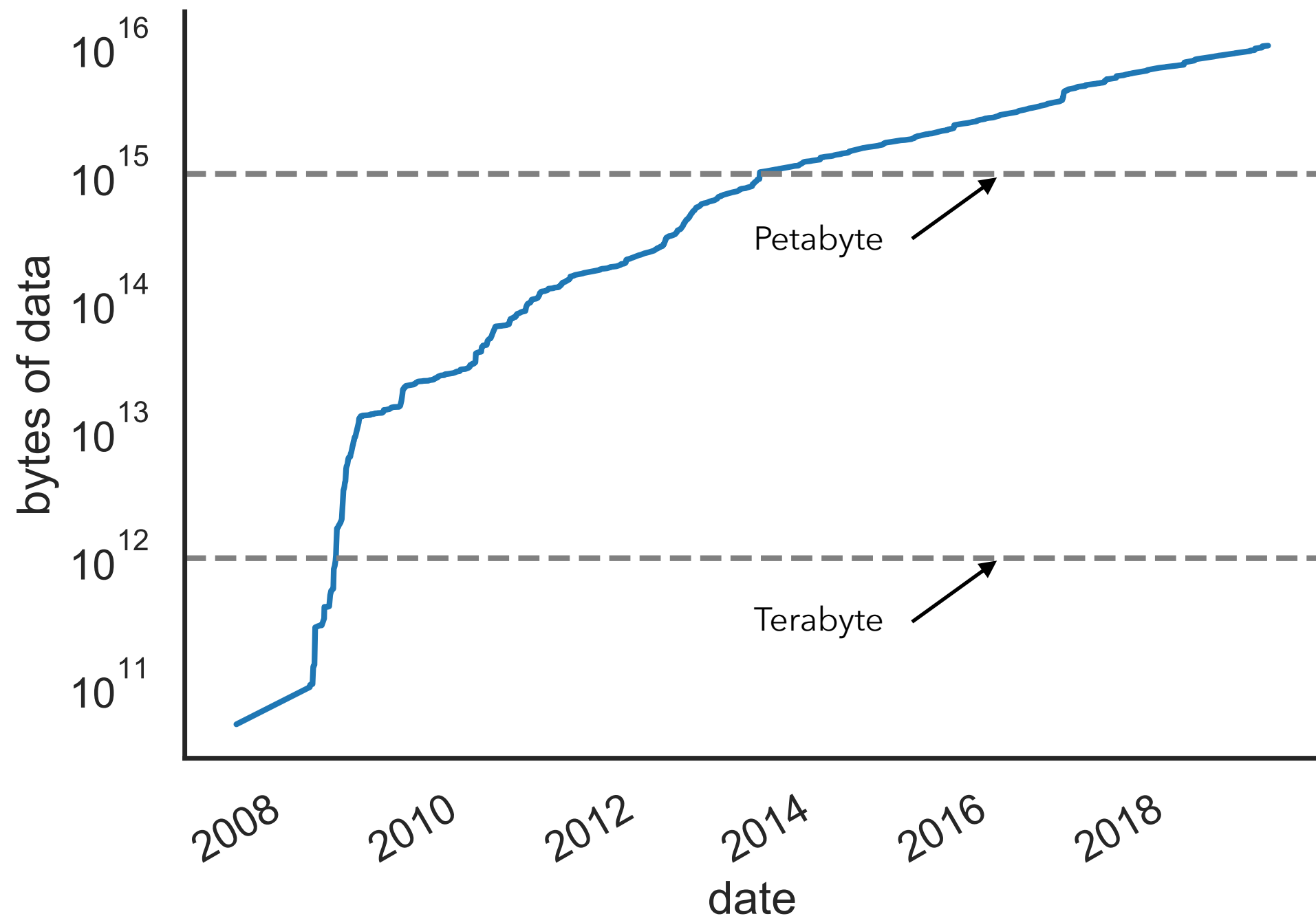


Scalability Challenges in Large-Scale Sequence Search

Prashant Pandey
School of Computing
University of Utah

Facing a New Challenge

The Sequence Read Archive (SRA) ...

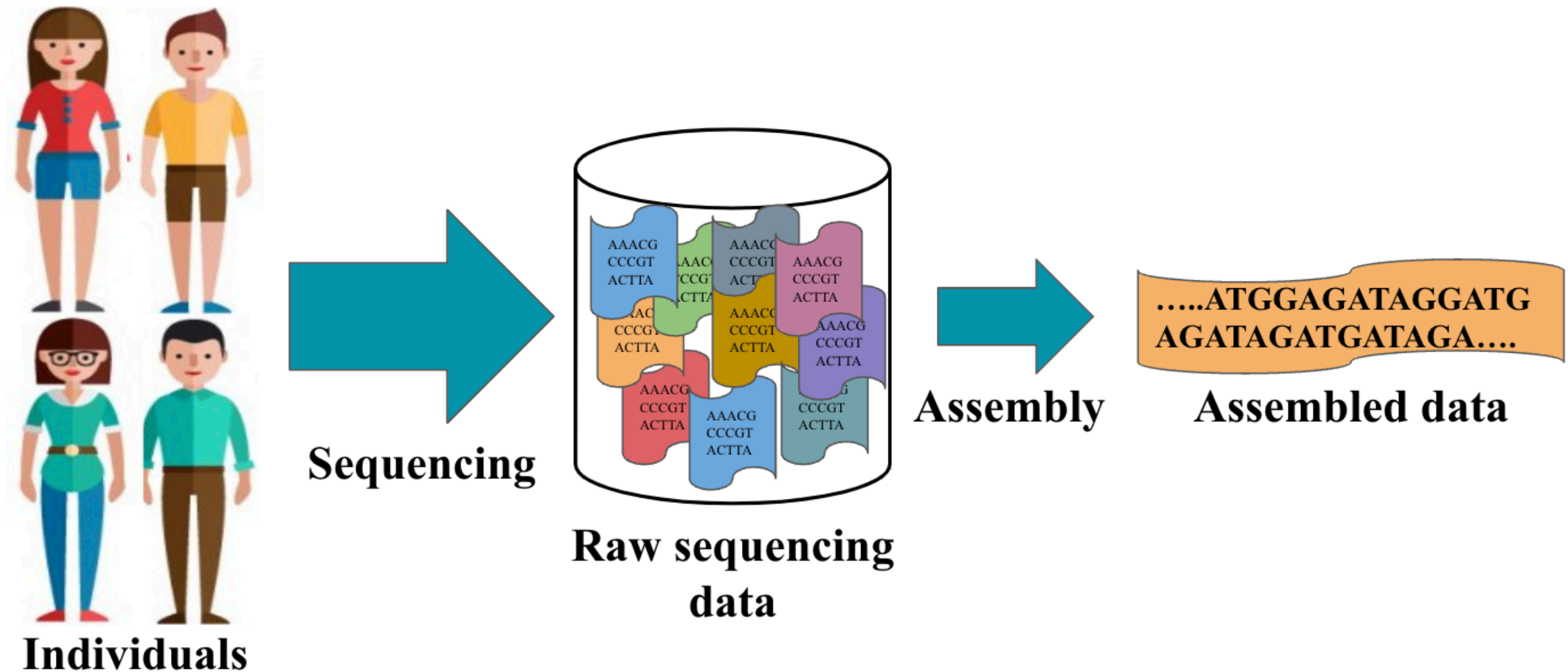


is not searchable by sequence* ! (Yes, I know!)

This renders what is otherwise an immensely valuable public resource **largely inert**

* there is an SRA BLAST, but functionality is limited

A Huge Amount of Information is Available in Raw Sequencing Data



Assembled data is hugely lossy. A lot of **variability information is lost during assembly**.

And a lot of raw sequencing data never gets assembled.

The Ability to Perform Searches on Raw Sequencing Data would Enable Us to Answer Lots of Questions

Q: What if I find a new putative disease-related transcript, and want to see if it appeared in other biological samples?

Q : What if I discover a new fusion event in a particular cancer subtype and want to know if it is common among samples with this subtype?

Q: What if I find an unexpected bacterial contaminant in my data; which other samples might contain this?

The ability to perform searches on raw sequencing data would enable us to answer lots of questions

Q: What if I find a new putative disease-related transcript, and want to see if it appeared in other biological samples?

Q : What if I discover a new fusion event in a particular cancer subtype and want to know if it is common among samples with this subtype?

Q: What if I find an unexpected bacterial contaminant in my data; which other samples might contain this?

A: I need to search through tons of raw sequencing data.

Facing a New Challenge

Contrast this situation with the task of searching *assembled, curated* genomes, For which we have an *excellent* tool; BLAST.

The screenshot shows the BLAST web interface. The 'Enter Query Sequence' section contains the sequence: TGAAAAAGGGTAACCTCAAAGCTAAAAAGCCCAAGAAGGGGAAGCCCCATTGCAGCCGCAAC CCTGTCCTTGTTCAGAGGAATTGGCAGGTATTCCCGATC. The 'BLAST' button is highlighted. The 'Sequences producing significant alignments:' section shows a list of results with columns: Max score, Total score, Query cover, E value, Ident, and Accession. The results include 'Eukaryotic synthetic construct chromosome 18' and several 'PREDICTED: Pan paniscus 60S ribosomal protein L6-like' entries.

	Max score	Total score	Query cover	E value	Ident	Accession
<input type="checkbox"/> Eukaryotic synthetic construct chromosome 18	185	371	100%	2e-43	100.00%	CP034496.1
<input type="checkbox"/> PREDICTED: Pan paniscus 60S ribosomal protein L6-like (LOC100976413), mRNA	185	185	100%	2e-43	100.00%	XM_008963989.2
<input type="checkbox"/> PREDICTED: Pan paniscus 60S ribosomal protein L6_pseudogene (LOC100995849), misc_RNA	185	185	100%	2e-43	100.00%	XR_610957.3
<input type="checkbox"/> PREDICTED: Pan paniscus 60S ribosomal protein L6 (LOC100995836), mRNA	185	185	100%	2e-43	100.00%	XM_003812574.3
<input type="checkbox"/> PREDICTED: Pan troglodytes 60S ribosomal protein L6_pseudogene (LOC737972), misc_RNA	185	185	100%	2e-43	100.00%	XR_680356.3
<input type="checkbox"/> PREDICTED: Pan troglodytes ribosomal protein L6 (RPL6), transcript variant X8, mRNA	185	185	100%	2e-43	100.00%	XM_024347583.1
<input type="checkbox"/> PREDICTED: Pan troglodytes ribosomal protein L6 (RPL6), transcript variant X7, mRNA	185	185	100%	2e-43	100.00%	XM_024347582.1
<input type="checkbox"/> Human ORFeome Gateway entry vector pENTR223-RPL6, complete sequence	185	185	100%	2e-43	100.00%	LT737273.1
<input type="checkbox"/> PREDICTED: Gorilla gorilla gorilla ribosomal protein L6 (RPL6), transcript variant X5, mRNA	185	185	100%	2e-43	100.00%	XM_019038370.1

Essentially, the “Google of genomics”:

Basic local alignment search tool

[SF Altschul](#), [W Gish](#), [W Miller](#), [EW Myers](#)... - Journal of molecular ..., 1990 - Elsevier [Paperpile](#)

A new approach to rapid sequence comparison, basic local alignment search tool (BLAST), directly approximates alignments that optimize a measure of local similarity, the maximal segment pair (MSP) score. Recent mathematical results on the stochastic properties of MSP ...

☆ [Cited by 76248](#) [Related articles](#) [Web of Science: 52272](#) [Import into BibTeX](#)

However, even the scale of *reference* databases requires algorithmic innovations:

_computational
BIOLOGY

Compressive genomics

Po-Ru Loh, Michael Baym & Bonnie Berger

Algorithms that compute directly on compressed genomic data allow analyses to keep pace with data generation.

Entropy-Scaling Search of Massive Biological Data

Y. William Yu,^{1,2,3} Noah M. Daniels,^{1,2,3} David Christian Danko,² and Bonnie Berger^{1,2,*}

¹Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

²Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

³Co-first author

*Correspondence: bab@mit.edu

<http://dx.doi.org/10.1016/j.cels.2015.08.004>

The Computational Problem

So, why can't we just use BLAST for searching "raw" data?

- Patterns of interest might be spread across many reads (no contiguous substring)
- The pattern we are looking for may not be present in an assembled genome (we have genomes for only a small fraction of the ~8.7 Million* species on the planet – most can't even be cultivated in labs)
- Even if we had those genomes, there is so much more information in raw data. A reference genome reduces entire populations (e.g. humans) to a single string – hugely lossy (gene expression could change wildly in the same genome)
- BLAST-like algorithms & data structures just don't seem to scale!

*Mora, Camilo, et al. "How many species are there on Earth and in the ocean?." PLoS biology 9.8 (2011): e1001127.

Reframing the problem

Some recent work reframes this problem slightly, and suggested a direction toward a potential solution ...



Proposal:

A hierarchical index of k-mer content represented approximately via Bloom filters.

Returns "yes/no" results for individual experiments → "yes" results can be searched using more traditional methods

K-mers as search primitives*



- For a given molecule (string), a k-mer is simply a k-length sub-string.
- Akin to n-grams as used in NLP (except DNA/RNA have no natural “tokens” ... this complicates things quite a bit)
- **Idea:** Similarity of k-mer composition → similar sequence

*Note: This is related to the way we sped up transcript expression estimation by >20x in our “sailfish” work.

Sample discovery problem

...ACACGTA...

Check if this new
transcript has
been seen before?

ACTGAGTGA
ACGTTGTGC
GTGCGTGCG
TAAACGTA
CGTCACGTA

ACTGAGTGA
ACGTTGTGC
GTGCGTGCG
TAAACGTA
CGTCACGTA

•
•
•

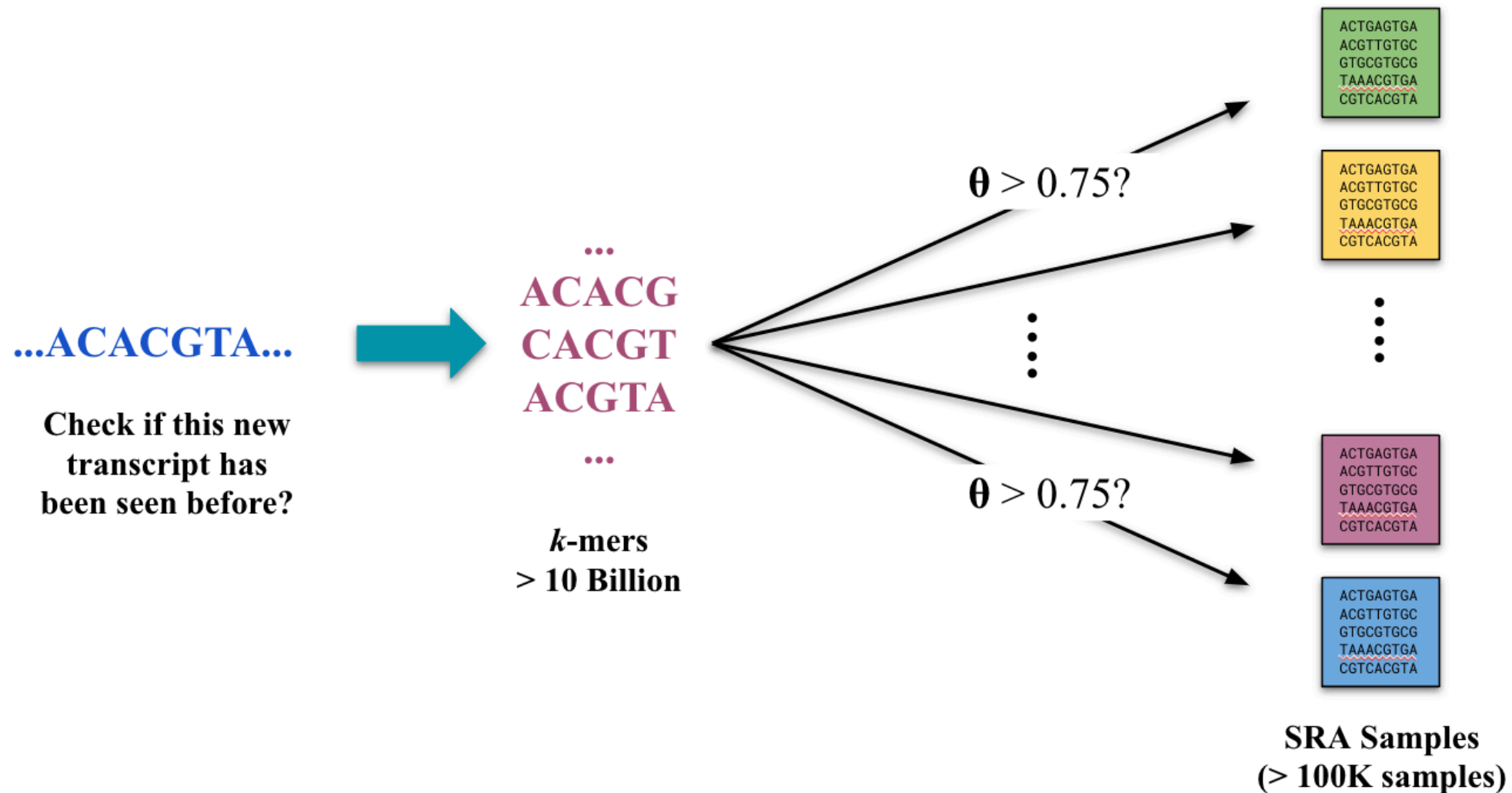
ACTGAGTGA
ACGTTGTGC
GTGCGTGCG
TAAACGTA
CGTCACGTA

ACTGAGTGA
ACGTTGTGC
GTGCGTGCG
TAAACGTA
CGTCACGTA

SRA Samples
(> 100K samples)

Return all samples that contain at least some user-defined fraction θ of k -mers present in the query string.

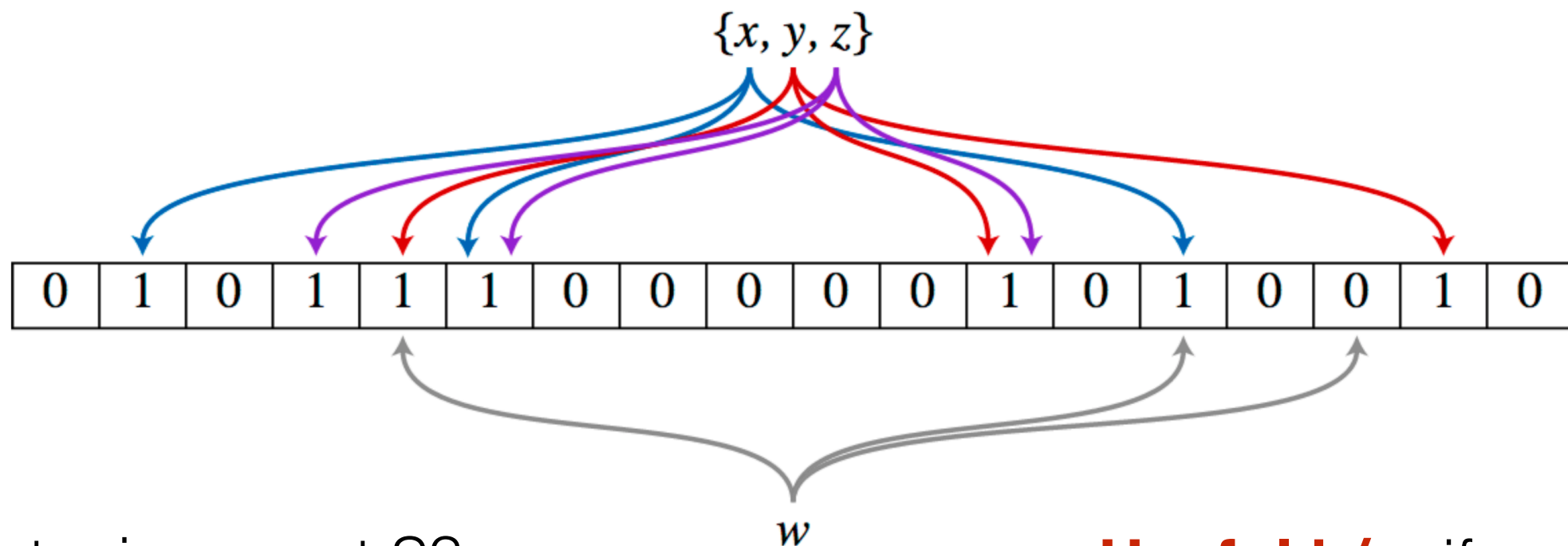
Sample discovery problem



Return all samples that contain at least some user-defined fraction θ of k -mers present in the query string.

Recall the Bloom Filter

- For a set of size N , store an array of M bits Use k different hash functions, $\{h_0, \dots, h_{k-1}\}$
- To insert e , set $A[h_i(e)] = 1$ for $0 < i < k$
- To query for e , check if $A[h_i(e)] = 1$ for $0 < i < k$



Is element e in my set S ?

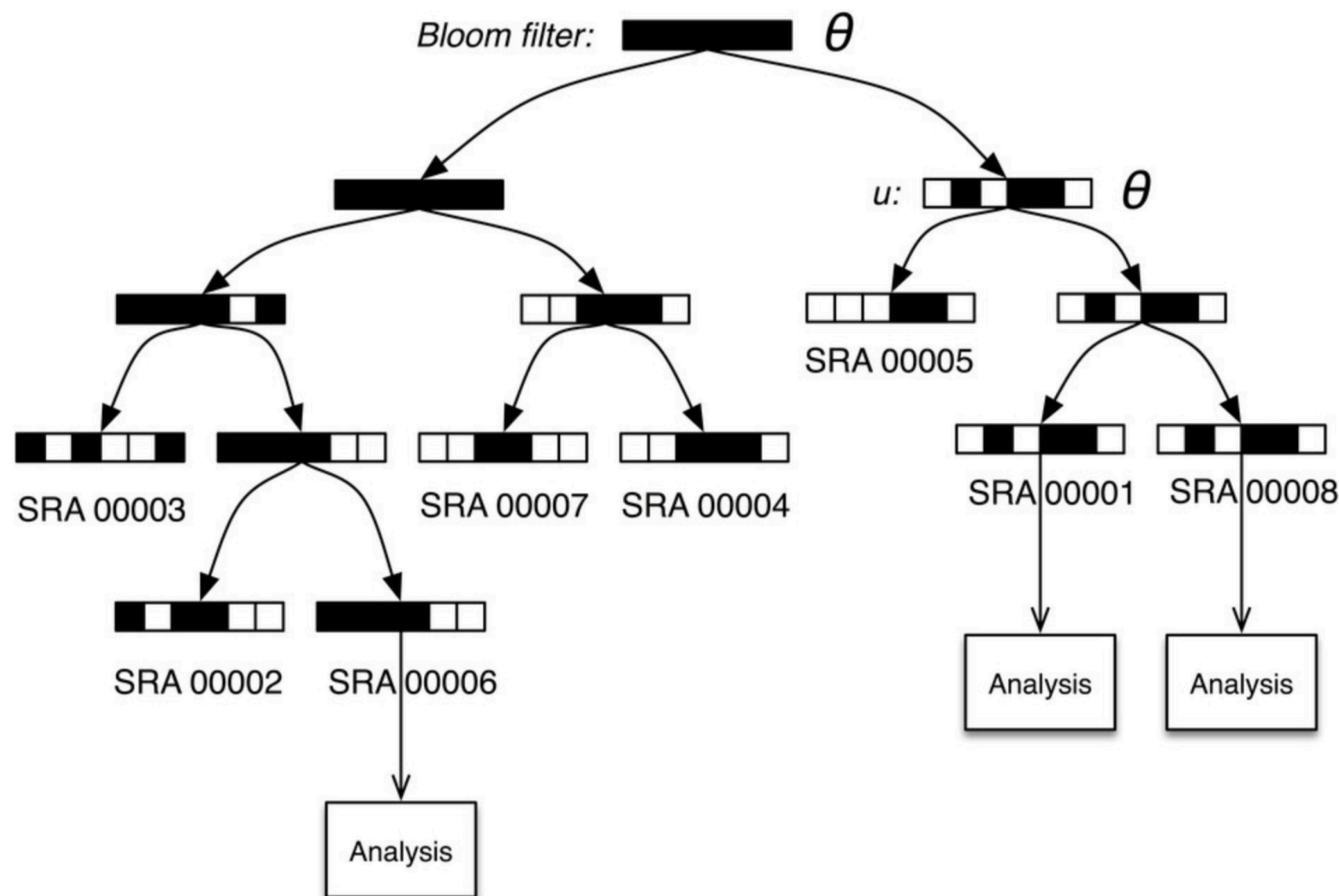
If yes, **always** say yes

If no, say no **with large probability**

Useful b/c: if we can tolerate false positives, we can query our set in very small space!

Sequence Bloom Trees (S&K '16)

- A binary tree of bloom filters, where leaves represent the k-mer set of a single sample.
- Bloom filter of parent is logical union (= bitwise OR) of children.
- Check both children, stop descending into tree when θ threshold is not satisfied



One inefficiency of this approach is that ***all Bloom filters must be the same size.***

Two improved SBT-related papers (RECOMB '17)

Improved Search of Large Transcriptomic Sequencing Databases Using Split Sequence Bloom Trees

Brad Solomon¹ and Carl Kingsford^{*1}

AllSome Sequence Bloom Trees

Chen Sun^{*1}, Robert S. Harris^{*2} Rayan Chikhi³, and Paul Medvedev^{†1,4,5}

Both papers share a very interesting core idea, but each also introduces its own, distinct improvements as well.

Happy to chat about details offline

Split Sequence Bloom Trees

Split Sequence Bloom Trees : Solomon & Kingsford (RECOMB '17)

Build

Data Index	SBT	Split SBT
Build Time	18 Hr	78 Hr
Compression Time	17 Hr	19 Hr
Compressed Size	200 GB	39.7 GB

Small enough
to fit in RAM
on a "reasonable"
server.

Build statistics for SBT & SSBT constructed from a 2652 experiment set. The sizes are the total disk space required to store a Bloom tree before or after compression.

In SSBT's case, this compression includes the removal of non-informative bits.

Query

Query Time:	$\theta=0.7$	$\theta=0.8$	$\theta=0.9$
SBT	20 Min	19 Min	17 Min
SSBT	3.7 Min	3.5 Min	3.2 Min
RAM SSBT	31 Sec	29 Sec	26 Sec

Starting to
approach
"interactive"

Comparison of query times using different thresholds θ for SBT and SSBT using the set of data at TPM 100 (i.e. high-expression transcripts).

A fundamentally different approach

Our initial idea: "The Bloom Filter is limiting. What can we get by replacing it with a *better* AMQ ?"



A General-Purpose Counting Filter: Making Every Bit Count

Prashant Pandey, Michael A. Bender, Rob Johnson, and Rob Patro

SIGMOD 2017

Interesting observation
about patterns of k-mer occurrence



Rainbowfish: A Succinct Colored de Bruijn Graph Representation*

Fatemeh Almodaresi¹, Prashant Pandey², and Rob Patro³

WABI 2017

"I bet we can exploit
that for large-scale search"



Mantis: A Fast, Small, and Exact Large-Scale Sequence-Search Index

Prashant Pandey¹, Fatemeh Almodaresi¹, Michael A. Bender¹, Michael Ferdman¹, Rob Johnson^{2,1}, and Rob Patro¹

RECOMB 2018 & Cell Systems

K-mer index



Squeakr: an exact and approximate κ -mer counting system

Prashant Pandey

¹Department of Computer Science, University of Maryland, USA, ²Department of Computer Science, Stony Brook University, USA and ³VMware Research, Palo Alto, CA 94301, USA

An Efficient, Scalable and Exact Representation of High-Dimensional Color Information Enabled via de Bruijn Graph Search

Fatemeh Almodaresi¹, Prashant Pandey¹, Michael Ferdman¹, Rob Johnson^{2,1}, and Rob Patro¹

RECOMB 2019

"I bet we can make
it even smaller"

An incrementally updatable and scalable system for large-scale sequence search using the Bentley-Saxe transformation

Fatemeh Almodaresi¹, Jamshed Khan¹, Sergey Madaminov², Michael Ferdman², Rob Johnson³, Prashant Pandey³ and Rob Patro^{1,*}

¹Department of Computer Science, University of Maryland, USA, ²Department of Computer Science, Stony Brook University, USA and ³VMware Research, Palo Alto, CA 94301, USA

Bioinformatics 2022

"I bet we can make
it scale and updatable"



The Counting Quotient Filter (COF)

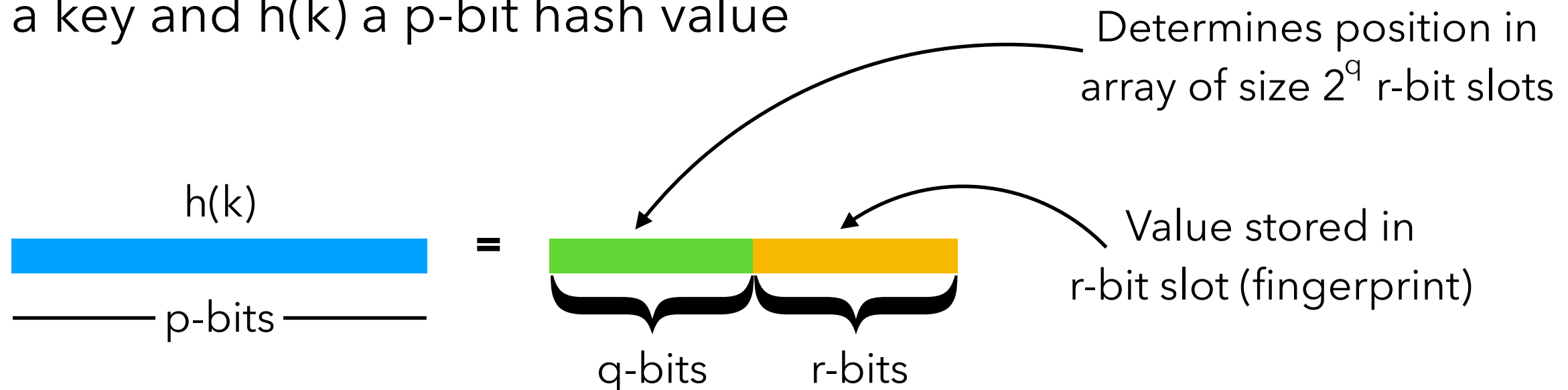
Approximate *Multiset* Representation

	0	1	2	3	4	5	6	7
occupied	0	1	0	1	0	0	0	1
runends	0	0	0	1	0	1	0	1
remainders		$h_1(a)$	$h_1(b)$	$h_1(c)$	$h_1(d)$	$h_1(e)$		$h_1(f)$

$\longleftarrow 2^q \longrightarrow$

Works based on quotienting* & fingerprinting keys

Let k be a key and $h(k)$ a p -bit hash value



Clever encoding allows low-overhead storage of element counts
(use *key* slots to store *values* in base 2^r-1 ; smaller values \Rightarrow fewer bits)

Careful engineering & use of efficient rank & select to resolve collisions leads to a **fast, cache-friendly** data structure

* Idea goes back at least to Knuth (TACOP vol 3)

Mantis

Observation 1 : If I want to index N k -mers over E experiments, there are $\leq \min(N, 2^{|E|})$ possible distinct “patterns of occurrence” of the k -mers ... there are usually *many* fewer.

Observation 2 : These patterns of occurrence are *far* from uniform. Specifically, k -mers don't occur independently; occurrences are *highly correlated*.

Why? Consider e.g. a gene G (~ 1000 k -mers). If it is present in an experiment at moderate to high abundance, we will likely observe *all of its k -mers*.

What if we add a layer of indirection: Store each distinct pattern (color class) only once. *Label* each pattern with an index, s.t. frequent patterns get small numbers (think Huffman encoding)

David Wheeler approves ... we think.

<https://github.com/splatlab/mantis>

The Mantis Index: Core Idea

Input Experiments

E ₁	E ₂	E ₃	E ₄
	ACTG	ACTG	
ACTT			
		CTTG	CTTG
	TTTC	TTTC	
	GCGT	GCGT	GCGT
	AGCC	AGCC	



Mantis Index

CQF

k-mer	Color ID
ACTG	0
ACTT	10
CTTG	1
TTTC	0
GCGT	11
AGCC	0

Color class table

	E ₁	E ₂	E ₃	E ₄
0	0	1	1	0
10	0	0	1	1
1	1	0	0	0
11	0	1	1	1

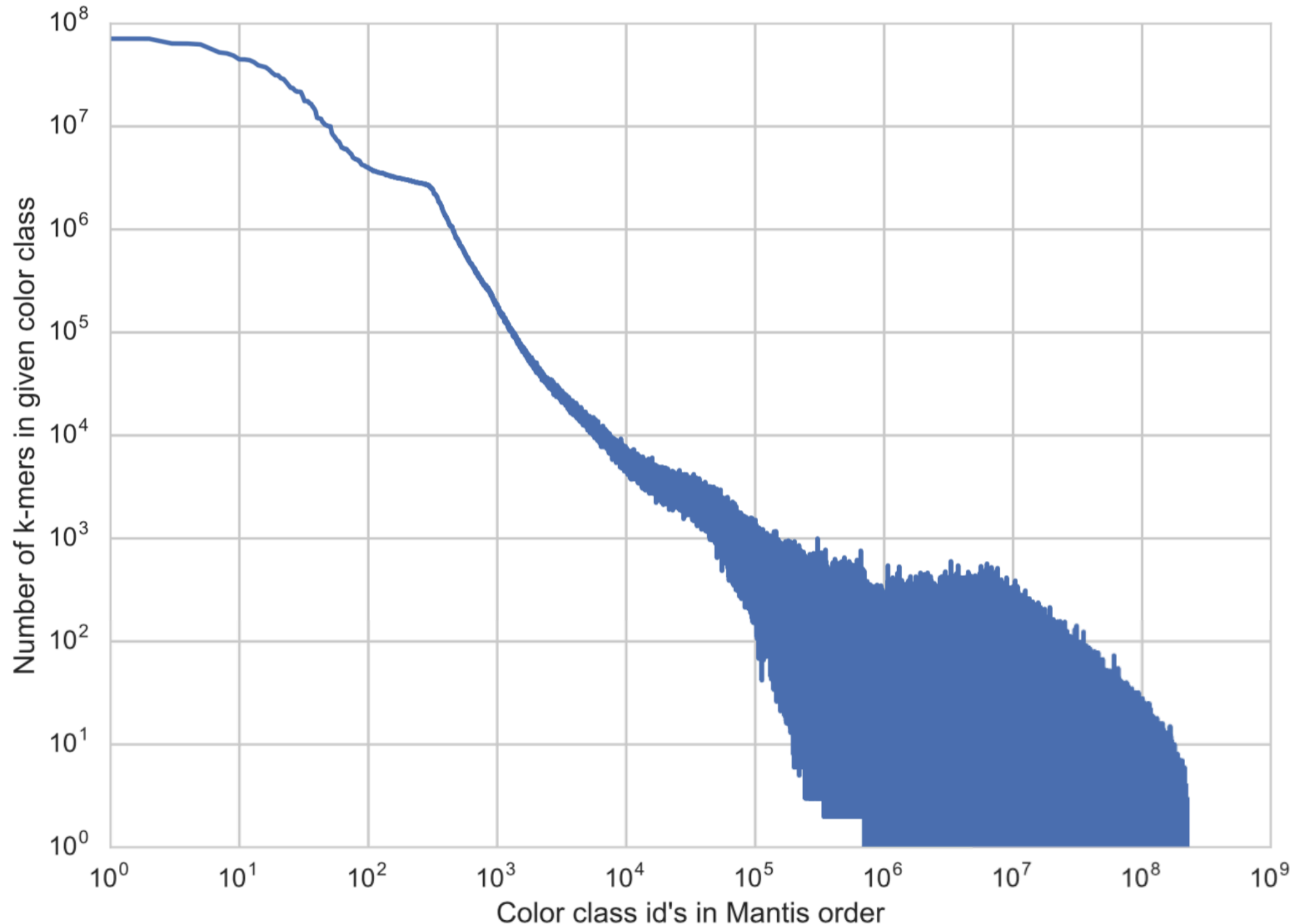
Compressed using RRR*

No tree!

- Build a CQF for each input experiment
(*can be different sizes, since CQFs of different sizes are mergeable*)
- Combine them via multi-way merge
- CQF : **key** = k-mer, **value** = color class ID
- *Estimate* a good ordering of color class IDs from first few million k-mers

Most k-mers have small IDs?

The distribution of k-mers / color class is *highly skewed*



~3.7 Billion k-mers from ~2,600 distinct sequencing experiments

Mantis : Comparing to SSBT

Construction Time – How long does it take to build the index?

Index Size – How large is the index, in terms of storage space?

Query Performance – How long does it take to execute queries?

Result Accuracy – How many FP positives are included in query results?

Bonus: If the **quotient** + **remainder** bits = **original key size** & we use an invertible hash, the CQF is *exact*.

Mantis is compact enough to **exactly** index all experiments.

This lets us ask useful questions about how other approaches perform.

Mantis : Construction Time & Index Size

Indexed 2,652 human RNA-seq (gene expression) experiments
~**4.5TB** of (Gzip compressed) data

Table 1. Time and Space Measurement for Mantis and SSBT

Tool	Mantis	SSBT
Build time	03 hr 56 min	97 hr
Representation size.	32 GB	39.7 GB

- Mantis can be constructed ~24x faster than a comparable SSBT
- The final Mantis representation is ~20% smaller than the comparable SSBT representation.

Note: both results assume you already have per-experiment AMQs (either Bloom Filters or CQFs)

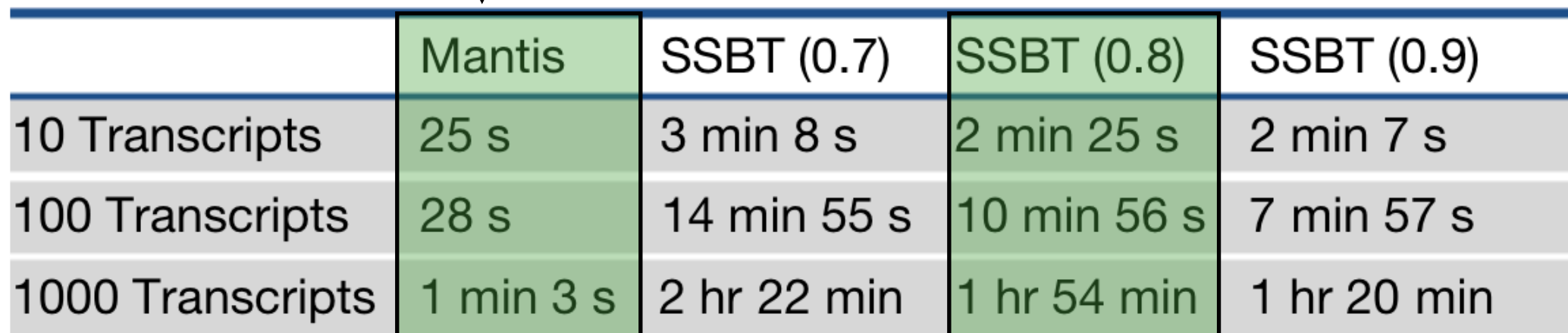
Mantis : Query Speed

Querying for the presence of randomly selected genes across all 2,652 experiments.

Query **includes index loading**

(will return to this later)

θ threshold for SSBT query



	Mantis	SSBT (0.7)	SSBT (0.8)	SSBT (0.9)
10 Transcripts	25 s	3 min 8 s	2 min 25 s	2 min 7 s
100 Transcripts	28 s	14 min 55 s	10 min 56 s	7 min 57 s
1000 Transcripts	1 min 3 s	2 hr 22 min	1 hr 54 min	1 hr 20 min

- Mantis is ~6 – 10⁹x faster than (in memory) SSBT

Mantis doesn't require a θ threshold for queries, though one can be applied *post hoc*.

Mantis returns the *fraction* (true θ) of query k-mers contained in the experiment.

Mantis : Query Accuracy

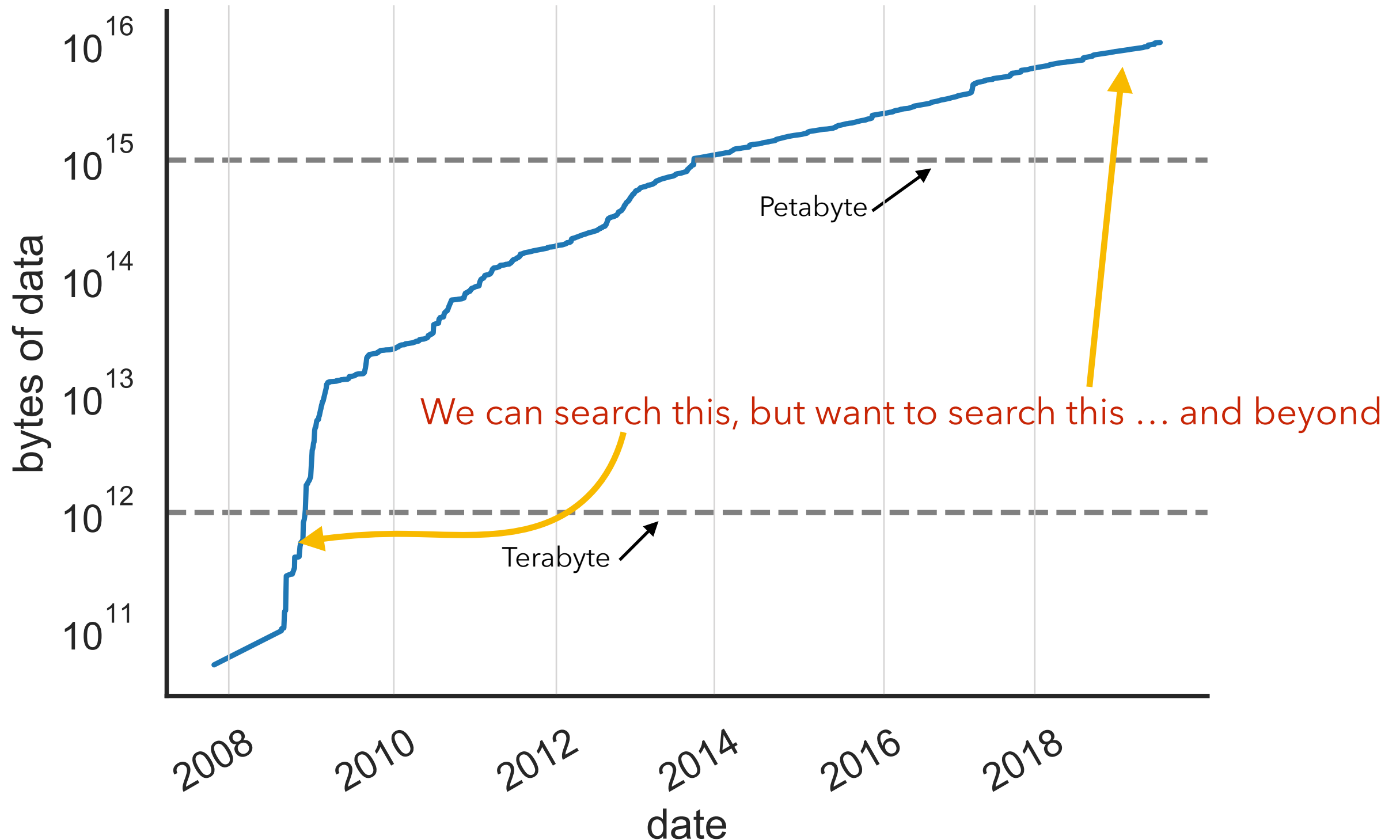
Querying for the presence of randomly selected genes across all 2,652 experiments. SSBT $\theta = 0.8$

	Both	Only Mantis	Only SSBT	Precision
10 Transcripts	2,018	19	1,476	0.577
100 Transcripts	22,466	146	10,588	0.679
1000 Transcripts	160,188	1,409	95,606	0.626

- Recall : Mantis is exact! Returns *only* experiments having $\geq \theta$ fraction of the query k-mers.

Due to a small number of corrupted SSBT filters – able to discover this b/c of Mantis' exact nature.

Where are we now?



"It seems that some essentially new ... ideas are here needed"

– Paul Adrien Maurice Dirac*

Data from: [https://](https://www.ncbi.nlm.nih.gov/)

www.ncbi.nlm.nih.gov/

Some Remaining Challenges

- It improves greatly upon existing solutions; takes a different approach
- We demonstrate indexing on the order of 10^3 experiments, we really want to index on the order of $10^5 - 10^6$
- Can be made approximate while providing strong bounds :

Theorem 1. *A query for q k -mers with threshold θ returns only experiments containing at least $\theta q - O(\delta q + \log n)$ queried k -mers w.h.p.*

but maybe not enough

Key Observation:

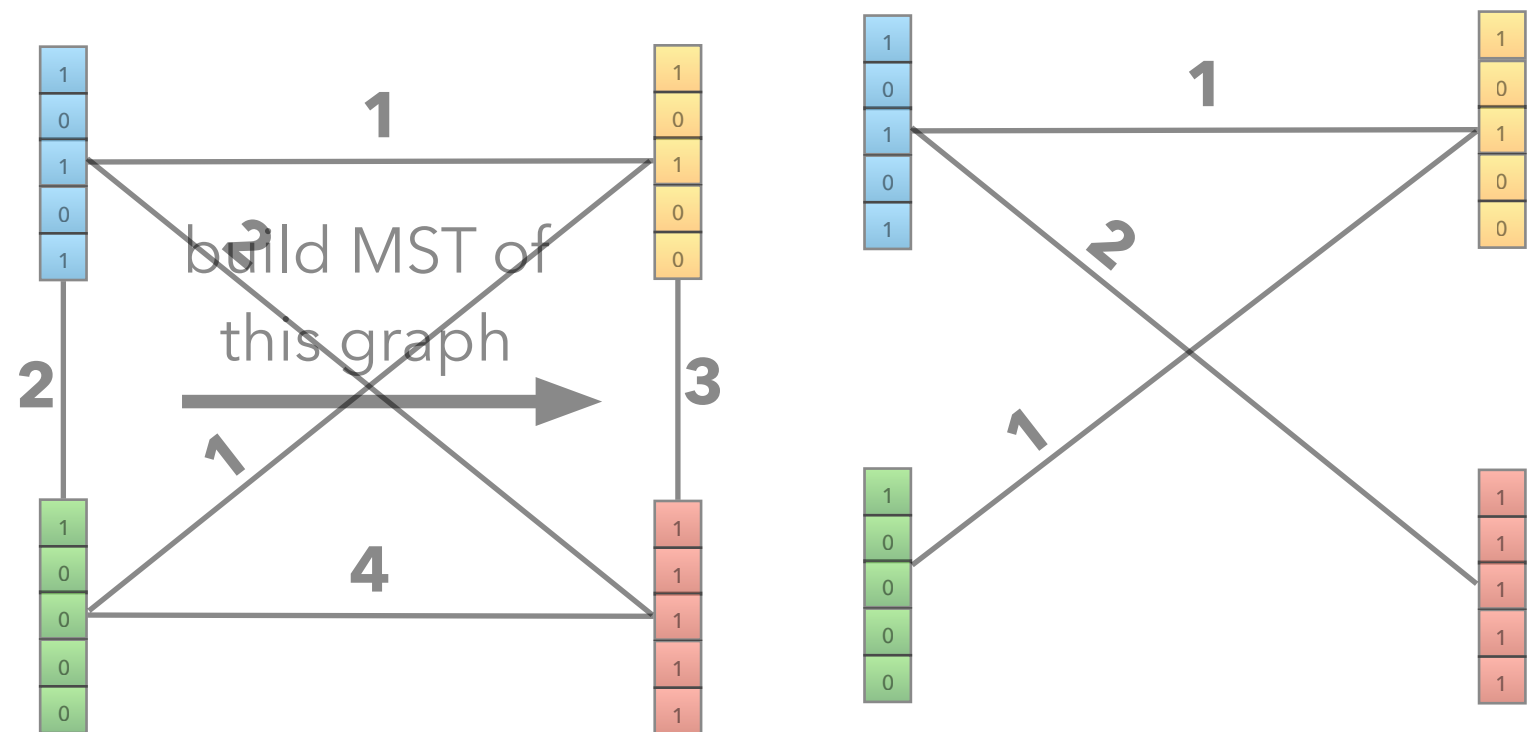
- K -mers grow at worst linearly
- Color classes increase super-linearly

Need a **fundamentally better** color class encoding; exploit *coherence* between rows of the color class matrix

Consider the following color class graph

Each color class is a vertex

Every pair of color classes is connected by an edge whose weight is the **hamming distance** between the color class vectors



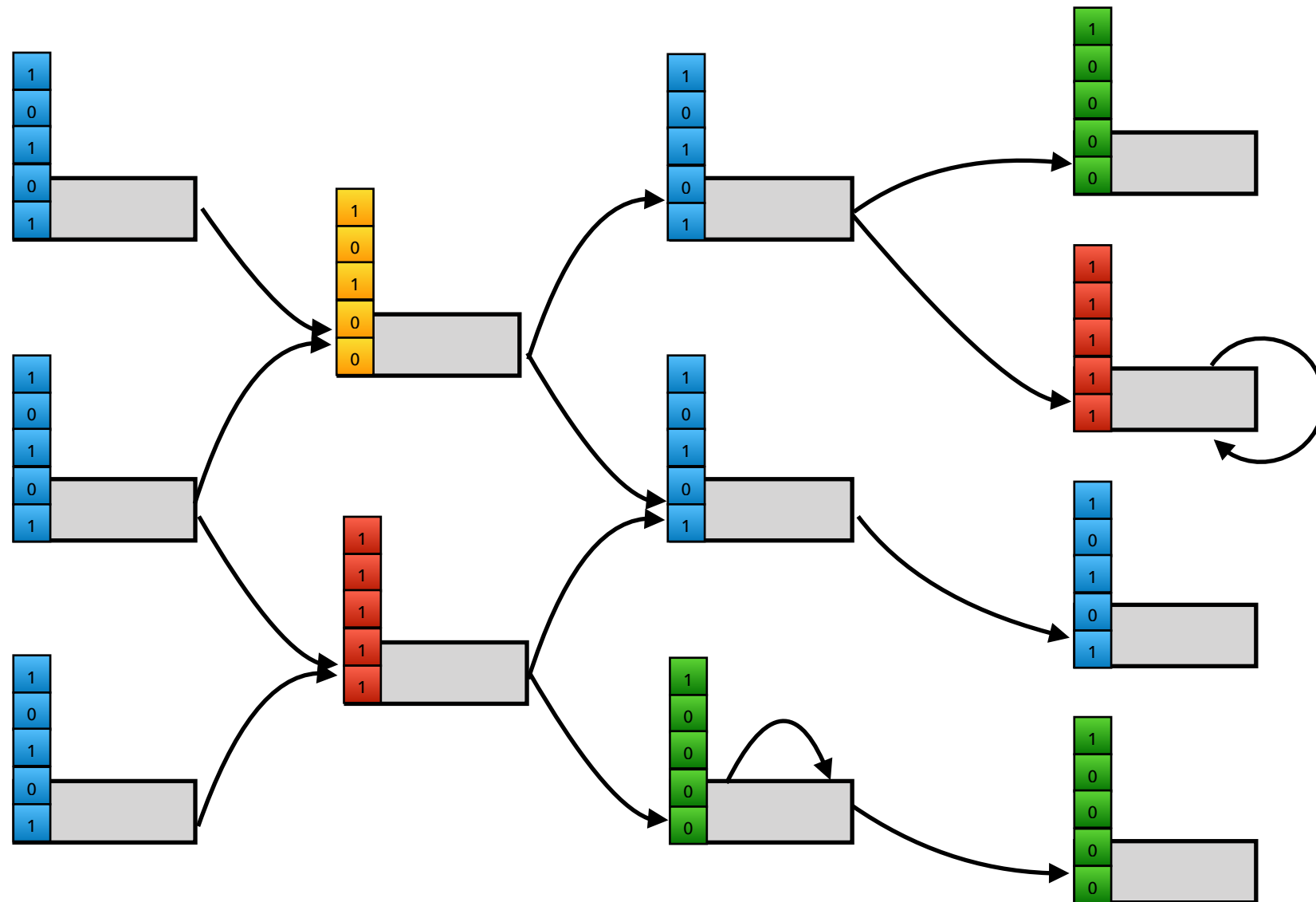
Unfortunately:

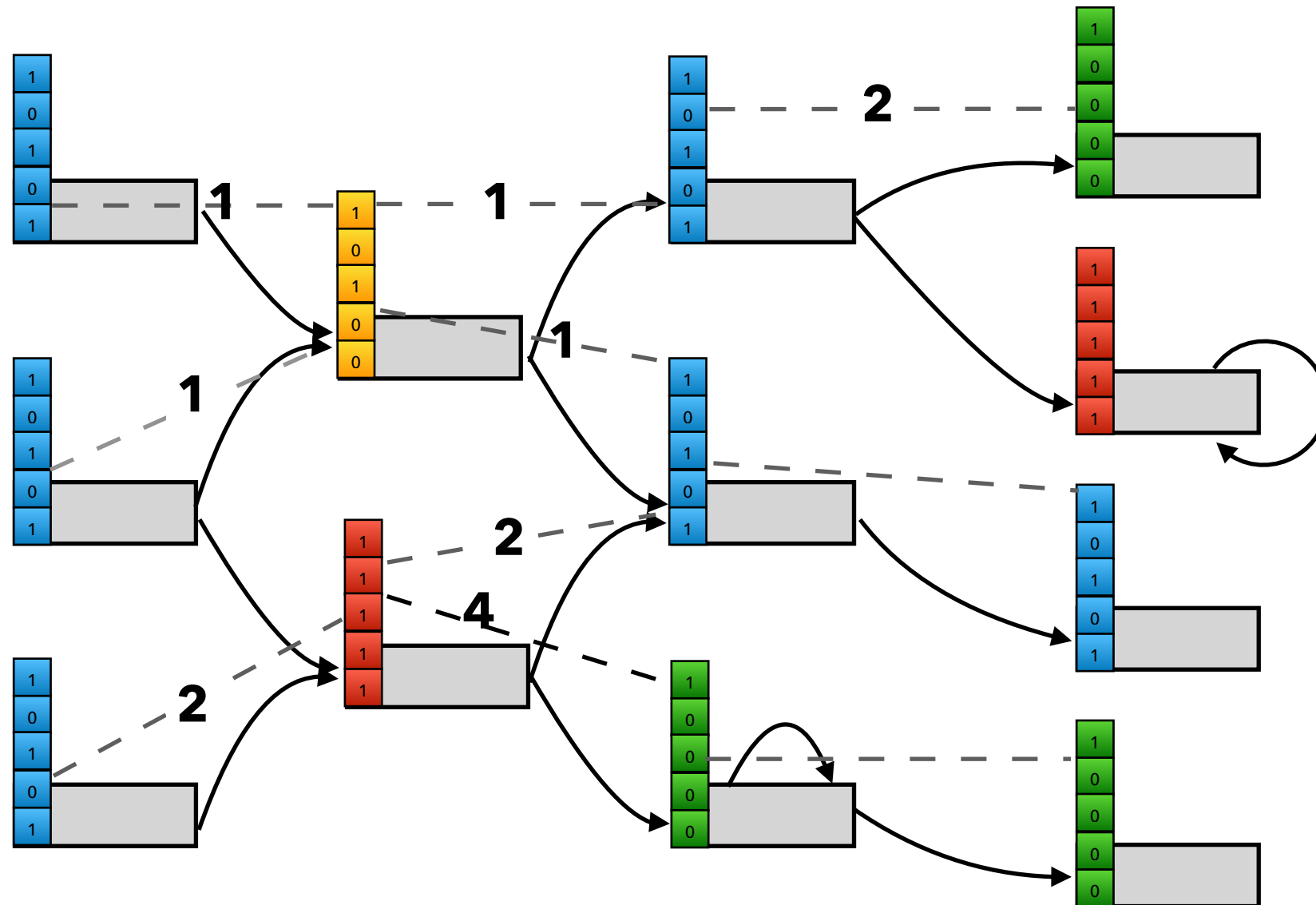
- 1) There are *many* color classes (full graph too big)
- 2) They are high-dimensional (# of experiments), neighbor search is very hard (LSH scheme seem to work poorly)

Mantis implicitly represents a colored dBG

Each CQF key represents a kmer \rightarrow can explicitly query neighbors

Each k-mer associated with color class id \rightarrow vector of occurrences





Use the **de Bruin graph** (dBG) as an efficient guide for near-neighbor search in the space of color classes!

1
0
1
0
1

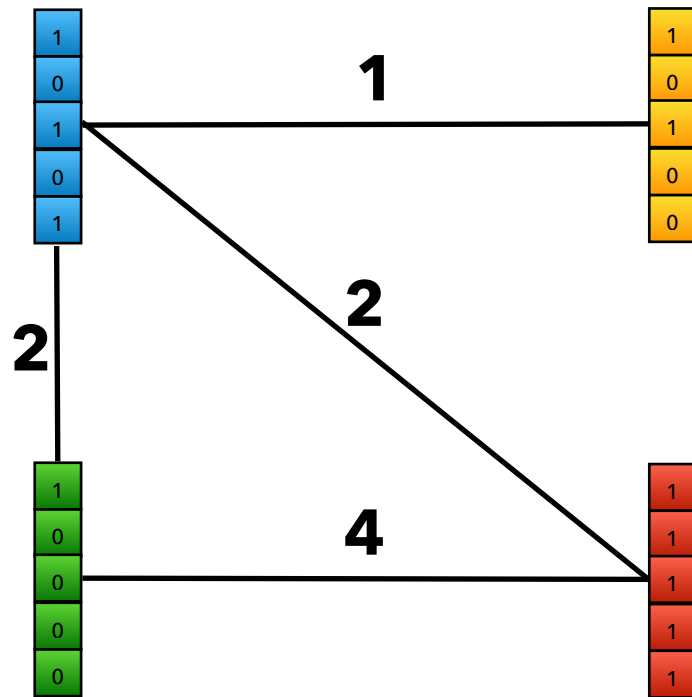
1
0
1
0
0

1
0
0
0
0

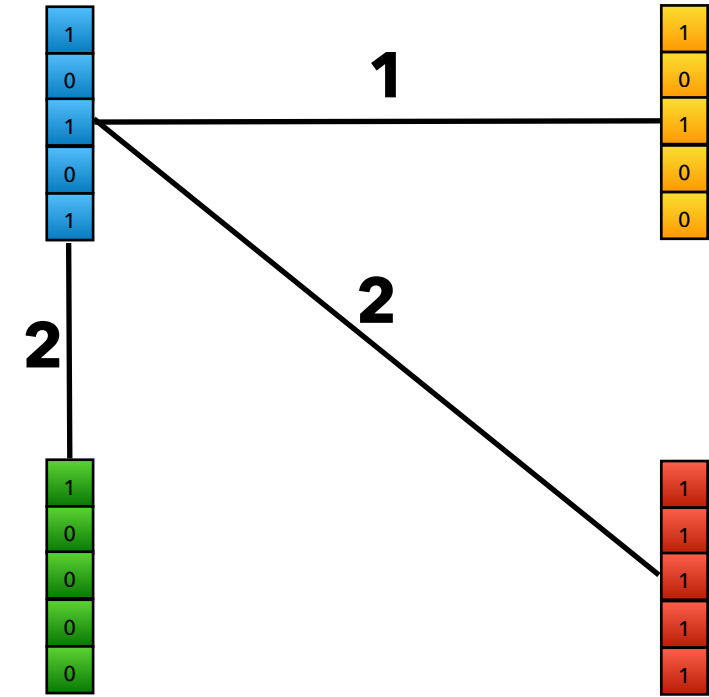
1
1
1
1
1

dBG common in genomics. Nodes u, v are k -mers & are *adjacent* if $k-1$ suffix of u is the same as $k-1$ prefix of v

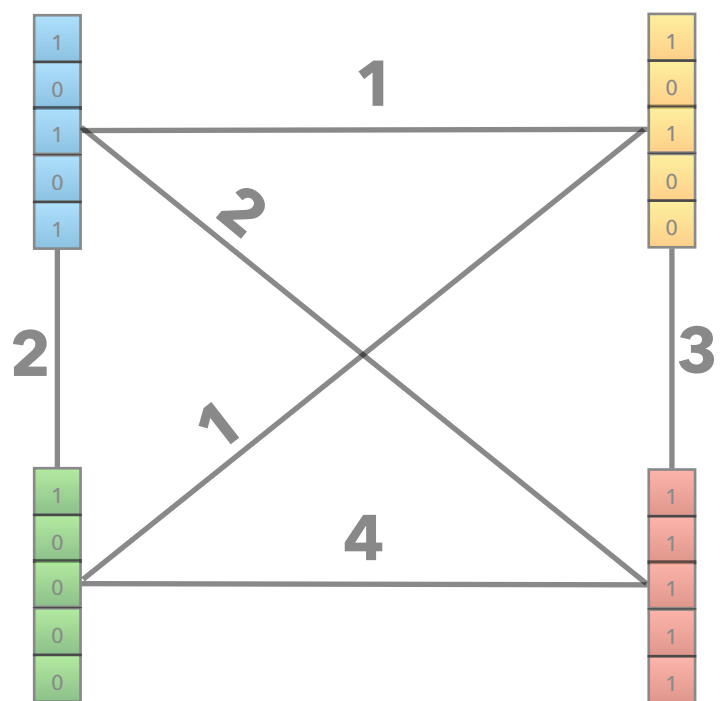
CCG derived from dbG



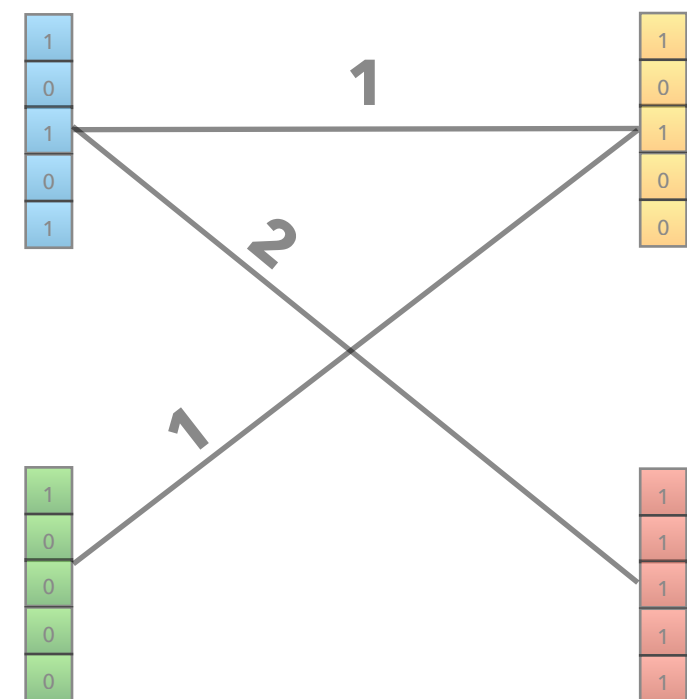
MST on our Graph



Complete CCG

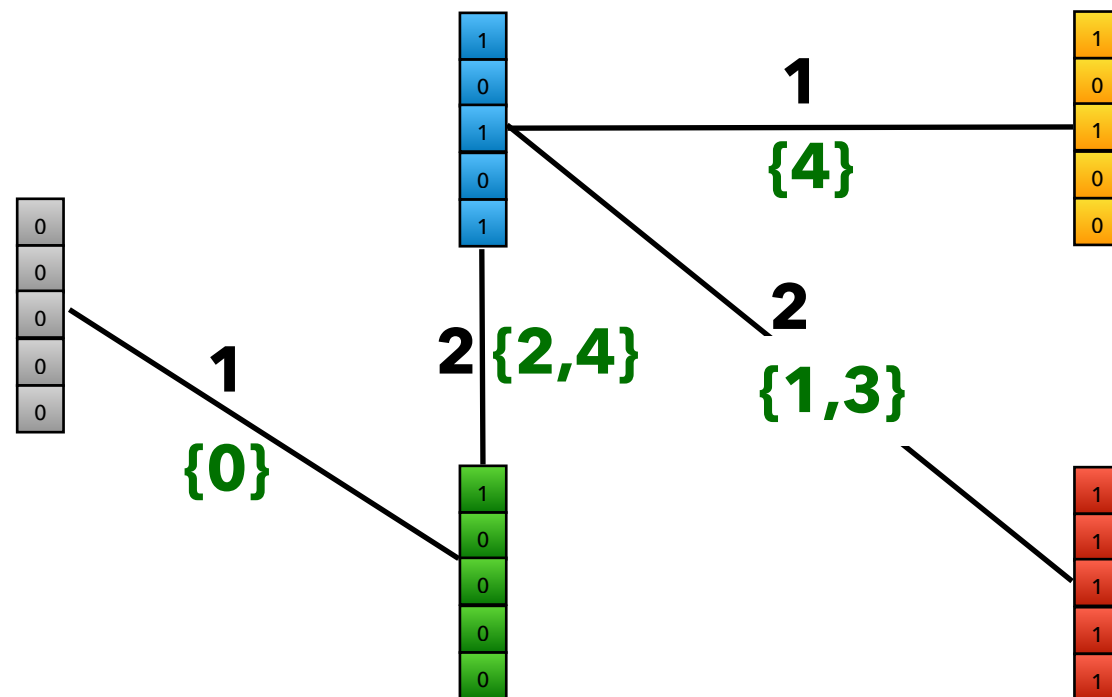


Optimal MST



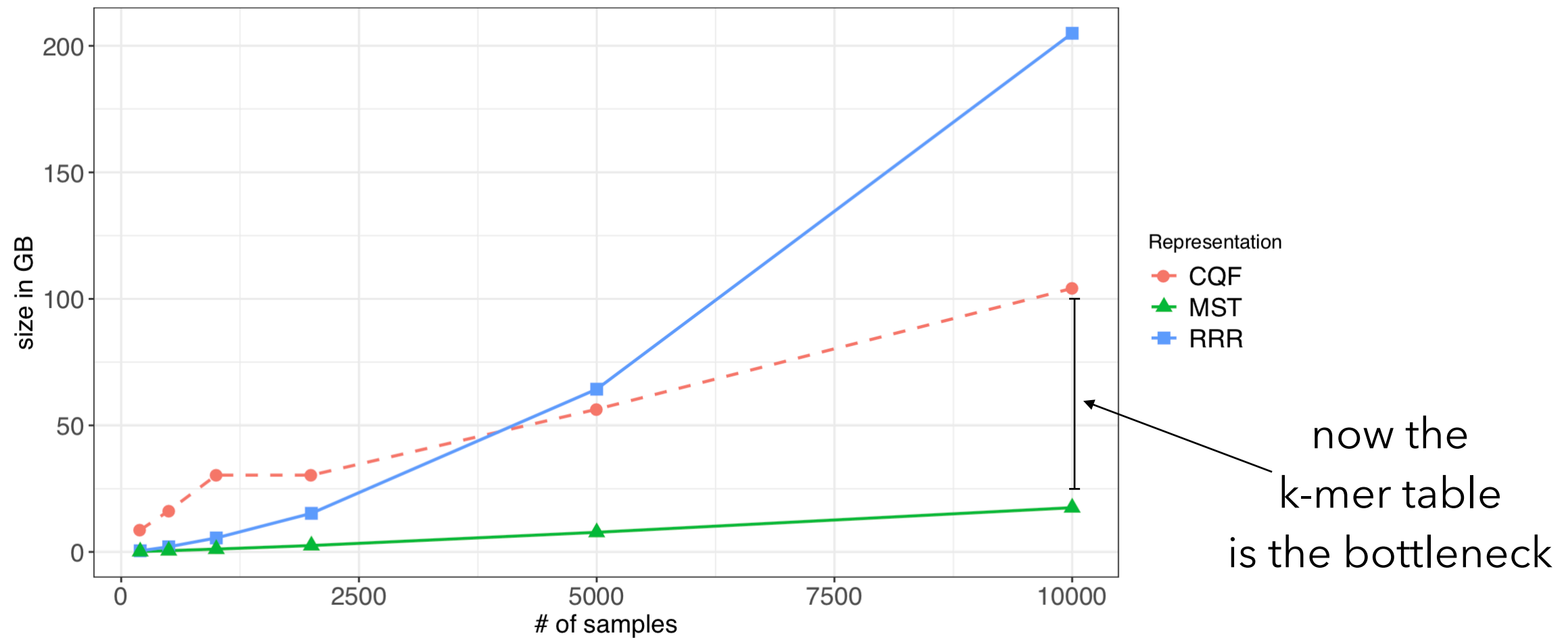
The MST efficiently encodes related color classes

Augment with all 0 color class to guarantee one, connected MST



To reconstruct a vector, walk from your node to the root, flipping the parity of the positions you encounter on each edge.

The MST approach scales very well



Dataset	# samples	MST					$\frac{\text{size}(MST)}{\text{size}(RRR)}$
		RRR matrix	Total space	Parent vector	Delta vector	Boundary bit-vector	
<i>H. sapiens</i> RNA-seq samples	200	0.42	0.15	0.08	0.06	0.01	0.37
	500	1.89	0.46	0.2	0.24	0.03	0.24
	1,000	5.14	1.03	0.37	0.6	0.06	0.2
	2,000	14.2	2.35	0.71	1.5	0.14	0.17
	5,000	59.89	7.21	1.72	5.1	0.39	0.12
	10,000	190.89	16.28	3.37	12.06	0.86	0.085
Blood, Brain, Breast (BBB)	2586	15.8	2.66	0.63	1.88	0.16	0.17

Improvement over RRR improves with # of samples

dataset from SBT / SSBT / Mantis paper

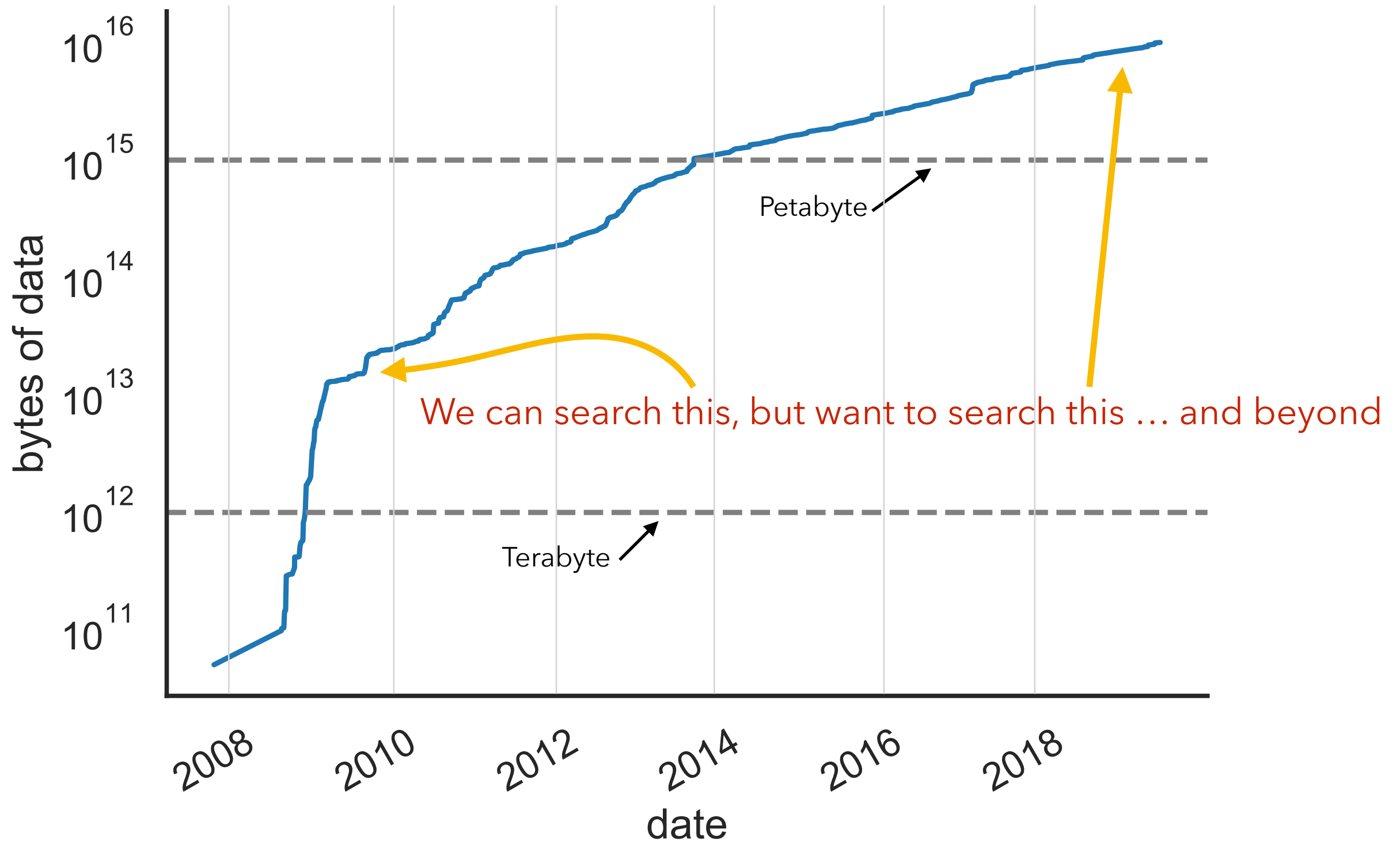
How does MST approach affect query time?

One concern is that replacing $O(1)$ lookup with MST-based decoding will make lookup slow; does it?

Turns out a caching strategy (an LRU over popular internal nodes) keeps it just as fast as lookup in the RRR matrix

	Mantis with MST			Mantis		
	index load + query	query	space	index load + query	query	space
10 Transcripts	1 min 10 sec	0.3 sec	118GB	32 min 59 sec	0.5 sec	290GB
100 Transcripts	1 min 17 sec	8 sec	119GB	34 min 33 sec	11 sec	290GB
1000 Transcripts	2 min 29 sec	79 sec	120GB	46 min 4 sec	80 sec	290GB

Where we are now?



"It seems that some essentially new ... ideas are here needed"

– Paul Adrien Maurice Dirac*

Data from: [https://](https://www.ncbi.nlm.nih.gov/)

www.ncbi.nlm.nih.gov/

Some Remaining Challenges

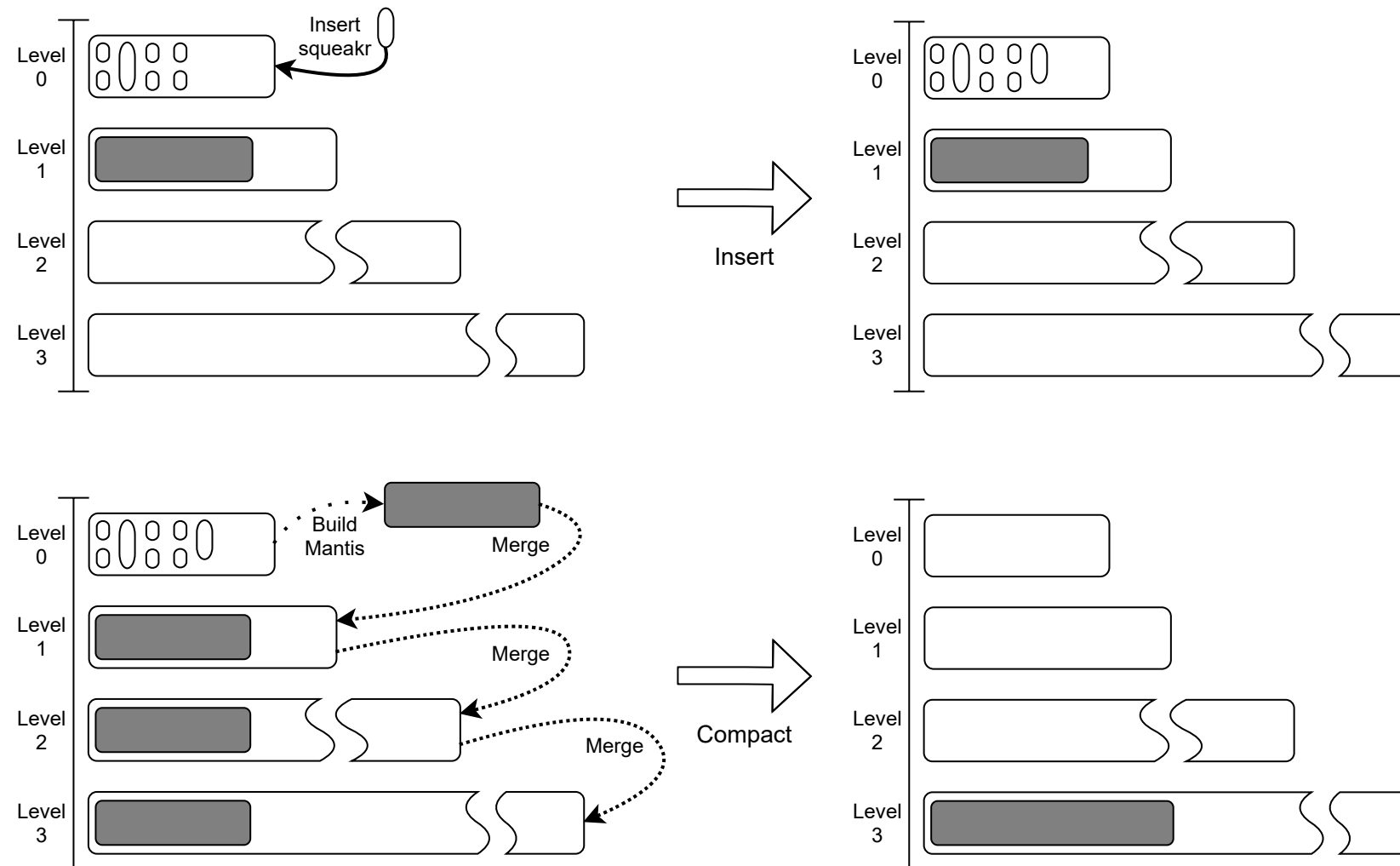
- We can scale to even larger datasets by compressing color class representation.
- We demonstrate indexing on the order of 10^3 experiments, we really want to index on the order of $10^5 - 10^6$
- We need to scale out of RAM and also support adding new experiments.

Key Observation:

- We can take a static representation and make it updatable using the Bentley-Saxe construction[Bentley and Saxe (1980).].
- We can reduce the memory usage using minimizers.

Need a **fundamentally better** construction which can support adding new experiments and can scale out of RAM to disk.

Mantis-LSM design



- Level 0 resizes in RAM
- L1...Ln remain on disk
- Level grow in size exponentially
- **Minimizers to partition the k-mer index on disk**
- **Helps to minimize RAM usage during merging and queries.**

Mantis-LSM design

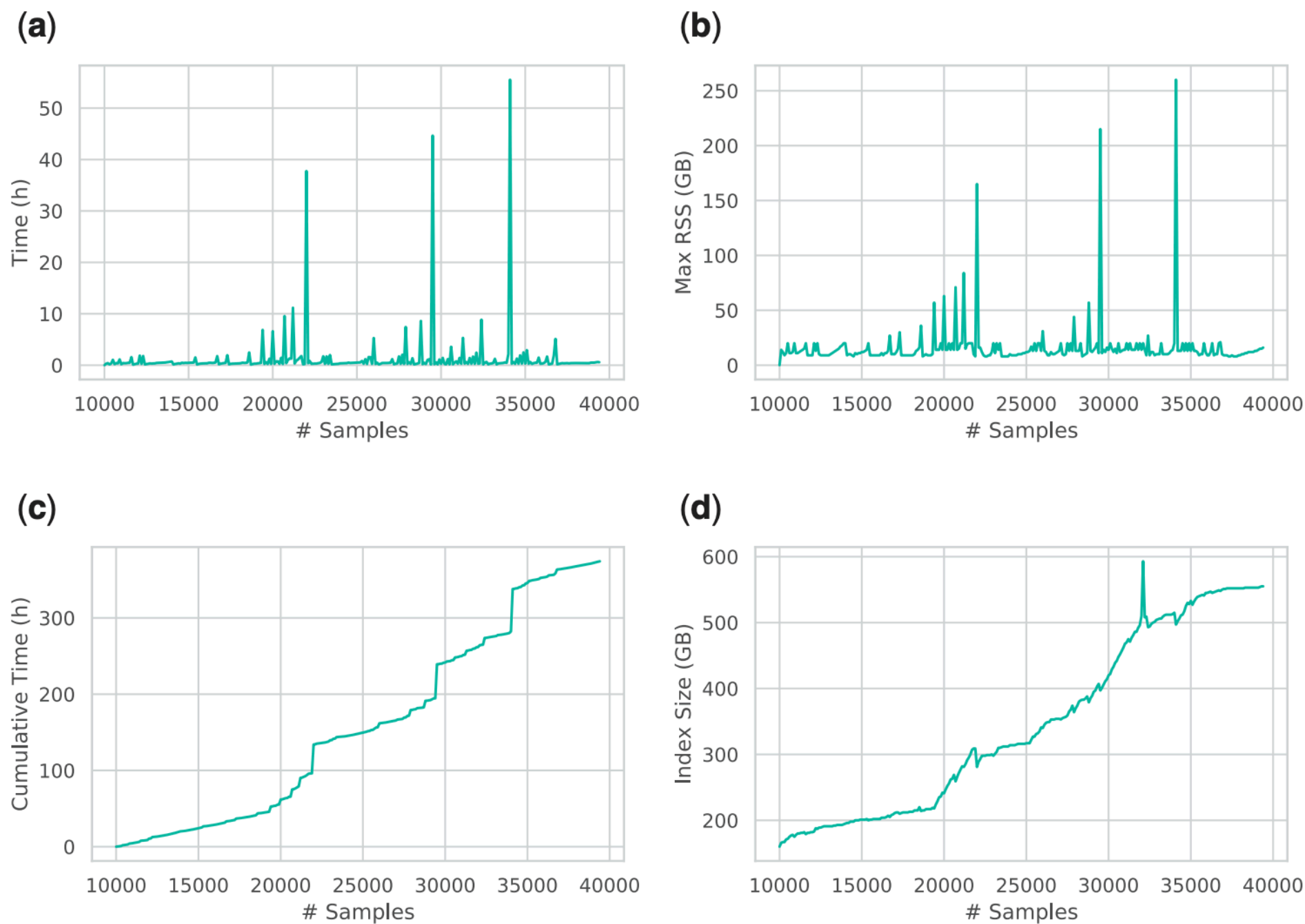


Fig. 4. Performance of the Dynamic Mantis update process. The spikes in time (Fig. a) and memory (Fig. b) happen when the cascading merge happens with deeper and thus larger indexes. Cumulative Time (Fig. c) shows the total time required to add all the samples up to the current one, and index size (Fig. d) is total size of the index

Where we are now?



"It seems that some essentially new ... ideas are here needed"

– Paul Adrien Maurice Dirac*

Data from: [https://](https://www.ncbi.nlm.nih.gov/)

www.ncbi.nlm.nih.gov/

*Principles of Quantum Mechanics 2nd edition, Chapter XIII, Section 81 (p. 297)

A special thanks to my collaborators!!

Funding:



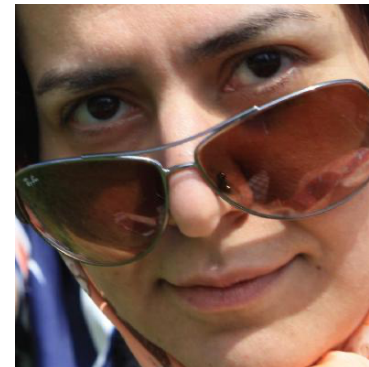
Jamshed Khan
(UMD)



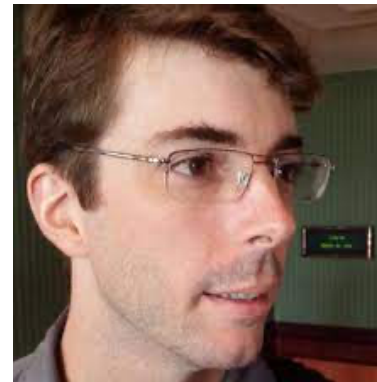
Mike Ferdman
(Stony Brook)



Fatemeh Almodaresi
(OICR)



Rob Johnson
(VMware Research)



Rob Patro
(UMD)



Michael Bender
(Stony Brook)



<https://prashantpandey.github.io/>