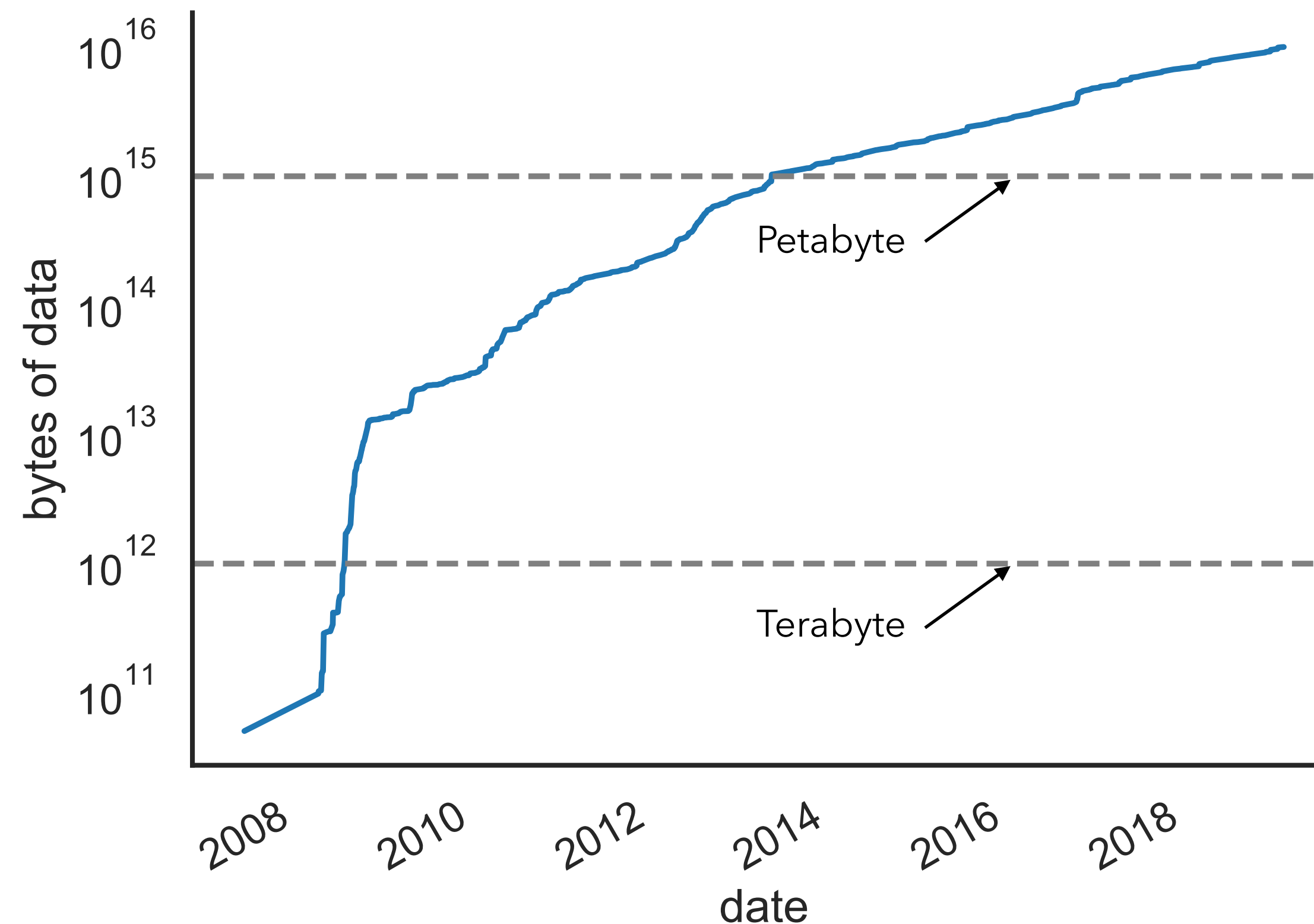


Hashing and AMQs for large-scale sequence search

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**

Q: What if I find e.g., a new disease-related gene, and want to see if it appeared in other experiments?

A: (basically) Too bad!

* there is an SRA BLAST, but functionality is limited

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 query sequence is: TGAAAAGGGTAACCTCAAAGCTAAAAGCCCAAGAAGGGGAAGCCCCATTGCAGCCGCAAC CCTGTCCTTGTGAGAGGAATTGGCAGGTATTCCCGATC. The search results table is as follows:

	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

COMMENTARY

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>

Fast search of thousands of short-read sequencing experiments.

SBT introduced by **Solomon and Kingsford.**

Nature biotechnology. 2016 doi: 10.1038/nbt.3442

Problem:

The vast repository of publicly-available data (e.g., the SRA) is essentially unsearchable by sequence. Current solutions (BLAST, STAR) too slow. What if I find a novel txp and want to search the SRA for it?

Solution:

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.

The bloom filter

Bloom Filters

Originally designed to answer *probabilistic* membership queries:

Is element e in my set S ?

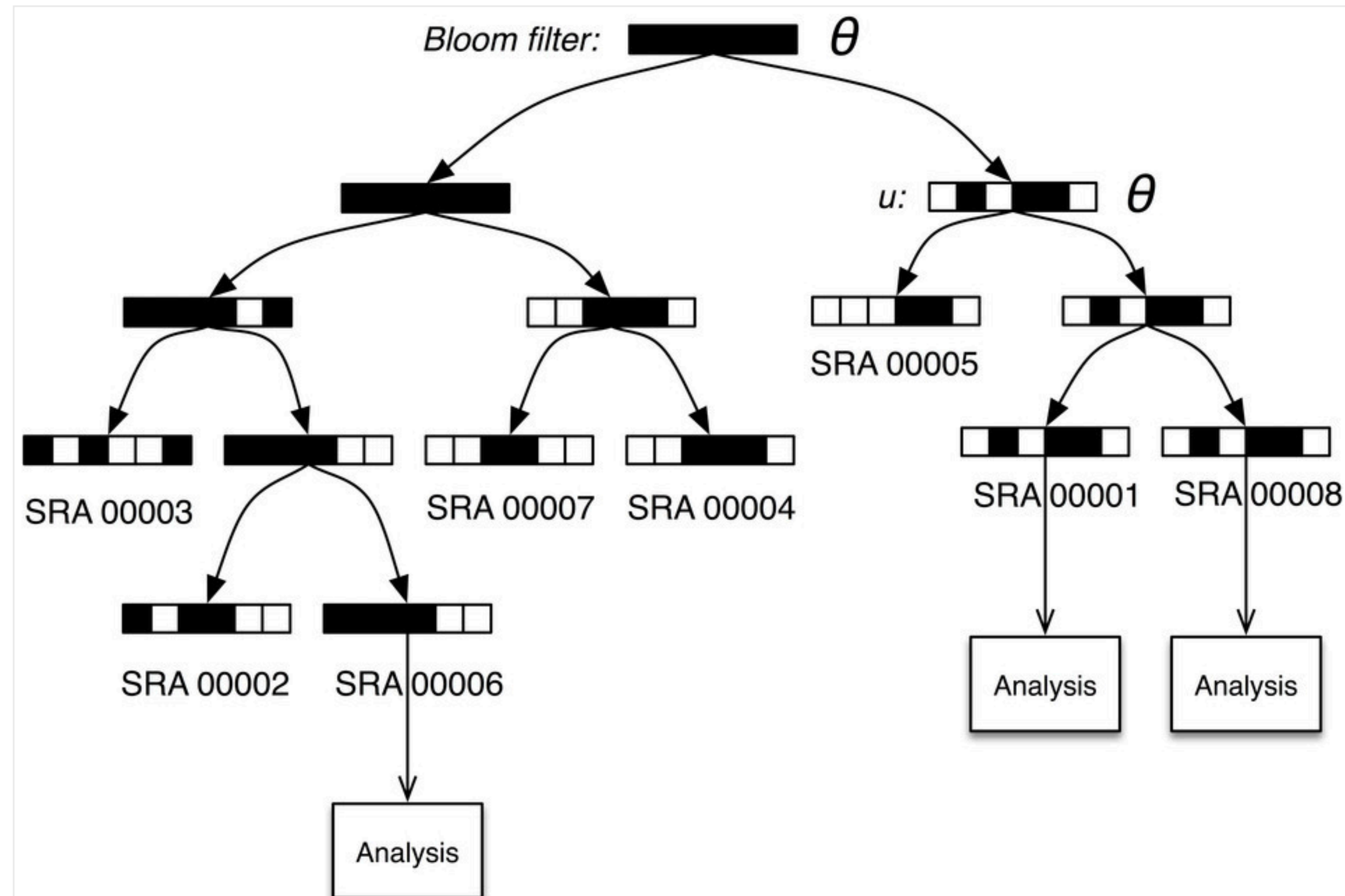
If yes, **always** say yes

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

False positives can happen; false negatives cannot.

SBT

An SBT is a binary tree of bloom filters, where leaves represent the k-mer set of a single *sample*.



SBT Operations

Construction (repeatedly insert samples s):

Let $b(s)$ be the bloom filter of sample s

Use $b(s)$ to walk from the root of T to the leaves

For a node u :

If u has a single child, insert $b(s)$ as the other child

If u has 2 children recurse into child with $<$ hamming dist to $b(s)$

If u is a leaf (an experiment), create a parent with filter $b(u) \cup b(s)$

SBT Operations

Query (given collection of k-mers K_q , parameter θ):

For a node u :

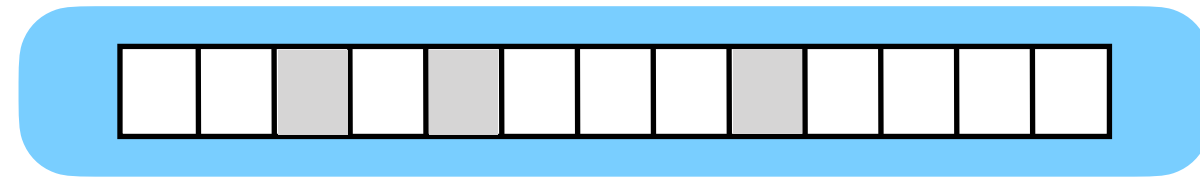
Hash elements of K_q and check if at least $\theta |K_q|$ k-mers exist

If not then this sub-tree cannot θ -match our query

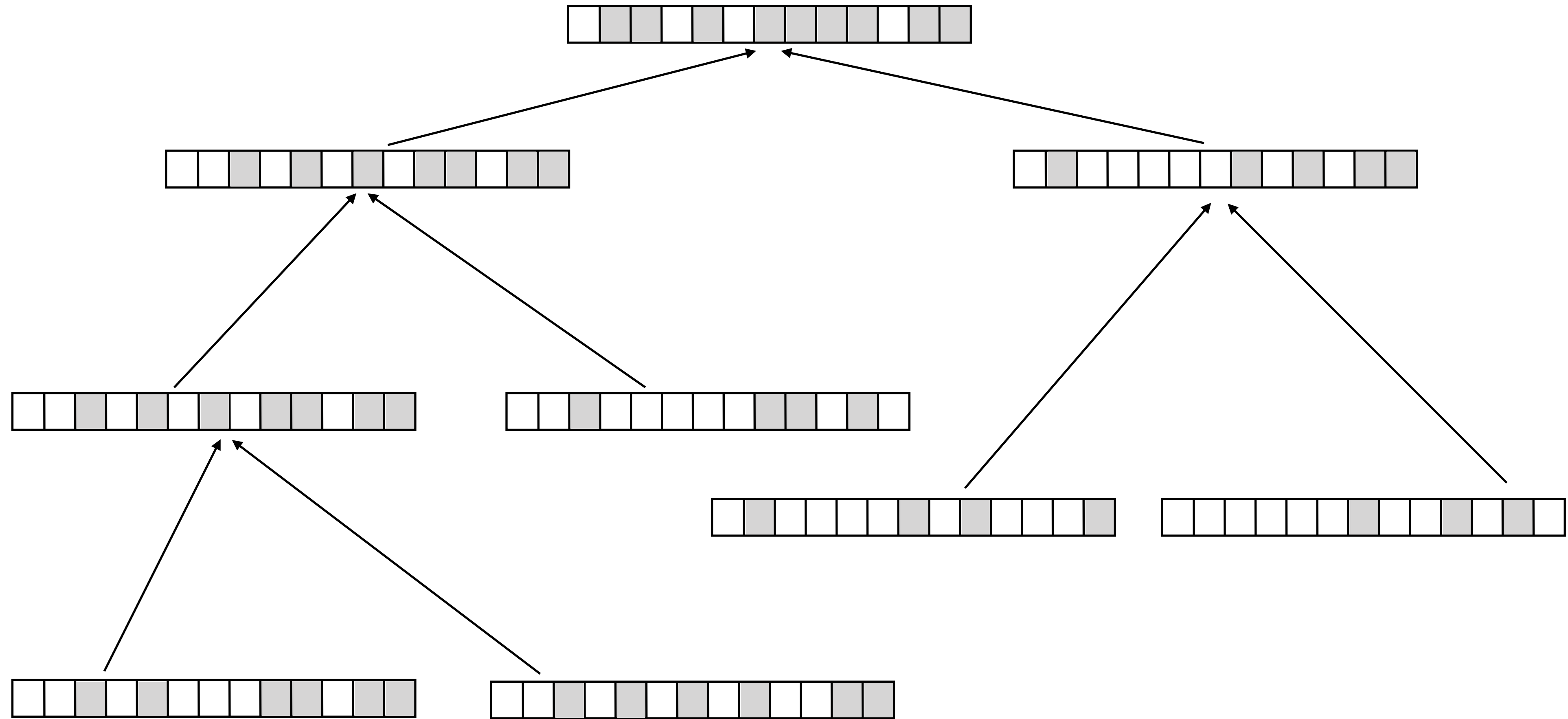
Else continue searching both children recursively

The implementation allows each query k-mer to be given a “weight” or importance.

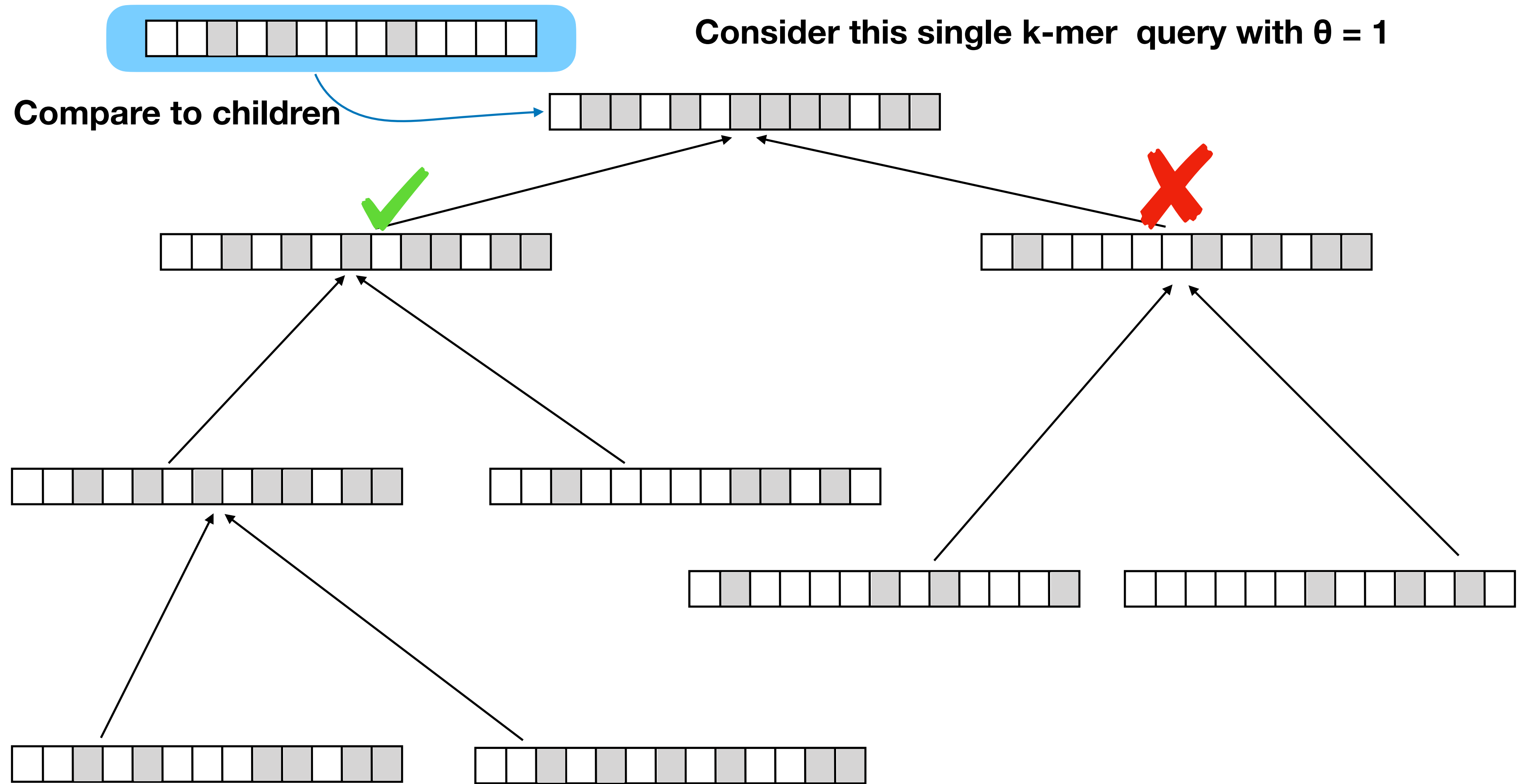
SBT Operations



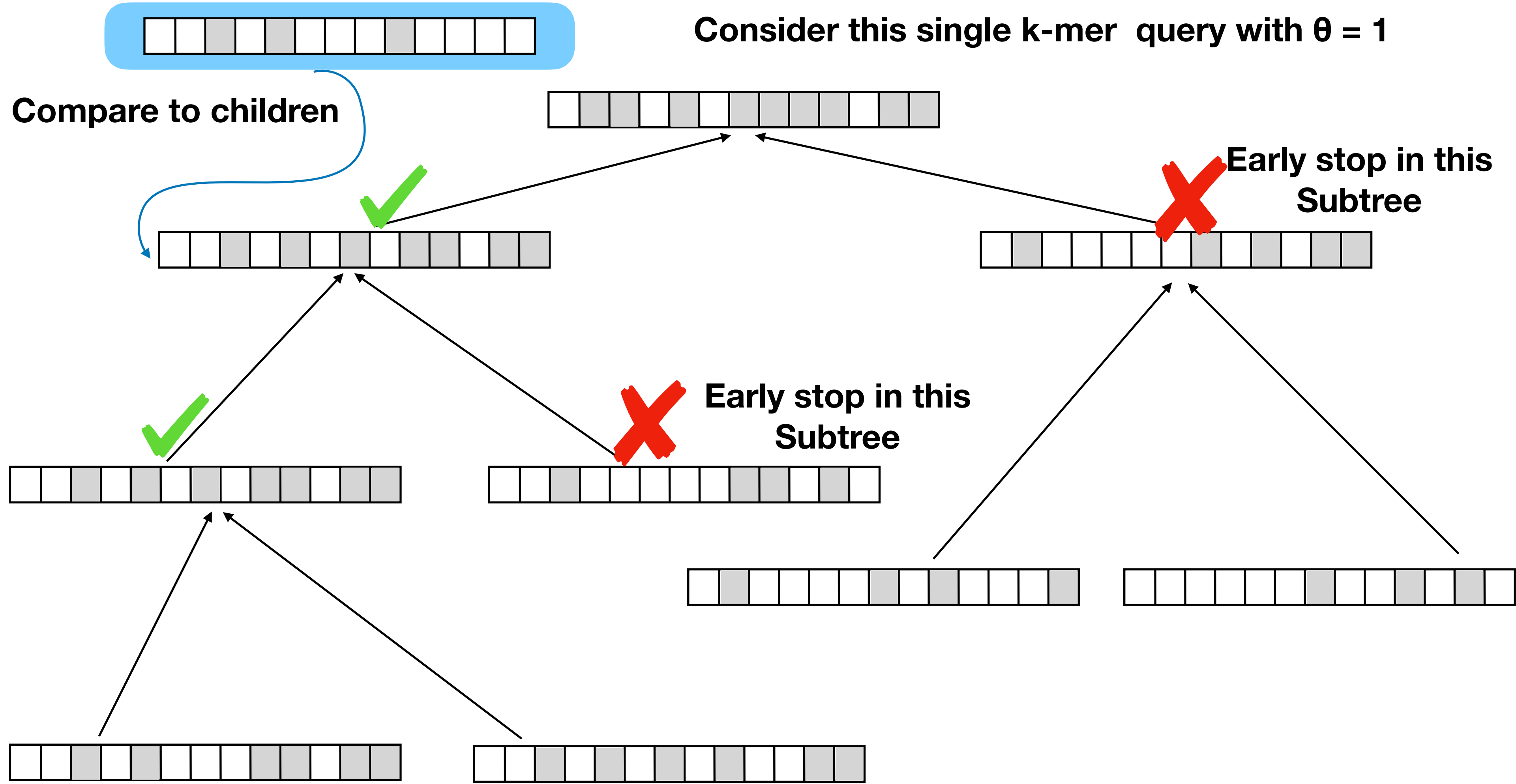
Consider this single k-mer query with $\theta = 1$



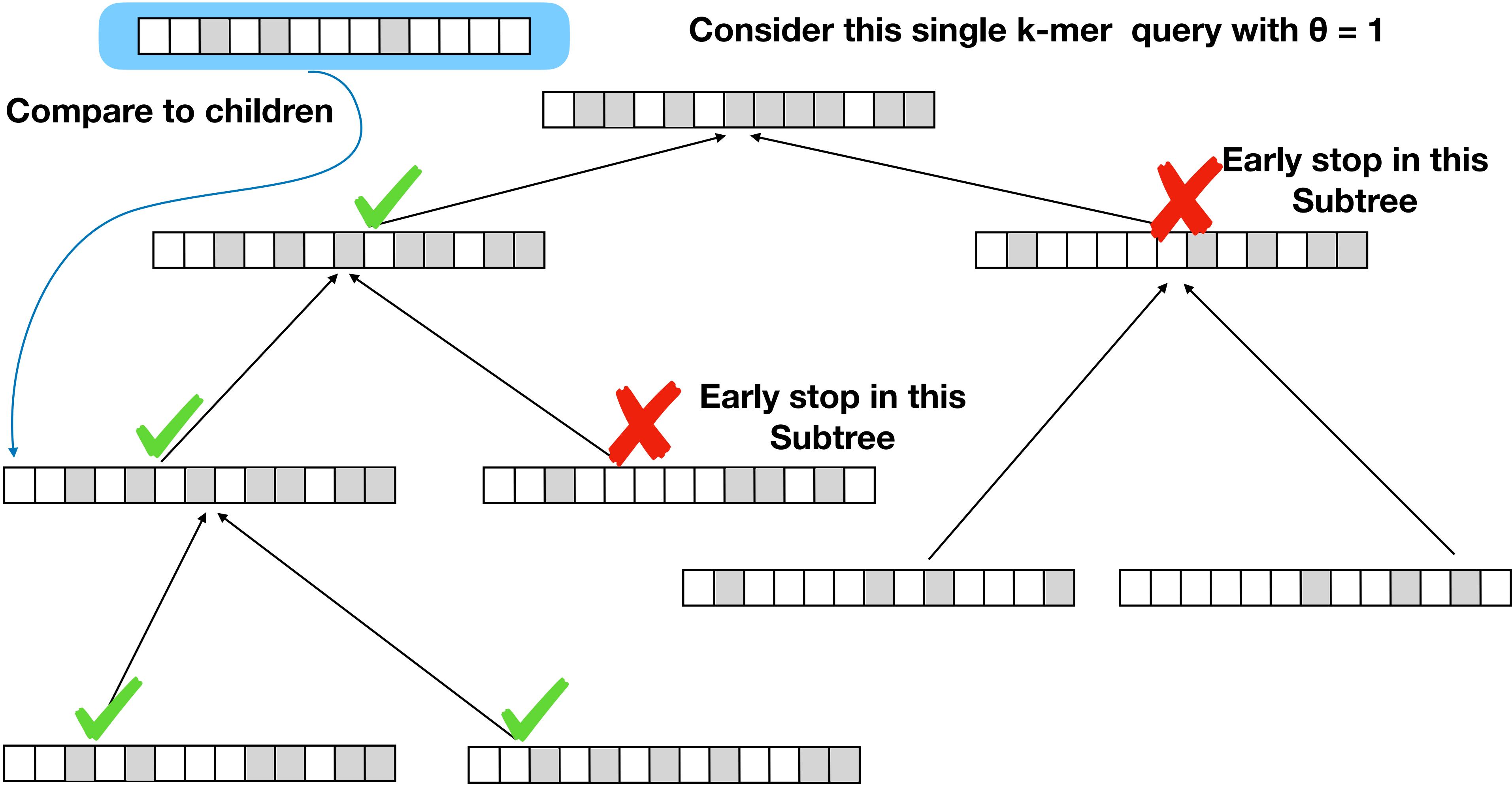
SBT Operations



SBT Operations



SBT Operations



SBT Operations

Query (given collection of k-mers K_q , parameter θ):

Thm 2.

Let q be a query string containing ℓ distinct k-mers. If we treat the k-mers of q as being independent, the probability that $> \lfloor \theta \ell \rfloor$ false-positive k-mers appear in a filter U with FPR ξ is

$$1 - \sum_{i=0}^{\lfloor \theta \ell \rfloor} \binom{\ell}{i} \xi^i (1 - \xi)^{\ell - i}$$

Prob of $> \lfloor \theta \ell \rfloor$ false positives is simply the difference from 1

Prob of $\leq \lfloor \theta \ell \rfloor$ false positives.

Because of assumed indolence of q , these are independent Binomial trials with “success” rate ξ

SBT Tricks

High false positive rate lets filters be small (& use only a single hash)

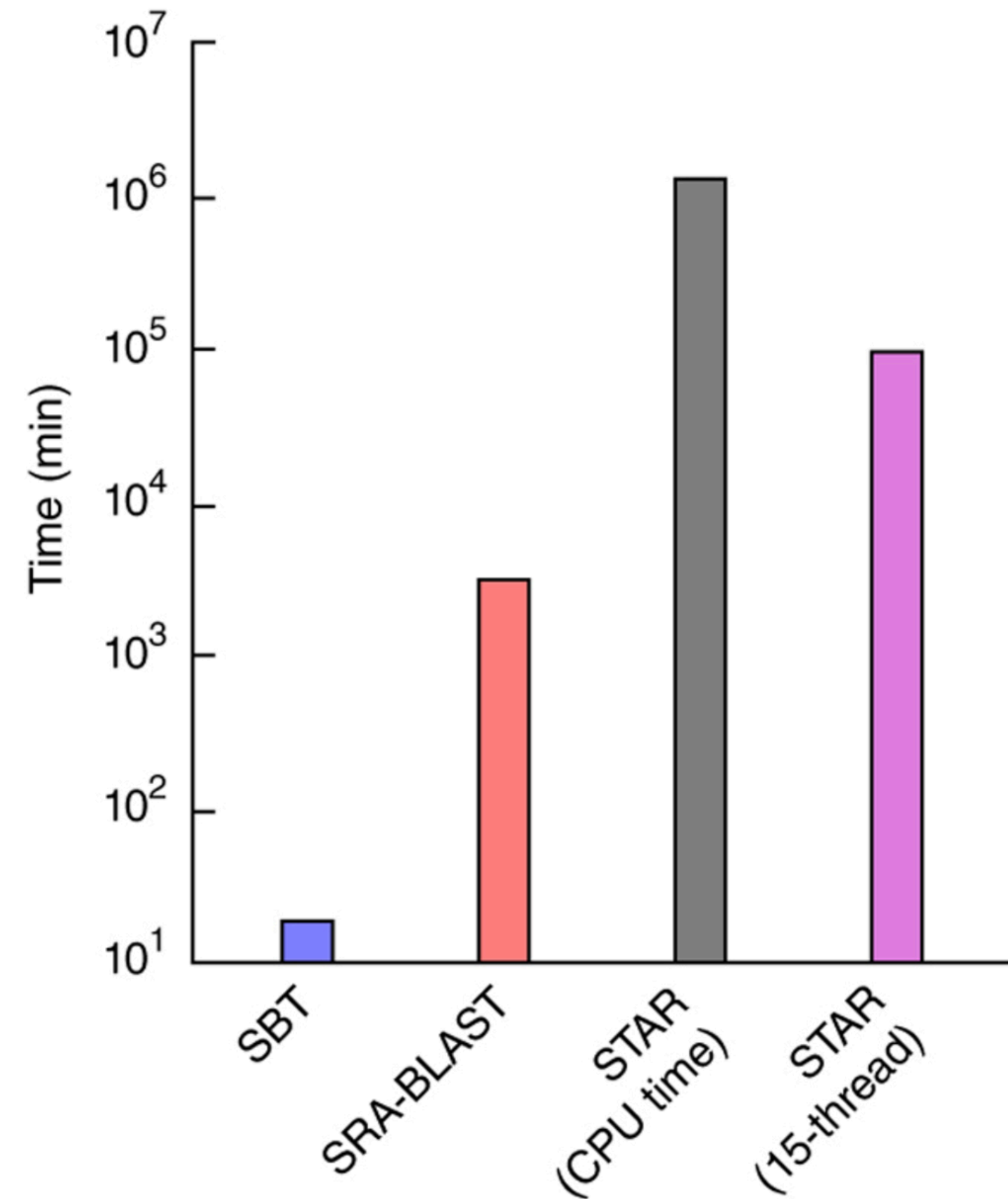
Insert in leaves only k-mers occurring $> c$ times, set

as follows: $count(s_i) = 1$ if s_i is 300 MB or less, $count(s_i) = 3$ for files of size 300–500 MB, $count(s_i) = 10$ for files of size 500 MB–1 GB, $count(s_i) = 20$ for files between 1 GB and 3 GB, and $count(s_i) = 50$ for files > 3 GB or larger FASTA files.

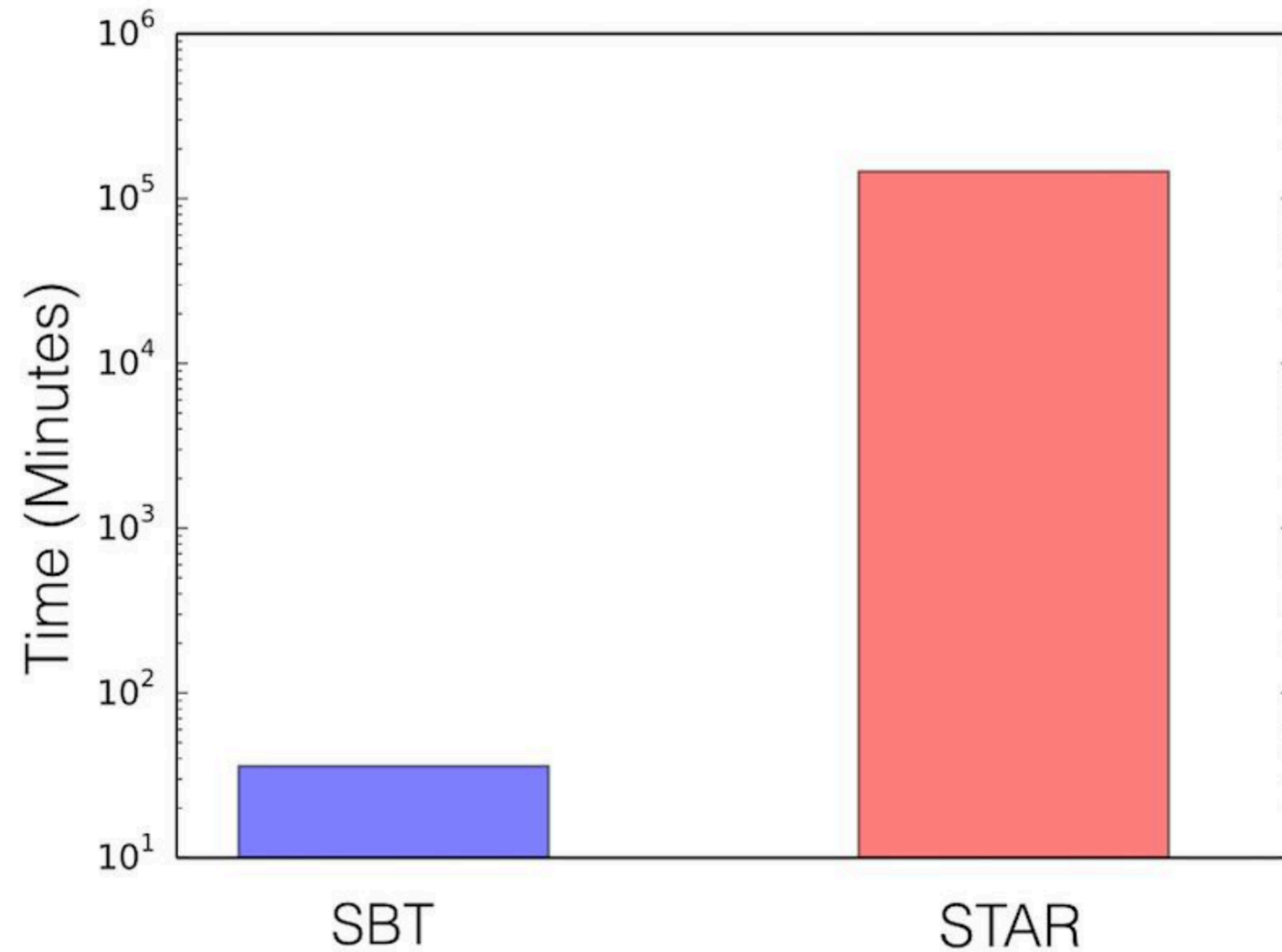
Store Bloom filter as RRR-compressed bit vectors. Greatly reduces storage space. Individual bits can be accessed *without* decompression in $O(\log m)$ time.

SBT Speed

Average search time for a single transcript over 2,652 RNA-seq experiments in the SRA for human blood, breast and brain tissues



SBT Speed

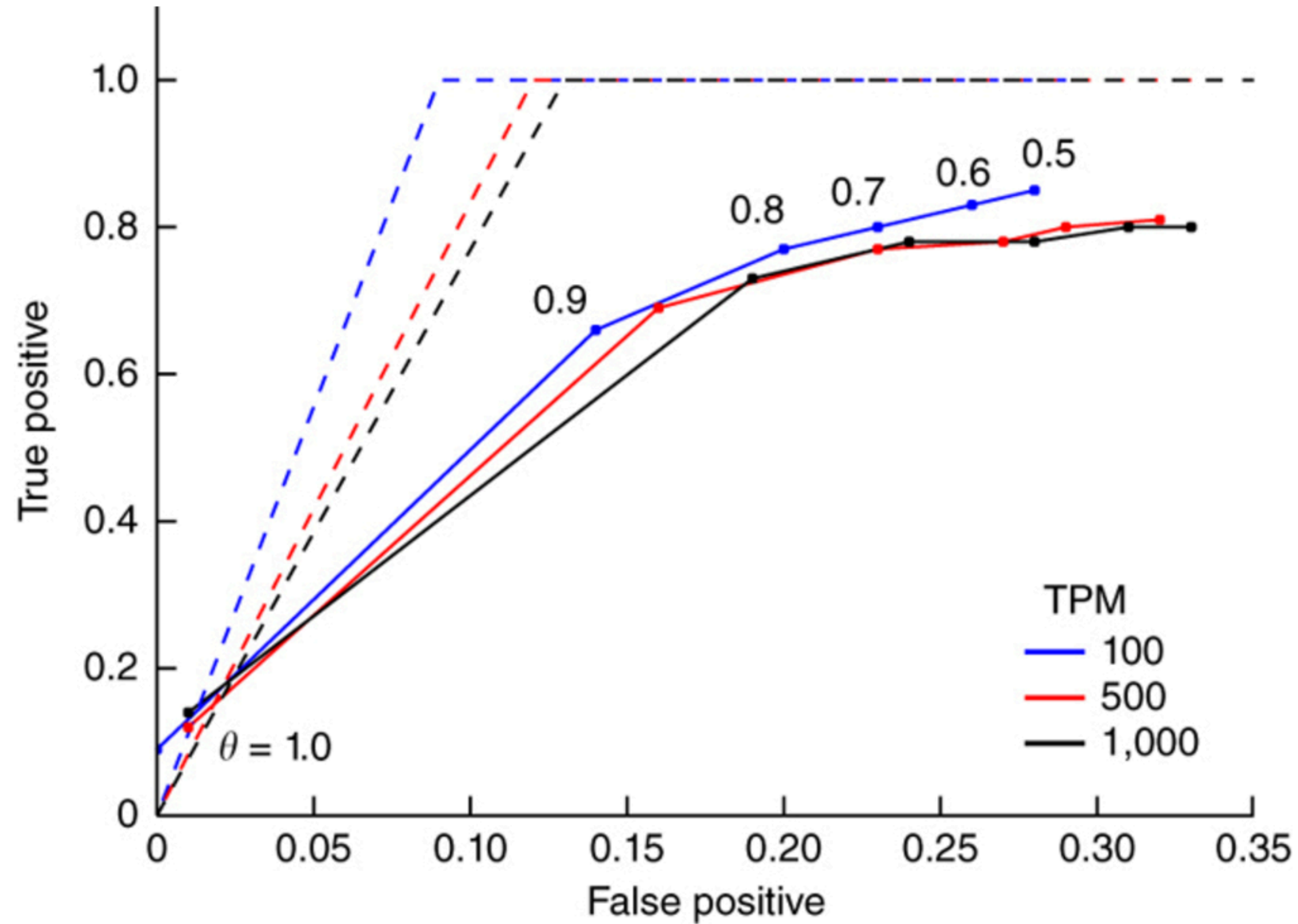


Supplementary Figure 2

Comparison with STAR on batched queries.

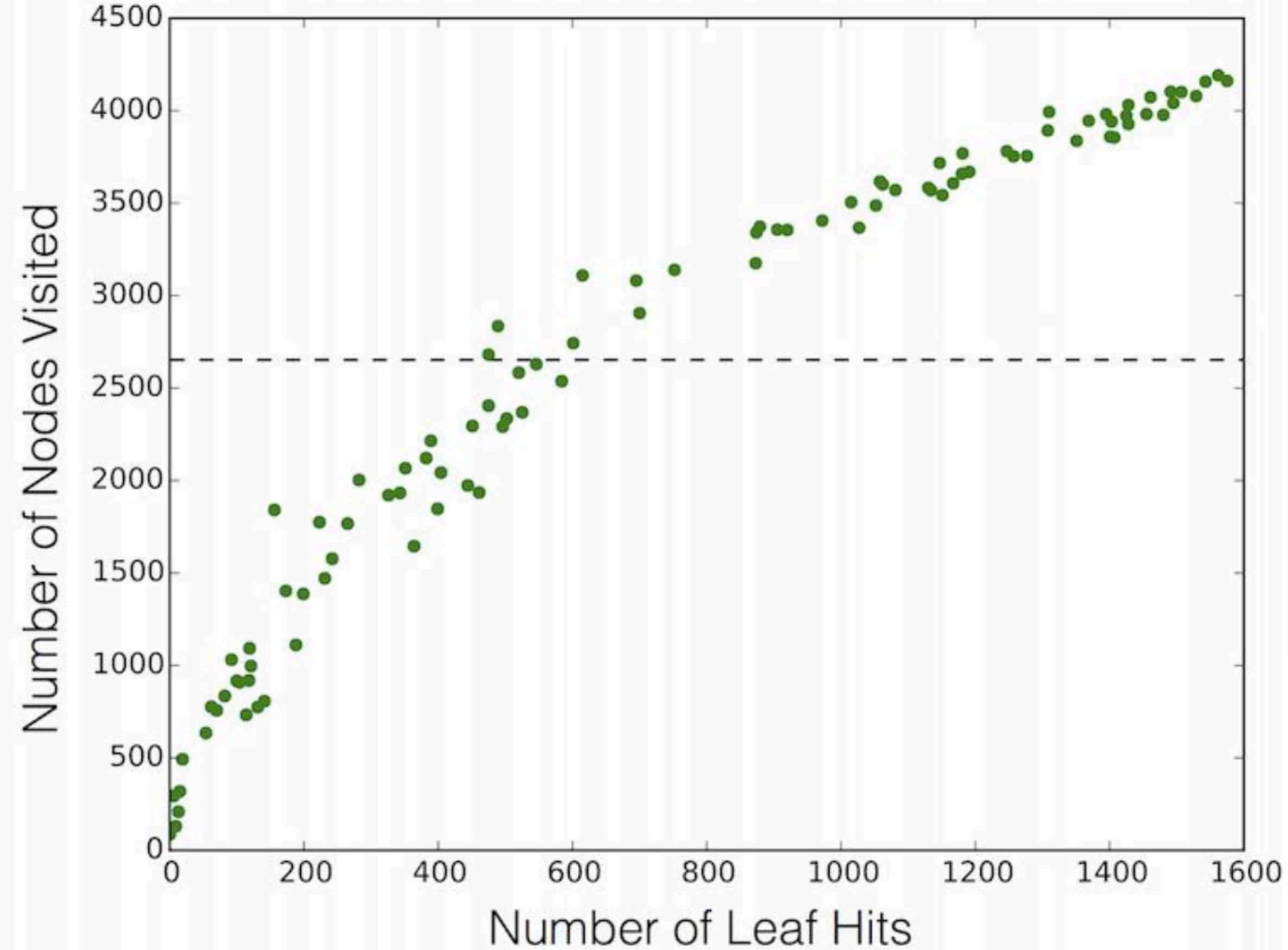
STAR was run using an index built from 100 batch-queries and a size 11 pre-index string. Both SBT and STAR were run using one thread and SBT was limited to a single filter in RAM. SBT is an estimated 4056 times faster than STAR under these conditions. STAR times are estimated from extrapolating from querying 100 random SRR files.

SBT Accuracy



Solid lines represent mean true-positive and false-positive rates, dashed lines represent the median rates on the same experiments. Relaxing θ leads to a higher sensitivity at the cost of specificity. In more than half of all queries, 100% of true-positive hits can be found with θ as high as 0.9.

SBT Efficiency

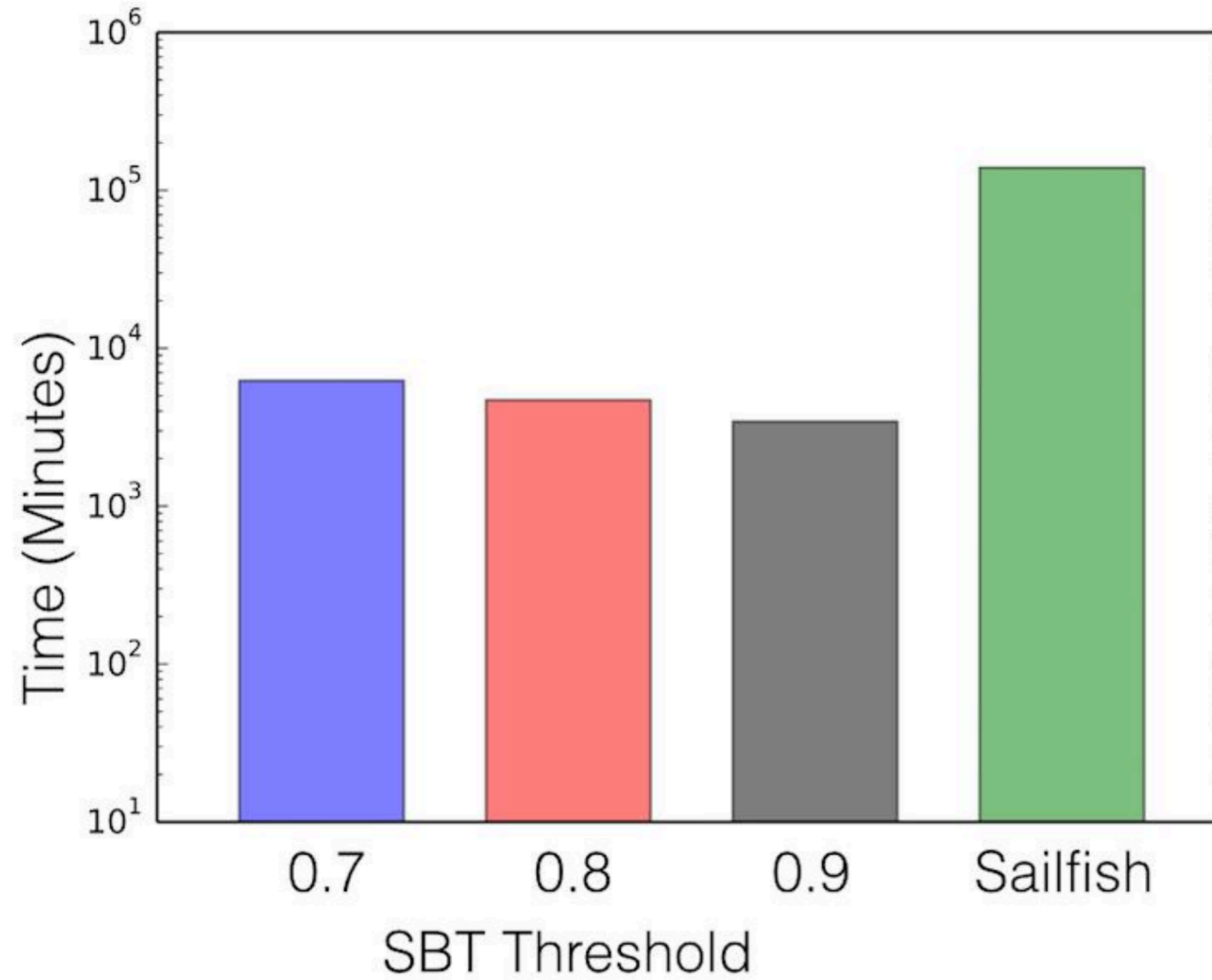


Supplementary Figure 5

Total number of Sequence Bloom Tree nodes visited as a function of the number of leaf hits when querying 100 random human transcripts in the Low query set.

Number of nodes includes both internal and leaf nodes of the SBT. Each point represents a single query. When a query is found in many of the leaves, the query must also visit a nearly equal number of internal tree nodes, and so the tree structure would not provide any benefit over merely searching all the leaf filters directly. On the other hand, when the query is found in only a few leaves, the total number of nodes visited can be significantly smaller than the number of leaves. For the SBT built here, we find that for queries that are found in 600 or fewer leaves, the tree structure and internal nodes result in an improvement of overall efficiency by visiting fewer than 2652 nodes. A naive approach that did not use the tree would require querying 2652 leaf filters for all queries (denoted by dashed line). Approximately half of the randomly selected queries known to be expressed in the included experiments fall below this threshold.

SBT Efficiency



Supplementary Figure 8

Time for querying all known human transcripts.

Total times (single-threaded) for querying all 214,293 human transcripts (in batch mode) against all publicly available blood, breast, and brain RNA-seq experiments in the SRA for $\theta = 0.7, 0.8, 0.9$ as well as the extrapolated time to run Sailfish on the full dataset. Sailfish is significantly faster than nearly all other algorithms for RNA-seq quantification.

Two improved SBT-related papers (RECOMB 2017)

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 share a core idea

All Some SBT:

$$B_{all}(u) = B_{\cap}(u) \setminus B_{\cap}(parent(u))$$

present in all leaves below u ,
but not in u 's parent

$$B_{some}(u) = B_{\cup}(u) \setminus B_{\cap}(u)$$

present in some (but not) all
leaves below u

Split sequence bloom tree (SSBT):

Store **2** filters at each node, r_{sim} and r_{rem}

$$r_{sim} = \bigcap_{i=0}^n b_i$$

present in all leaves below r

$$r_{rem} = \bigcup_{i=0}^n (b_i - r_{sim})$$

present in some (but not) all
leaves below r

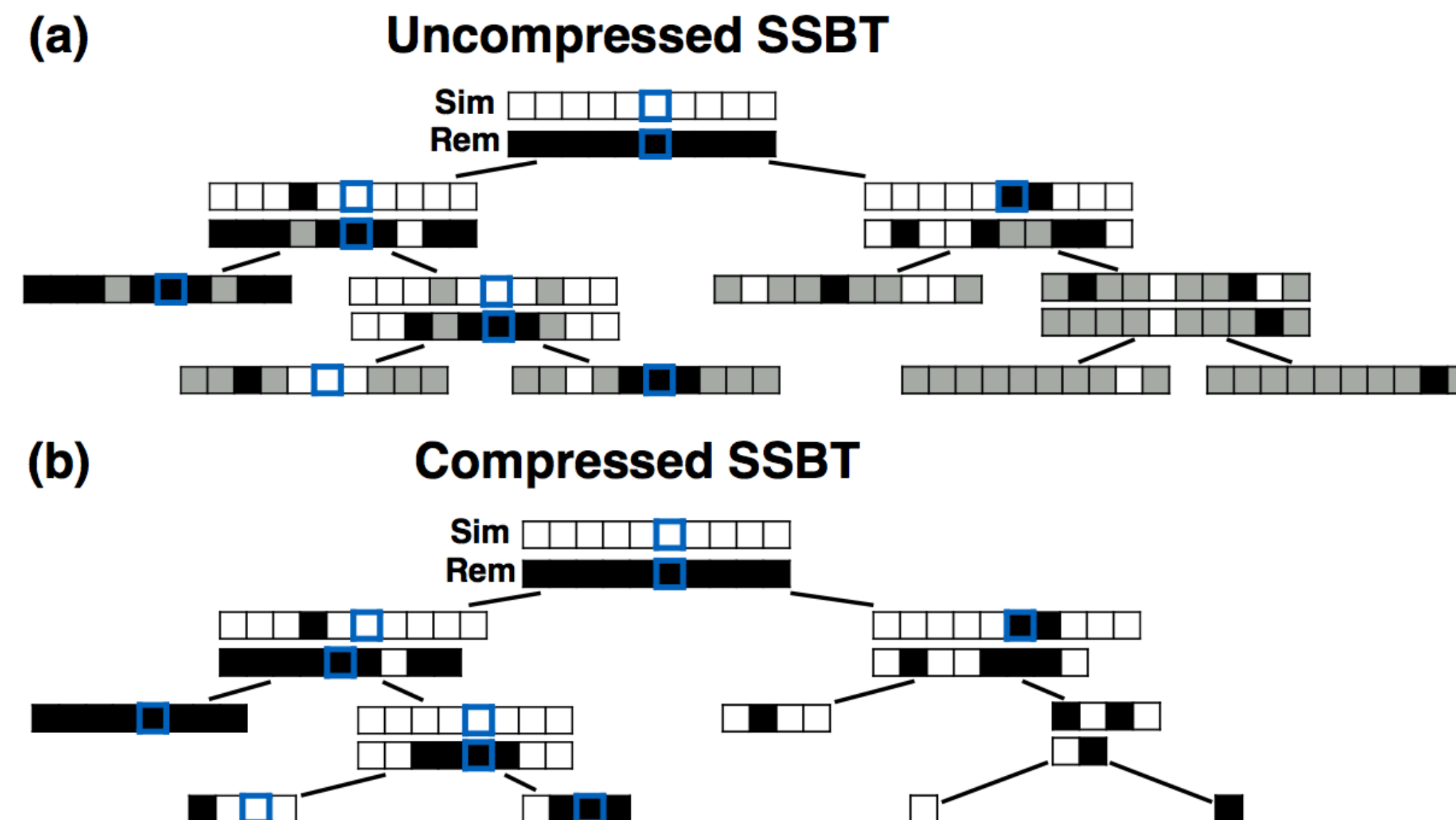
Both share a core idea

This allows an immediate optimization

if r_{sim} or B_{all} match the query, we can add all leaves below r/u without explicitly continuing the search

The details differ

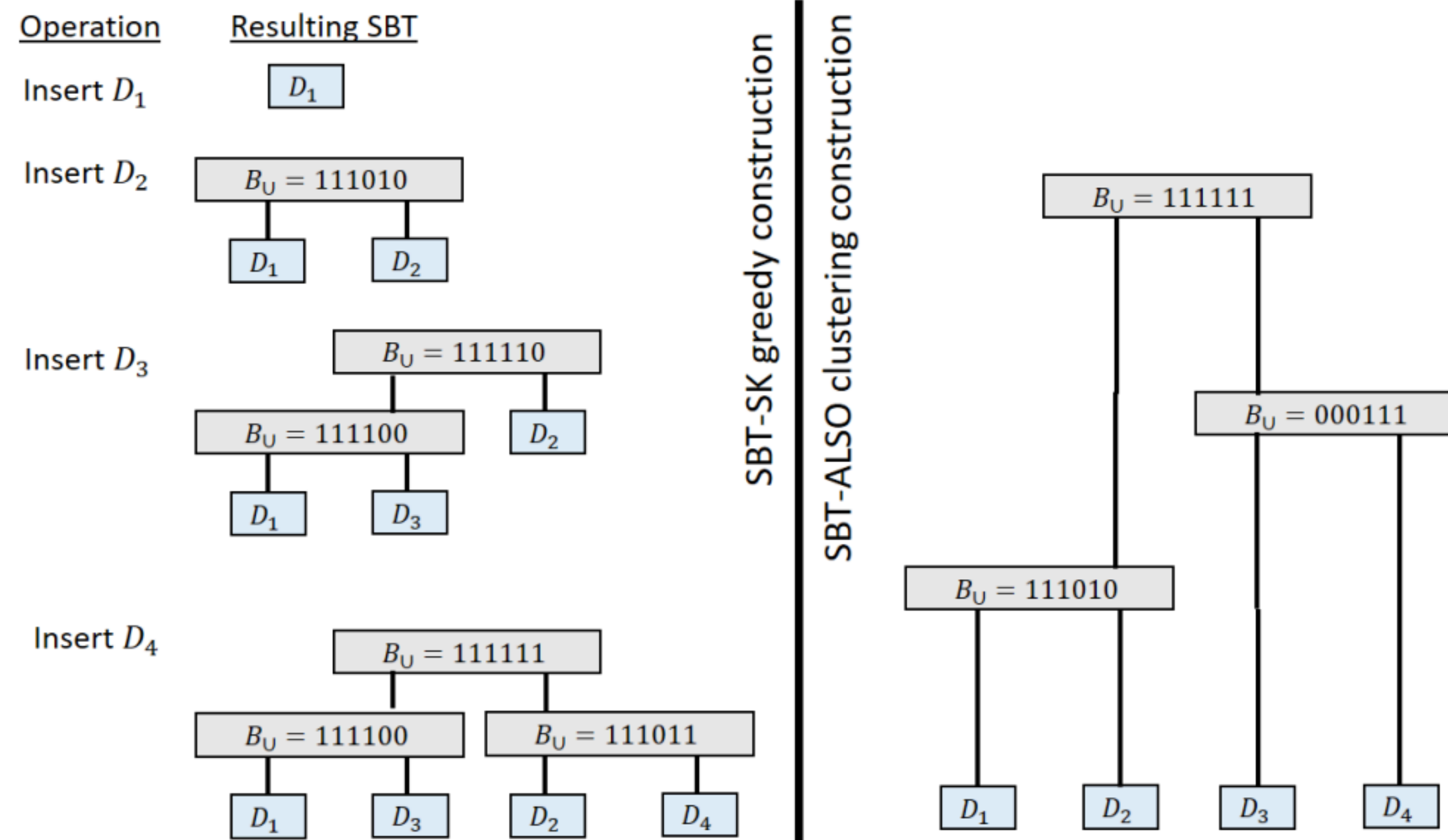
SSBT *explicitly* removed redundant elements



Both share a core idea

The details differ

SBT-AllSome don't explicitly remove these bits, but they optimize tree construction to put similar filters together (agglomerative clustering)



SSBTs take longer to build than the original but require considerably less memory to store.

Data Index	BFT	SBT	SSBT
Build Time (Min)	195	6	19
Compression Time (Min)	-	6.5	17
Total Time (Min)	195	12.5	36

Table 4: Build and compression times for SBT, SSBT, and BFT constructed from a 50 experiment set. As SBT and SSBT were designed to be queried from a compressed state, we compare the time to build and compress against BFT’s time to build.

Data Index	SBT	Split SBT
Build Time	18 Hr	78 Hr
Compression Time	17 Hr	19 Hr
Uncompressed Size	1295 GB	1853 GB
Compressed Size	200 GB	39.7 GB

Table 2: Build statistics for SBT and 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.

Data Index	BFT	SBT	SSBT
Build Peak RAM (GB)	23	21.5	15.6
Compress Peak RAM (GB)	-	24.2	16.2
Uncompressed Size (GB)	9.2	24	35
Compressed Size (GB)	-	3.9	0.94

Table 3: Build and compression peak RAM loads and on-disk storage costs for SBT, SSBT, and BFT constructed from a 50 experiment set. BFT does not have a built-in compression tool and cannot be queried when compressed. For these reasons, the uncompressed BFT is compared against the compressed SBT/SSBT.

SSBTs are also faster to query than SBTs

Index	TPM ≥ 100	TPM ≥ 500	TPM ≥ 1000
BFT	75 Sec (11.8 GB)	75 Sec (11.8 GB)	75 Sec (11.8 GB)
SBT	19 Sec (2.9 GB)	21 Sec (3.1 GB)	22 Sec (3.2 GB)
SSBT	5.8 Sec (0.64 GB)	6.2 Sec (0.65 GB)	6.3 Sec (0.66 GB)

Table 5: Comparison in query timing (and average peak memory) between SBT, SSBT, and BFT indices for 50 experiments.

Index	TPM ≥ 100	TPM ≥ 500	TPM ≥ 1000
SBT	19.7 Min	20.7 Min	20 Min
SSBT	3.7 Min	3.8 Min	3.6 Min

Table 6: Comparison in query timing between SBT and SSBT for 2652 experiments.

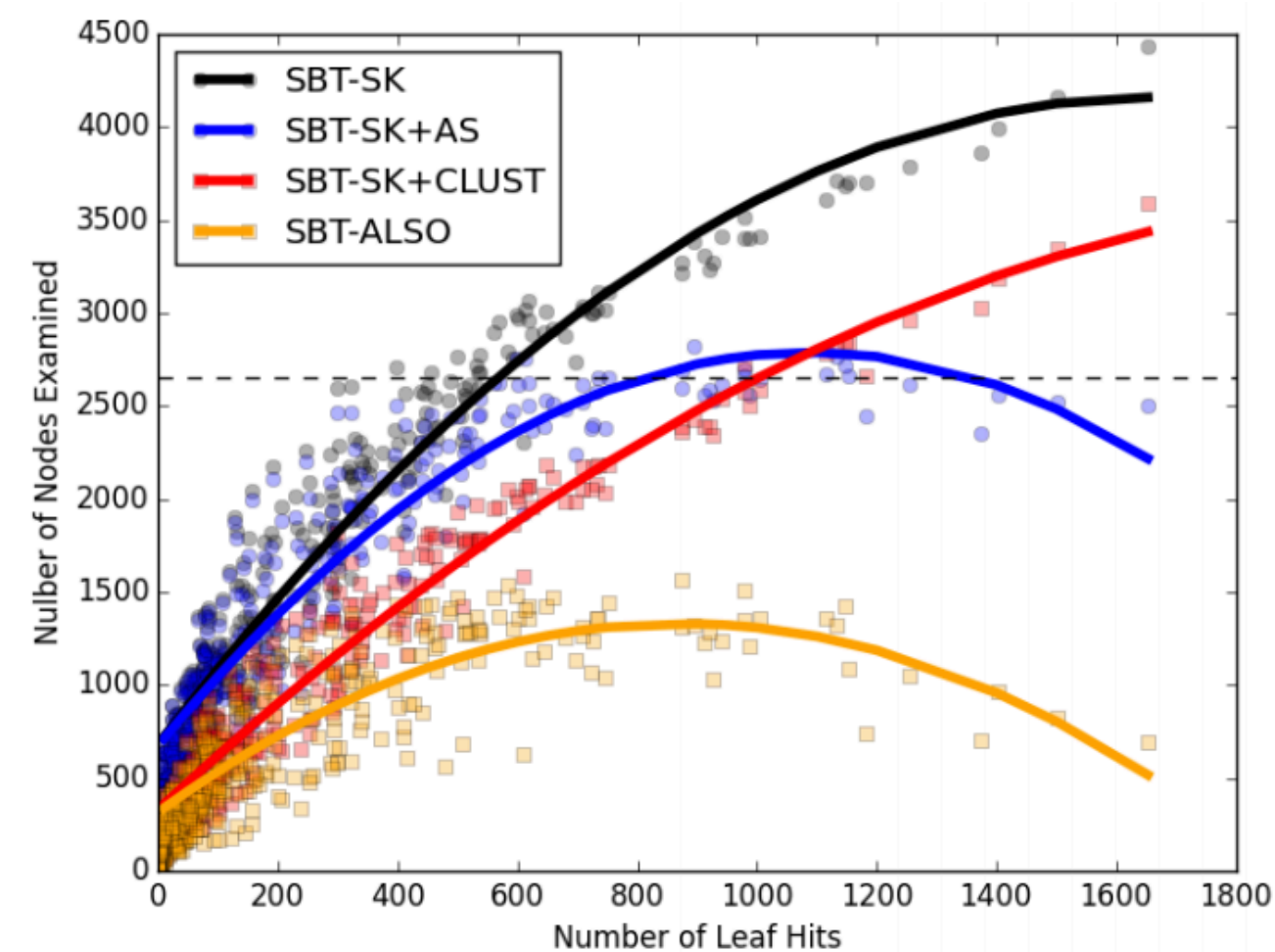
Query Time:	$\theta=0.7$	$\theta=0.8$	$\theta=0.9$
SBT	20 Min	19 Min	17 Min
SSBT	3.7 Min	3.5 Min	3.2 Min
RAM SSBT	31 Sec	29 Sec	26 Sec

Table 7: Comparison of query times using different thresholds θ for SBT and SSBT using the set of data at TPM 100.

AllSome SBTs are faster to construct than SBT (called SBT-SK here), but not much smaller.

	SBT-SK	SBT-ALSO
construction of tree topology (i.e. clustering)	N/A	27m
construction of internal nodes	56h 54m	26h 3m
temporary disk space	1,235 GB	2,469 GB
final disk space	200 GB	177 GB

Table 1. Construction time and space. Times shown are wall-clock times. A single thread was used. Note the SBT-SK tree that was constructed for the purposes of this Table differs from the tree used in [36] and in our other experiments because the insertion order during construction was not the same as in [36] (because it was not described there).



They examine fewer nodes than the original SBT too

Fig. 3. Number of nodes examined per query for SBT-SK, SBT-ALSO, as well two intermediate SBTs. A set of 1,000 transcripts were chosen at random from Gencode set, and each one queried against the four different trees. A dot represents a query and shows the number of matches in the database (x-axis) compared to the number of nodes that had to be loaded from disk and examined during the search (y-axis). For each tree (color), we interpolated a curve to show the pattern. The dashed horizontal line represents the hypothetical algorithm of simply checking if the query θ -matches against each of the database entries, one-by-one. For θ , we used the default value in the SBT software ($\theta = 0.9$).

Which makes them faster to query than the original SBT as well.

	SBT-SK	SBT-SK+CLUST	SBT-ALSO
1 query	1m 11s / 301 MB	56s / 299 MB	34s / 301 MB
10 queries	4m 4s / 305 MB	3m 17s / 304 MB	2m 4s / 313 MB
100 queries	7m 44s / 315 MB	6m 31s / 317 MB	4m 44s / 353 MB
1,000 queries	25m 31s / 420 MB	17m 22s / 418 MB	8m 23s / 639 MB
198,074 queries	3081m 42s / 22 GB	-	462m 39s / 63 GB

Table 2. Query wall-clock run times and maximum memory usage, for batches of different sizes. For the batch of 1,000 queries, we used the same 1,000 queries as in Figure 3. For the batch of 100 queries, we generated three replicate sets, where each set contains 100 randomly sampled transcripts without replacement from the 1,000 queries set. For the batch of 10 queries, we generated 10 replicate sets by partitioning one of the 100 query sets into 10 sets of 10 queries. For the batch of 1 query, we generated 50 replicate sets by sampling 50 random queries from Gencode set. The shown running times are the averages of these replicates. A dash indicates we did not run the experiment. For θ , we used the default value in the SBT software ($\theta = 0.9$).

	SBT-SK	SBT-ALSO		
	regular alg	regular alg	large exact alg	large heuristic alg
query time	1397m 18s	195m 33s	10m 35s	8m 32s
query memory	2.3 GB	4.7 GB	1.3 GB	1.2 GB

Table 3. Performance of different trees and query algorithms on a large query. We show the performance of SBT-SK and three query algorithms using SBT-ALSO compressed with ROAR: the regular algorithm, the large exact algorithm, and the large heuristic algorithm. We show the wall-clock run time and maximum RAM usage. We used $\theta = 0.8$ for this experiment. The ROAR compressed tree was 190 GB (7.3% larger than the RRR tree).

Even further improvements of SBT-like ideas

Bioinformatics, 36(3), 2020, 721–727
doi: 10.1093/bioinformatics/btz662
Advance Access Publication Date: 22 August 2019
Original Paper



Sequence analysis

Improved representation of sequence bloom trees

Robert S. Harris¹ and Paul Medvedev^{2,3,4,*}

¹Department of Biology, ²Department of Computer Science and Engineering, ³Department of Biochemistry and Molecular Biology and ⁴Center for Computational Biology and Bioinformatics, The Pennsylvania State University, University Park, PA 16801, USA

*To whom correspondence should be addressed.

Associate Editor: Inanc Birol

Received on March 25, 2019; revised on August 15, 2019; editorial decision on August 19, 2019; accepted on August 20, 2019

Table 1. File sizes (in GiB) and build times

	HowDe-SBT	ALLSOME-SBT	SSBT
Construction time (h)	9	25	57
Intermediate space	602	602	602
Index size	14	142	23

Notes: The intermediate space refers to the initial experiment BFs. The construction time is the time to build the index from the BFs and does not include k -mer counting. All times shown are on one processor.

Table 2. Query times (s)

	HowDe-SBT	ALLSOME-SBT	SSBT
Single	5.4	36.7	29.4
Ten	44.5	353.7	159.6
Hundred	171.3	817.8	400.8
Thousand	719.7	1168.6	3509.4

Notes: Values shown are the median over all the replicates. The cache was cleared prior to each run.

HowDe (How-Determined) SBT