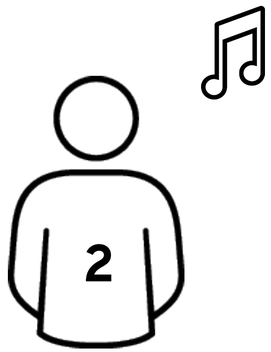
[Taylor Swift, Pulp, The Smiths,
Joy Division, Billy Bragg]



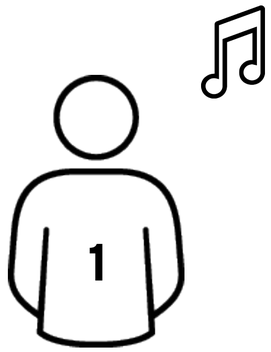
[Taylor Swift, Pulp, The Smiths,
Joy Division, New Order]



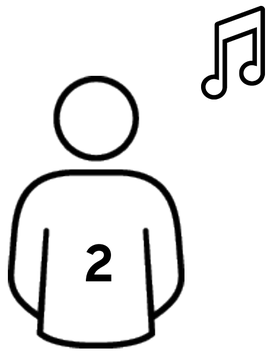
[Taylor Swift, Pulp, The Smiths,
Joy Division, Billy Bragg]



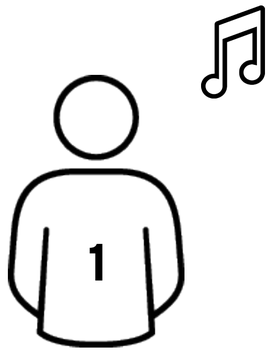
[Taylor Swift, Pulp, The Smiths,
Joy Division, New Order]



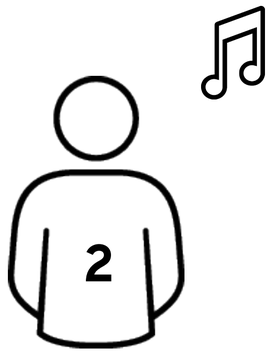

```
[16, 19, 91, 93, 43]
```



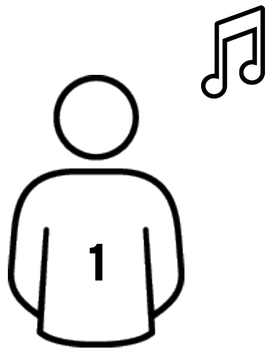

```
[89, 19, 48, 95, 43]
```

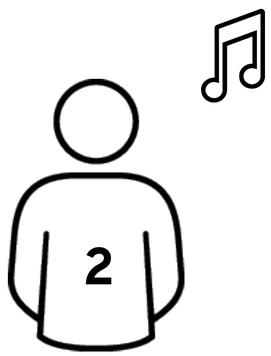


[16, 19, 91, 93, 43]

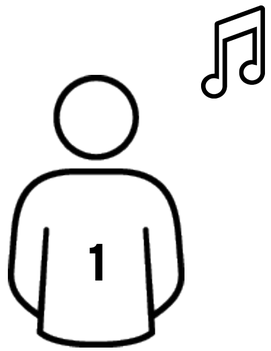


[89, 19, 48, 95, 43]

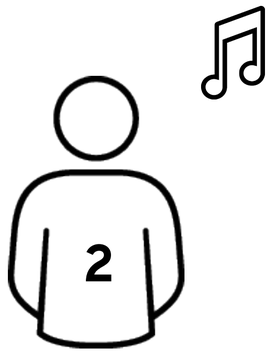




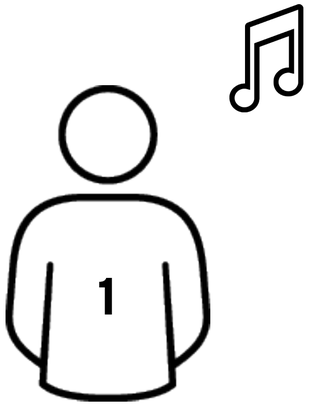

```
[16, 19, 91, 93, 43]

[89, 19, 48, 95, 43]
```

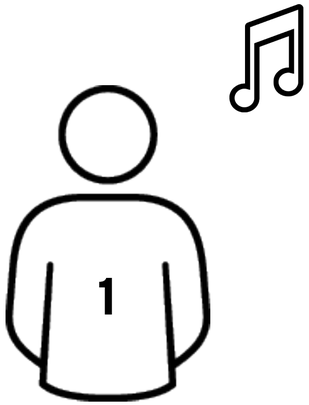


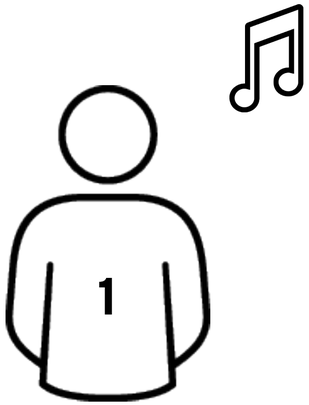
[Taylor Swift, Pulp, The Smiths,
Joy Division, Billy Bragg]

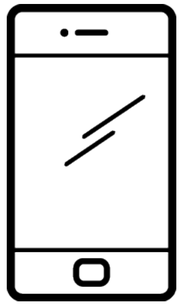


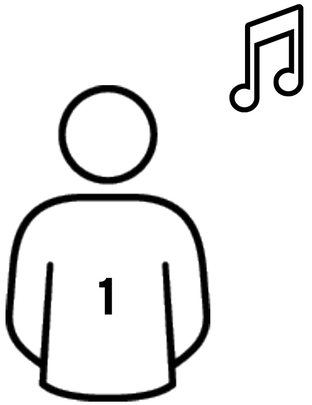
[Taylor Swift, Pulp, The Smiths,
Joy Division, New Order]


1

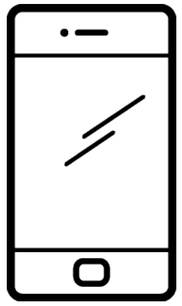

1


1

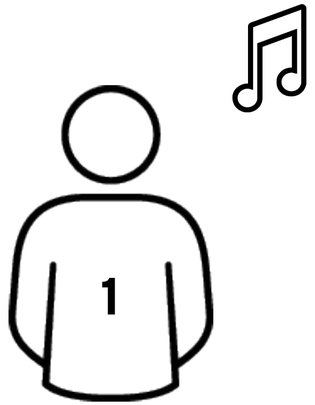




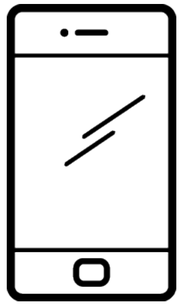
[16, 19, 91, 93, 43]



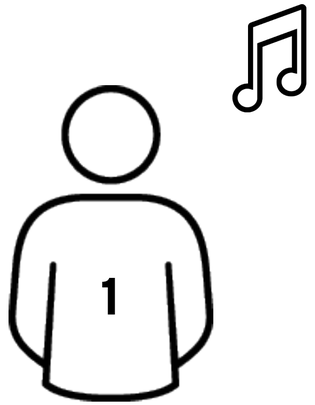
[89, 19, 48, 95, 43]

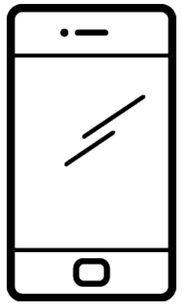


```
[16, 19, 91, 93, 43]
```

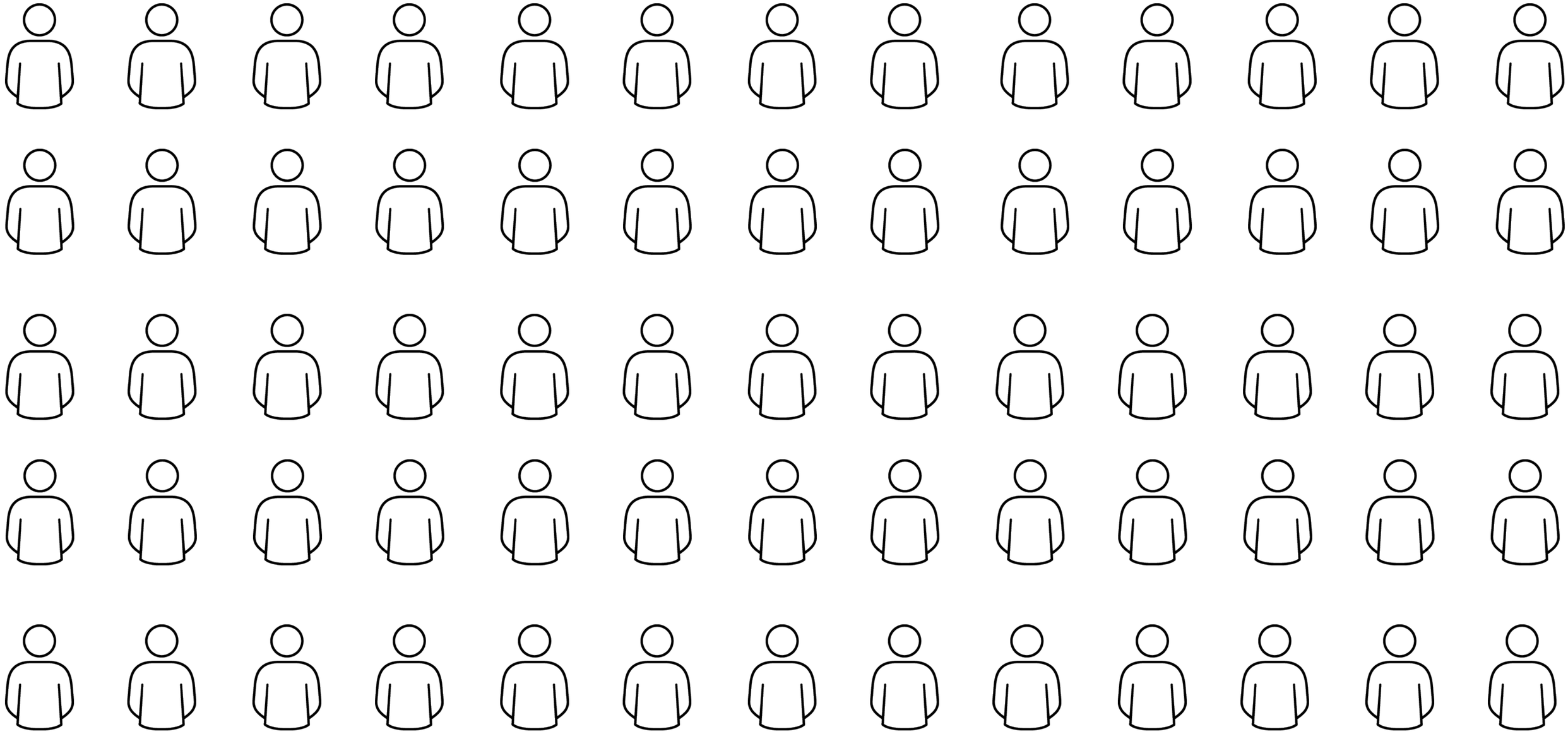


```
[89, 19, 48, 95, 43]
```

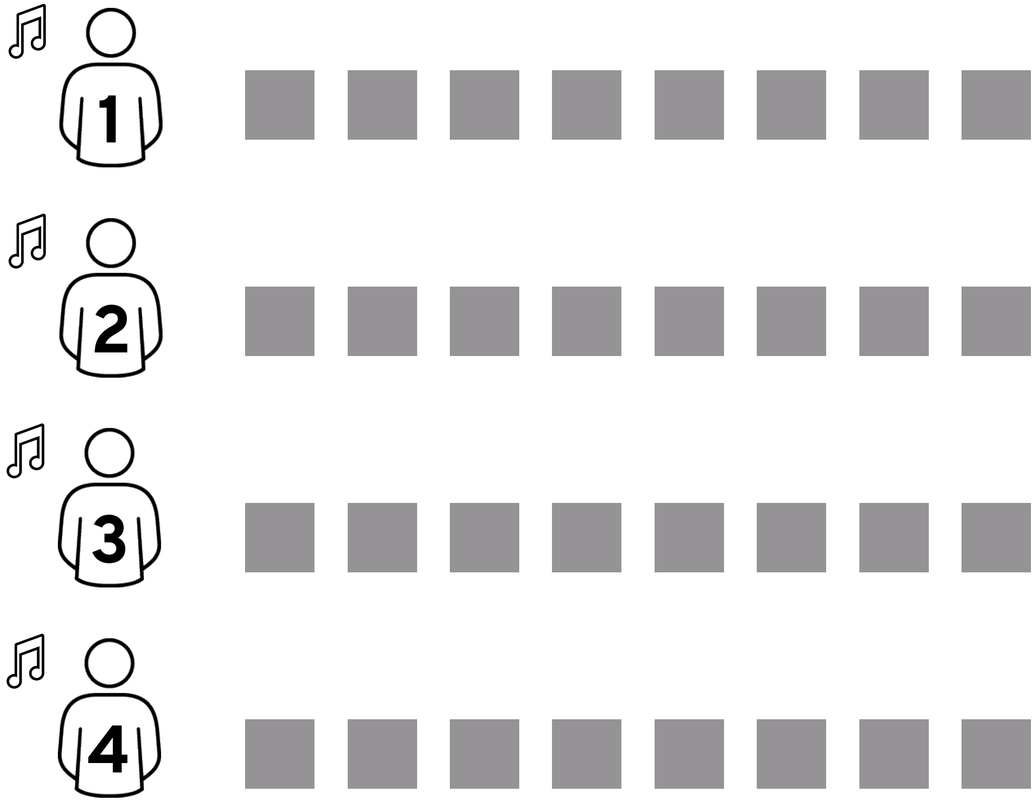




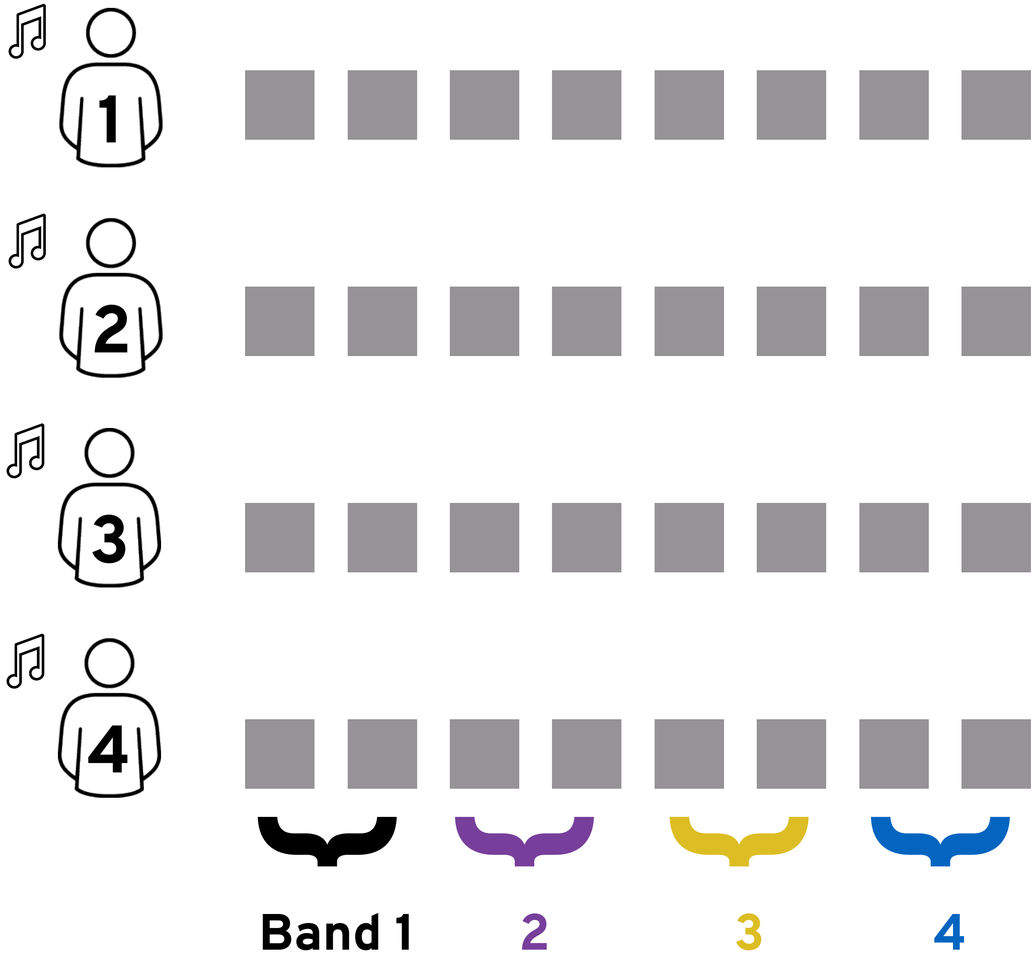
```
[16, 19, 91, 93, 43]
[16, 19, 48, 93, 43]
[89, 19, 48, 95, 43]
```



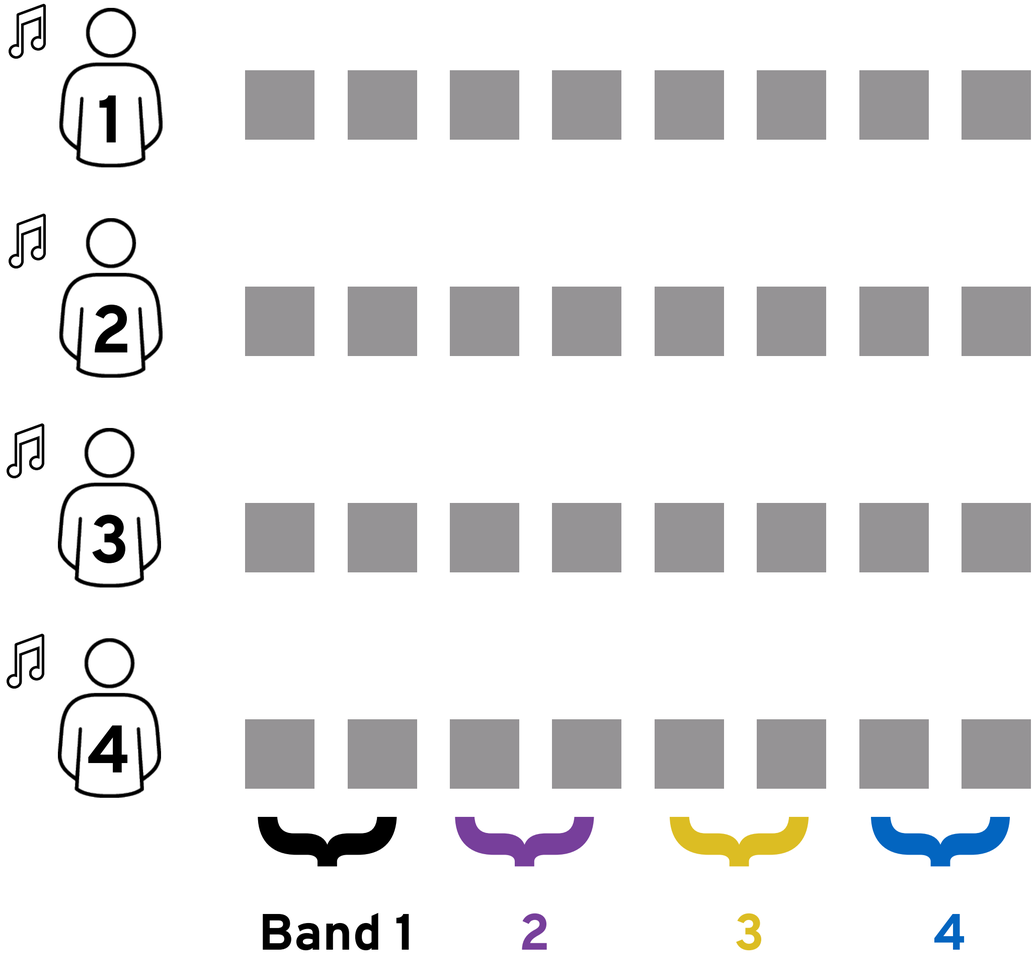
LOCALITY-SENSITIVE MINHASH



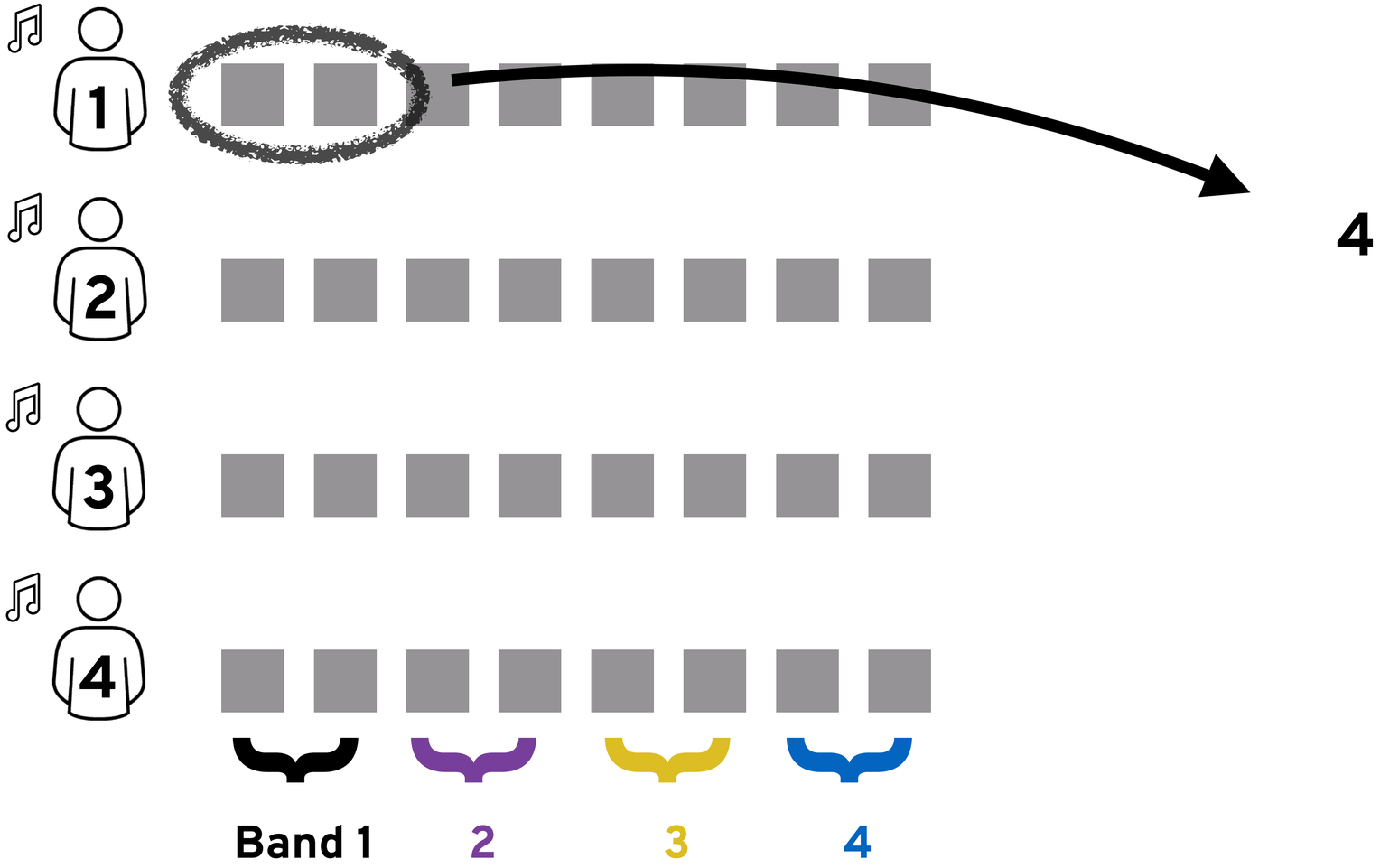
Candidate pairs:



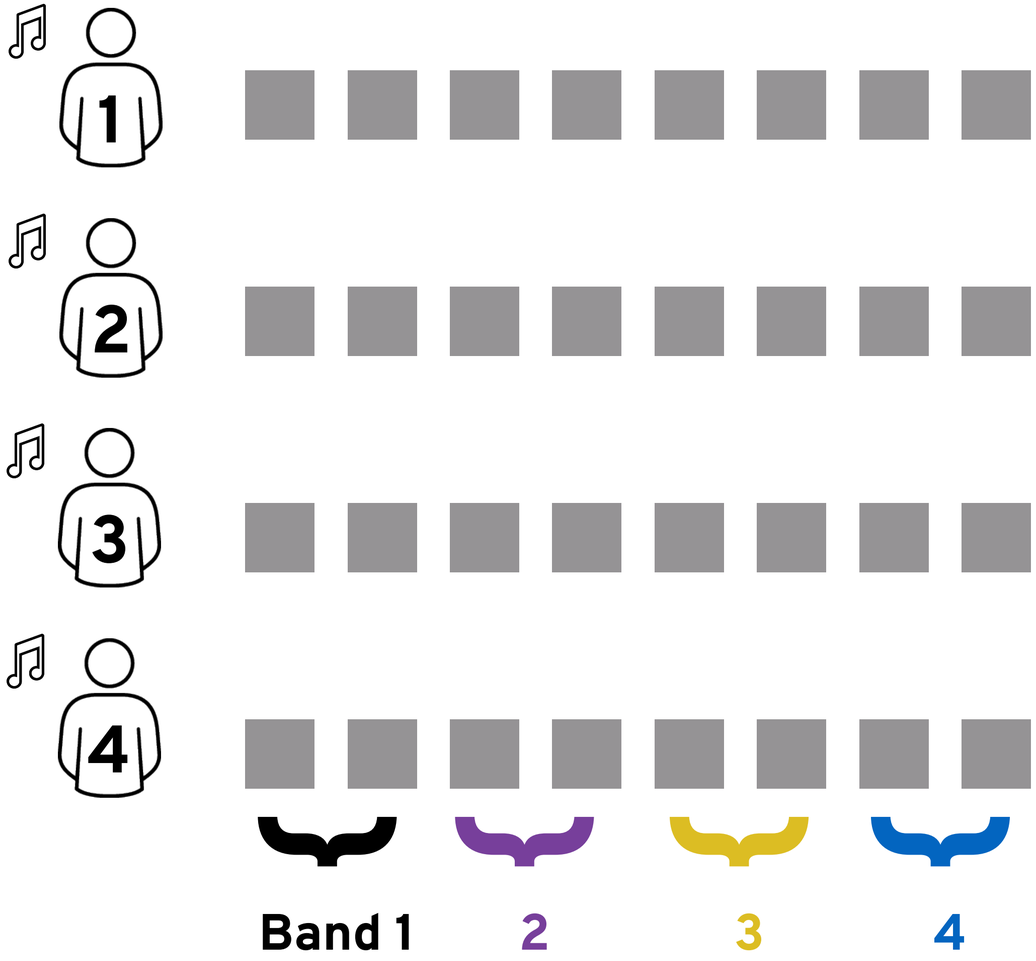
Candidate pairs:



Candidate pairs:

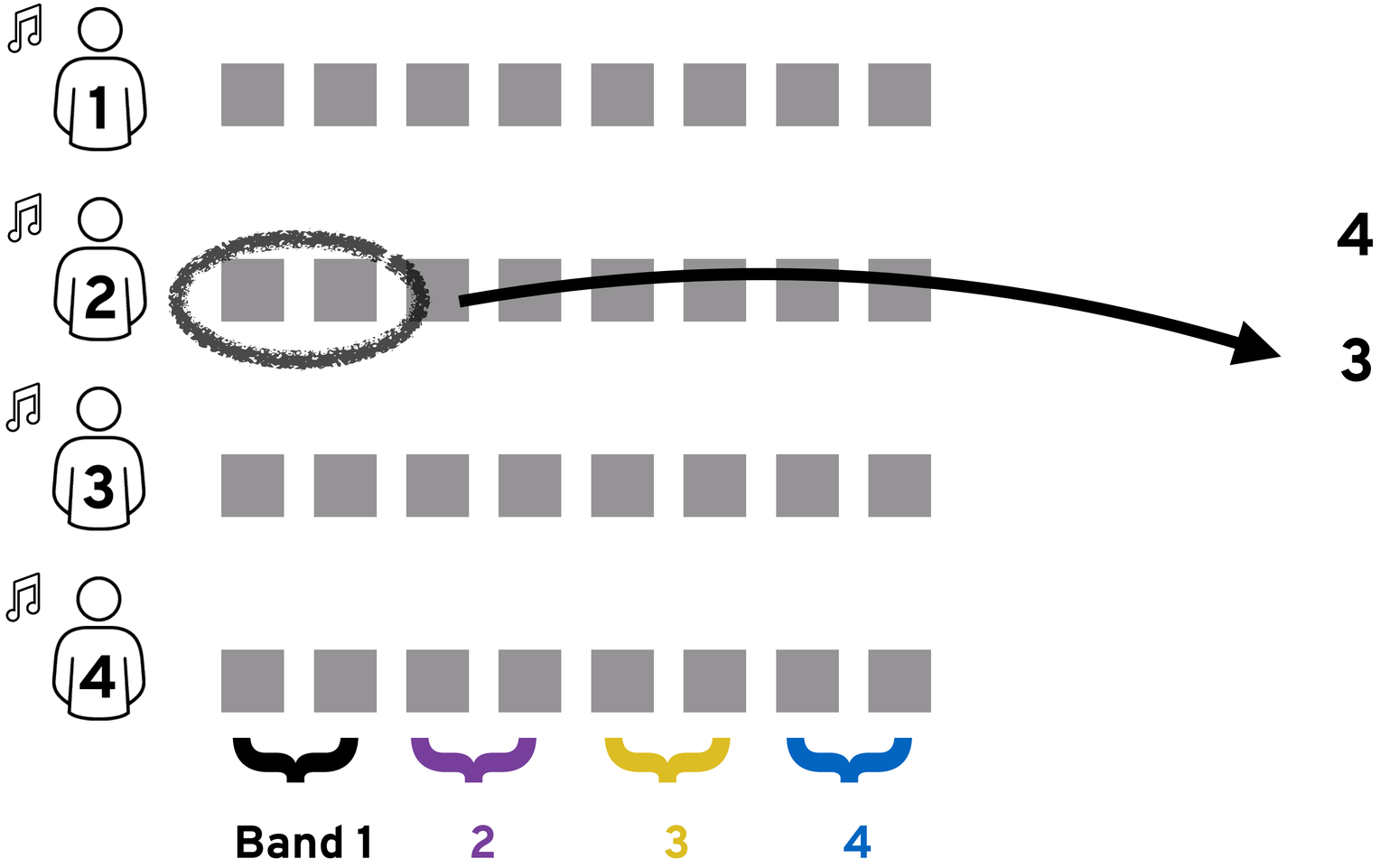


Candidate pairs:

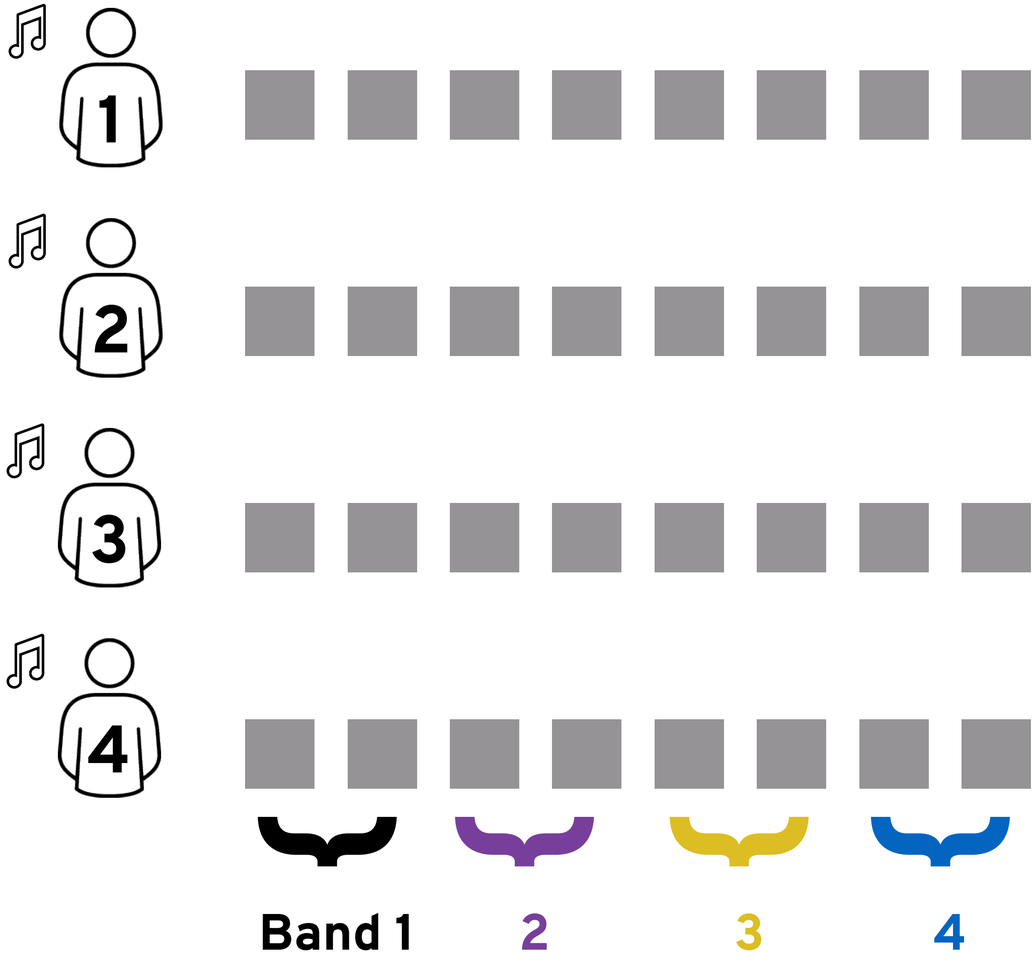


4

Candidate pairs:



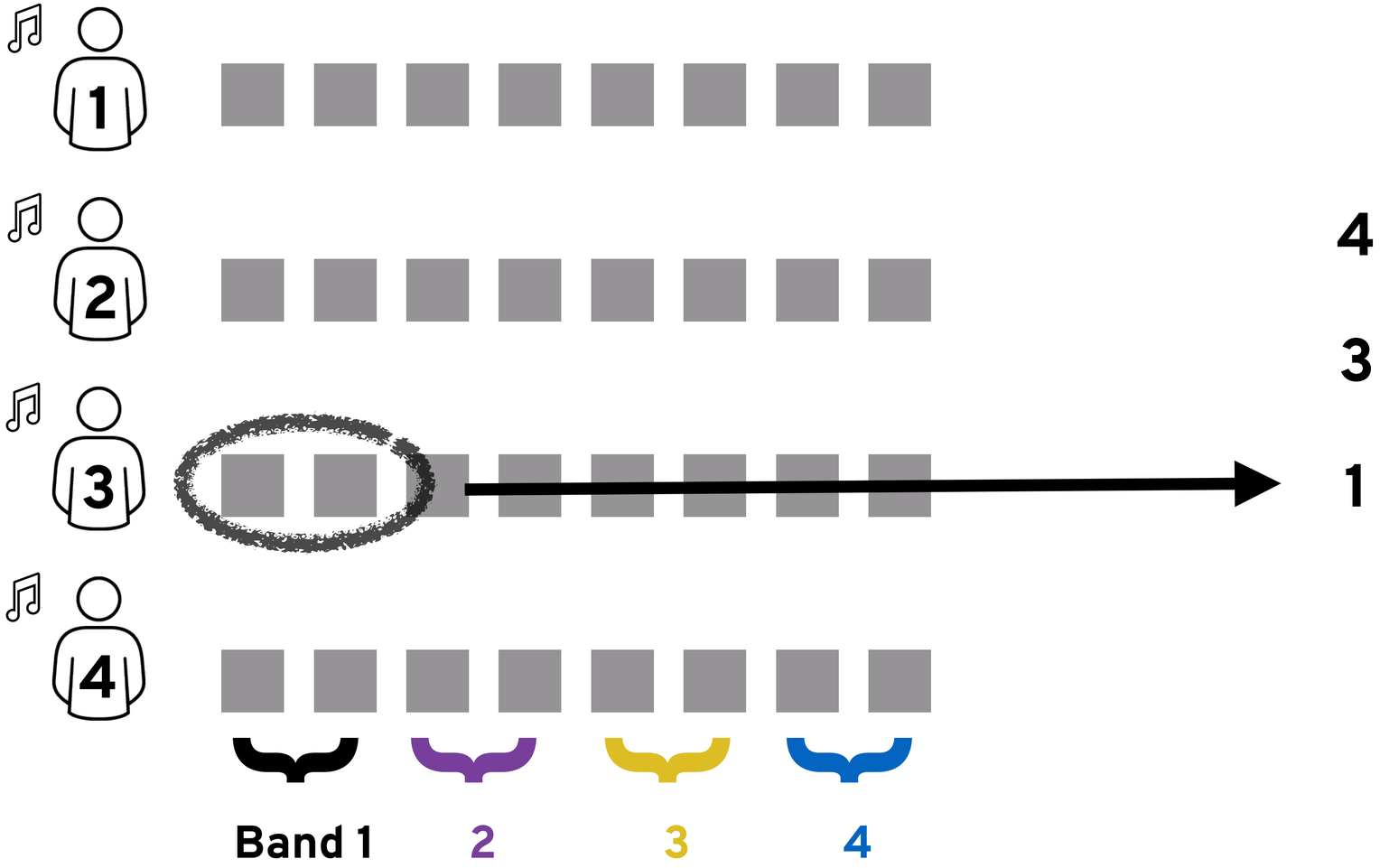
Candidate pairs:



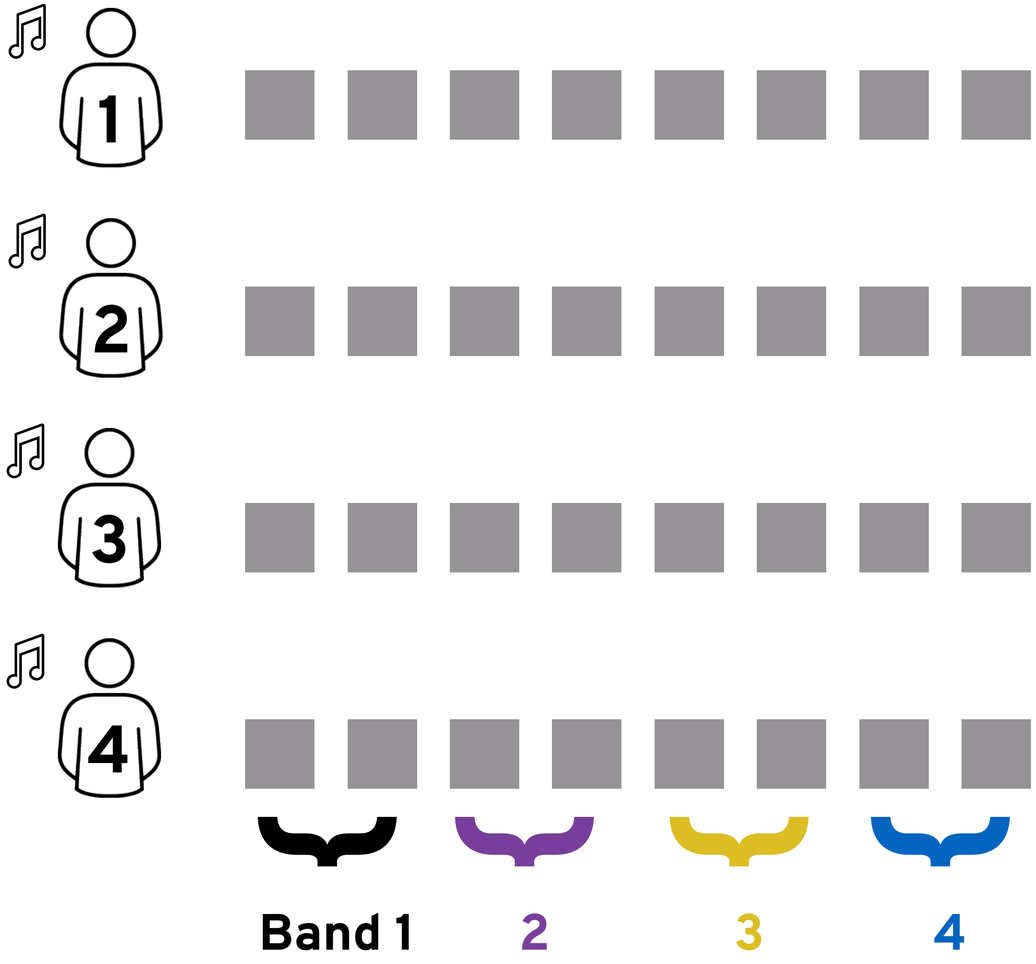
4

3

Candidate pairs:



Candidate pairs:

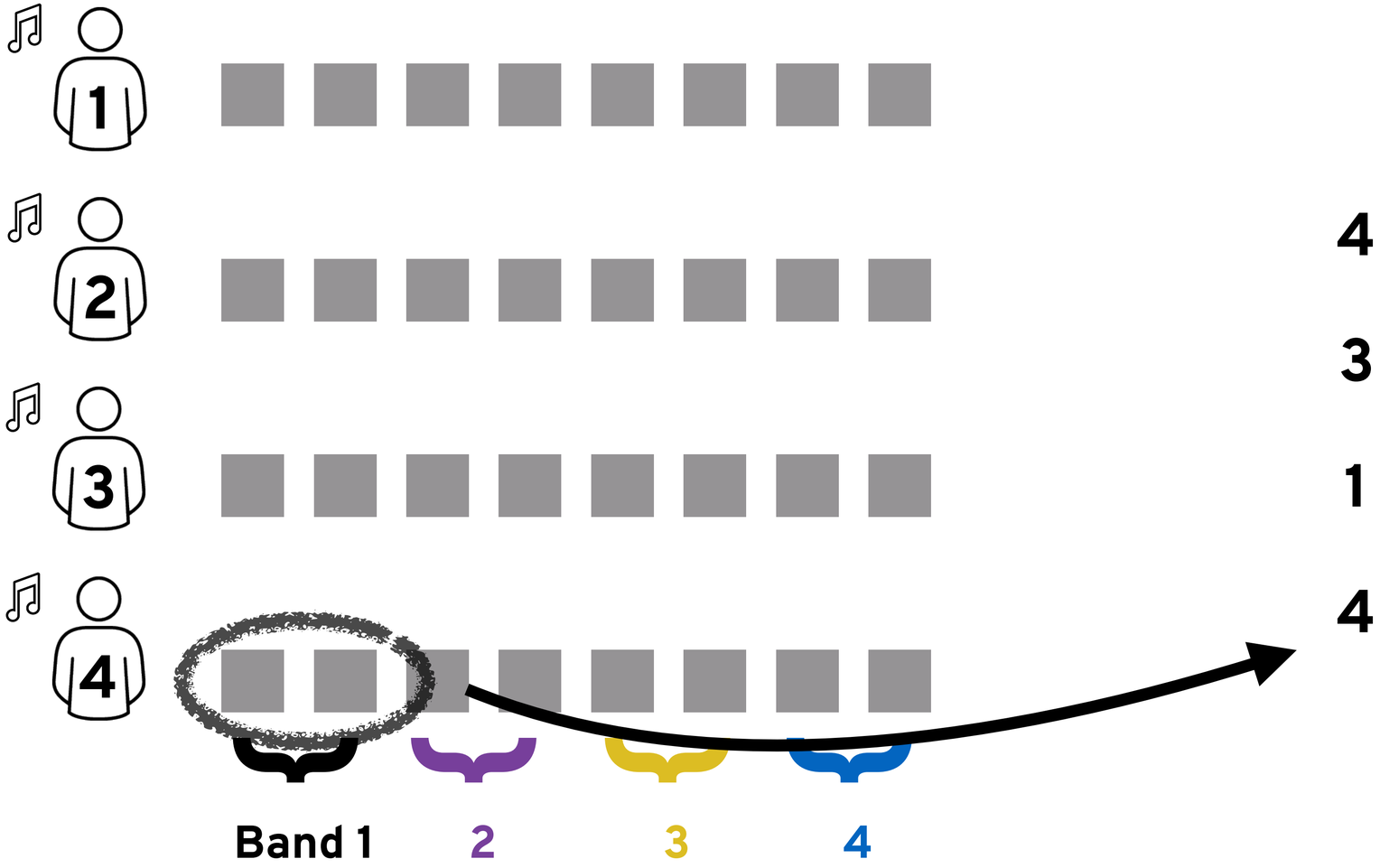


4

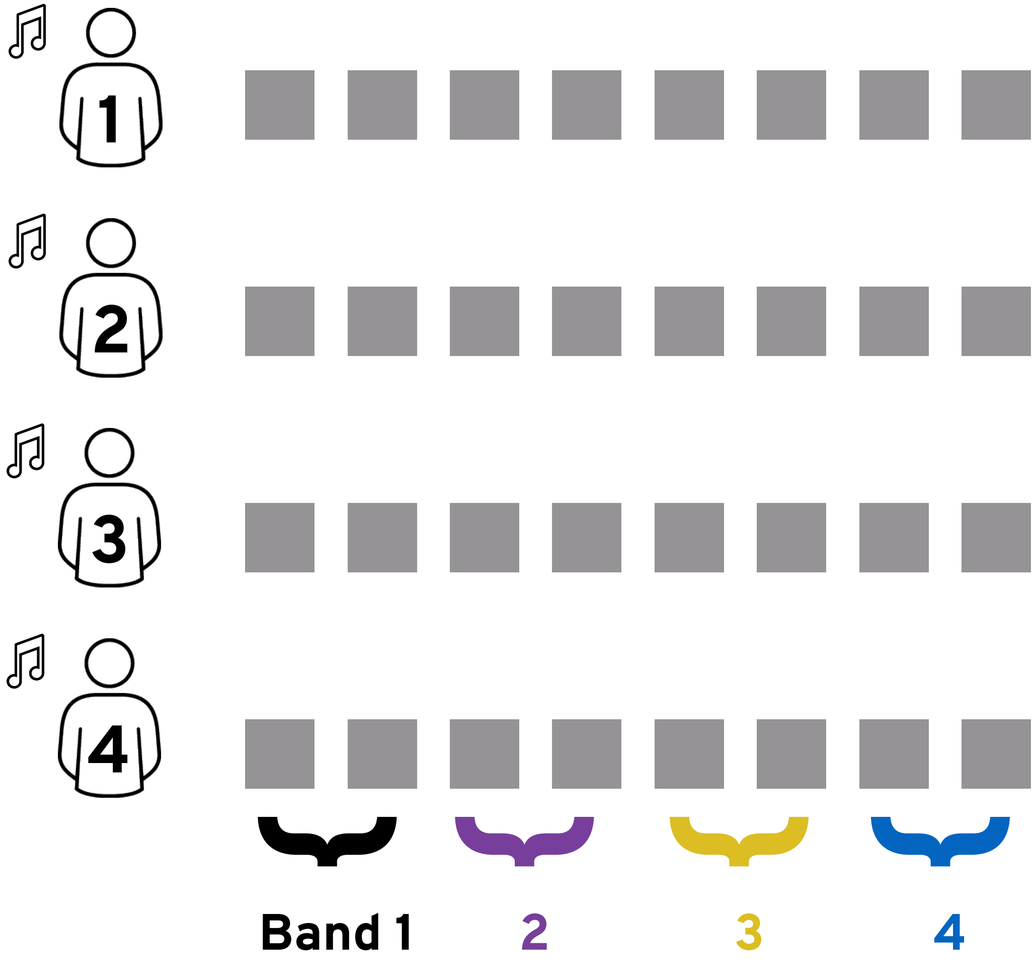
3

1

Candidate pairs:



Candidate pairs:



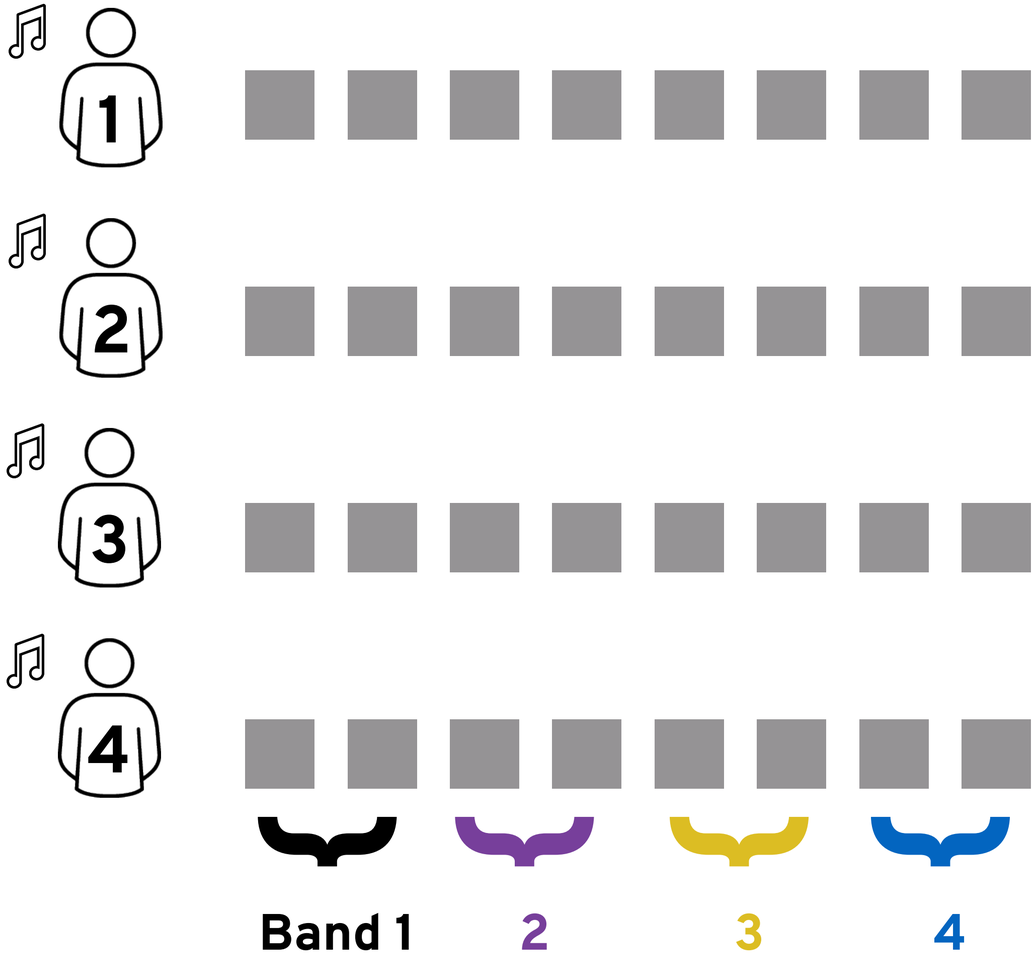
4

3

1

4

Candidate pairs:



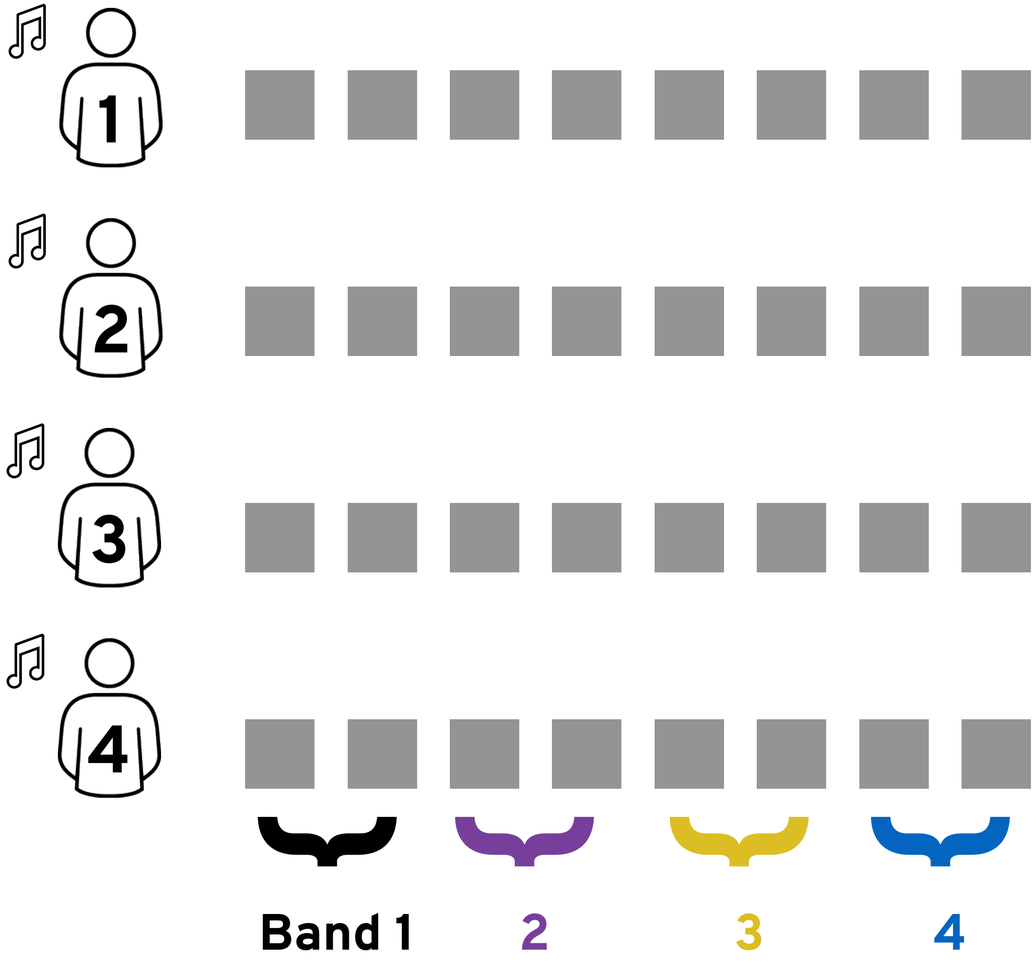
4

3

1

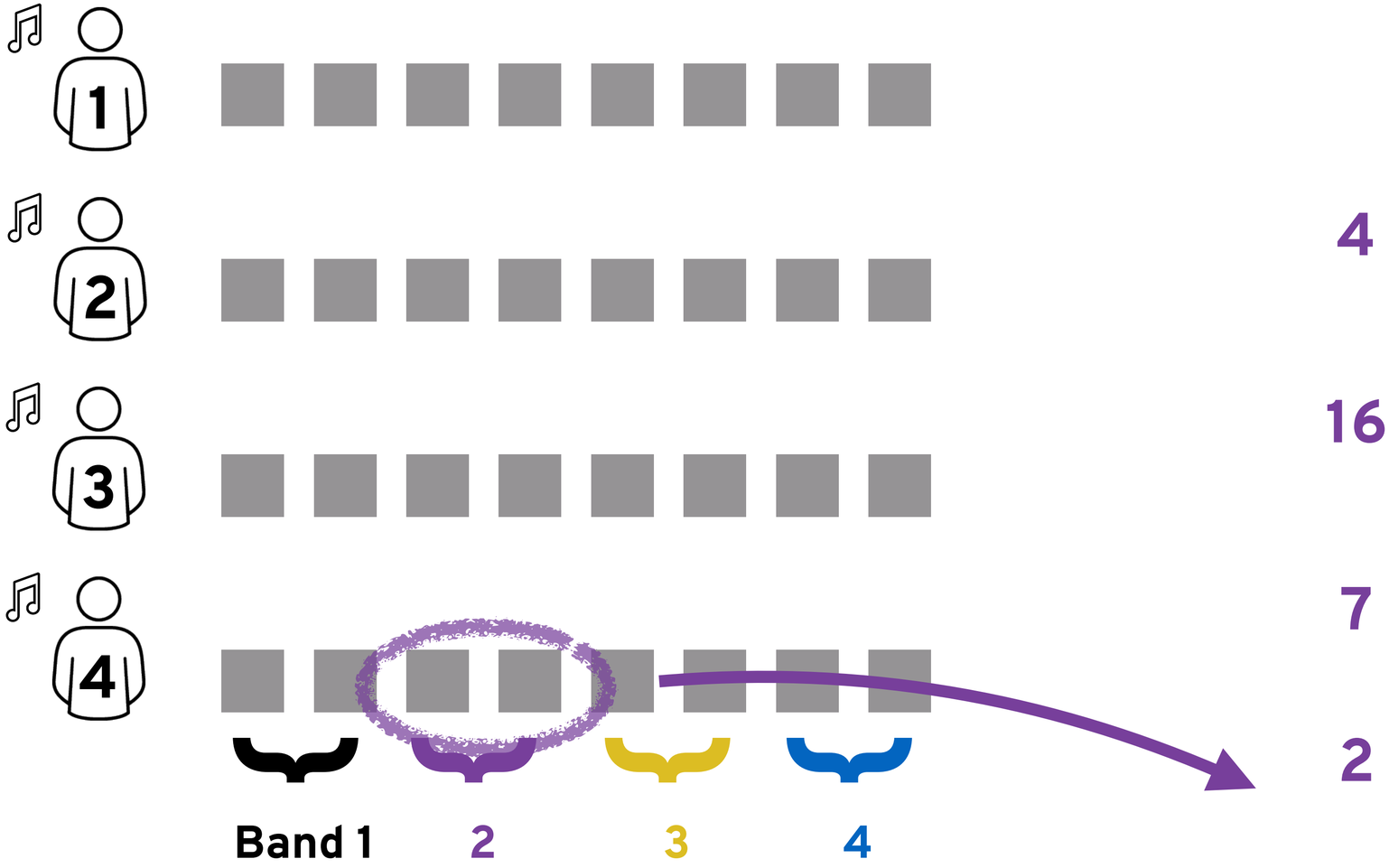
4

Candidate pairs: (1, 4)

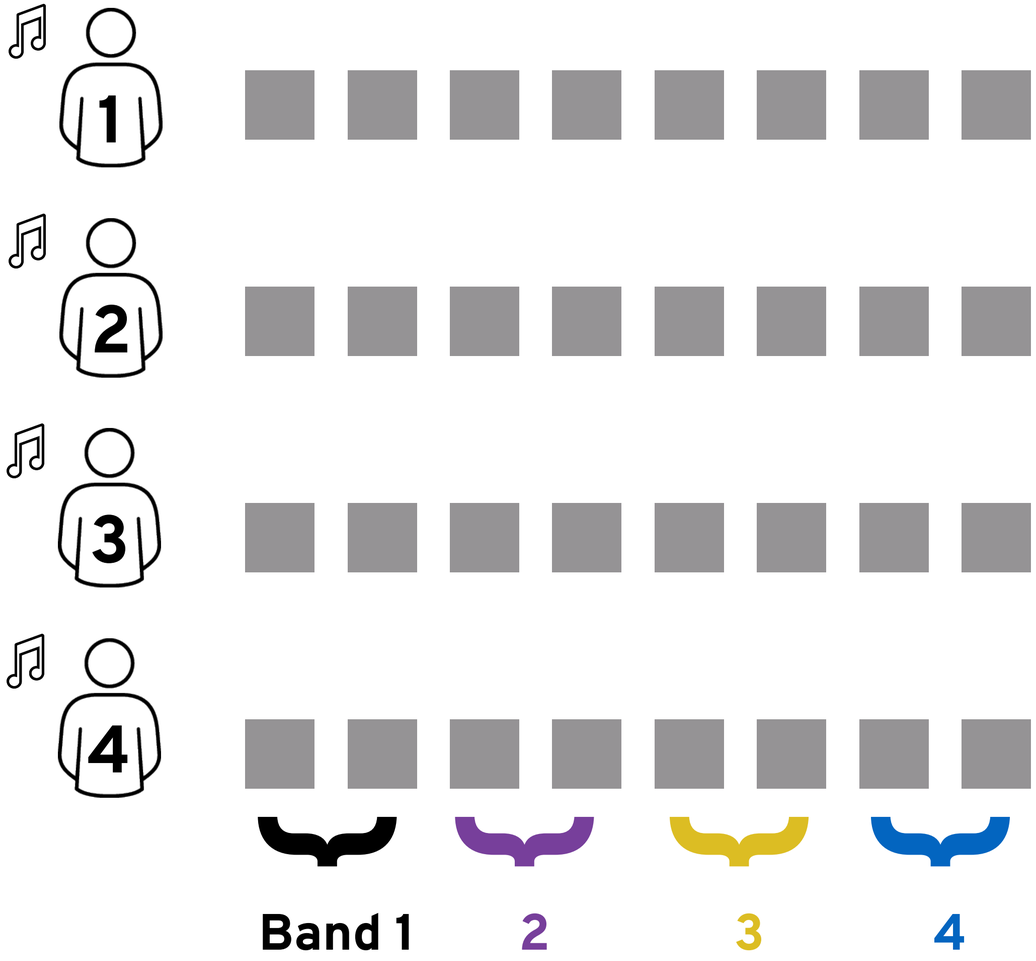


Candidate pairs: (1, 4)

Red Hat

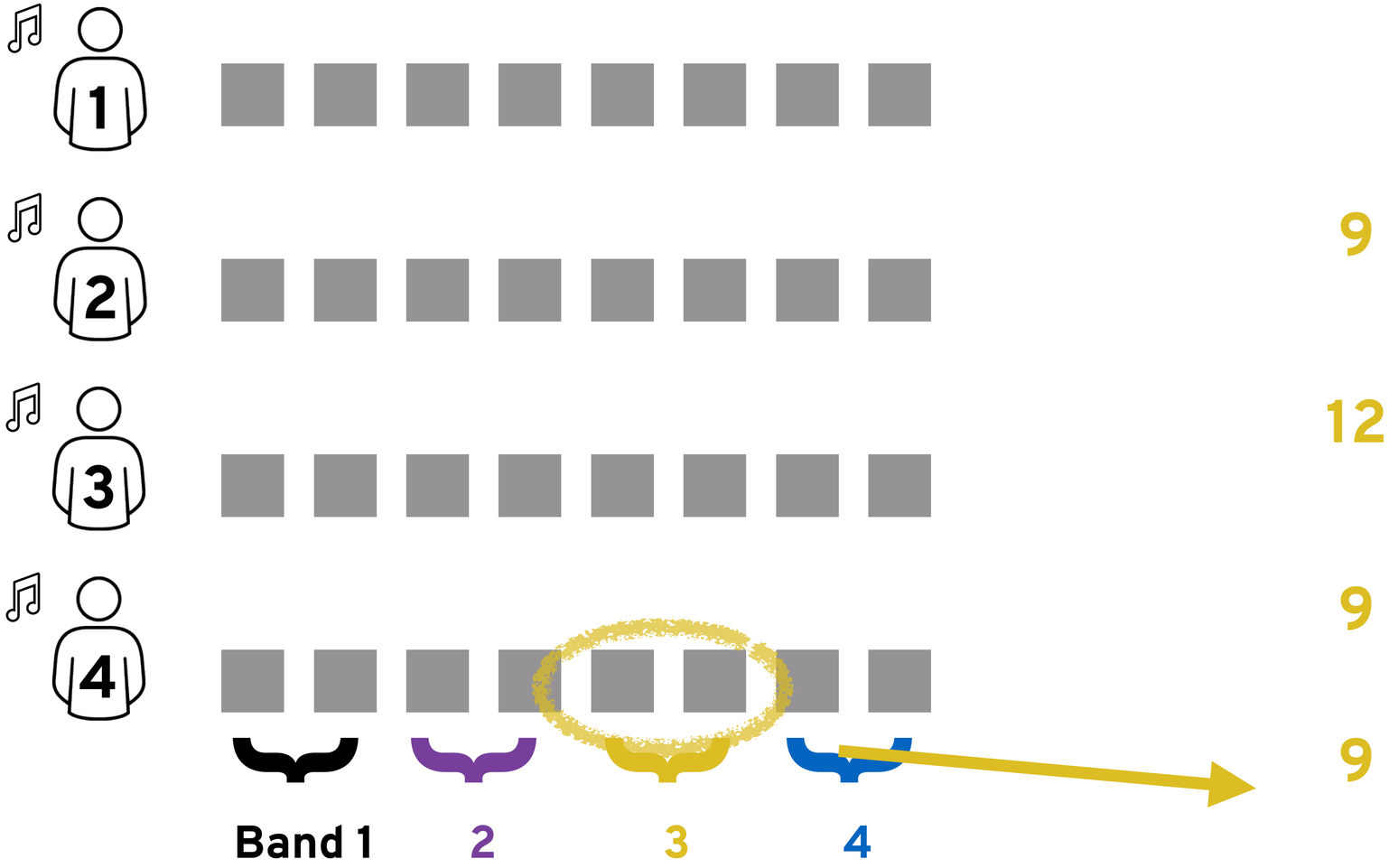


Candidate pairs: (1, 4)

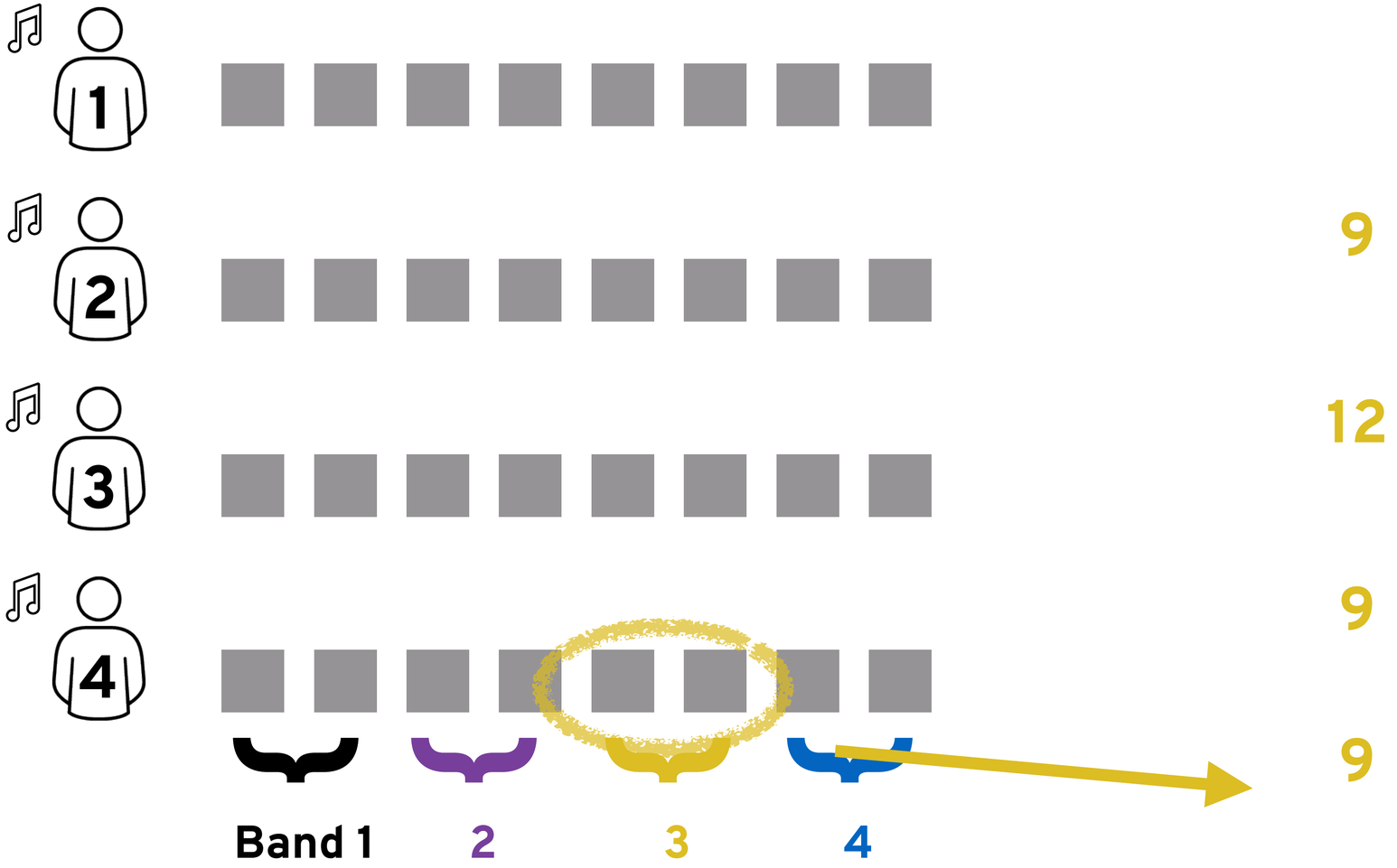


Candidate pairs: (1, 4)

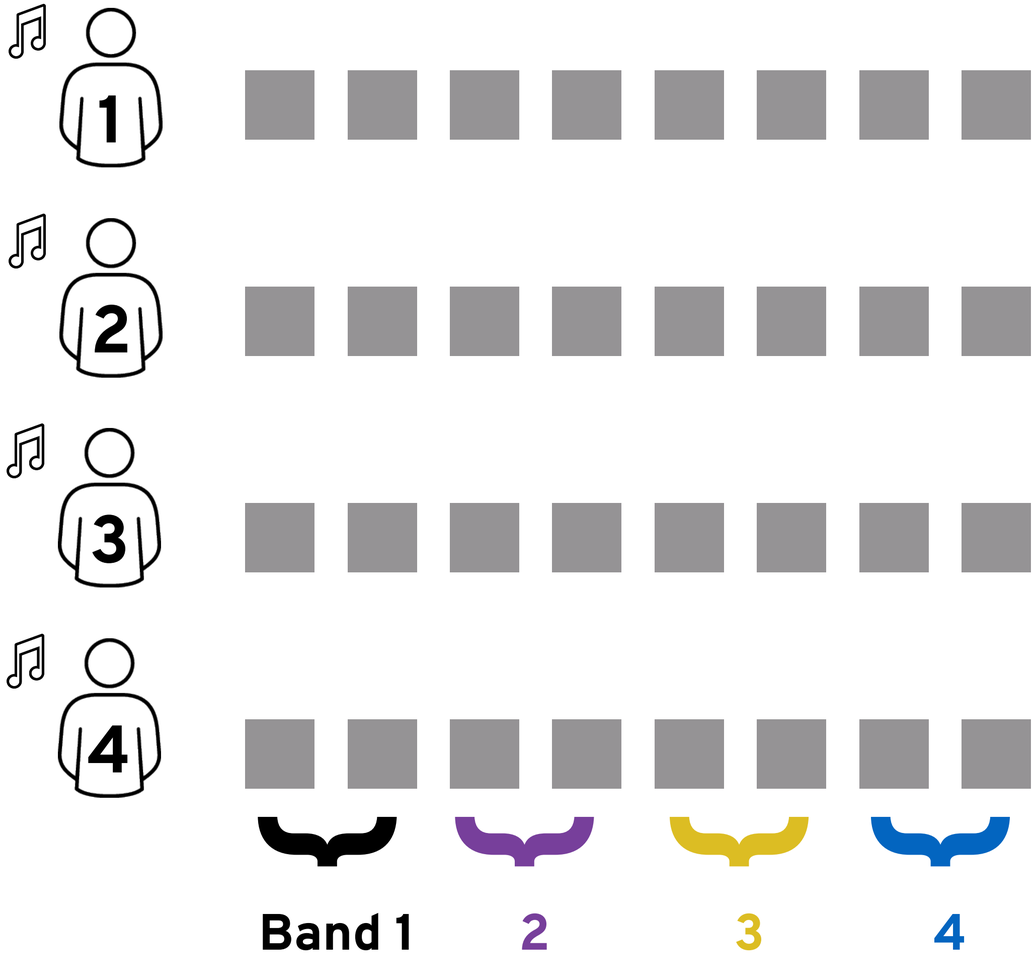
Red Hat



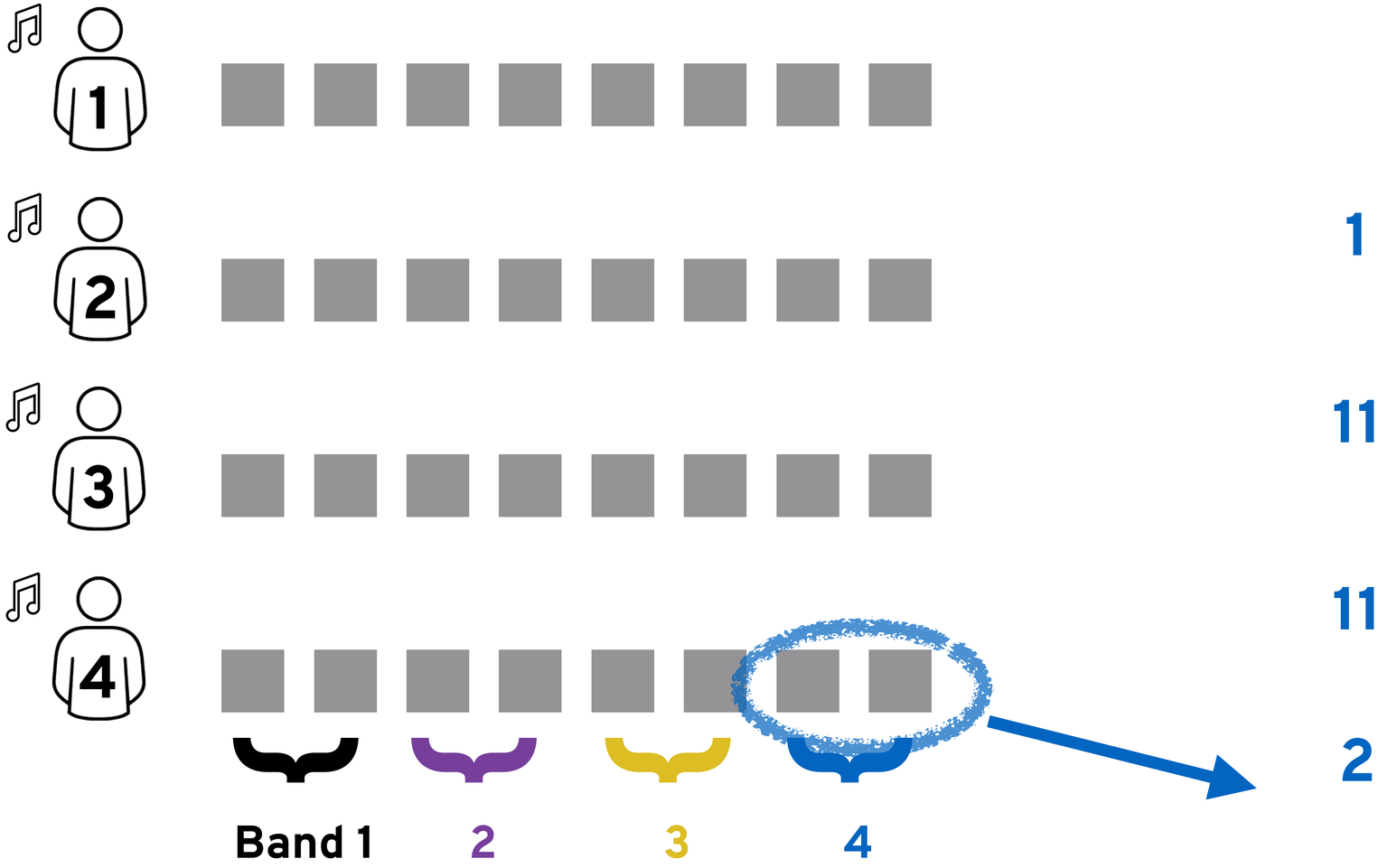
Candidate pairs: (1, 4)



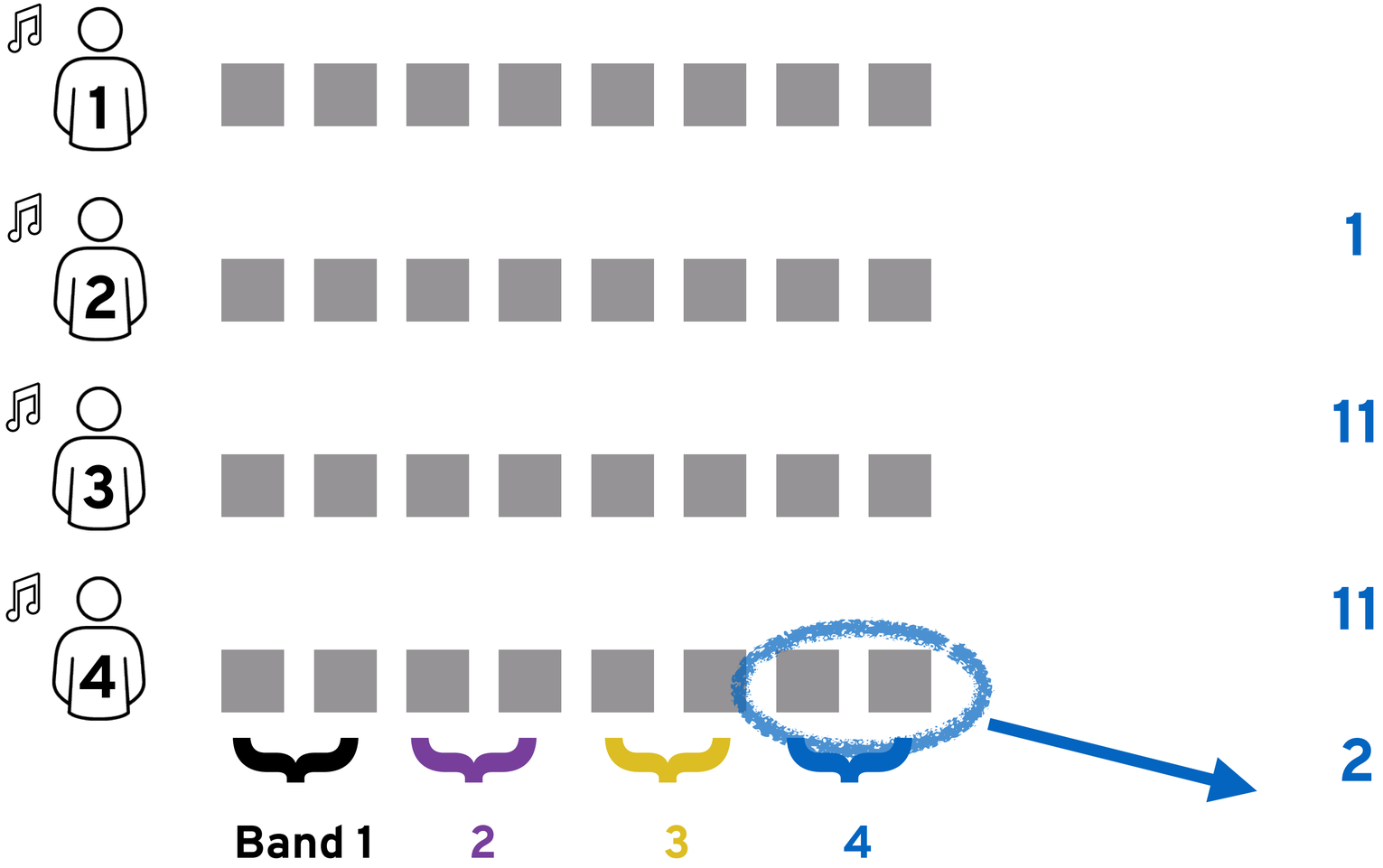
Candidate pairs: (1, 4) (1, 3) (3, 4)



Candidate pairs: (1, 4) (1, 3) (3, 4)

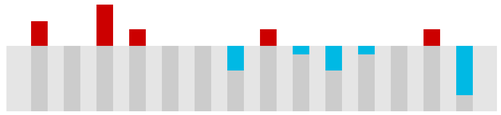


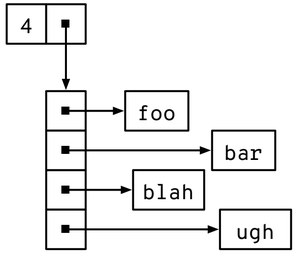
Candidate pairs: (1, 4) (1, 3) (3, 4)

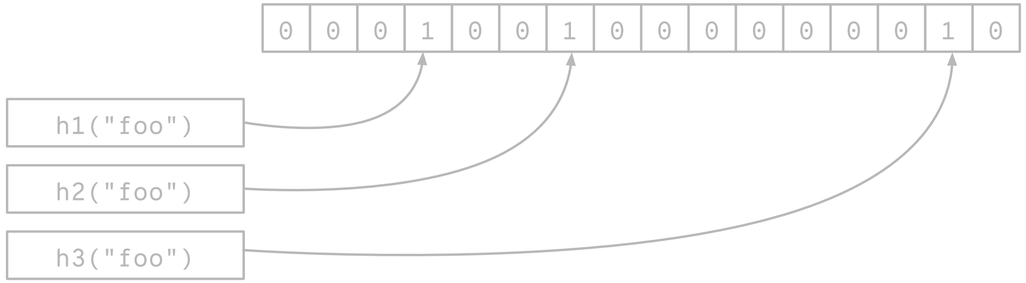


Candidate pairs: (1, 4) (1, 3) (3, 4) (2, 3)

What have we learned?



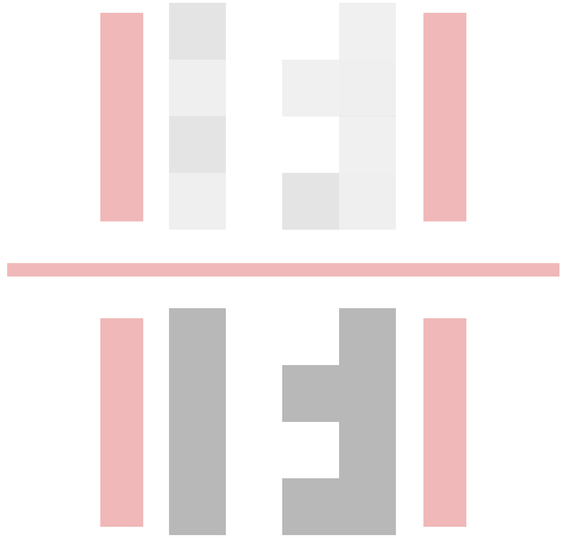




0	5	0	31	0	57	12	0	40	31	2	0	45	7	0	31
5	20	17	0	2	46	20	6	0	20	42	31	33	5	7	54
1	4	21	30	0	0	43	7	52	10	17	20	0	0	51	8

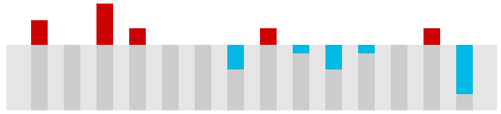
put("bar")

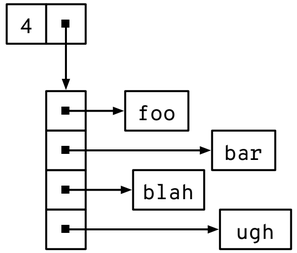
(42, goofy)	(20, foo)	(7, blah)	(2, ugh)	

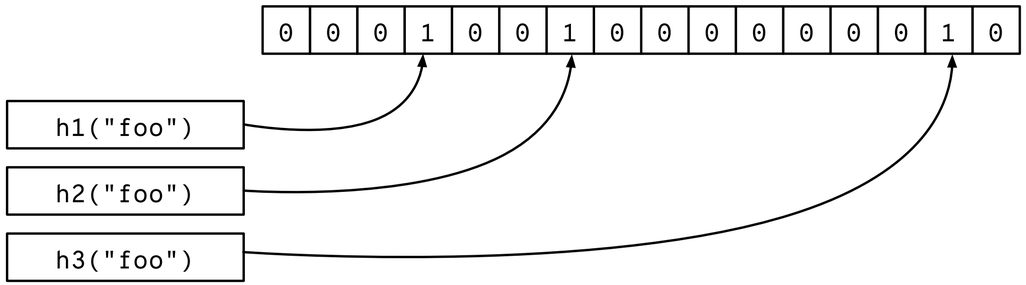


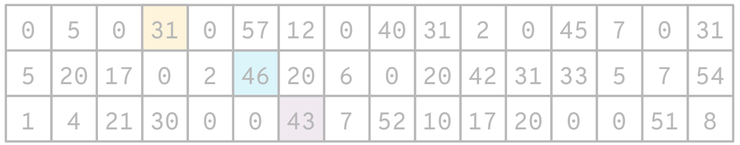
{sophie, willb}@redhat.com

@sophwats and @willb

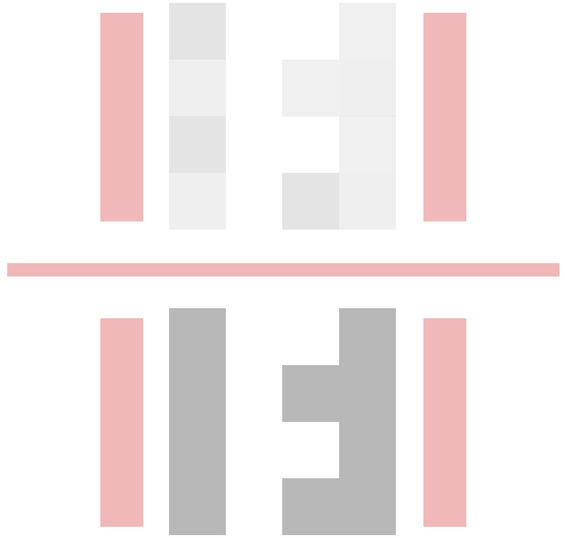






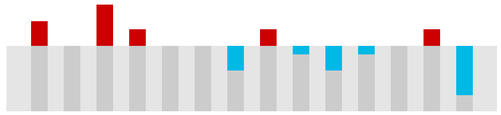


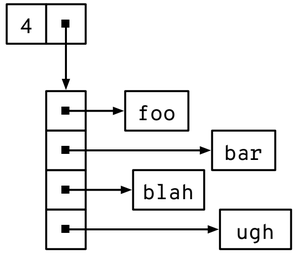
put("bar")

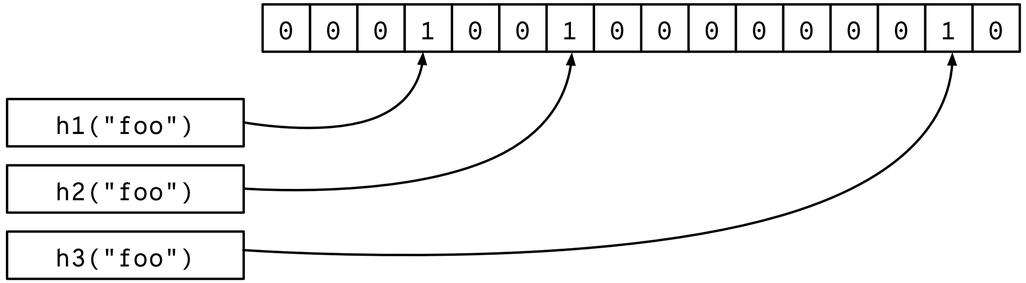


{sophie, willb}@redhat.com

@sophwats and @willb







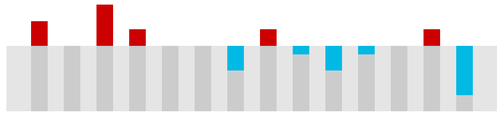
0	5	0	31	0	57	12	0	40	31	2	0	45	7	0	31
5	20	17	0	2	46	20	6	0	20	42	31	33	5	7	54
1	4	21	30	0	0	43	7	52	10	17	20	0	0	51	8

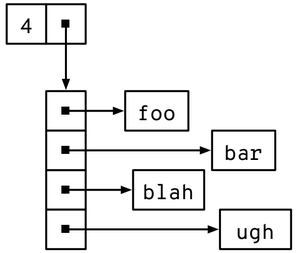
put("bar")

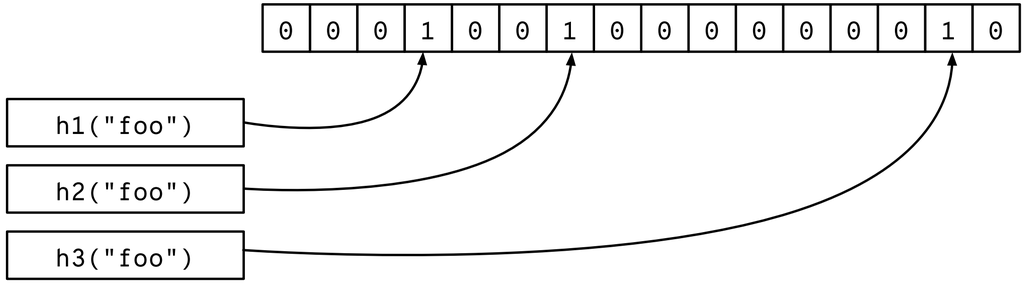
(42, goofy)	(20, foo)	(7, blah)	(2, ugh)	

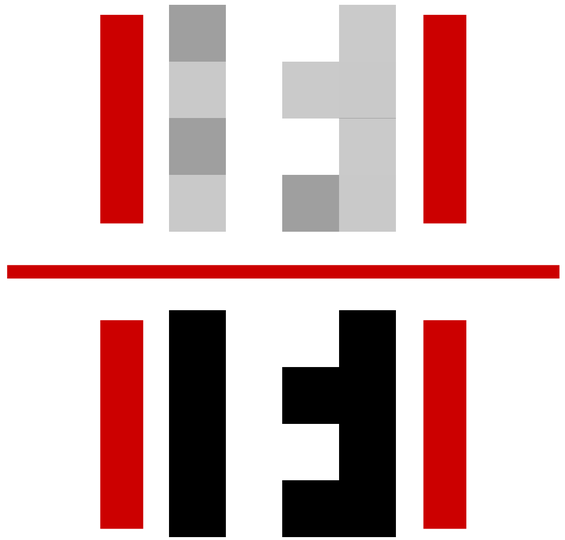
{sophie, willb}@redhat.com

@sophwats and @willb









0	5	0	31	0	57	12	0	40	31	2	0	45	7	0	31
5	20	17	0	2	46	20	6	0	20	42	31	33	5	7	54
1	4	21	30	0	0	43	7	52	10	17	20	0	0	51	8

put("bar")

(42, goofy)	(20, foo)	(7, blah)	(2, ugh)	

{sophie, willb}@redhat.com

@sophwats and @willb

LET'S GO TO THE NOTEBOOK

http://bit.ly/data-sketching-binder