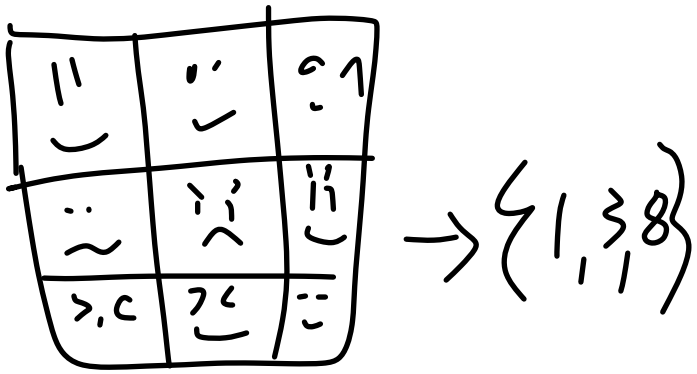
Algorithms and Data Structures for Data Science

MinHash Sketch

CS 277

Brad Solomon

April 24, 2024



UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science

Learning Objectives

Review bloom filters

Introduce the concept of cardinality and cardinality estimation

Demonstrate the relationship between cardinality and similarity

Introduce the MinHash Sketch for set similarity detection

Bloom Filters

A probabilistic data structure storing a set of values $h_{\{1,2,3,\ldots,k\}}$

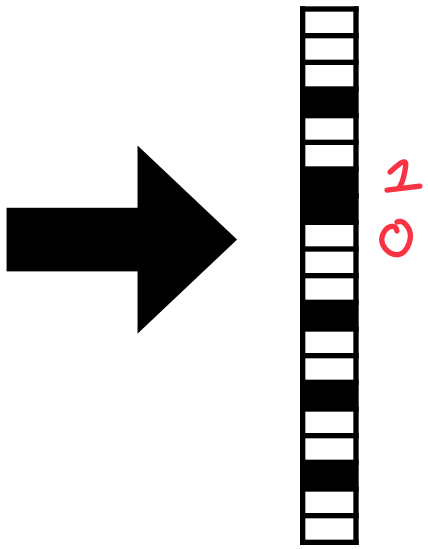
Has three key properties:

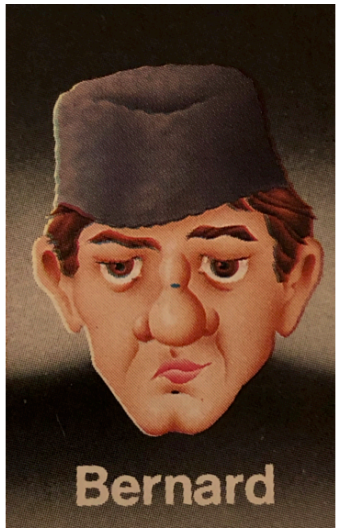
- k, number of hash functions
- n, expected number of insertions
- m, filter size in bits

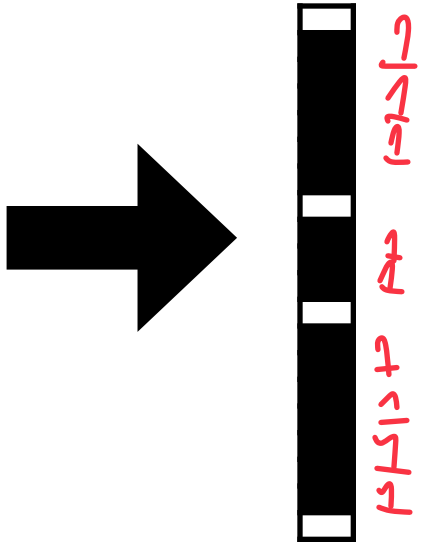
Expected false positive rate: $\left(1-\left(1-\frac{1}{m}\right)^{nk}\right)^{k} \approx \left(1-e^{\frac{-nk}{m}}\right)^{k}$

Optimal accuracy when: $k^{*} = \ln 2 \cdot \frac{m}{n}$

The hidden problem with (most) sketches…







m = BF size → pick based on size of human genome

"Human genes" + "Cancer genes"

Cardinality

Cardinality is a measure of how many unique items are in a set

2
4
9
3
7
9
7
8
5
6

↳ put in set & get length

↳ set allows no duplicates

↳ Hash table!

Applied Cardinalities

Cardinalities

$|A|$

$|B|$

$|A \cup B|$

$|A \cap B|$

Set similarities

$$O = \frac{|A \cap B|}{min(|A|, |B|)}$$

$$J = \frac{|A \cap B|}{|A \cup B|}$$

Real-world Meaning

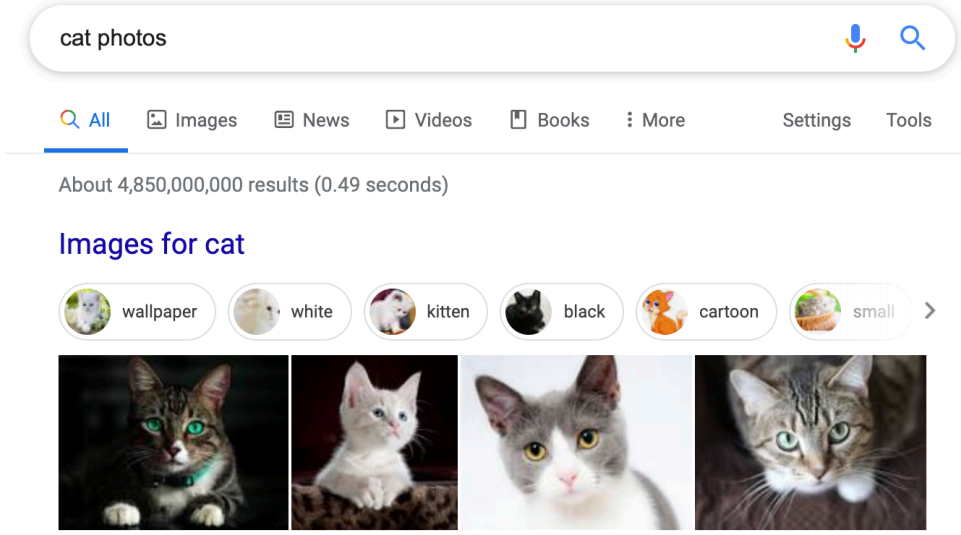
```
AGGCCACAGTGTATTATGACTG
|||||||||||  |||||||||
AGGCCACAGTGAGTTATGACTG

AAAAAAAAAAAGATGT-AAGTA
|||||||||||||||| |||||
AAAAAAAAAAAGATGTAAAGTA

GAGG--TCAGATTCACAGCCAC
||||  ||||||||||||||||
GAGGGGTCAGATTCACAGCCAC
```

Cardinality

Sometimes its not possible or realistic to count all objects!



Estimate: 60 billion — 130 trillion

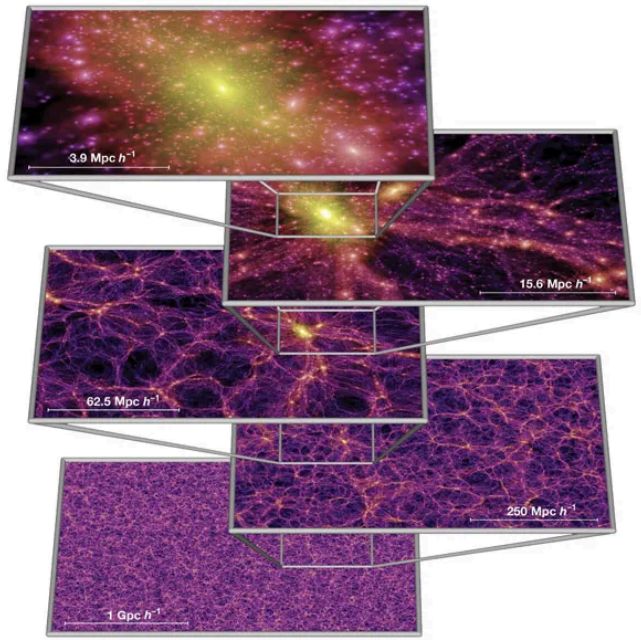


Image: https://doi.org/10.1038/nature03597

5581
8945
6145
8126
3887
8925
1246
8324
4549
9100
5598
8499
8970
3921
8575
4859
4960
42
6901
4336
9228
3317
399
6925
2660
2314

Applied Cardinalities

$$k^* = \ln 2 \cdot \frac{m}{n}$$

Given any two values, we can optimize the third

$n = 100$ items $\quad k = 3$ hashes $\quad m =$

$m = 100$ bits $\quad n = 20$ items $\quad k =$

$m = 100$ bits $\quad k = 2$ items $\quad n =$

Cardinality Estimation

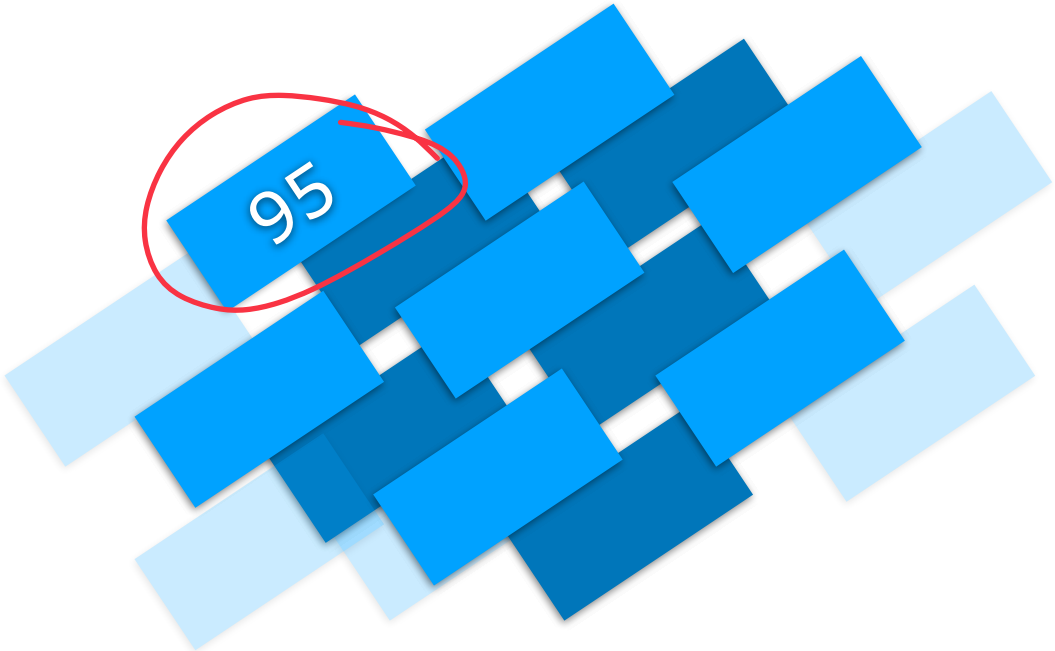
Imagine I fill a hat with numbered cards and draw one card out at random.

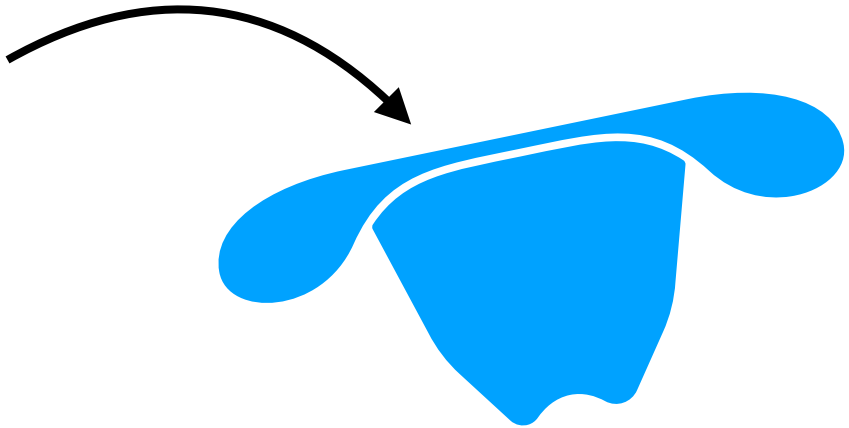
If I told you the value of the card was 95, what have we learned?

↳ 95 exists in dataset

↳ min card is at most 95

max least 95





Analogy from Ben Langmead

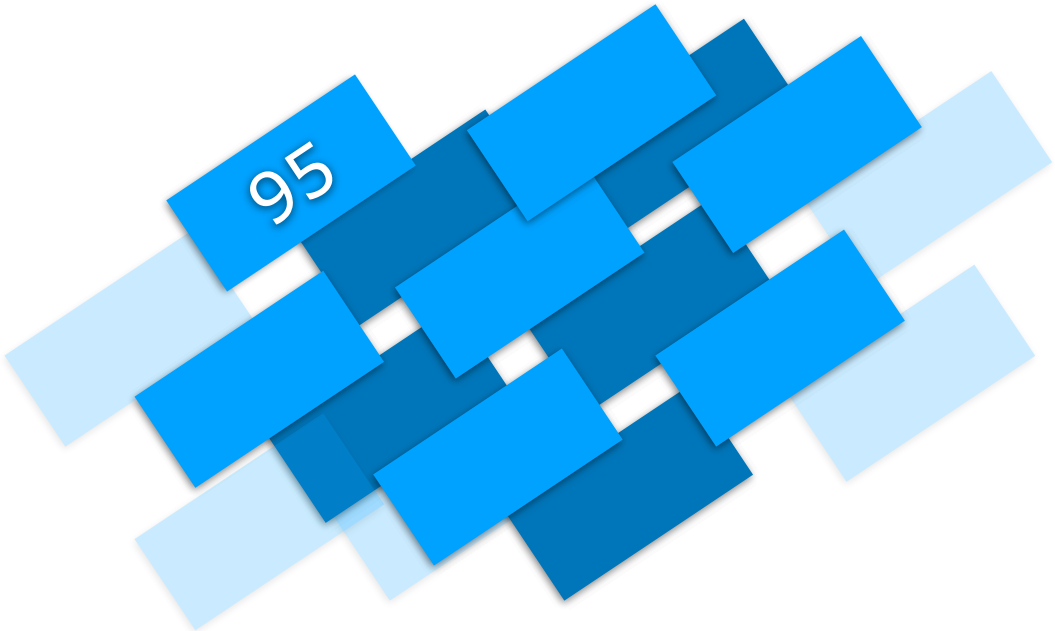
Cardinality Estimation

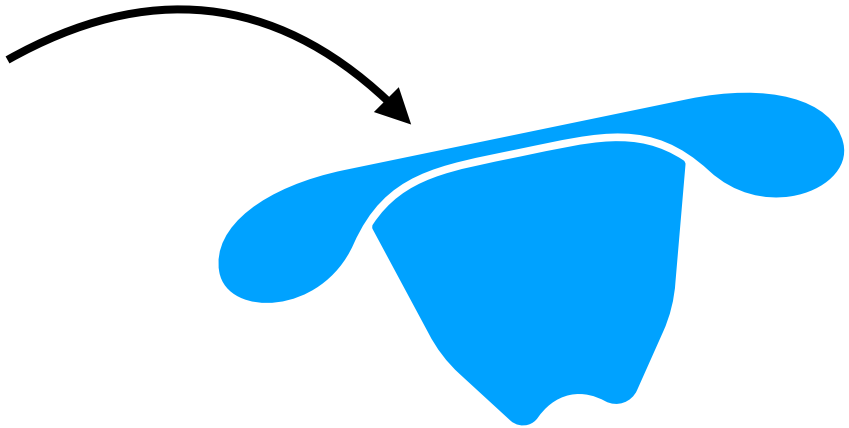
Universe

Imagine I fill a hat with **a random subset** of numbered cards **from 0 to 999**

If I told you that the **minimum** value was 95, what have we learned?

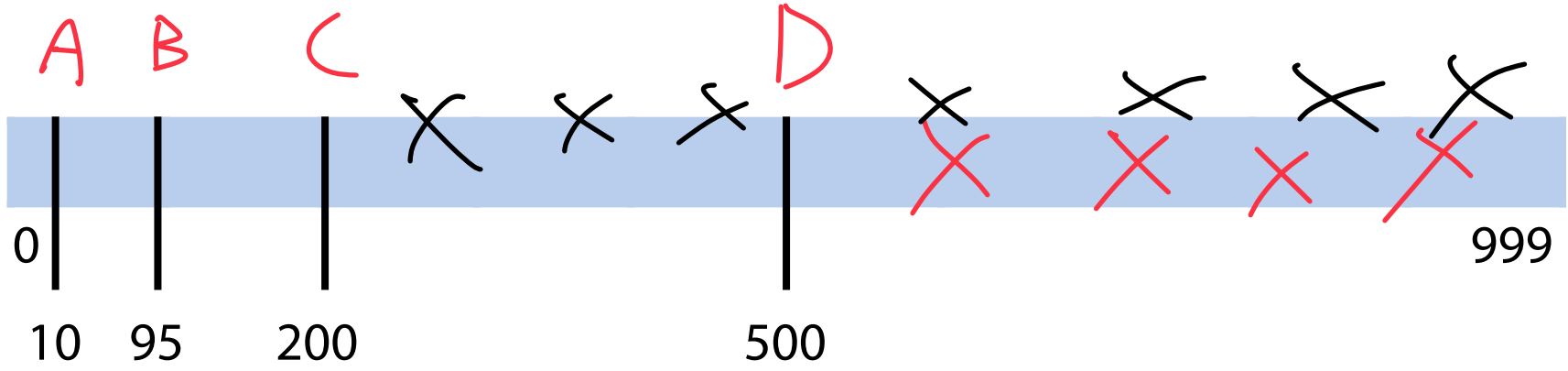
Claim: there are 10 cards in the hat.





Cardinality Estimation

Imagine we have multiple sets (multiple minimums).



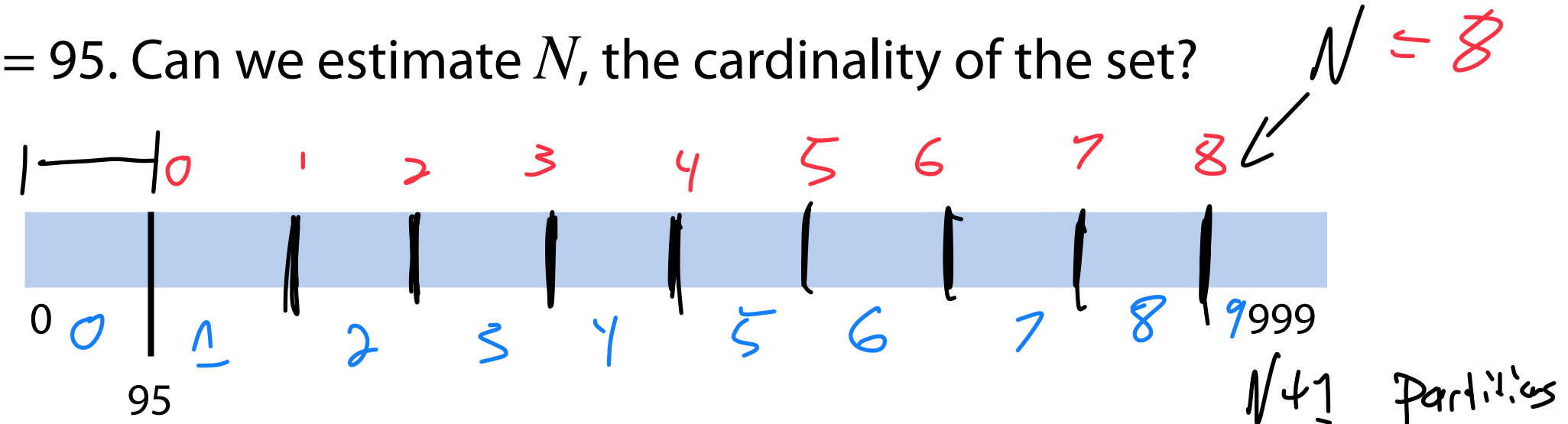
Guess: D has smallest # of cards

↳ each card had 50% chance of being < 500

↳ each card in C had 20% chance of being < 200

Cardinality Estimation

Let min = 95. Can we estimate N, the cardinality of the set?



$N = 8$

$N+1$ Partitions $= 9$

$95 \approx \frac{1000}{N+1}$

$N+1 = \frac{1000}{95}$

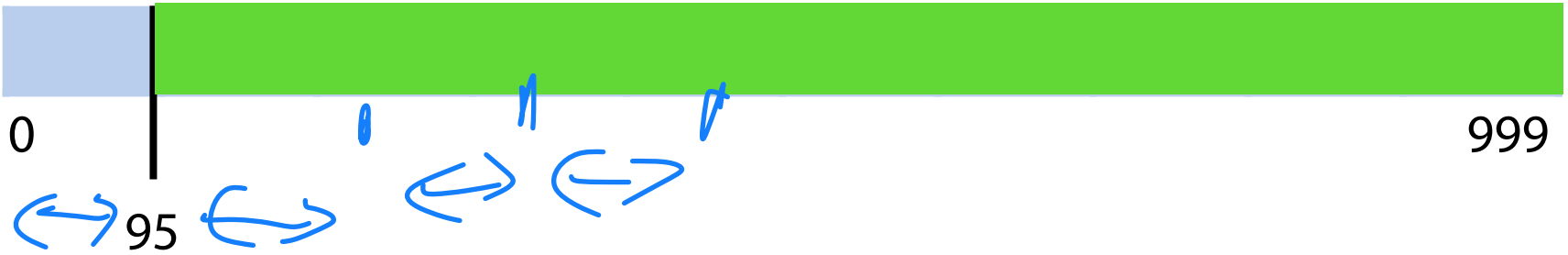
$N = \frac{1000}{95} - 1 = 10.5 - 1 = 9.5$

Cardinality Estimation

Let min = 95. Can we estimate N, the cardinality of the set?

Claim: $95 \approx \frac{1000}{(N+1)}$

Cardinality Estimation



Let min = 95. Can we estimate N, the cardinality of the set?

Conceptually: If we scatter N points randomly across the interval, we end up with $N + 1$ partitions, each about $1000/(N + 1)$ long

Assuming our first 'partition' is about average:

$$95 \approx 1000/(N+1)$$

$$N+1 \approx 10.5$$

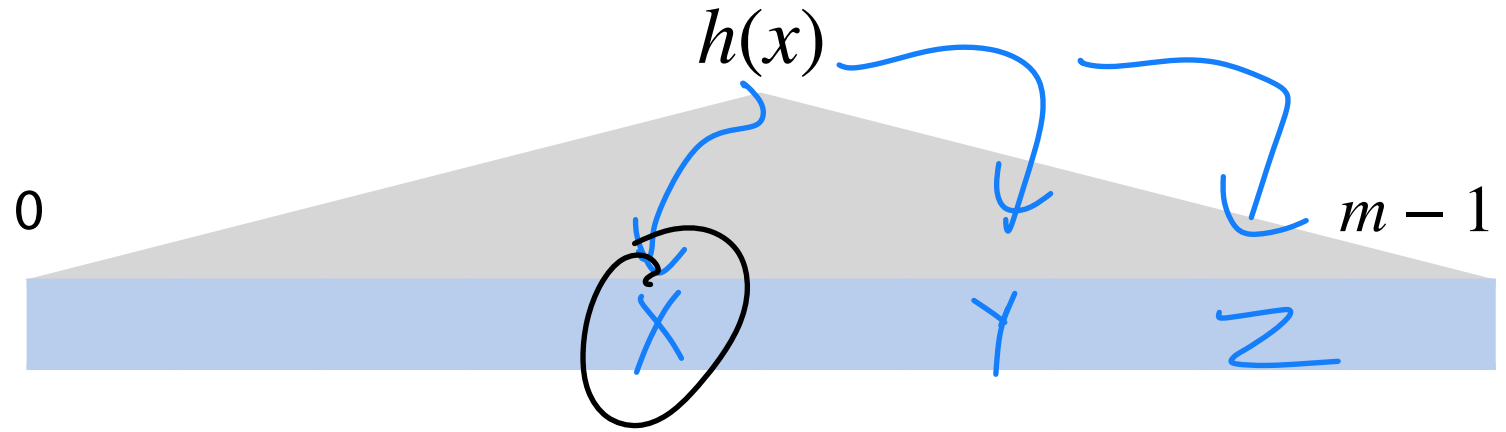
$$N \approx 9.5$$

Cardinality Estimation

Imagine we have a SUHA hash h over a range m.

Inserting a new value is equivalent to adding a card to our hat!

Tracking only the minimum value can let us estimate the cardinality!



Cardinality Estimation

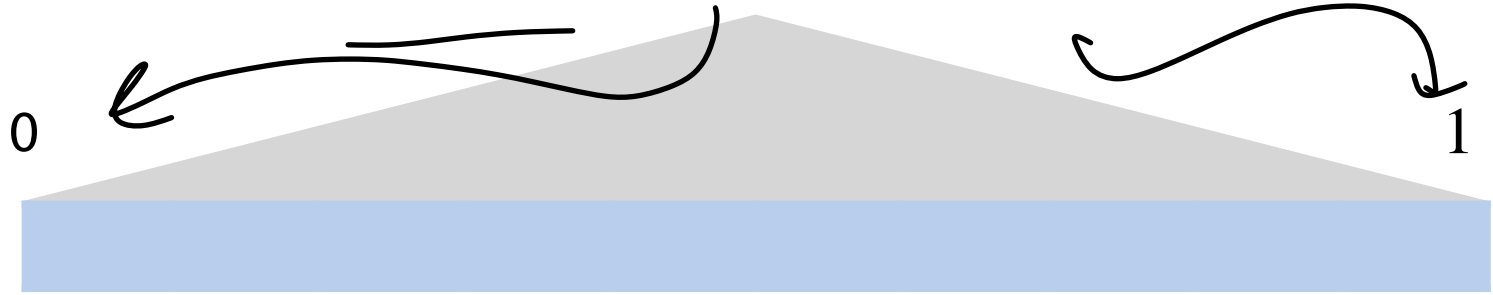
Imagine we have a SUHA hash h over a range m.

Inserting a new value is equivalent to adding a card to our hat!

Tracking only the minimum value can let us estimate the cardinality!

To make the math work out, lets normalize our hash…

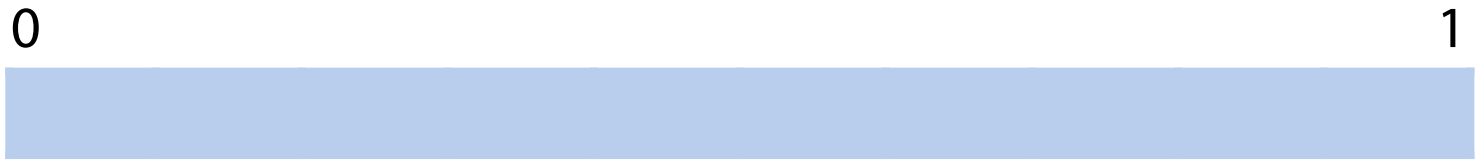
$$h'(x) = h(x) \ / \ (m - 1)$$



Cardinality Sketch

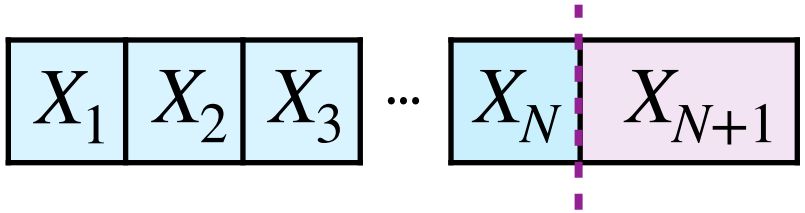
Let $M = min(X_1, X_2, \ldots, X_N)$ where each $X_i \in [0, 1]$ is an uniform independent random variable

Claim: $\mathbf{E}[M] = \frac{1}{N+1}$



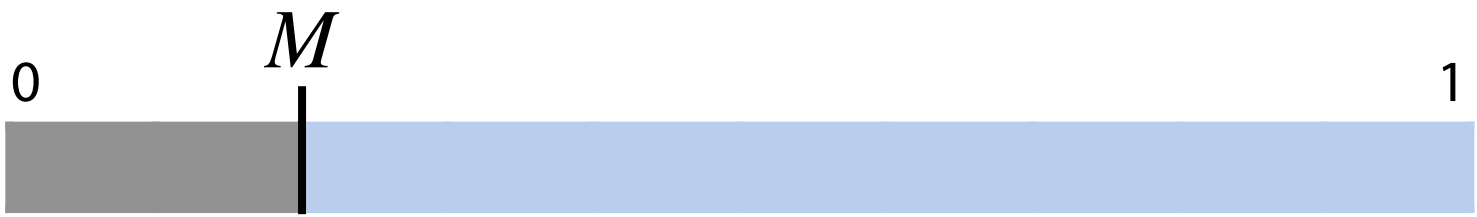
Cardinality Sketch

Consider an $N + 1$ draw:



$$M = \min_{1 \le i \le N} X_i$$

X_{N+1} can end up in one of two ranges:

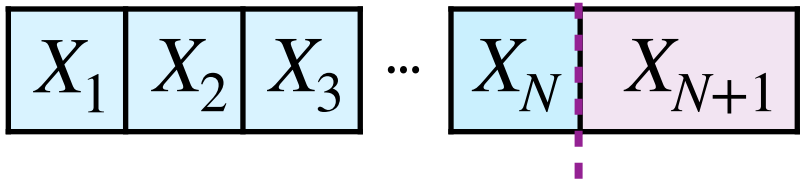


Cardinality Sketch

random indep

0 – 1

Consider an $N+1$ draw:



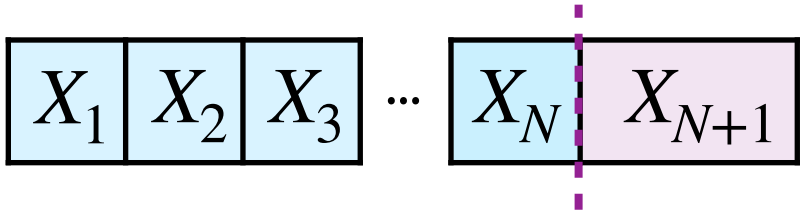
$$M = \min_{1 \le i \le N} X_i$$

X_{N+1} can end up in one of two ranges:

X_{N+1} will be the new minimum with probability M

Cardinality Sketch

Consider an $N+1$ draw:



$$M = \min_{1 \le i \le N} X_i$$

X_{N+1} can end up in one of two ranges:

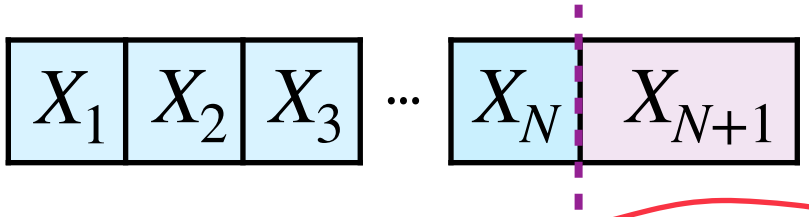
X_{N+1} will be the new minimum with probability M

X_{N+1} will not change minimum with probability $1 - M$

Prob sums to 1

Cardinality Sketch

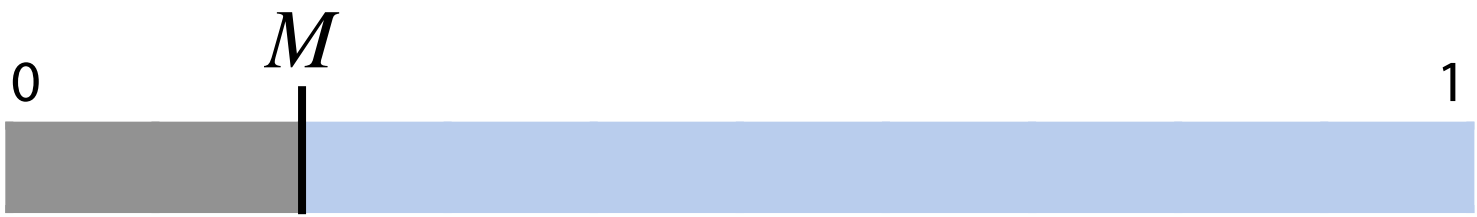
Consider an $N+1$ draw:



$$M = \min_{1 \le i \le N} X_i$$

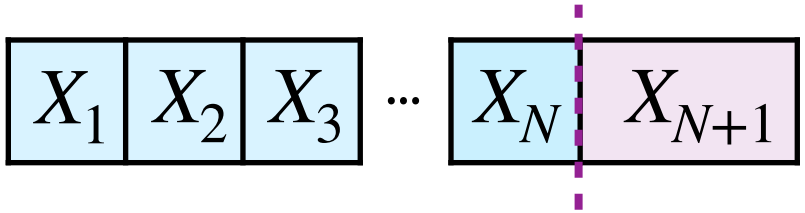
X_{N+1} will be the new minimum with probability M

By definition of SUHA, X_{N+1} has a $\frac{1}{N+1}$ chance of being smallest item



Cardinality Sketch

Consider an $N+1$ draw:

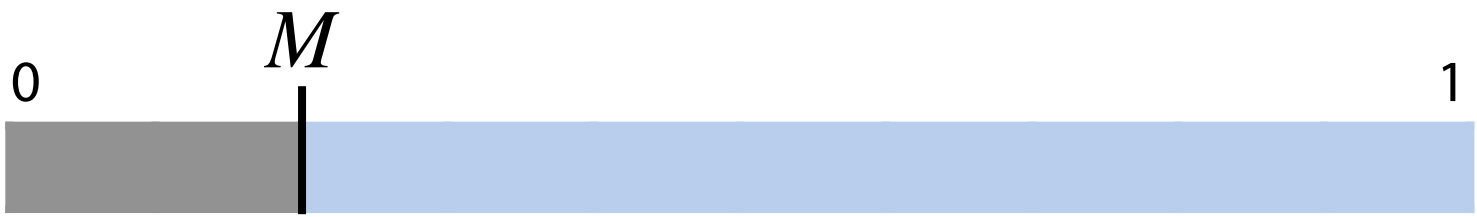


$$M = \min_{1 \le i \le N} X_i$$

X_{N+1} will be the new minimum with probability M

By definition of SUHA, X_{N+1} has a $\frac{1}{N+1}$ chance of being smallest item

Thus, $\mathbf{E}[M] = \frac{1}{N+1}$



Cardinality Sketch

Claim: $\mathbf{E}[M] = \dfrac{1}{N+1}$

$$N \approx \frac{1}{M} - 1$$

Ground truth $N = 5$

Attempt 1

0.962	0.328	0.771	0.952	0.923

$\frac{1}{0.328} - 1 \approx 2$

Attempt 2

0.253	0.839	0.327	0.655	0.491

$N \approx 3$

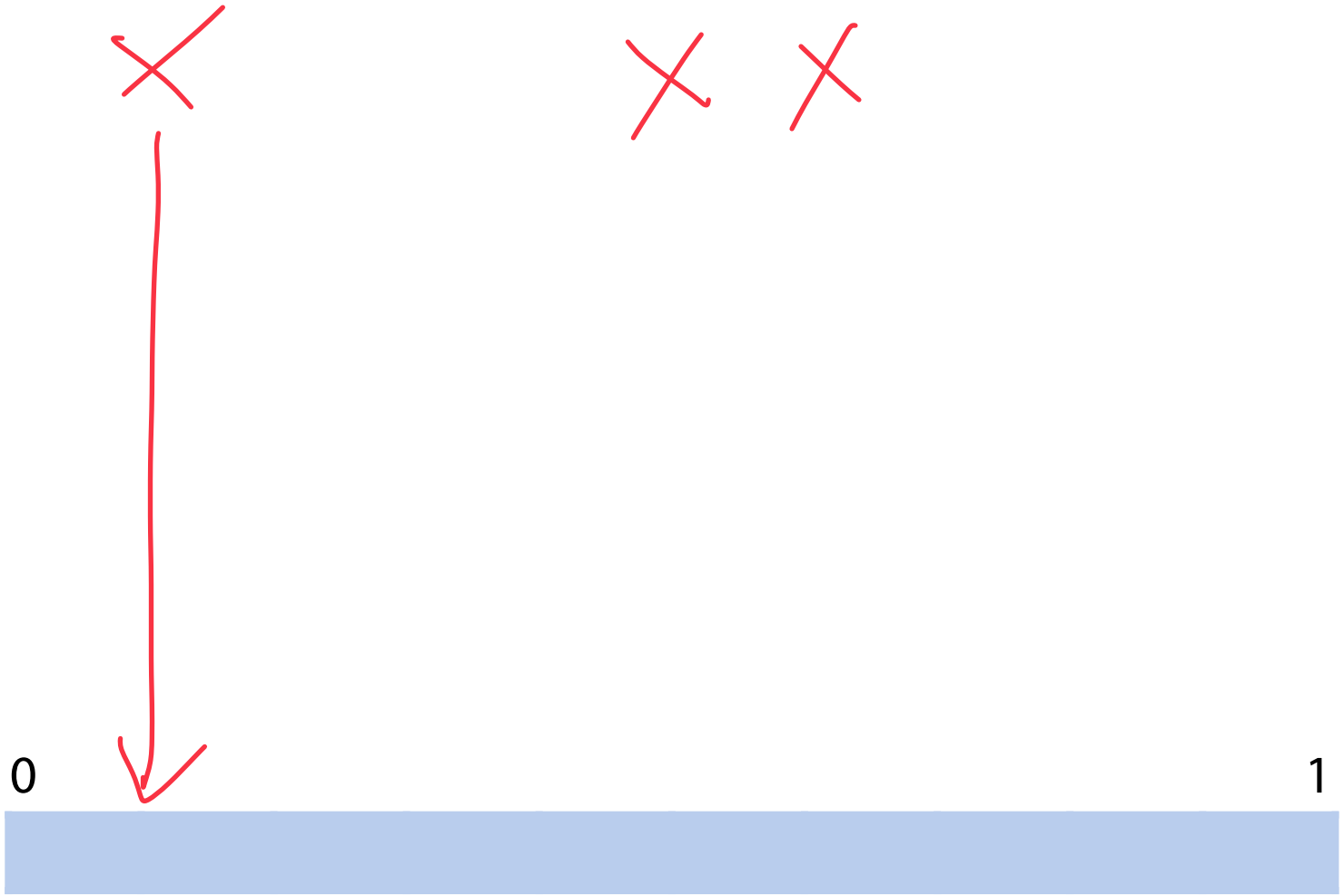
Attempt 3

0.134	0.580	0.364	0.743	0.931

$N \approx 6.5$

Cardinality Sketch

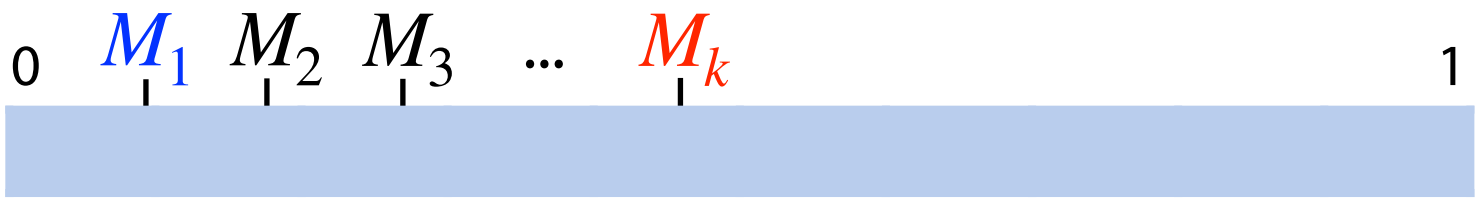
The minimum hash is a valid sketch of a dataset but can we do better?



Cardinality Sketch

Claim: Taking the k^{th}-smallest hash value is a better sketch!

Claim: $\mathbf{E}[M_k] = \frac{k}{N+1}$

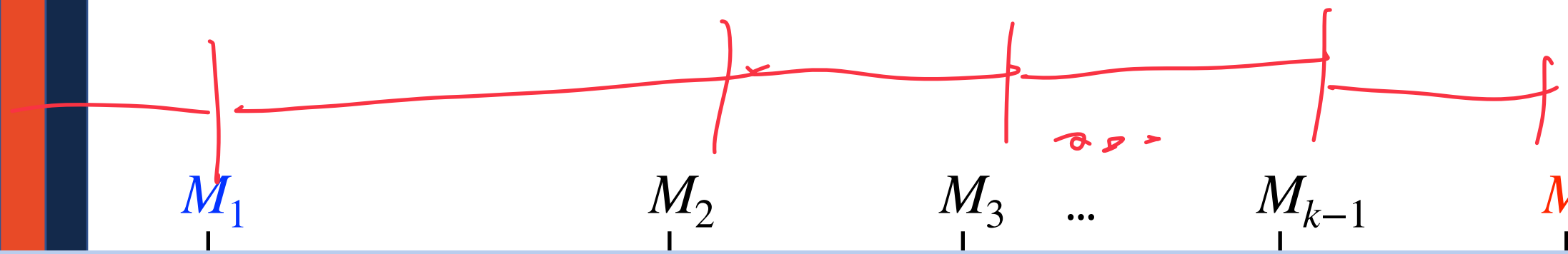


Cardinality Sketch

Claim: Taking the k^{th}-smallest hash value is a better sketch!

Claim: $$\frac{\mathbf{E}[M_k]}{k} = \frac{1}{N+1}$$

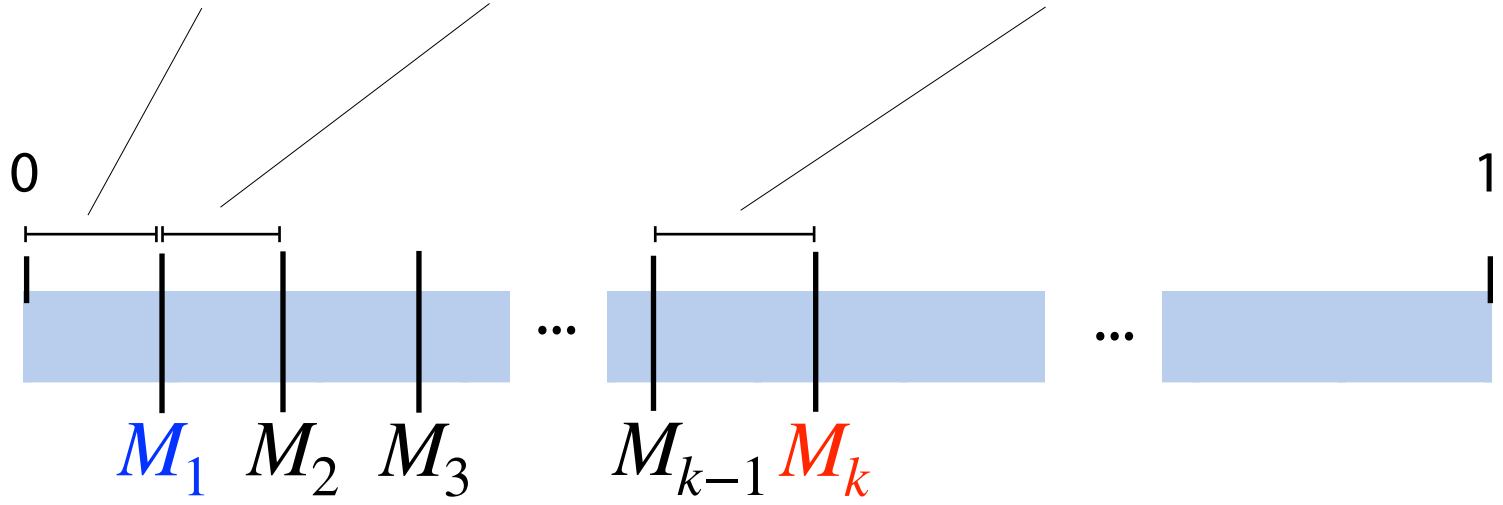
$$= \left[\mathbf{E}[M_1] + (\mathbf{E}[M_2] - \mathbf{E}[M_1]) + \ldots + (\mathbf{E}[M_k] - \mathbf{E}[M_{k-1}])\right] \cdot \frac{1}{k}$$



Cardinality Sketch

$$\frac{1}{N+1} = \frac{\mathbf{E}[M_k]}{k}$$

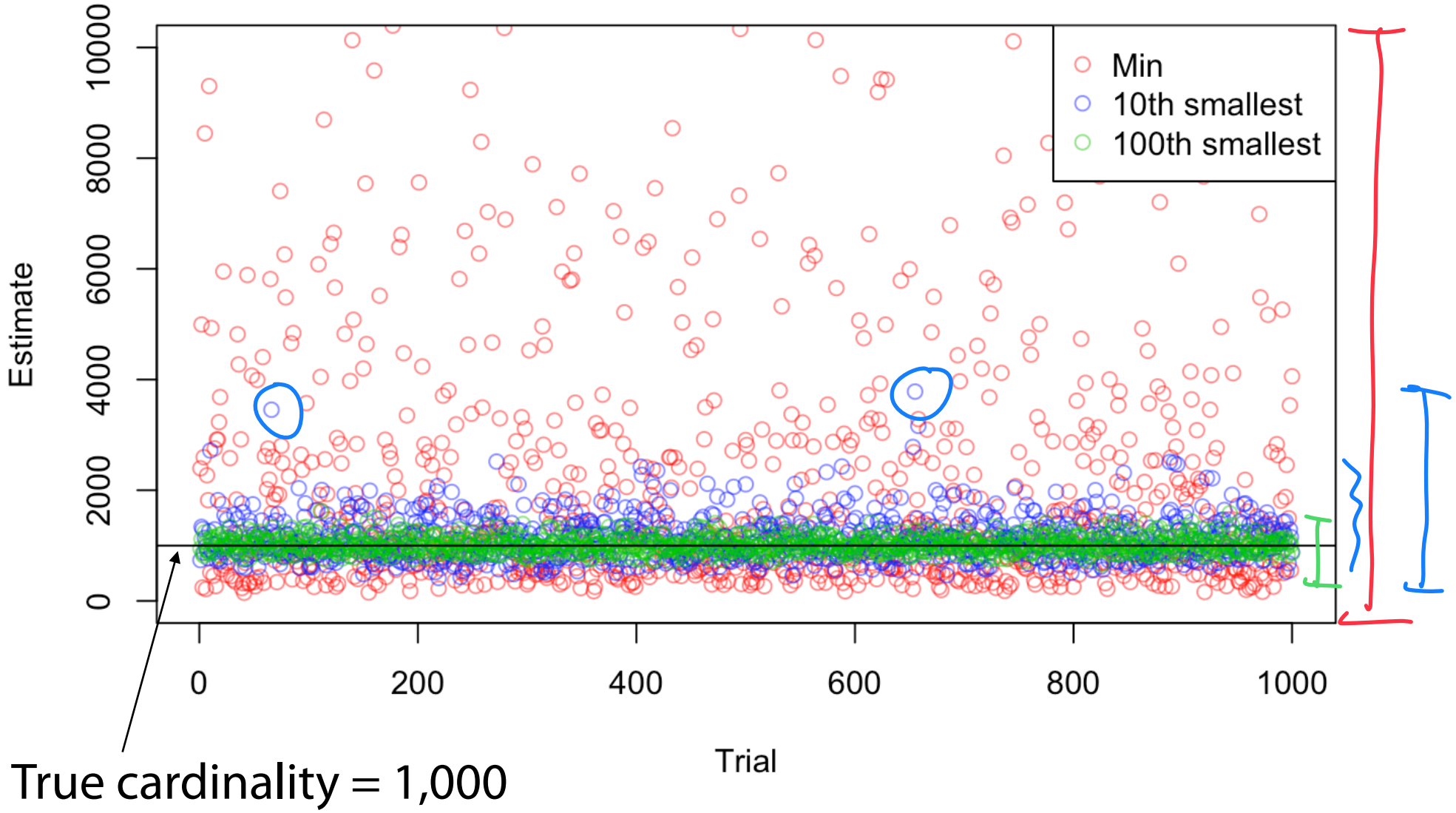
$$= \left[\mathbf{E}[M_1] + (\mathbf{E}[M_2] - \mathbf{E}[M_1]) + \ldots + (\mathbf{E}[M_k] - \mathbf{E}[M_{k-1}])\right] \cdot \frac{1}{k}$$



k^{th} minimum value (KMV)

Averages k estimates for $\frac{1}{N+1}$

Cardinality Sketch



Cardinality Sketch

Given any dataset and a SUHA hash function, we can **estimate the number of unique items** by tracking the **k-th minimum hash value**.

Hash

0.253	0.839	0.327	0.655	0.491

5

1st smallest $\frac{1}{.253} - 1 \approx 5$

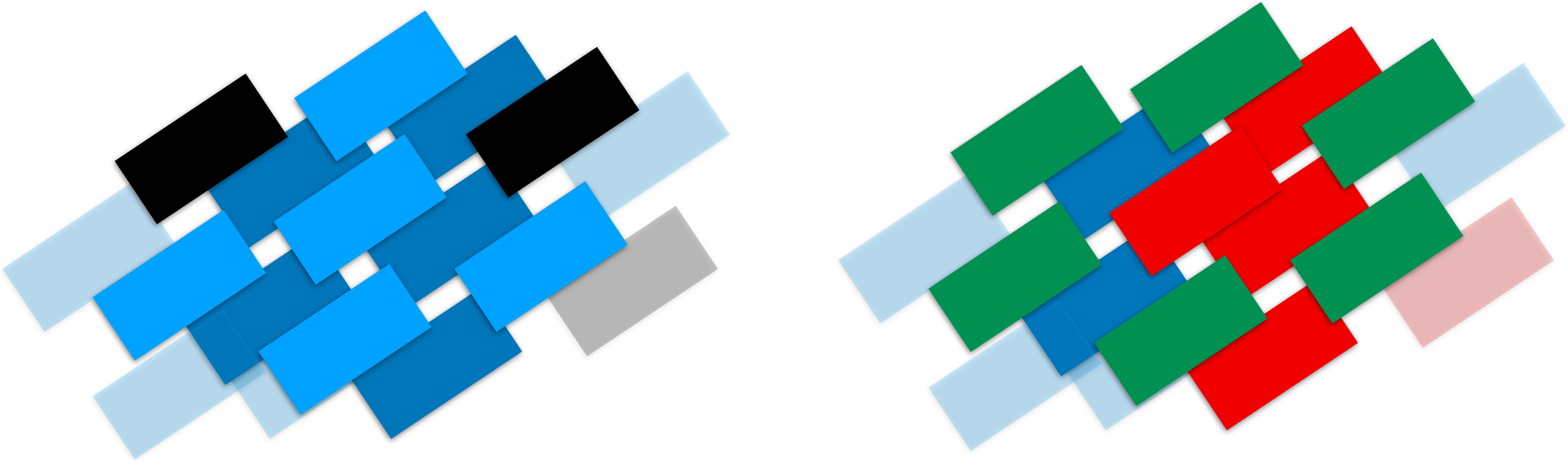
2nd smallest $\frac{2}{.327} - 1 \approx 5.1$

3rd $\frac{3}{.491} - 1 \approx 5.1$

MinHash Sketch

The **k-th minimum value sketch** is built by tracking k minima but only uses one value (the k-th minima) to get **cardinality!**

We can extend this approach into a full **MinHash sketch** that can also estimate **set similarities.**



Set "Data Structure"

A **set** stores an unordered collection of objects with no duplicates

Genome assembly databases are growing rapidly. The sequence content in each new assembly can be largely redundant with previous ones, but this is neither conceptually nor algorithmically easy to measure. We propose new methods and a new tool called DandD that addresses the question of how much new sequence is gained when a sequence collection grows. DandD can describe how much human structural variation is being discovered in each new human genome assembly and when discoveries will level off in the future. DandD uses a measure called δ ("delta"), developed initially for data compression. Computing δ directly requires counting k-mers, but DandD can rapidly estimate it using genomic sketches. We also propose δ as an alternative to k-mer-specific cardinalities when computing the Jaccard coefficient, avoiding the pitfalls of a poor choice of k. We demonstrate the utility of DandD's functions for estimating δ, characterizing the rate of pangenome growth, and computing allpairs similarities using k-independent Jaccard. DandD is open source software available at: https://github.com/jessicabonnie/dandd.

Sets in Python

A set can be initialized with a list or a tuple

```
mySet = set(<list>)
```

Add(x) adds object x to the set; it does nothing if x is already present

```
mySet = set()

mySet.add(<value>)
```

Remove(x) removes the object x. It will crash if x doesn't exist

```
mySet = set()

mySet.add(<value>)
```

Sets in Python: Data Access

Sets have no indices (no order of objects). We can only access by:

1. Looping through a set for each element

2. Looking up a specific element in our set

```
 1 mySet = set([1,2,3,4,5])
 2
 3
 4
 5 for obj in mySet:
 6     print(obj)
 7
 8
 9 print(10 in mySet)
10
11
12
13
```

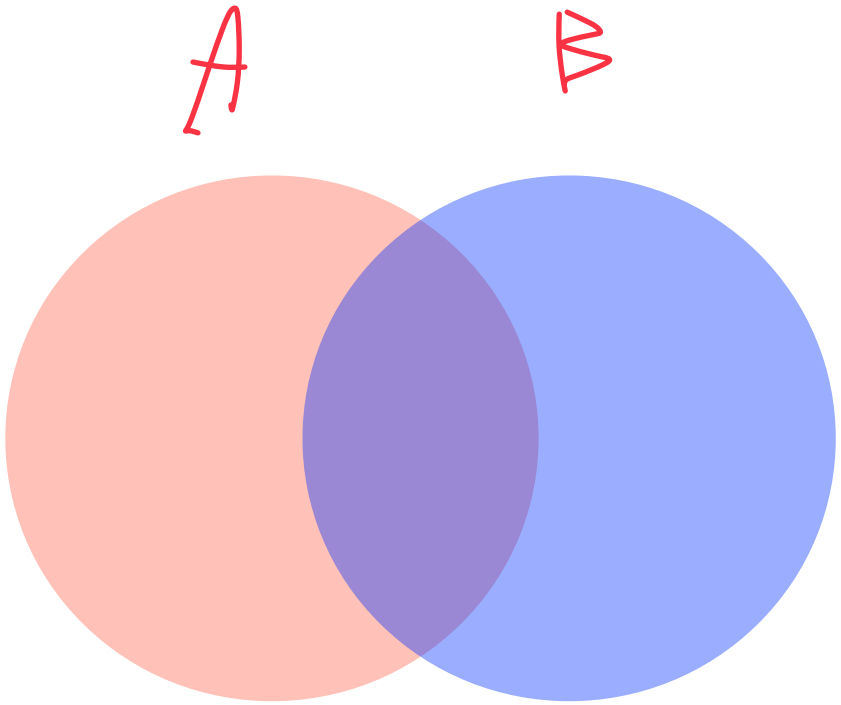
Set Operations

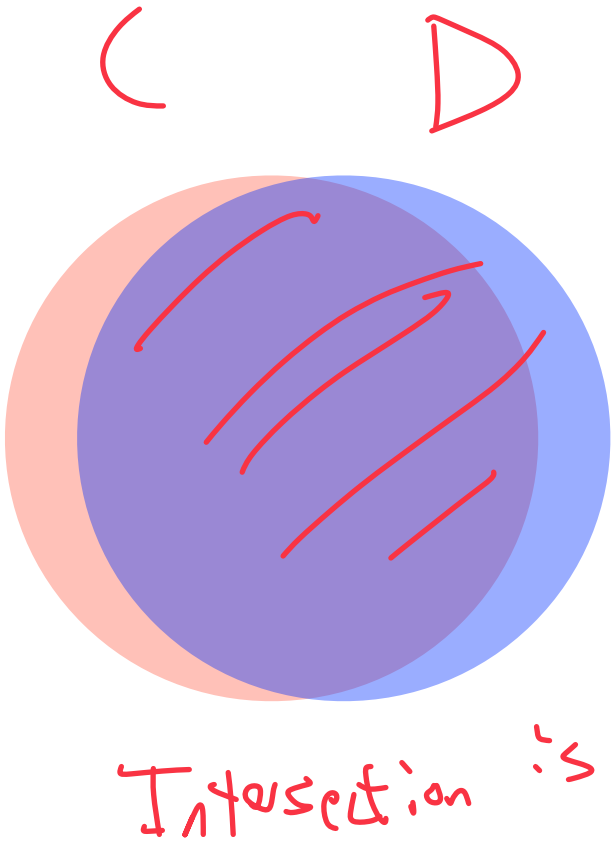
$A = \{1, 2, 3, 4\}$ $B = \{3, 4, 5, 6, 7\}$

Union	$A \cup B$	1, 2, 3, 4, 5, 6, 7
Intersection	$A \cap B$	3, 4
Difference	$A \setminus B$	1, 2
Symmetric difference	$A \triangle B$	1, 2, 5, 6, 7

Set Similarity Review

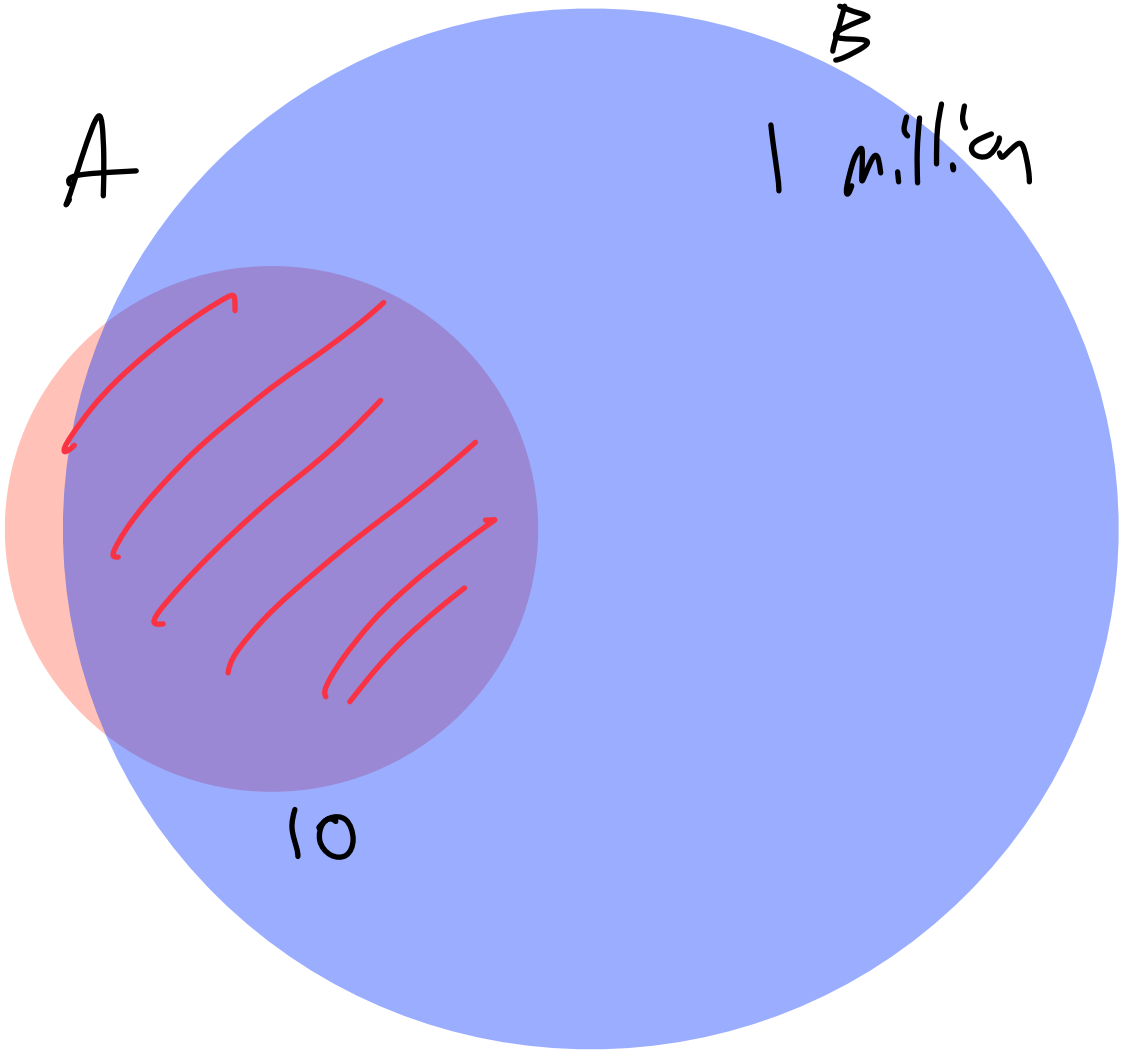
How can we describe how ***similar*** two sets are?

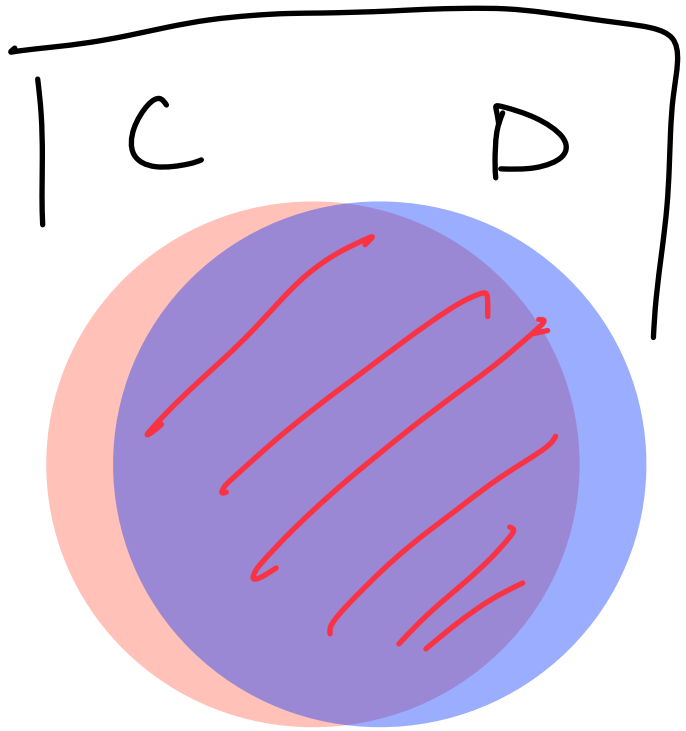




Set Similarity Review

How can we describe how ***similar*** two sets are?

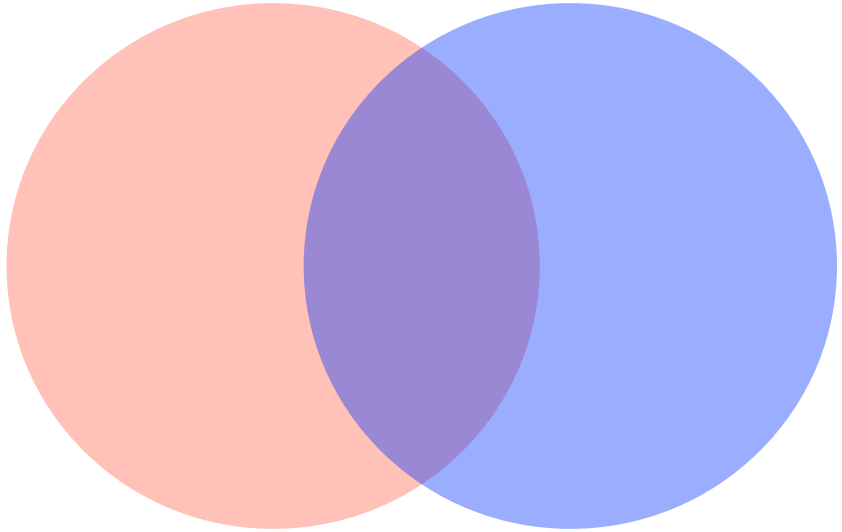




Assume intersection size is same

Set Similarity Review

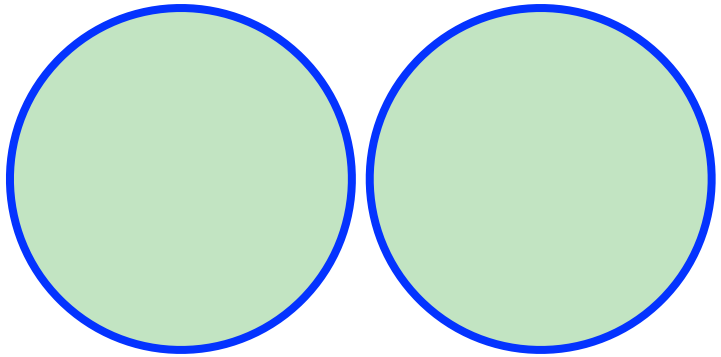
To measure **similarity** of A & B, we need both a measure of how similar the sets are but also the total size of both sets.



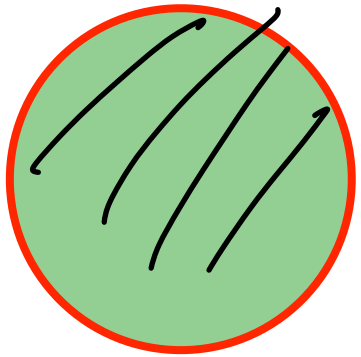
$$J = \frac{|A \cap B|}{|A \cup B|}$$

J is the ***Jaccard coefficient***

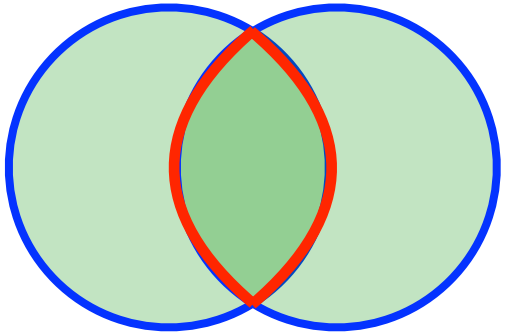
Set Similarity Review



$$\frac{|A \cap B|}{|A \cup B|} = 0 \quad = 0$$



$$\frac{|A \cap B|}{|A \cup B|} = 1 \quad = |A| \quad \geq |A|$$



$$0 < \frac{|A \cap B|}{|A \cup B|} < 1$$

Similarity Sketches

Imagine we have two datasets represented by their kth minimum values

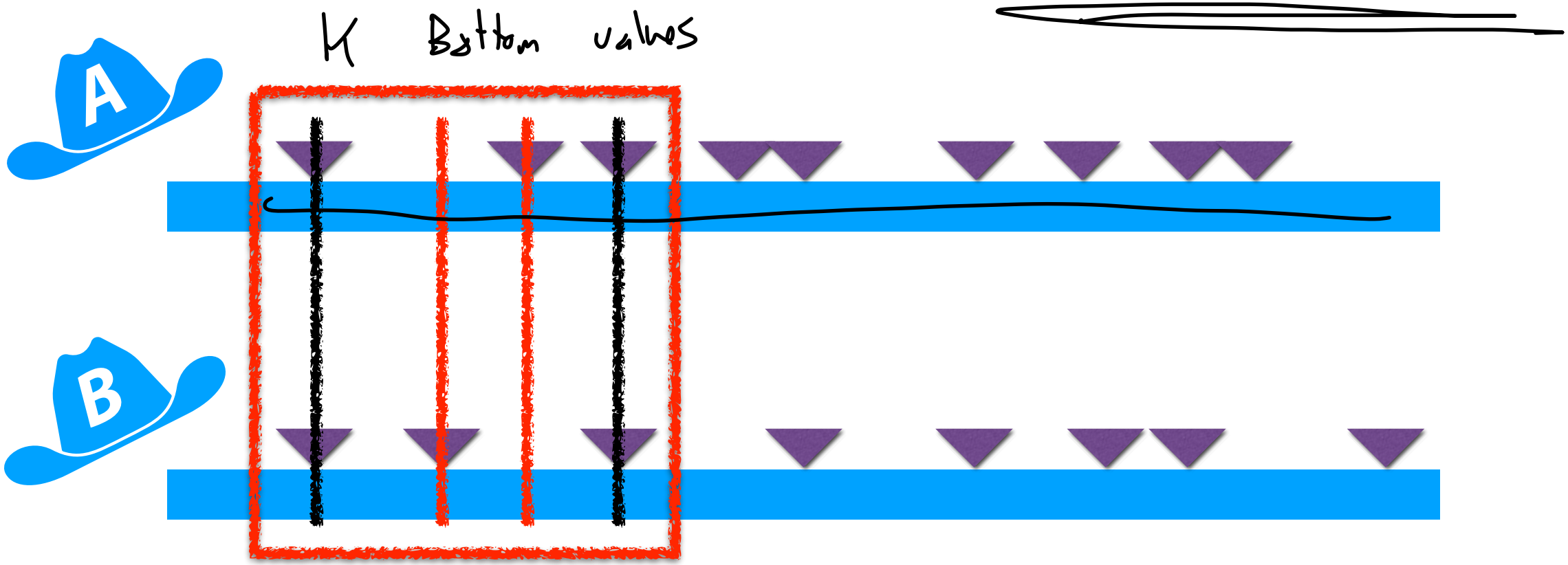


Image inspired by: Ondov B, Starrett G, Sappington A, Kostic A, Koren S, Buck CB, Phillippy AM. **Mash Screen: high-throughput sequence containment estimation for genome discovery**. *Genome Biol* 20, 232 (2019)

Similarity Sketches

Claim: Under SUHA, comparing all k minima can estimate set similarity!

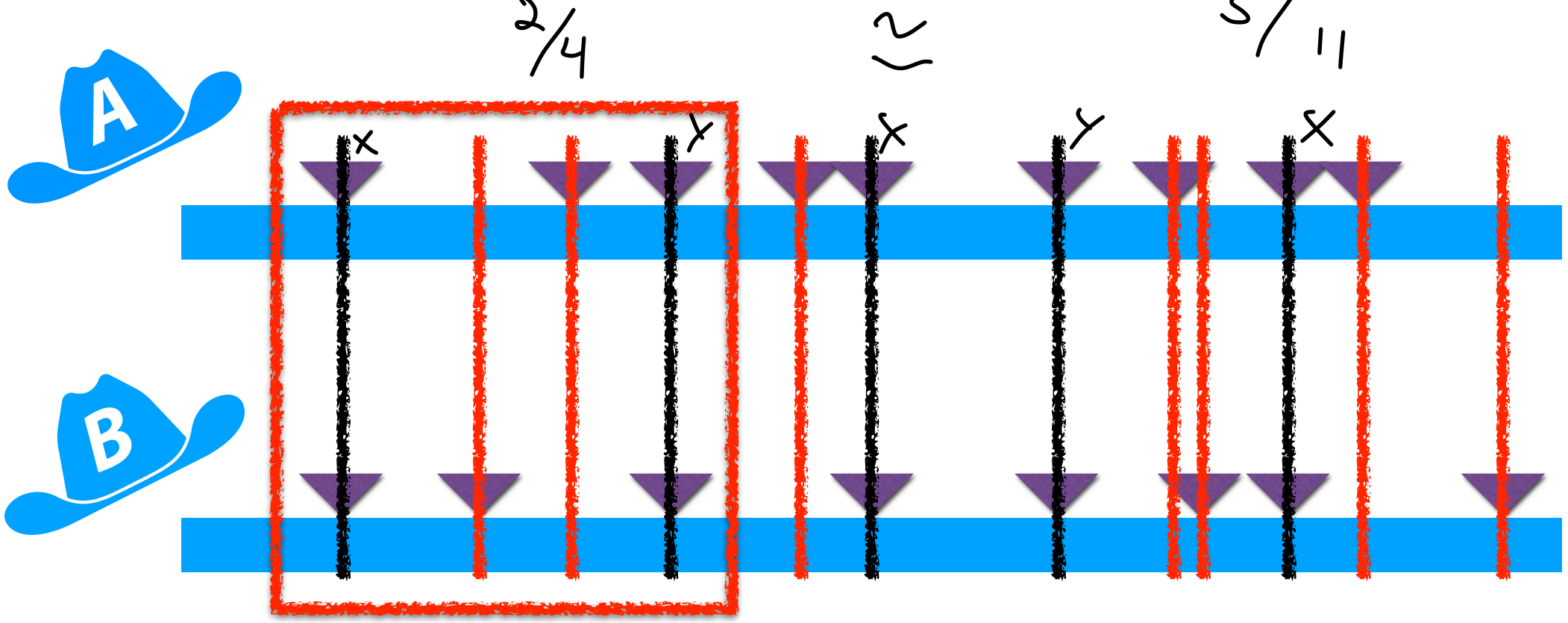


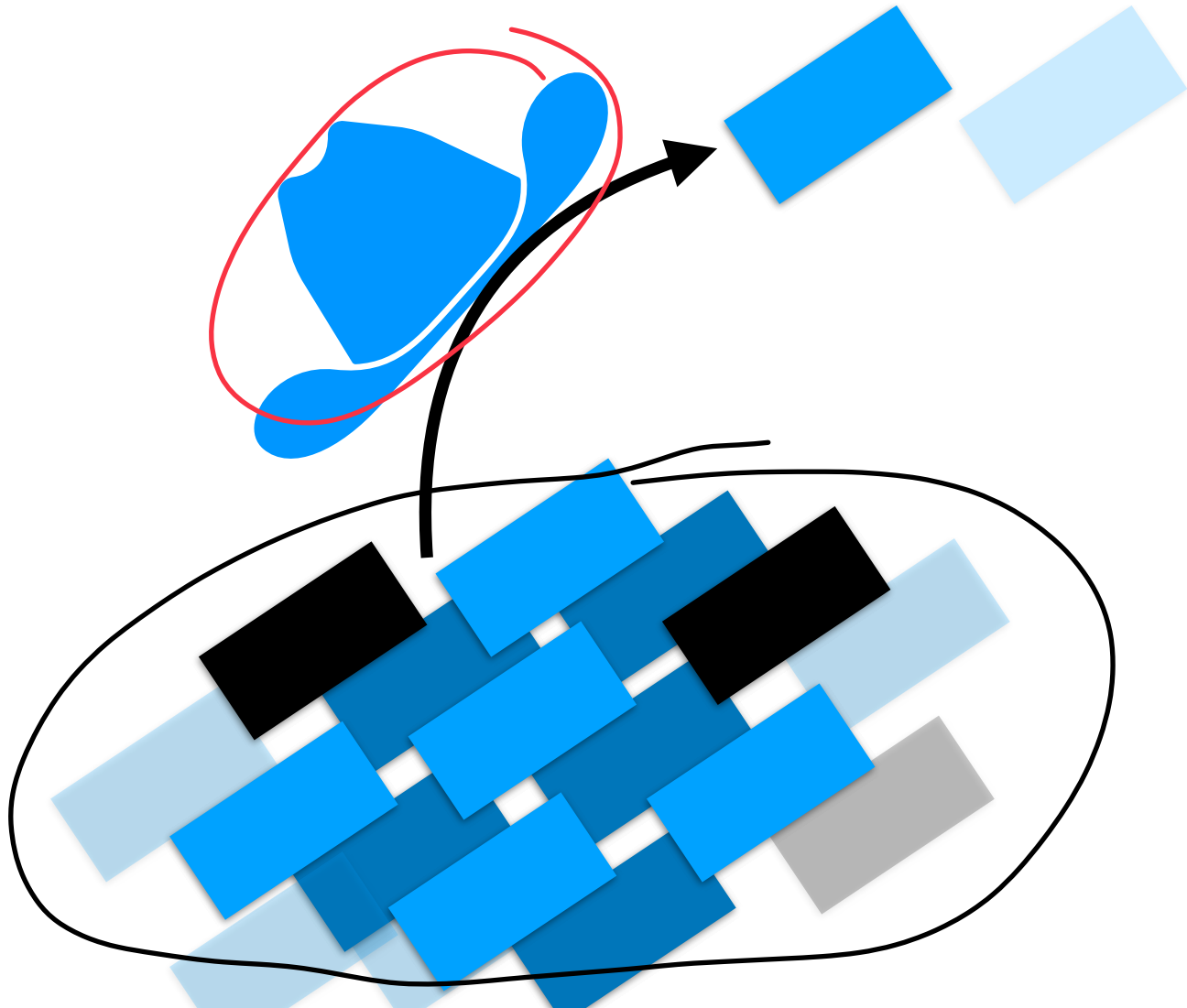
Image inspired by: Ondov B, Starrett G, Sappington A, Kostic A, Koren S, Buck CB, Phillippy AM. **Mash Screen: high-throughput sequence containment estimation for genome discovery**. *Genome Biol* 20, 232 (2019)

MinHash Construction

A MinHash sketch has three required inputs:

1. A hashable dataset
2. A hash function
3. k, the # of minima

$k=2$



MinHash Construction

S = { 16, 8, 4, 13, 15}

h(x) = x % 7

k = 3

Keys → Hash values

16 → 2

8 → 1

4 → 4

13 → 6

15 → I

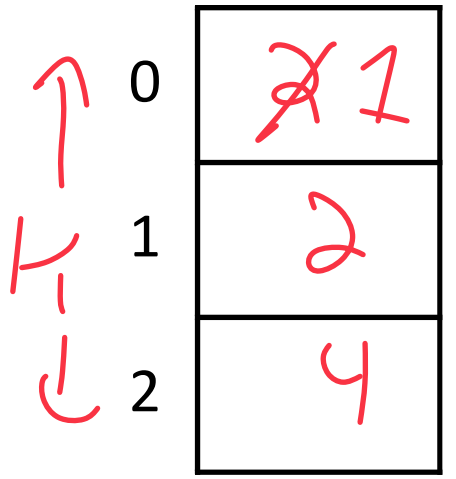
Algorithm is trivial:

1. Hash each item — O(1)
2. Keep the k-minimum values in memory
   (Ignore collisions / duplicates)

$n \times \log(k)$

$O(n)$ time!

Sorted list



MinHash Jaccard Estimation

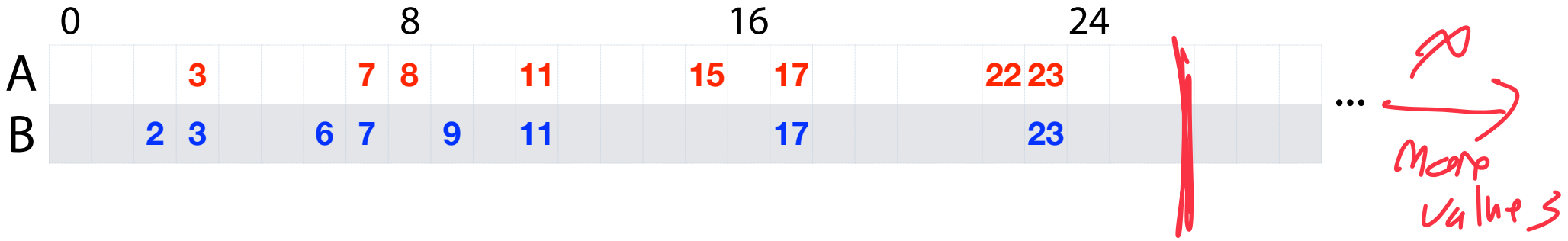
Given sets A and B sampled uniformly from [0, 100], store the bottom-8 **MinHash:**

Sketch A

3	15
7	17
8	22
11	23

Sketch B

2	9
3	11
6	17
7	23



MinHash Jaccard Estimation

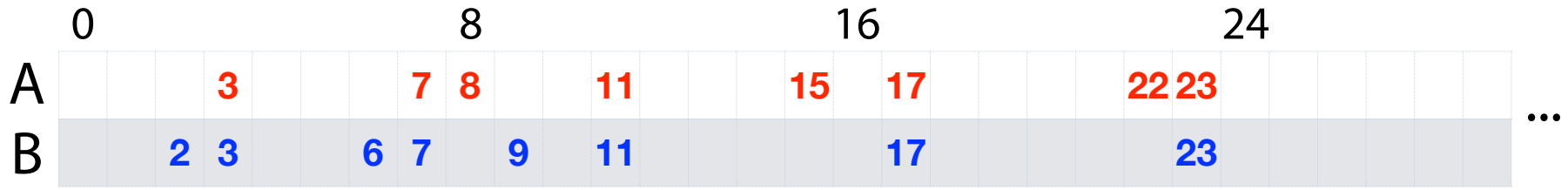
We want to estimate the Jaccard Coefficient: $\frac{|A \cup B|}{|A \cap B|}$

Sketch A

3	15
7	17
8	22
11	23

Sketch B

2	9
3	11
6	17
7	23



MinHash Jaccard Estimation

What do we know about $A \cup B$?

Sketch A

3	15
7	17
8	22
11	23

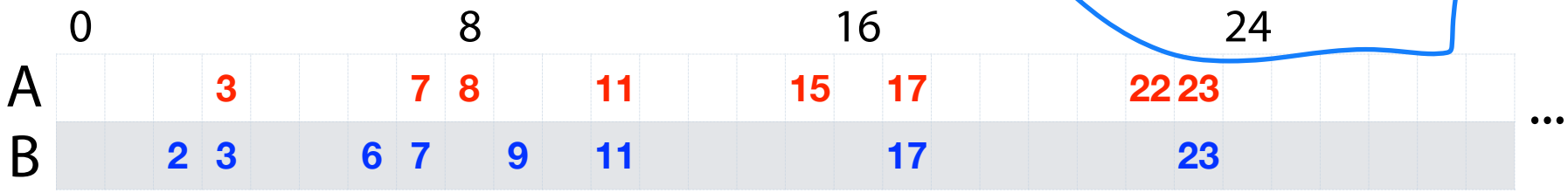
Sketch B

2	9
3	11
6	17
7	23

$A \cup B$

2	8	17
3	9	22
6	11	23
7	15	???

...



MinHash Jaccard Estimation

We dont *know* $A \cup B$, but we can make a sketch!

Sketch A

3	15
7	17
8	22
11	23

$\cup$

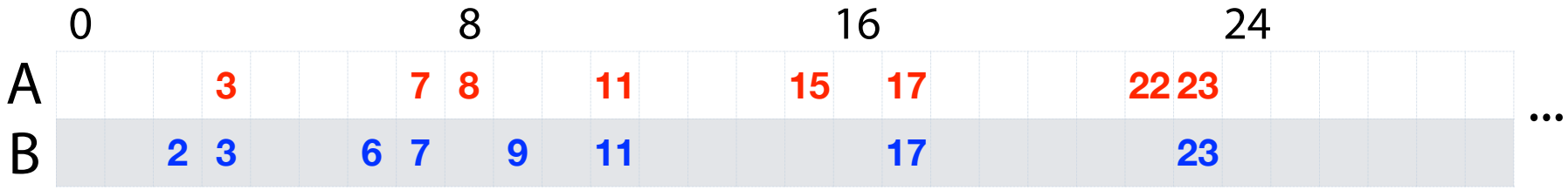
Sketch B

2	9
3	11
6	17
7	23

$=$

Sketch $A \cup B$

2	8
3	9
6	11
7	15



MinHash Jaccard Estimation

Estimate $|A \cup B|$ (the cardinality of the union) from sketch:

Sketch $A \cup B$

2	8
3	9
6	11
7	15

Our sets sampled from [0, 100].

$$\frac{15}{100} = \frac{8}{N+1} \rightarrow N+1 = \frac{800}{15}$$

$$N = \frac{800}{15} - 1$$

$$\approx 52$$

MinHash Jaccard Estimation

Can we build a 8-Minhash of $A \cap B$?

Sketch A

3	15
7	17
8	22
11	23

$\cap$

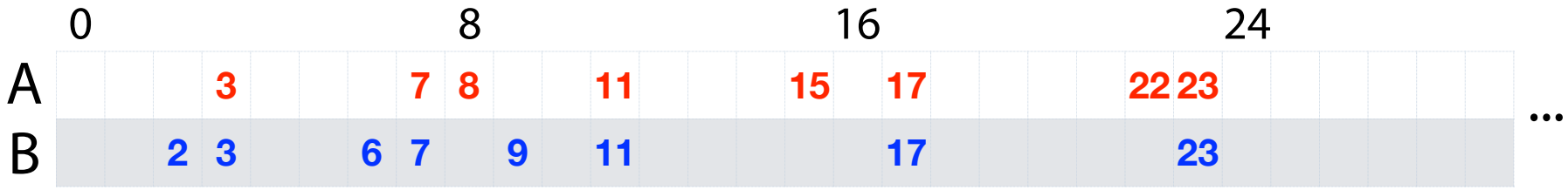
Sketch B

2	9
3	11
6	17
7	23

$=$

Sketch $A \cap B$

3	23
7	9
11	17
15	23



MinHash Jaccard Estimation

Not guaranteed to be able to get a full sketch of the intersection!

Sketch A

3	15
7	17
8	22
11	23

$\cap$

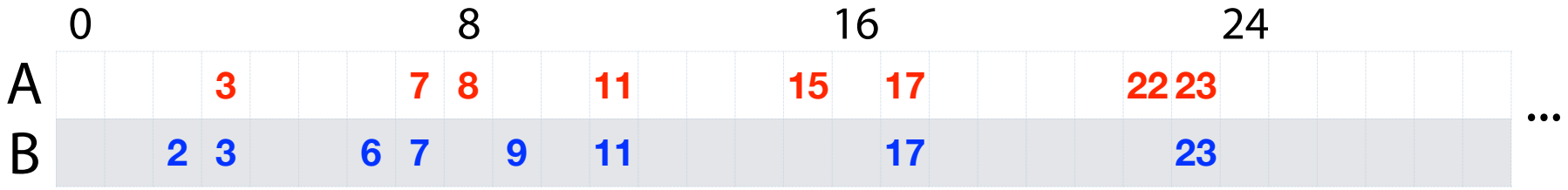
Sketch B

2	9
3	11
6	17
7	23

$=$

Sketch $A \cap B$

3	23
7	
11	
17	



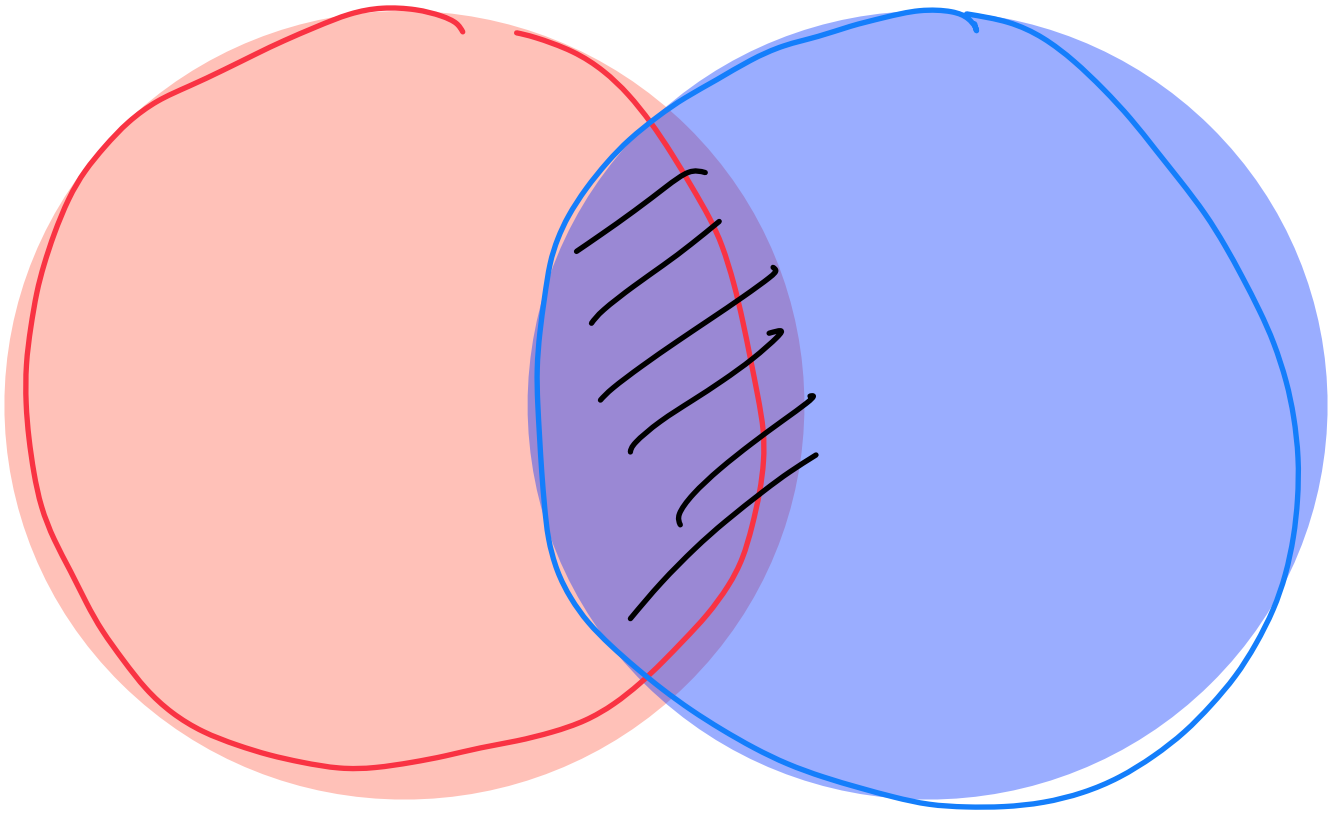
MinHash Jaccard Estimation

Using MinHash sketches, we can estimate $|A|$, $|B|$, and $|A \cup B|$

Is this enough to estimate the Jaccard?

Inclusion-Exclusion Principle

$$|A \cap B| = |A| + |B| - |A \cup B|$$



MinHash Indirect Jaccard Estimation

$$\frac{|A| \cap |B|}{|A| \cup |B|} = \frac{|A| + |B| - |A \cup B|}{|A \cup B|}$$

$k = 8$ MinHash sketches

Our sets sampled from [0, 100]

Sketch A	
3	15
7	17
8	22
11	23

Sketch B	
2	9
3	11
6	17
7	23

Sketch of $\lvert A \cup B \rvert$	
2	8
3	9
6	11
7	15

$$= \frac{(800/23-1) + (800/23-1) - (800/15-1)}{800/15-1}$$

$$= \frac{34.782 + 34.782 - 53.333 - 1}{53.333 - 1} \approx 0.29$$

MinHash Direct Jaccard Estimate

We can also estimate cardinality directly using our sketches!

Sketch A

3	15
7	17
8	22
11	23

Sketch B

2	9
3	11
6	17
7	23

Intersection

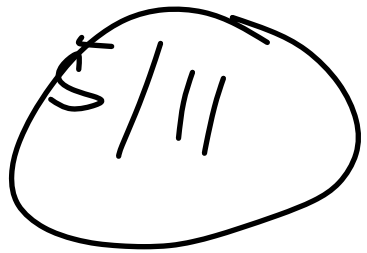
3	23
7	
11	
17	

5

Union

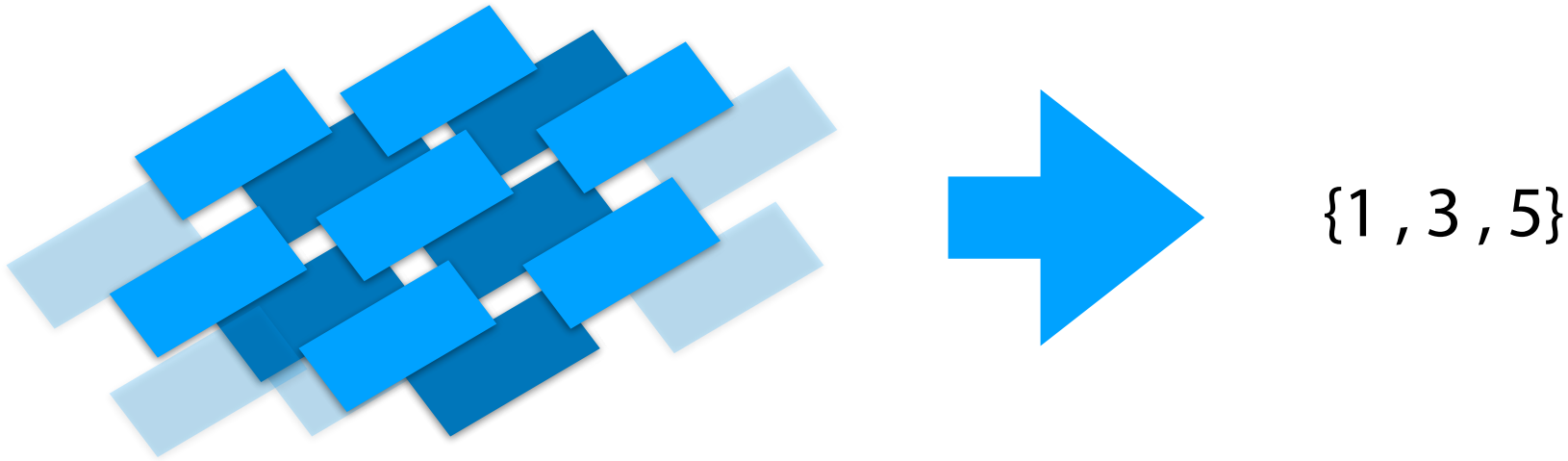
2	8	17
3	9	22
6	11	23
7	15	

11



MinHash Sketch

We can convert any hashable dataset into a **MinHash sketch**



We lose ~~our original~~ dataset, but we can still estimate two things:

1. # of unique items
2. Sylt similarity between two minhash stretches

Alternative MinHash Sketch Approaches

Rather than use one single hashes and take bottom-k, we can also use k hashes — **if you have access to that many independent hashes**!

1) Sequence decomposed into **kmers**

S_1 : CATGGACCGACCAG GCAGTACCGATCGT : S_2

CAT GAC GAC GTA CGA CGT

ATG ACC ACC AGT CCG TCG

TGG CCG CCA CAG ACC ATC

GGA CGA CAG GCA TAC GAT

Assembling large genomes with single-molecule sequencing and locality-sensitive hashing
Berlin et al (2015) *Nature Biotechnology*

1) Sequence decomposed into **kmers**

2) Multiple hash functions (**Γ**) map kmers to values.

S_1 : CATGGACCGACCAG

CAT GAC GAC

ATG ACC ACC

TGG CCG CCA

GGA CGA CAG

Γ_1	Γ_2	Γ_3	Γ_4	
19	14	57	36	CAT
14	57	36	19	ATG
58	37	16	15	TGG
40	23	2	61	GGA
33	28	11	54	GAC
5	48	47	26	ACC
22	1	60	43	CCG
24	7	50	45	CGA
33	28	11	54	GAC
5	48	47	26	ACC
20	3	62	41	CCA
18	13	56	39	CAG

GCAGTACCGATCGT : S_2

GTA CGA CGT

AGT CCG TCG

CAG ACC ATC

GCA TAC GAT

	Γ_1	Γ_2	Γ_3	Γ_4
GCA	36	19	14	57
CAG	18	13	56	39
AGT	11	54	33	28
GTA	44	27	6	49
TAC	49	44	27	6
ACC	5	48	47	26
CCG	22	1	60	43
CGA	24	7	50	45
GAT	35	30	9	52
ATC	13	56	39	18
TCG	54	33	28	11
CGT	27	6	49	44

Assembling large genomes with single-molecule sequencing and locality-sensitive hashing
Berlin et al (2015) *Nature Biotechnology*

1) Sequence decomposed into **kmers**

2) Multiple hash functions (**Γ**) map kmers to values.

3) The smallest values for each hash function is chosen

S_1 : CATGGACCGACCAG

CAT GAC GAC

ATG ACC ACC

TGG CCG CCA

GGA CGA CAG

Γ_1	Γ_2	Γ_3	Γ_4	
19	14	57	36	CAT
14	57	36	19	ATG
58	37	16	15	TGG
40	23	2	61	GGA
33	28	11	54	GAC
5	48	47	26	ACC
22	1	60	43	CCG
24	7	50	45	CGA
33	28	11	54	GAC
5	48	47	26	ACC
20	3	62	41	CCA
18	13	56	39	CAG

[5, 1, 2, 15]

Sketch (S_1)

GCAGTACCGATCGT : S_2

GTA CGA CGT

AGT CCG TCG

CAG ACC ATC

GCA TAC GAT

	Γ_1	Γ_2	Γ_3	Γ_4
GCA	36	19	14	57
CAG	18	13	56	39
AGT	11	54	33	28
GTA	44	27	6	49
TAC	49	44	27	6
ACC	5	48	47	26
CCG	22	1	60	43
CGA	24	7	50	45
GAT	35	30	9	52
ATC	13	56	39	18
TCG	54	33	28	11
CGT	27	6	49	44

[5, 1, 6, 6]

Sketch (S_2)

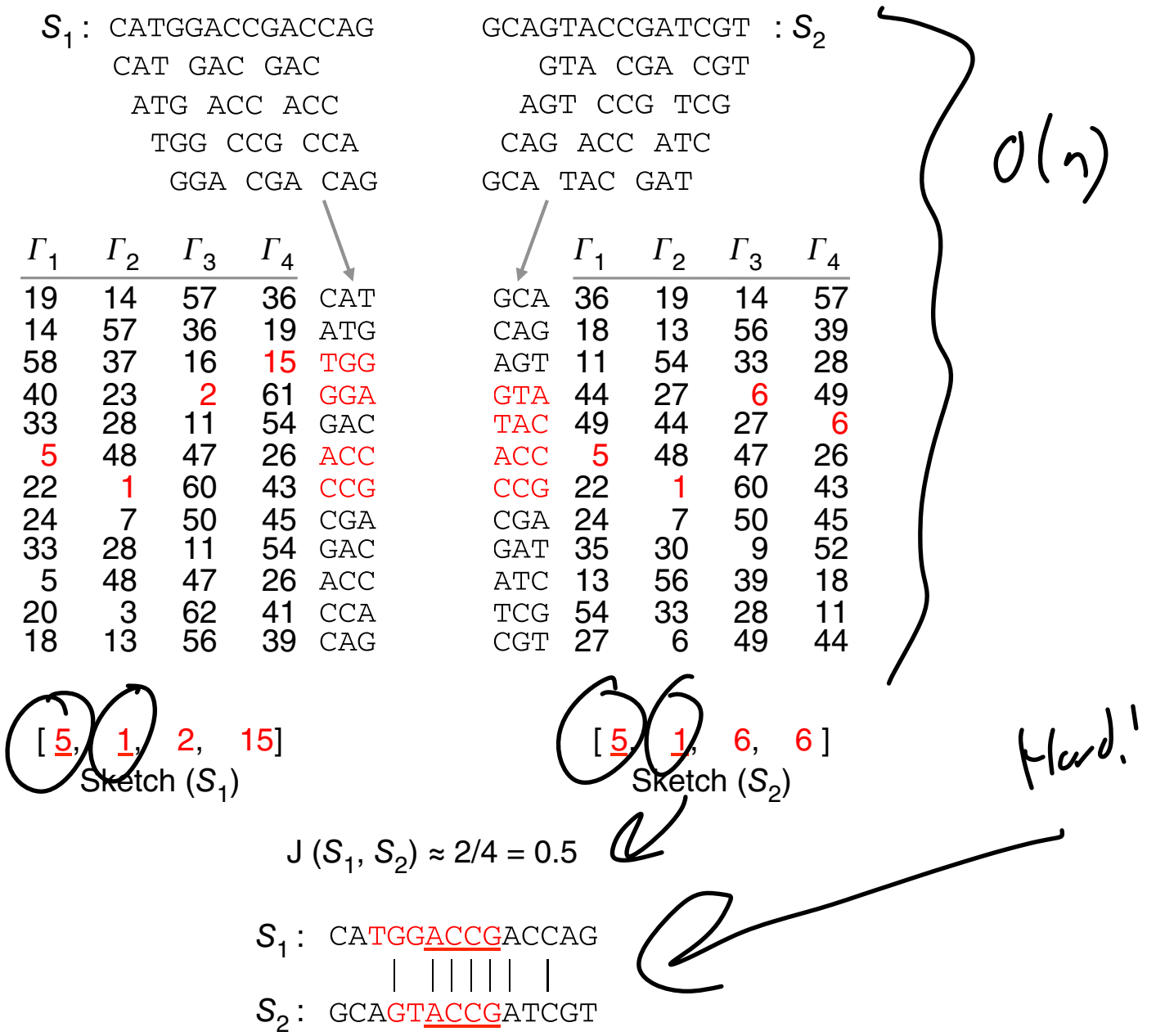
Assembling large genomes with single-molecule sequencing and locality-sensitive hashing
Berlin et al (2015) *Nature Biotechnology*

1) Sequence decomposed into **kmers**

2) Multiple hash functions (**Γ**) map kmers to values.

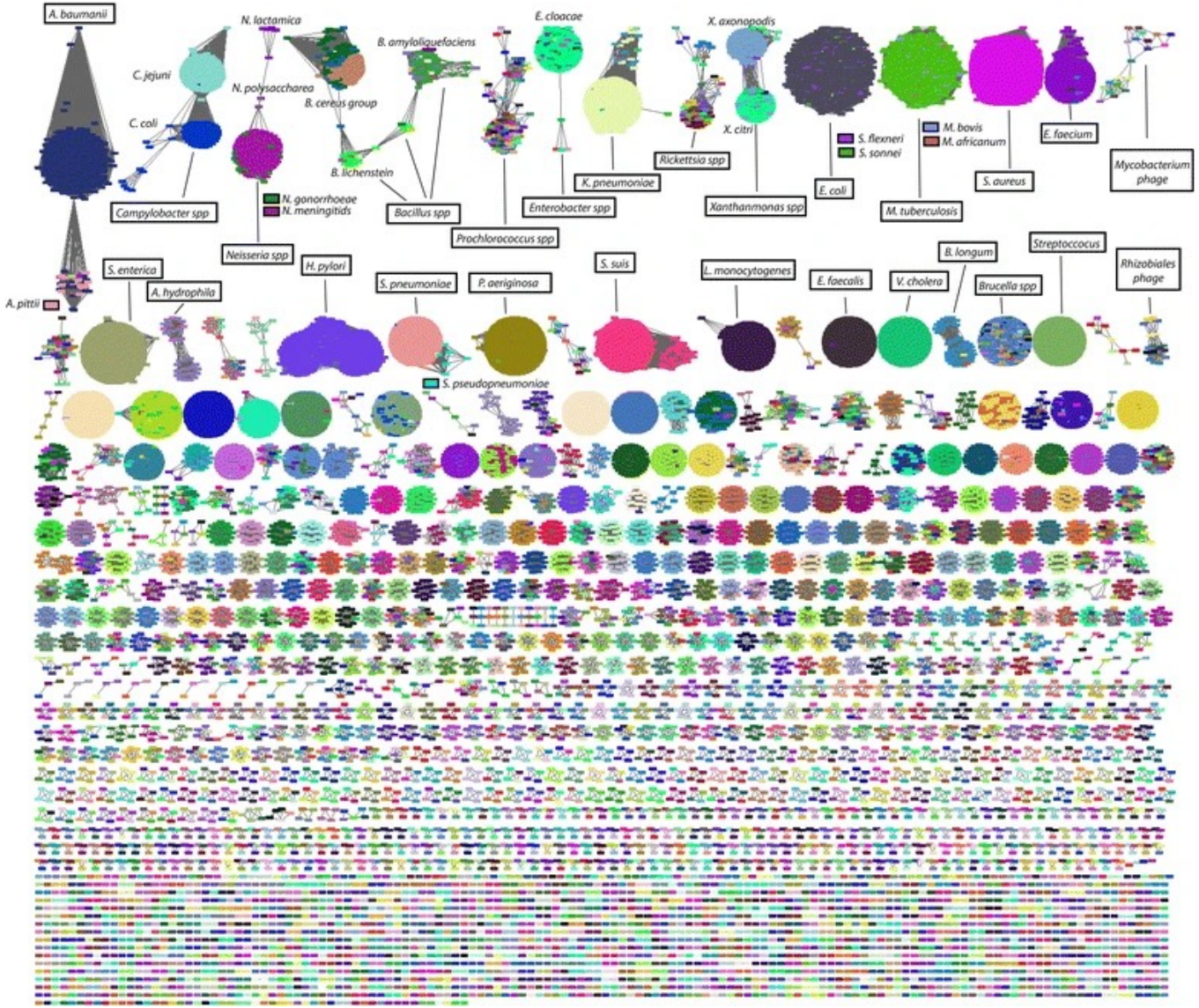
3) The smallest values for each hash function is chosen

4) The Jaccard similarity can be estimated by the overlap in the **Min**imum **Hash**es (**MinHash)**



Assembling large genomes with single-molecule sequencing and locality-sensitive hashing
Berlin et al (2015) *Nature Biotechnology*

MinHash in practice



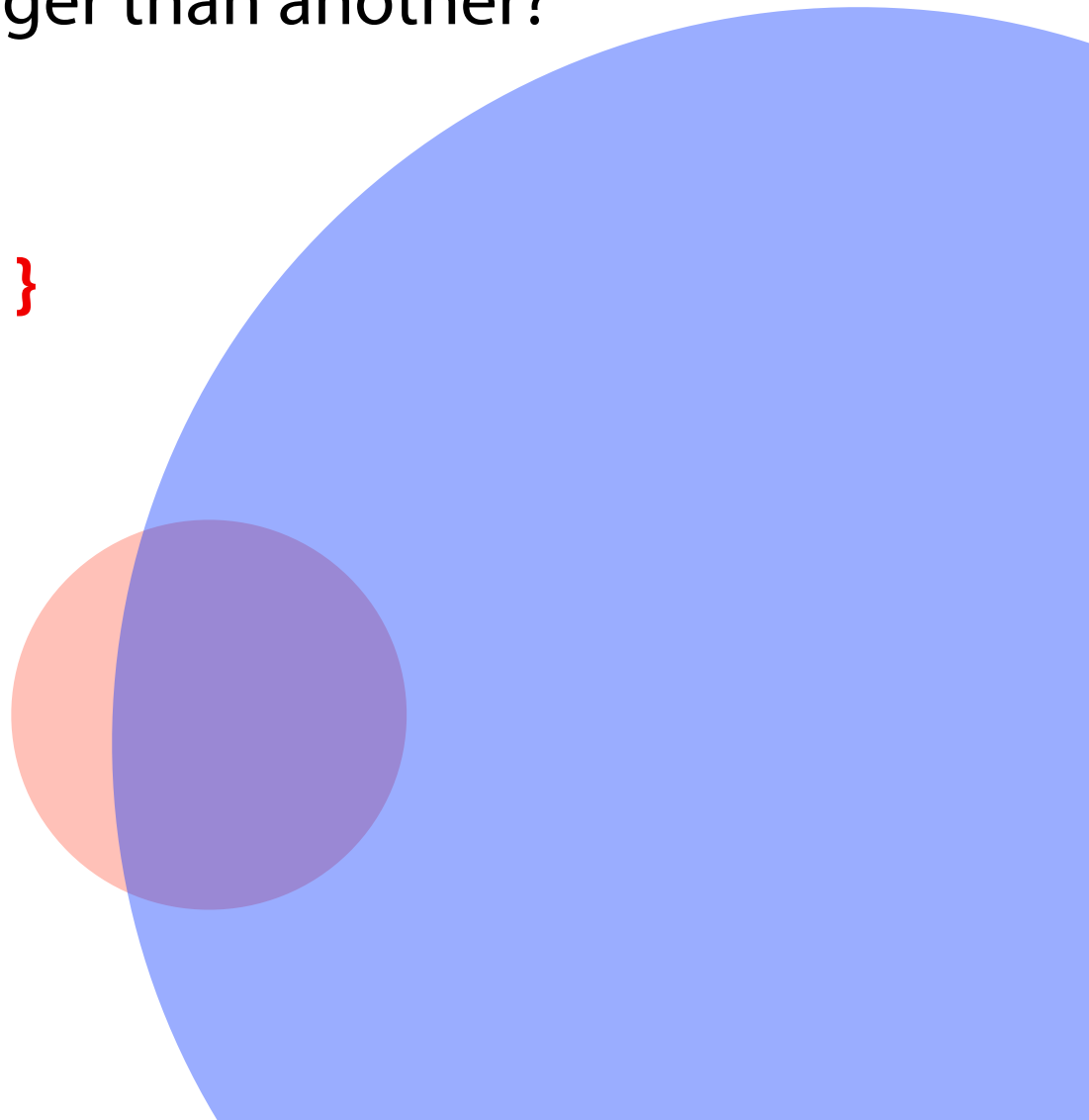
Mash: fast genome and metagenome distance estimation using MinHash
Ondov et al (2016) *Genome Biology*

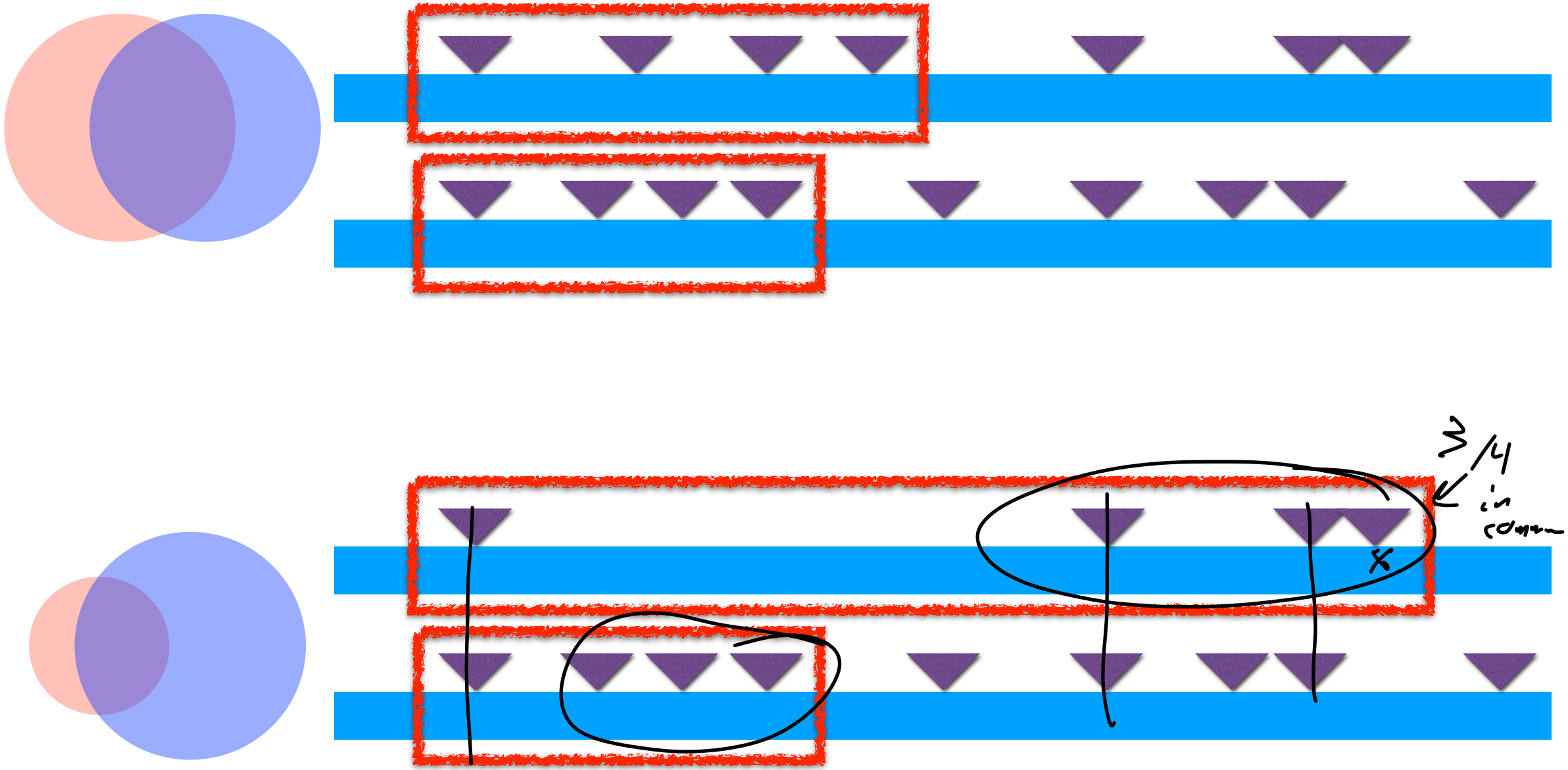
Alternative MinHash Sketch Approaches

What if I have a dataset which is **much** larger than another?

S_1 = { 1, 3, 40, 59, 82, 101 }

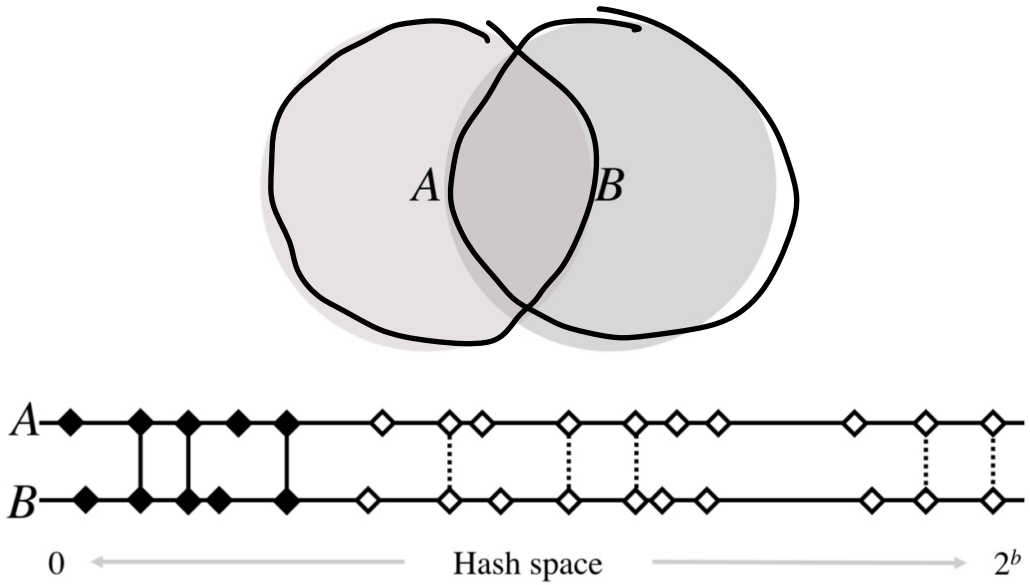
S_2 = { 1, 2, 3, 4, 5, 6, 7, … 59, 82, 101, … }

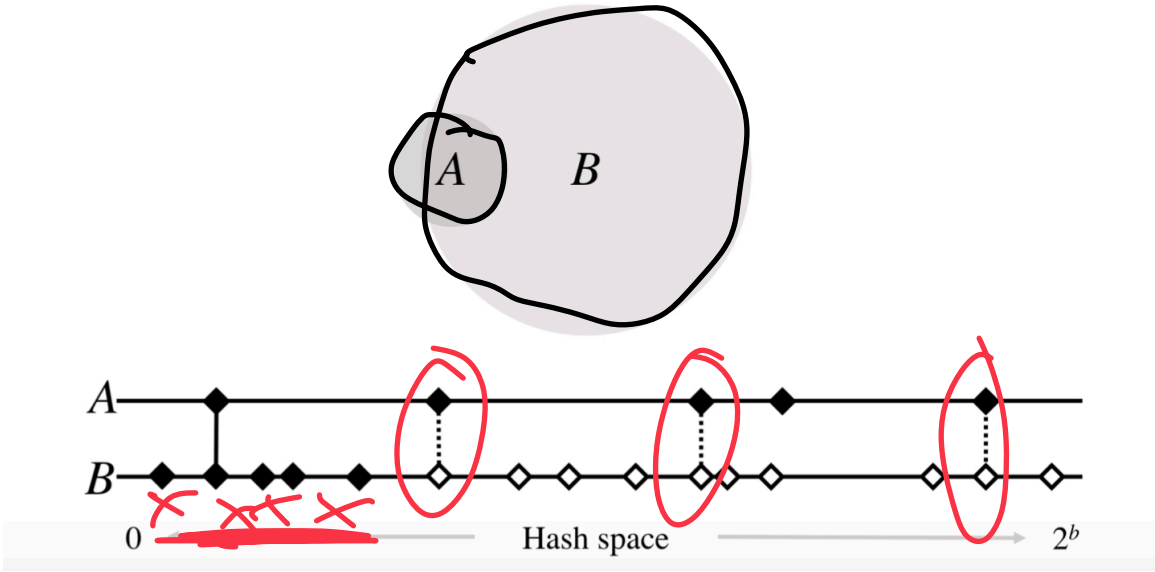



3/4
in
common

Alternative MinHash sketches

Bottom-k minhash has low accuracy if the cardinality of sets are skewed

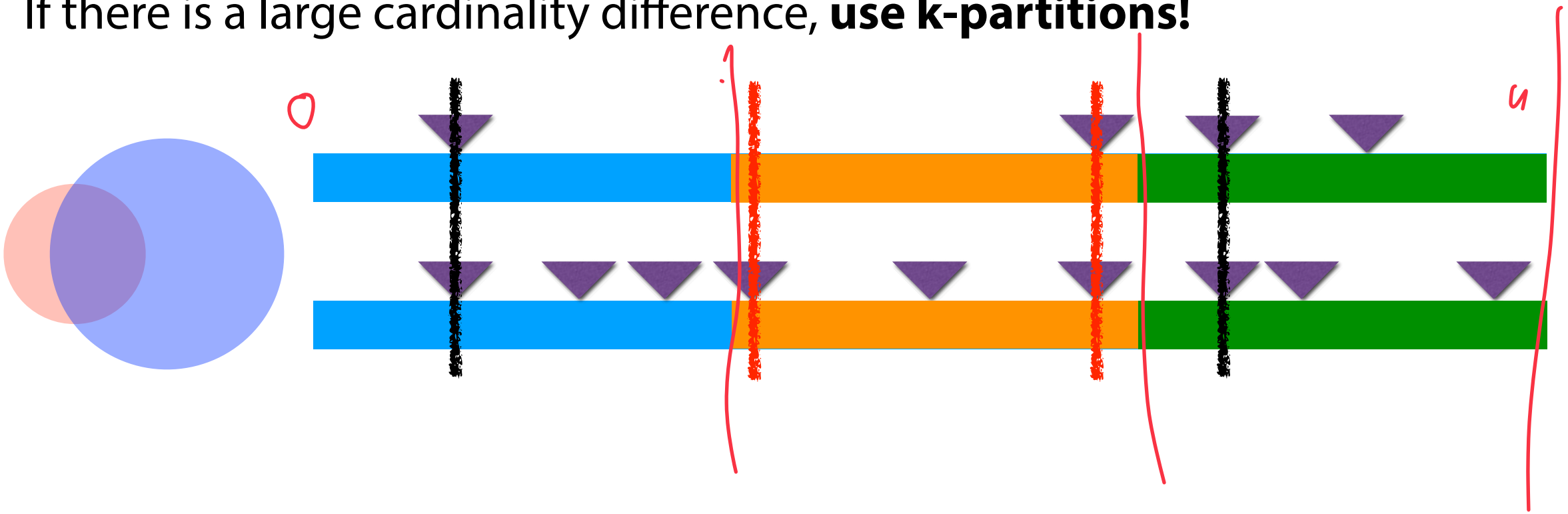




Ondov, Brian D., Gabriel J. Starrett, Anna Sappington, Aleksandra Kostic, Sergey Koren, Christopher B. Buck, and Adam M. Phillippy. **Mash Screen: High-throughput sequence containment estimation for genome discovery**. *Genome biology* 20.1 (2019): 1-13.

Alternative MinHash Sketch Approaches

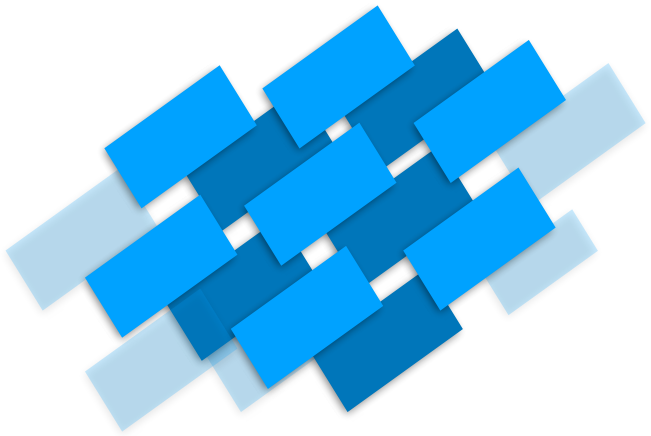
If there is a large cardinality difference, **use k-partitions!**



K-Partition Minhash

2 bits

$2^2 \rightarrow 4$ partitions



Hash

1010110101

0001111010

1101101011

1011010110

0101100000

0010001101

Partition

00

01111010

10001101

01

01100000

10

10110101

11010110

11

01101011

Probabilistic Data Structures

Probabilistic data structures trade accuracy for efficiency

Most can maintain surprisingly good accuracy

“Cheat” Big O limitations on conventional data analysis