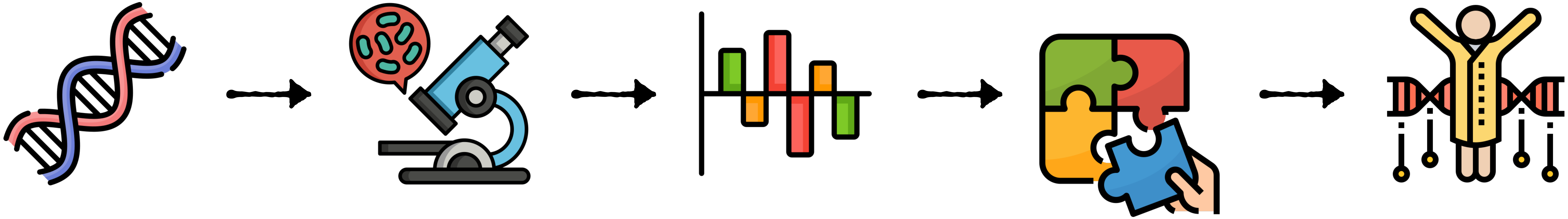
Vector Search for Large-Scale Genomic Discovery

VecDB@ICML2025

Prashant Pandey, Northeastern University, Boston USA
https://prashantpandey.github.io/

A typical genomic pipeline



DNA/RNA extraction and fragmentation into smaller pieces suitable for sequencing

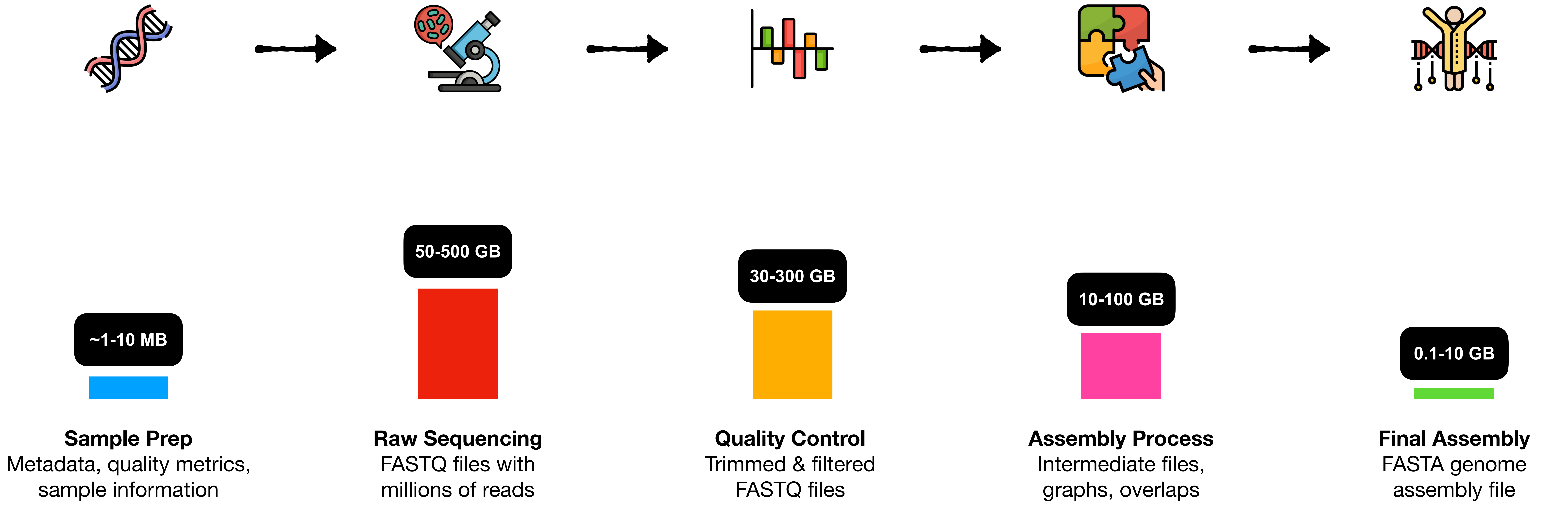
High-throughput sequencing generates millions of short/long reads from DNA fragments

Raw sequencing reads undergo quality assessment and preprocessing

Overlapping reads are assembled into contiguous sequences (contigs) and scaffolds

Complete or draft genome assembly ready for downstream analysis

Assembled data is hugely lossy

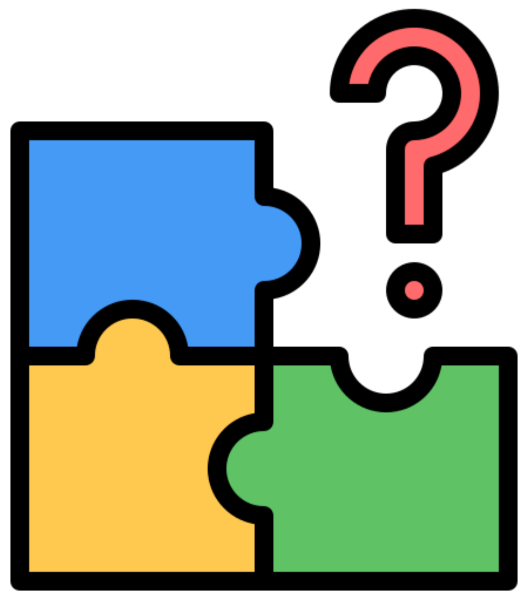


Data Reduction: 500 GB → 1 GB (500x compression)

Assembled data is hugely lossy

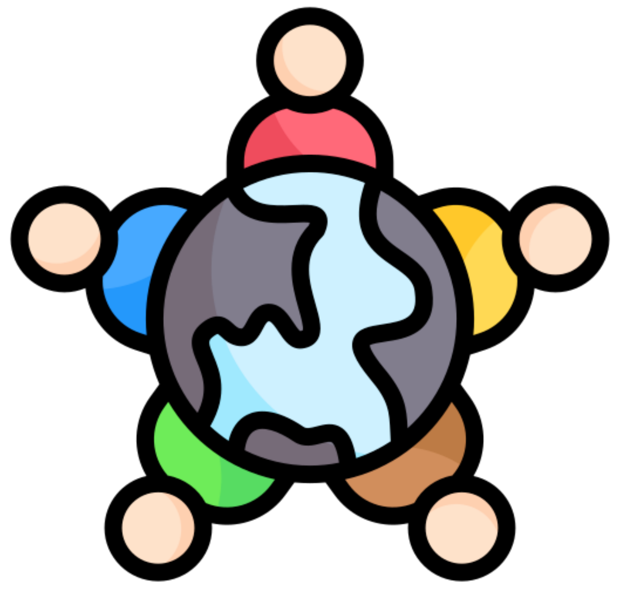
Data Reduction: 500 GB → 1 GB (500x compression)

Raw sequencing data contains biological diversity information



Missing Species

We only have assembled genomes for a tiny fraction of Earth's ~8.7 million species. Most can't even be grown in labs!



Population Diversity

Reference genomes reduce populations to single sequences, losing massive amounts of genetic variation and expression data.

A lot of variability information is lost during assembly. And a lot of raw sequencing data never gets assembled.

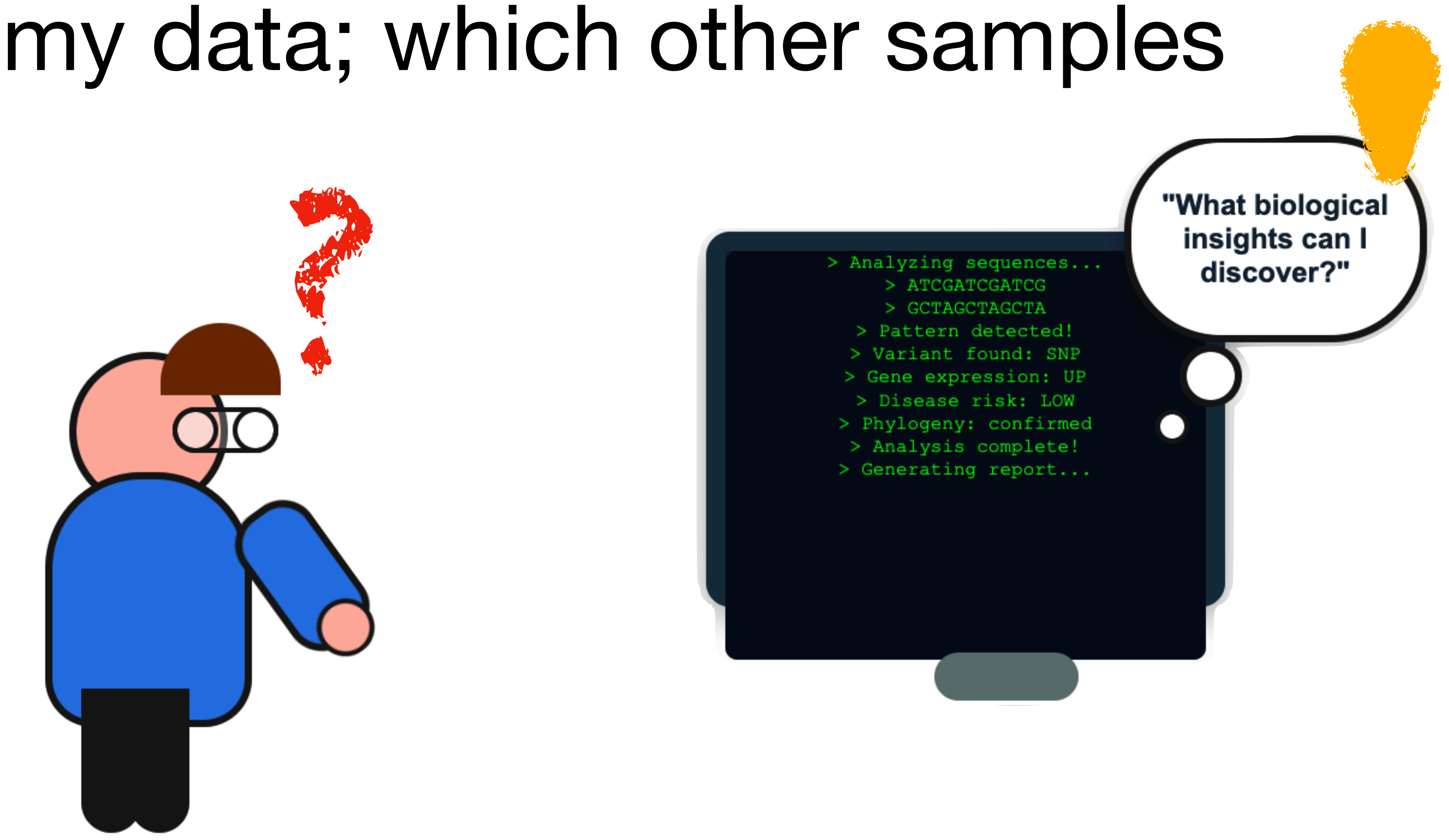
*Mora, Camilo, et al. "How many species are there on Earth and in the ocean?." PLoS biology https://doi.org/10.1371/journal.pbio.1001127

Raw sequencing data can unlock biological insights

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?



Raw sequencing data can unlock biological insights

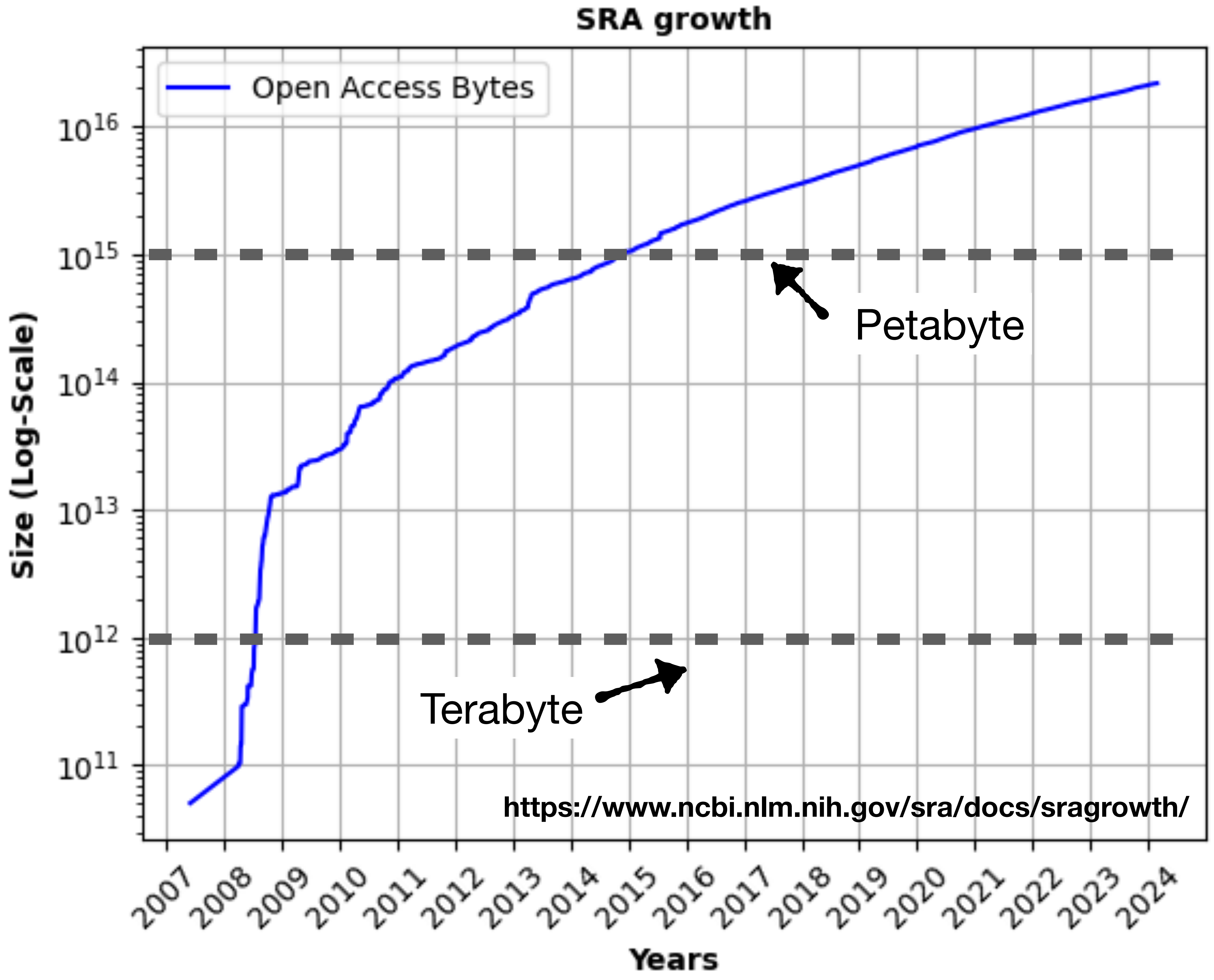
Q. What if I find a new putative disease-related transcript, and want to see if it appeared in other biological samples?

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 perform string searches through tons of raw sequencing data.

Sequence Read Archive (SRA) is growing rapidly



SRA is a publicly available dataset from NIH containing **raw sequencing data**

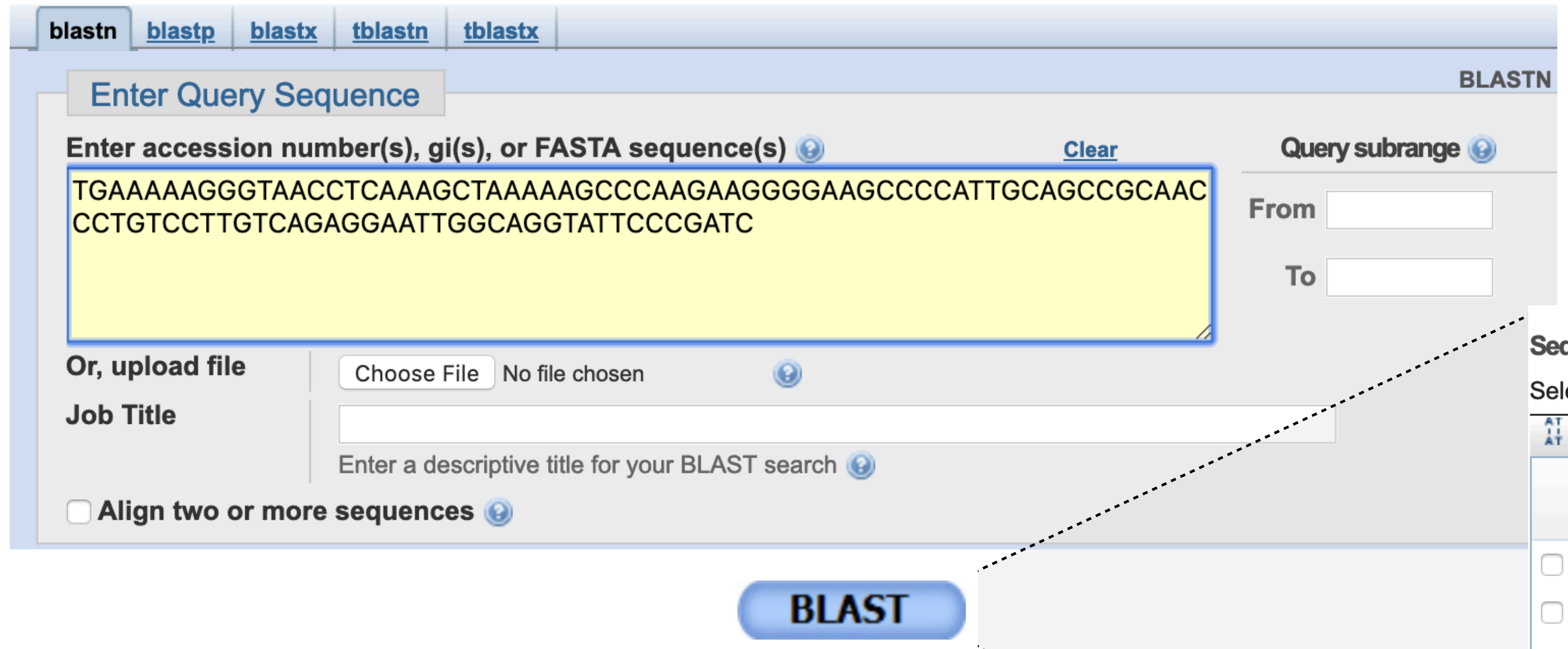
Basic Local Alignment Search Tool (BLAST)

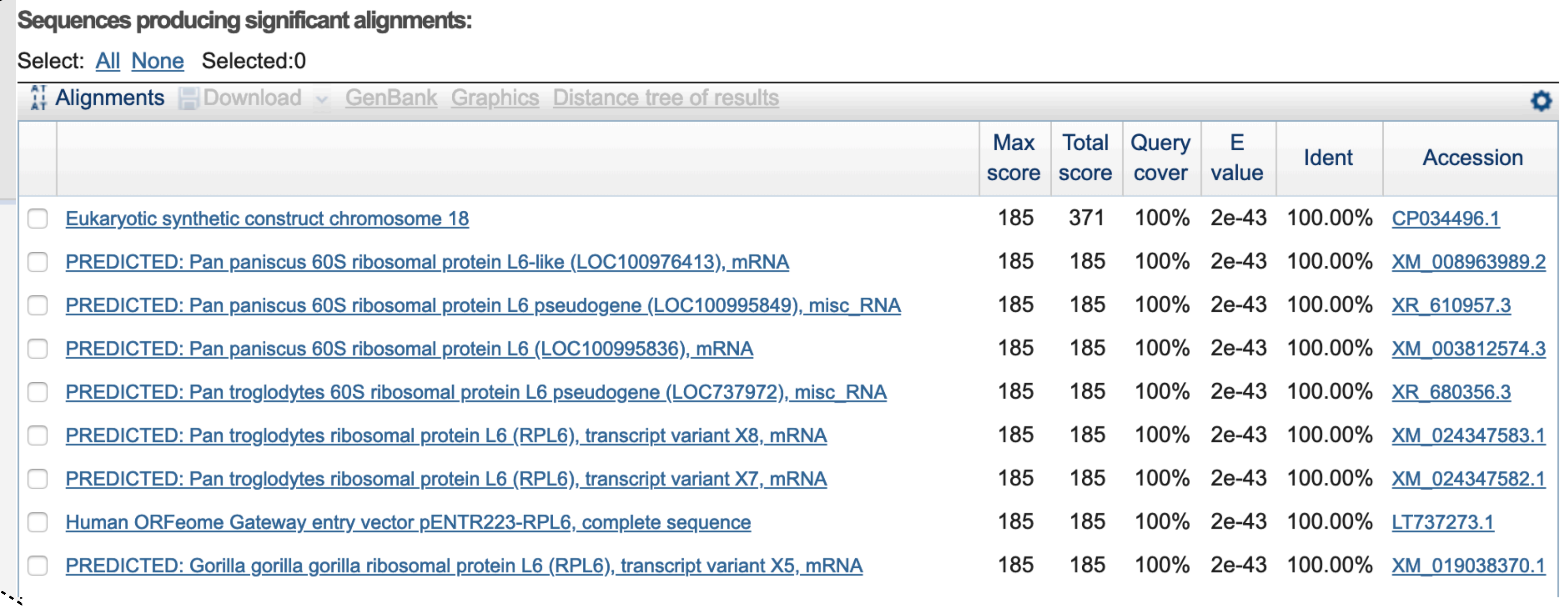
Basic Local Alignment Search Tool (BLAST)

Contrast this situation with searching assembled, curated genomes, for which we have an excellent tool; BLAST

Basic Local Alignment Search Tool (BLAST)

Contrast this situation with searching assembled, curated genomes, for which we have an excellent tool; BLAST





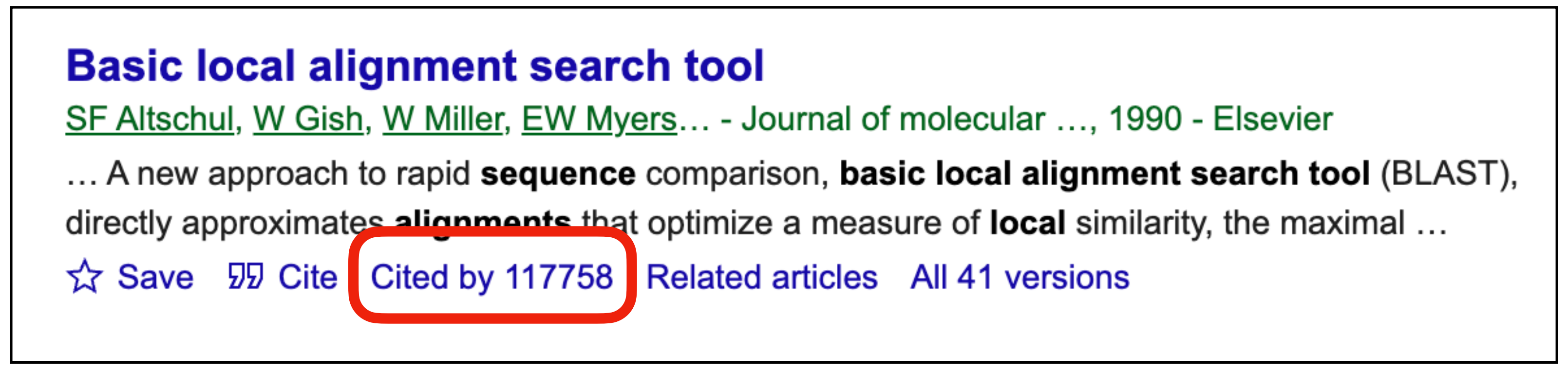
Sequences producing significant alignments:

Select: All None Selected:0

Alignments Download GenBank Graphics Distance tree of results

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

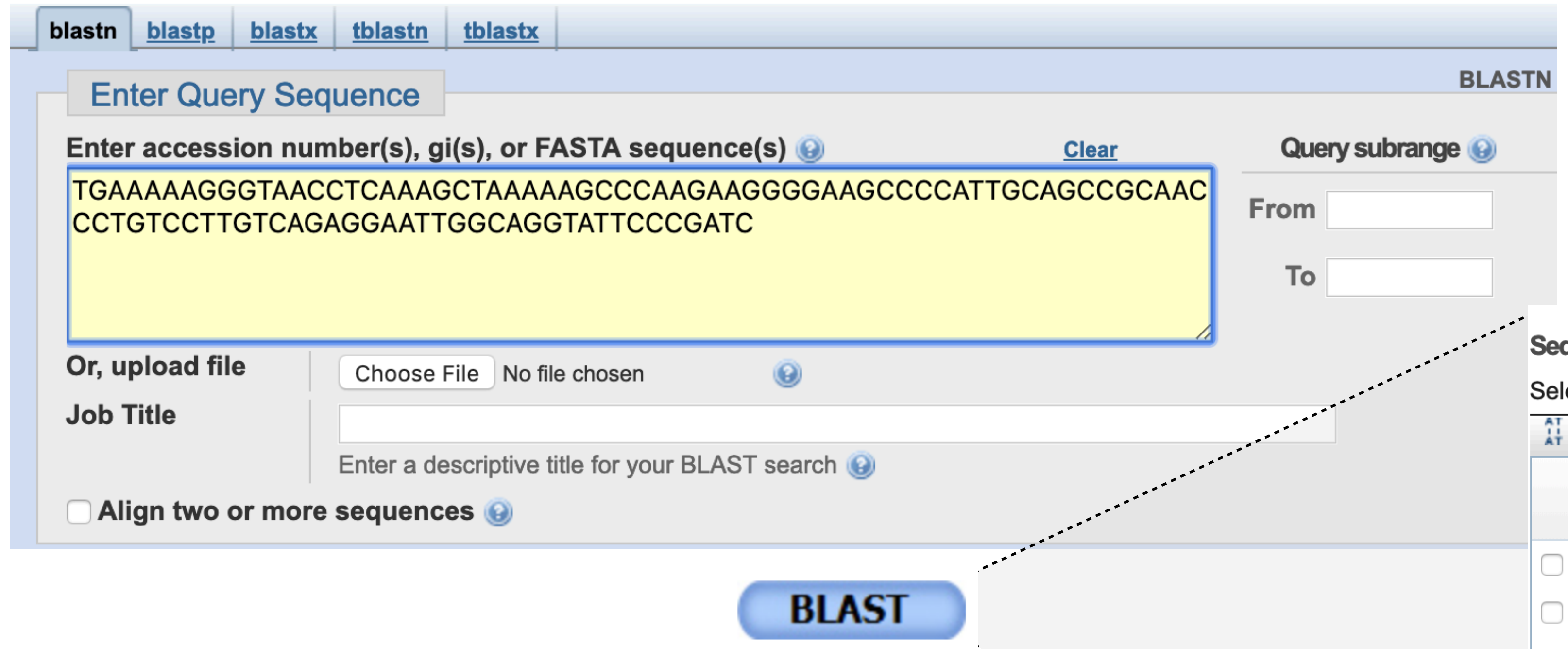
SF Altschul, W Gish, W Miller, EW Myers… - Journal of molecular …, 1990 - Elsevier

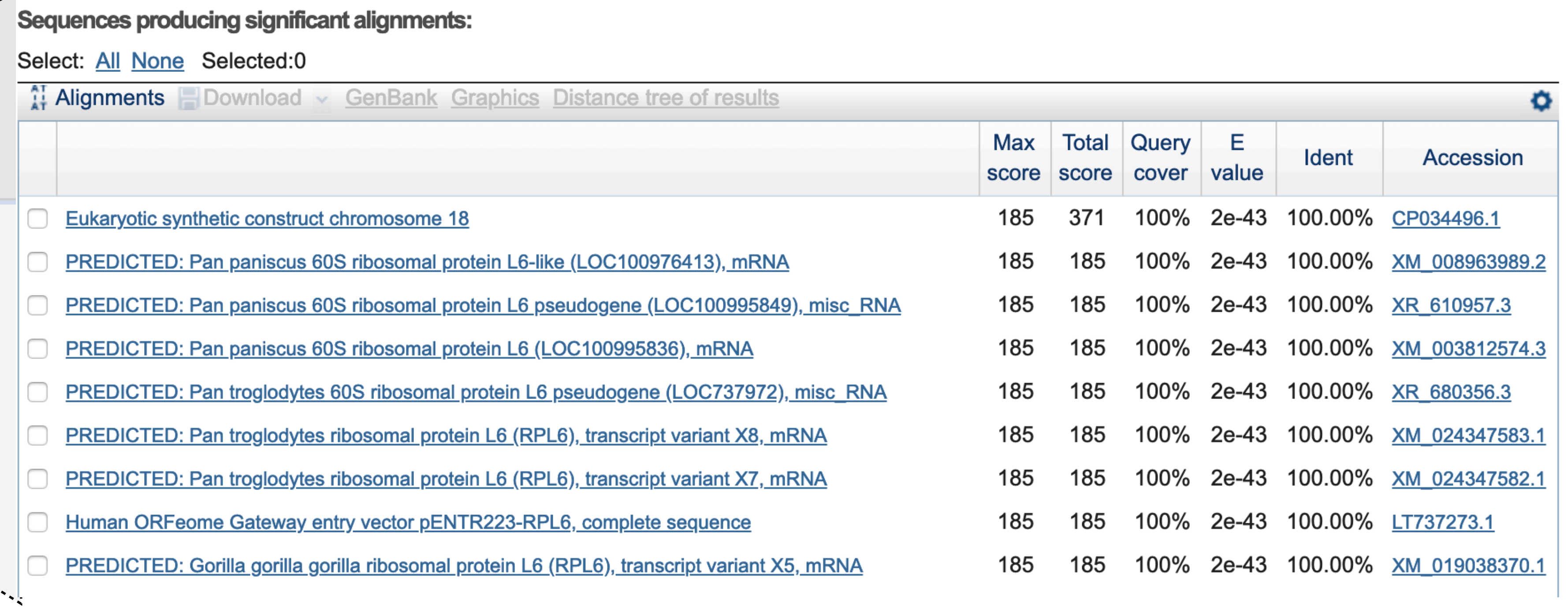
… A new approach to rapid **sequence** comparison, **basic local alignment search tool** (BLAST), directly approximates **alignments** that optimize a measure of **local** similarity, the maximal …

☆ Save 99 Cite Cited by 117758 Related articles All 41 versions

Basic Local Alignment Search Tool (BLAST)

Contrast this situation with searching assembled, curated genomes, for which we have an excellent tool; BLAST





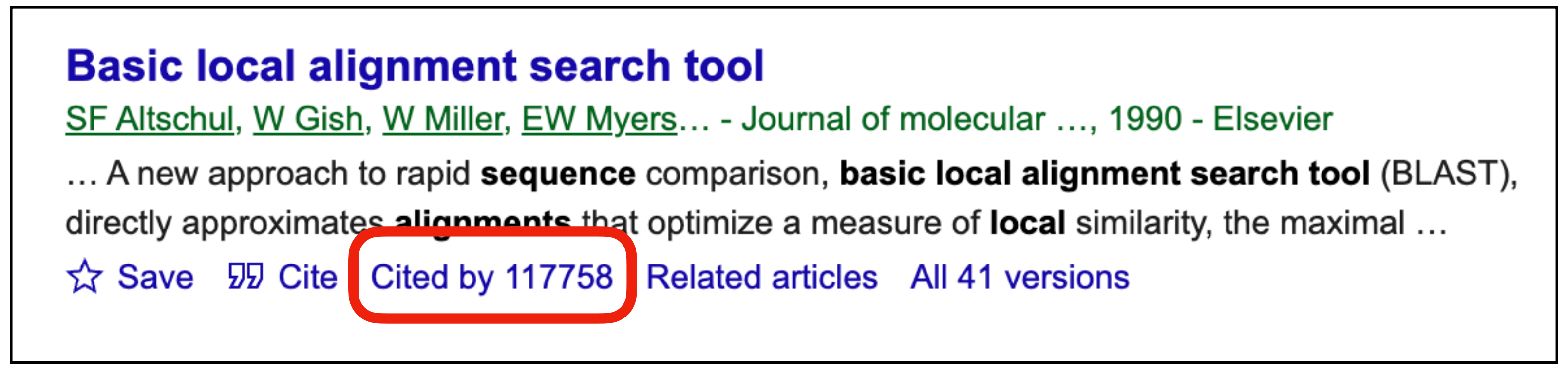
Sequences producing significant alignments:

Select: All None Selected:0

Alignments Download GenBank Graphics Distance tree of results

	Max score	Total score	Query cover	E value	Ident	Accession
Eukaryotic synthetic construct chromosome 18	185	371	100%	2e-43	100.00%	CP034496.1
PREDICTED: Pan paniscus 60S ribosomal protein L6-like (LOC100976413), mRNA	185	185	100%	2e-43	100.00%	XM_008963989.2
PREDICTED: Pan paniscus 60S ribosomal protein L6 pseudogene (LOC100995849), misc_RNA	185	185	100%	2e-43	100.00%	XR_610957.3
PREDICTED: Pan paniscus 60S ribosomal protein L6 (LOC100995836), mRNA	185	185	100%	2e-43	100.00%	XM_003812574.3
PREDICTED: Pan troglodytes 60S ribosomal protein L6 pseudogene (LOC737972), misc_RNA	185	185	100%	2e-43	100.00%	XR_680356.3
PREDICTED: Pan troglodytes ribosomal protein L6 (RPL6), transcript variant X8, mRNA	185	185	100%	2e-43	100.00%	XM_024347583.1
PREDICTED: Pan troglodytes ribosomal protein L6 (RPL6), transcript variant X7, mRNA	185	185	100%	2e-43	100.00%	XM_024347582.1
Human ORFeome Gateway entry vector pENTR223-RPL6, complete sequence	185	185	100%	2e-43	100.00%	LT737273.1
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
… 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 …
☆ Save Cite Cited by 117758 Related articles All 41 versions

BLAST performs substring matching (based on seed-search-align) using traditional succinct string data structures.

Why can’t we use BLAST for searching “raw” data?

Fragmented Patterns

The sequence you're looking for might be spread across multiple reads, making it impossible for BLAST to find as a contiguous match.

Scale Limitations

BLAST algorithms and data structures simply don't scale to handle millions of raw sequencing experiments efficiently.

Reframing the problem as vector search

Solomon and Kingsford 2016 reframed the problem, and suggested a direction…

nature biotechnology

Fast search of thousands of short-read sequencing experiments

Brad Solomon & Carl Kingsford

Nature Biotechnology **34**, 300–302 (2016)
doi:10.1038/nbt.3442
Download Citation

Received: 28 April 2015
Accepted: 23 November 2015
Published online: 08 February 2016

Proposal:

Represent each sample from SRA as a set of *tokens*

Similarity of *token* composition — similar sequence — small edit distance

Returns “yes/no” results for individual samples — “yes” results can be searched using 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 used in NLP (except DNA/RNA have no natural “tokens”)
- **Idea:** Similarity of *k*-mer composition — similar sequence

Sample discovery problem

Sample discovery problem

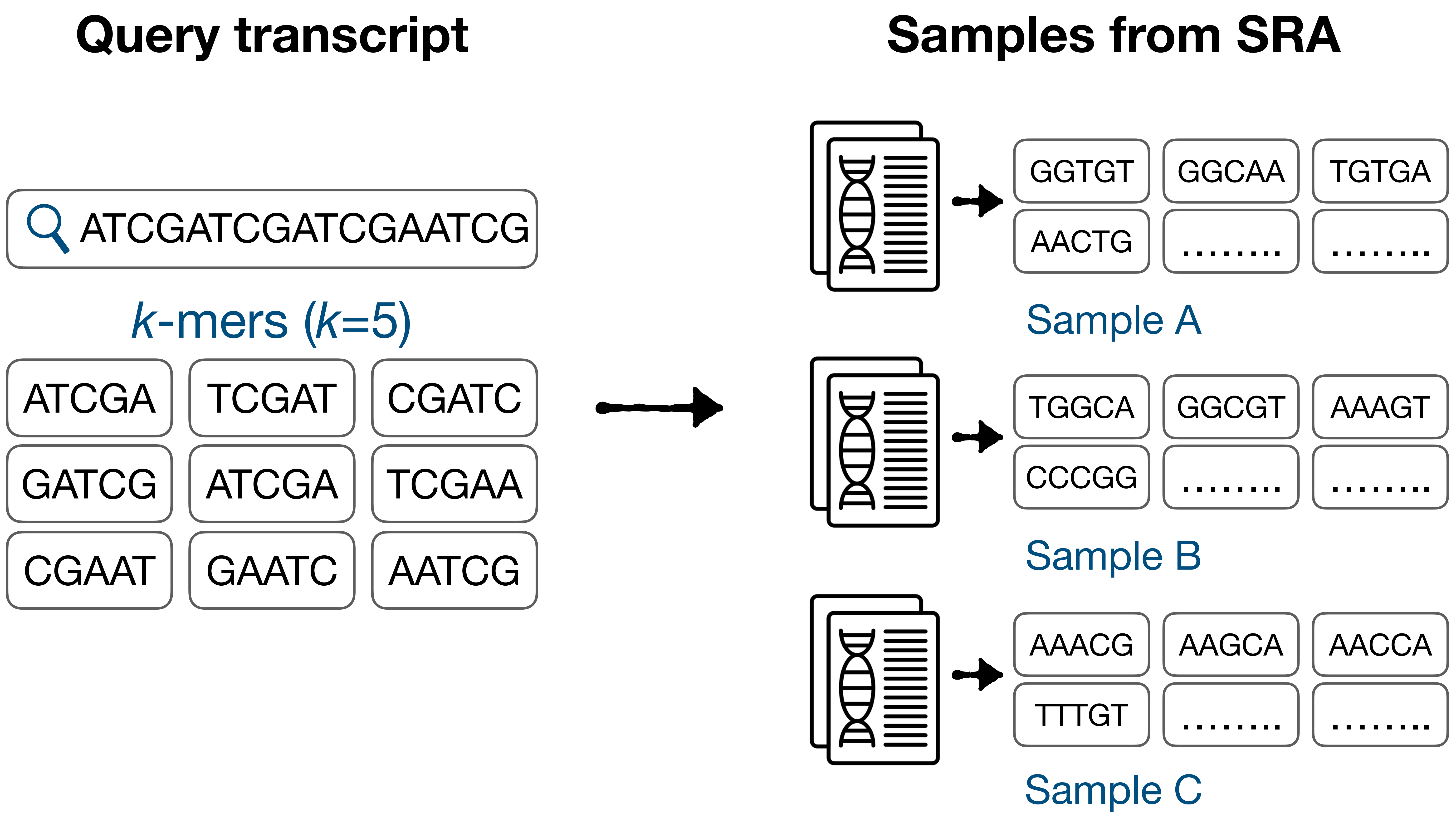
Query transcript

ATCGATCGATCGAATCG

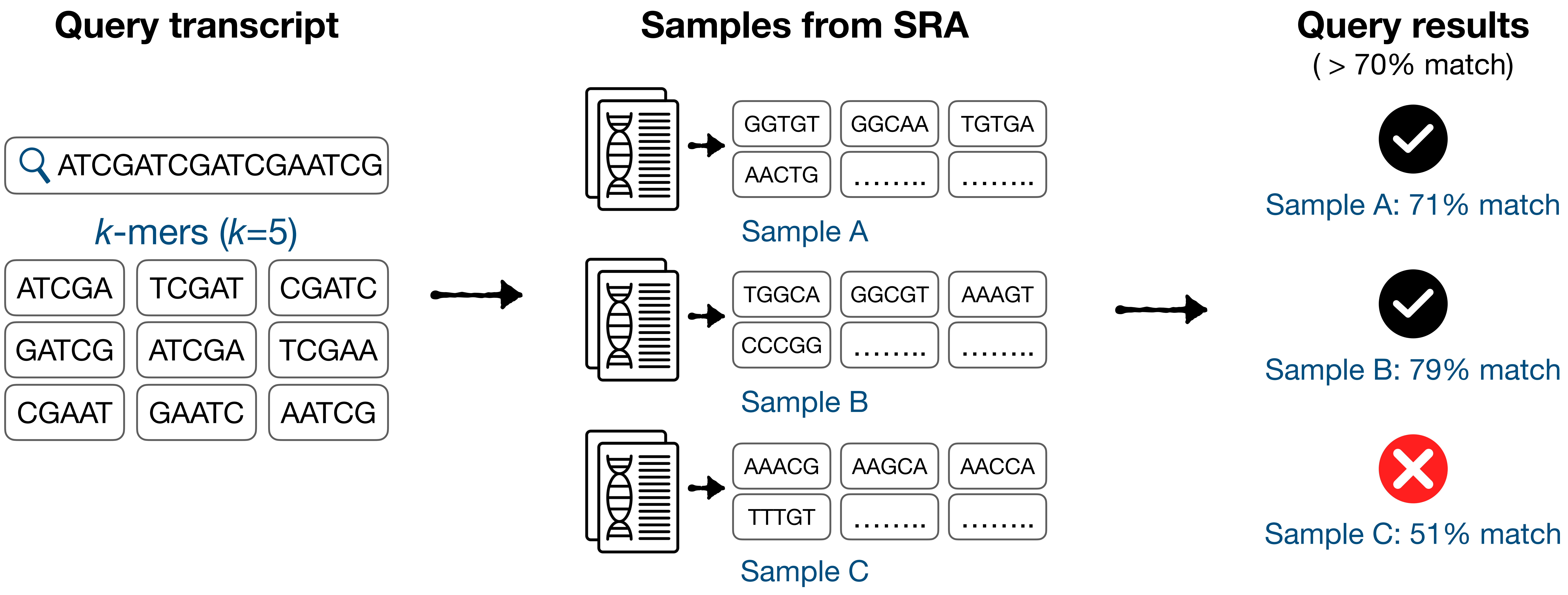
k-mers (*k*=5)

ATCGA	TCGAT	CGATC
GATCG	ATCGA	TCGAA
CGAAT	GAATC	AATCG

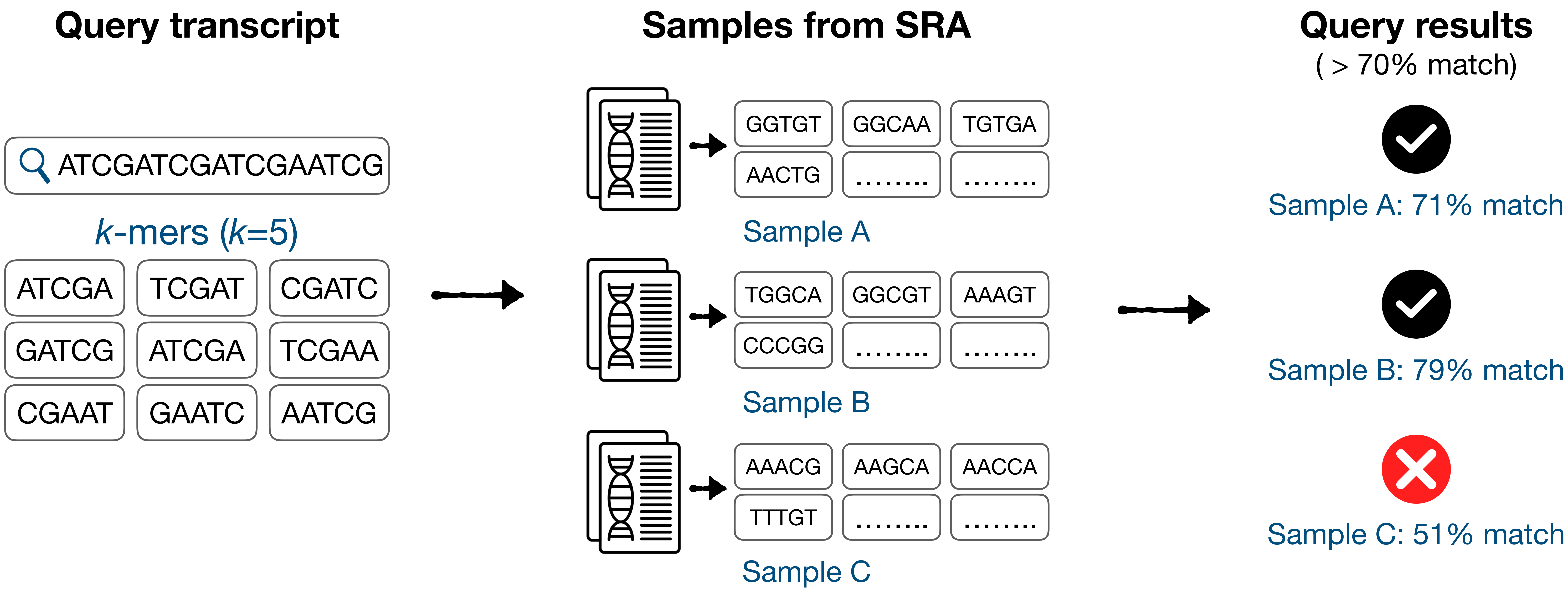
Sample discovery problem



Sample discovery problem

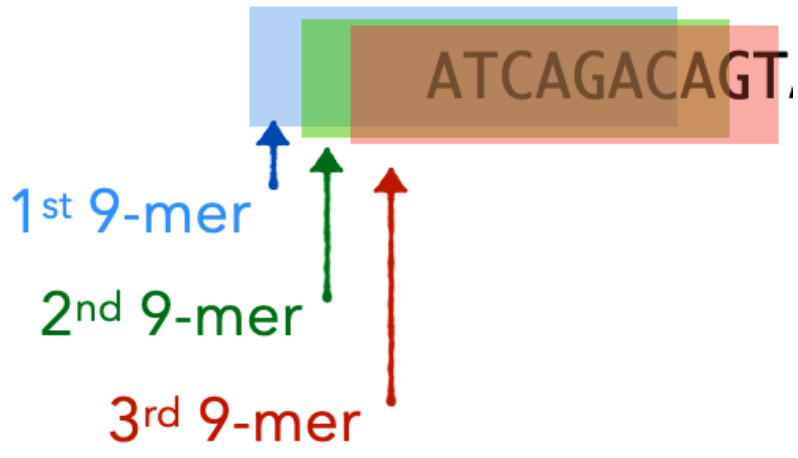


Sample discovery problem



Return all samples that contain at least some θ fraction of k-mers present in the query transcript

Sample discovery ⟶ Vector search



K-mer representation of sequences is employed as vector embeddings. *k* ranges between 21-31

Hamming distance between *k*-mer embeddings is employed as a proxy for sequence similarity

Distance threshold is employed as a proxy for the top-*k* nearest neighbor

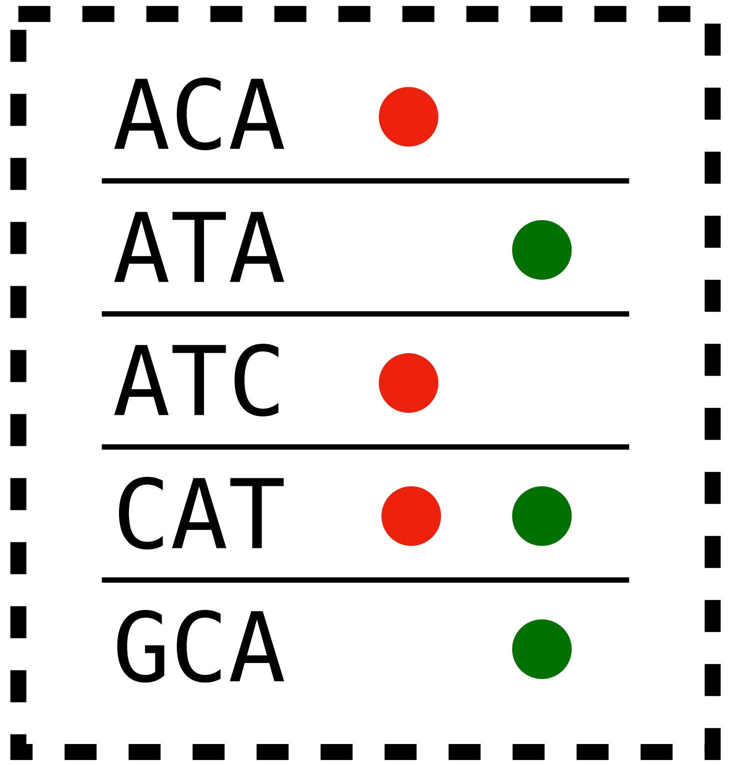
Underlying biological generative model claims that similar species have similar *k*-mer content.

Hamming distance is often not a good proxy for sequence similarity*. (More on this later in the talk.)

*Guillaume Marcais, Dan DeBlasio, Prashant Pandey, Carl Kingsford "Locality Sensitive Hashing for the Edit Distance" ISMB 2019

Several indexing methods for raw sequence search

Color aggregative methods



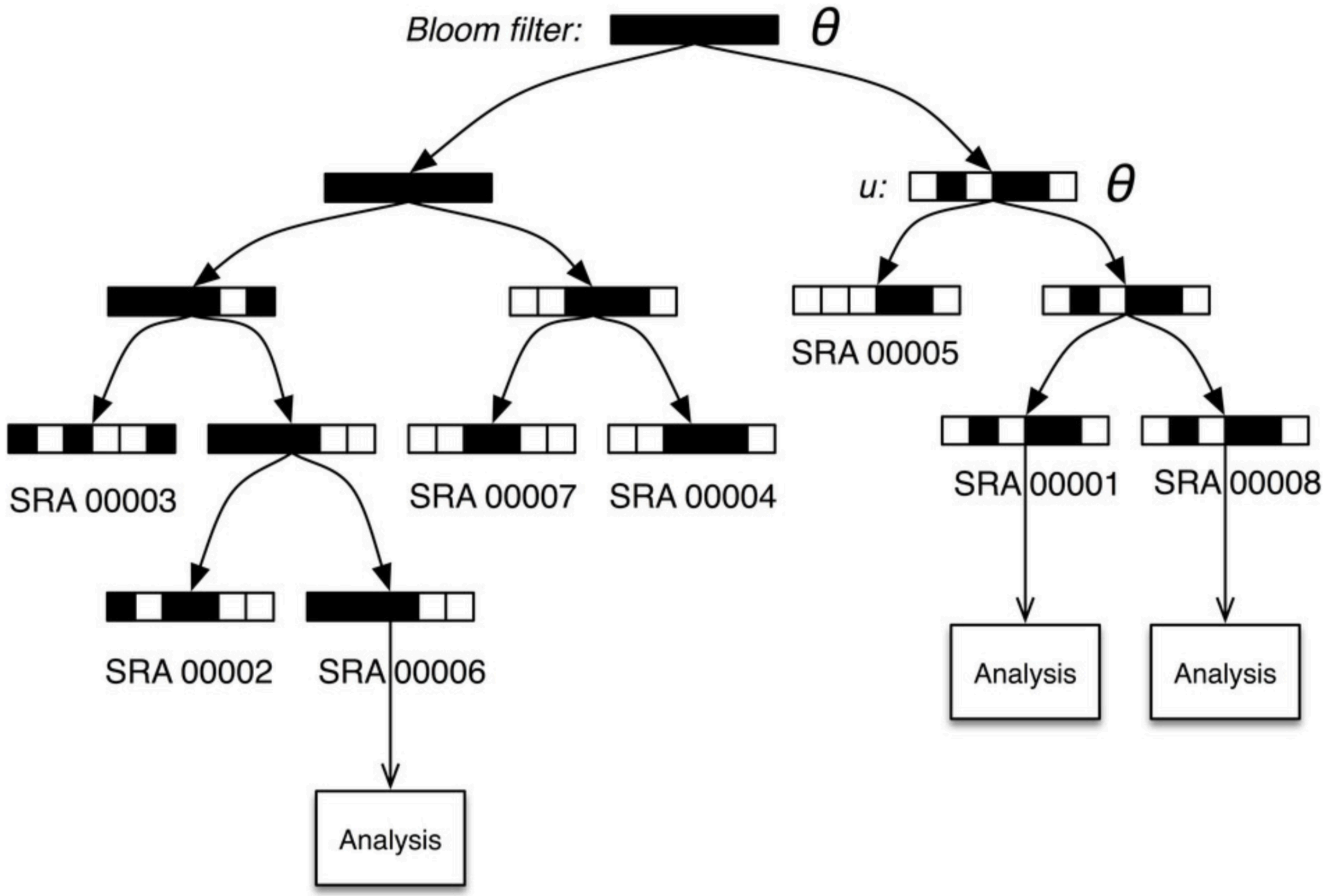
k-mer aggregative methods

	Color aggregative methods	*k*-mer aggregative methods
***K*-mer set data structure**	Hash tables, CQF, BWT, BF trie	Bloom filters
Aggregation data structure	Color matrices	Search tree/forests, Bloom filter matrices
Method names	Mantis, SeqOthello, Bifrost, Metanot, BFT, VARI,	SBT (variants), BIGSI, COBS, RAMBO

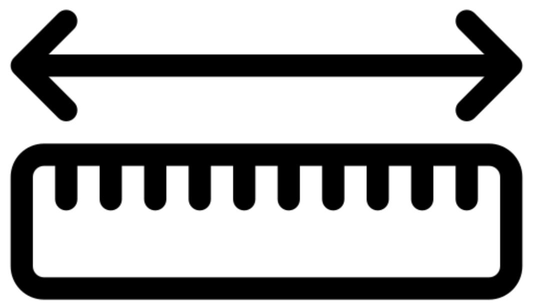
Vibrant area with exciting work over the past several years; excellent review by Marchet et al. 2021.

K-mer aggregative methods: SBT [Solomon & Kingsford 2016]

- 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

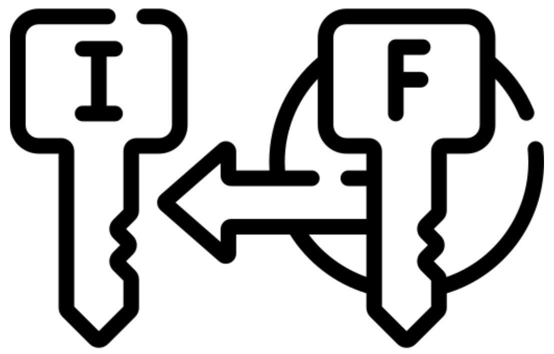


Several limitations in Bloom filter-based methods



All Bloom filter are of same size — creating skewness in the query results

Bloom filters cause false positives in the results



Hard to associate any attributes such as abundance, position etc.

SRA sample sizes range between few MBs to a few GBs. Results in high variance in Bloom filter false-positive rates.

A fundamentally different approach

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

A fundamentally different approach

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

A General-Purpose Counting Filter: Making Every Bit Count

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

SIGMOD 2017

A fundamentally different approach

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

A General-Purpose Counting Filter: Making Every Bit Count

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

SIGMOD 2017

K-mer index

Squeakr: an exact and approximate *k*-mer counting system

Bioinformatics 2018

Prashant Pandey[1,*], Michael A. Bender[1], Rob Johnson[1,2] and Rob Patro[1]

[1]Department of Computer Science, Stony Brook University, Stony Brook, NY 11790, USA and [2]VMware Research, Palo Alto, CA 94304, USA

A fundamentally different approach

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

A General-Purpose Counting Filter: Making Every Bit Count

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

SIGMOD 2017

K-mer index

Interesting observation
about patterns of k-mer occurrence

Rainbowfish: A Succinct Colored de Bruijn Graph Representation*

Fatemeh Almodaresi[1], Prashant Pandey[2], and Rob Patro[3]

WABI 2017

Squeakr: an exact and approximate *k*-mer counting system

Bioinformatics 2018

Prashant Pandey[1,*], Michael A. Bender[1], Rob Johnson[1,2] and Rob Patro[1]

[1]Department of Computer Science, Stony Brook University, Stony Brook, NY 11790, USA and [2]VMware Research, Palo Alto, CA 94304, USA

A fundamentally different approach

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

A General-Purpose Counting Filter: Making Every Bit Count

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

SIGMOD 2017

K-mer index

Squeakr: an exact and approximate *k*-mer counting system

Bioinformatics 2018

Prashant Pandey[1,*], Michael A. Bender[1], Rob Johnson[1,2] and Rob Patro[1]

[1]Department of Computer Science, Stony Brook University, Stony Brook, NY 11790, USA and [2]VMware Research, Palo Alto, CA 94304, USA

Interesting observation
about patterns of k-mer occurrence

Rainbowfish: A Succinct Colored de Bruijn Graph Representation*

Fatemeh Almodaresi[1], Prashant Pandey[2], and Rob Patro[3]

WABI 2017

“I bet we can exploit
that for large-scale search”

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

Prashant Pandey[1], Fatemeh Almodaresi[1], Michael A. Bender[1], Michael Ferdman[1], Rob Johnson[2,1], and Rob Patro[1]

RECOMB 2018 & Cell Systems

A fundamentally different approach

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

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 Almodaresi[1], Prashant Pandey[2], and Rob Patro[3]

WABI 2017

K-mer index

Squeakr: an exact an
counting system

Prashant Pandey[1,*], Michael A

[1]Department of Computer Science, Stony Brook
Palo Alto, CA 94304, USA

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

Fatemeh Almodaresi[1], Prashant Pandey[1], Michael Ferdman[1], Rob Johnson[2,1], and Rob Patro[1]

RECOMB 2019 & JCB 2020

“I bet we can exploit
that for large-scale search”

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

Prashant Pandey[1], Fatemeh Almodaresi[1], Michael A. Bender[1], Michael Ferdman[1], Rob Johnson[2,1], and Rob Patro[1]

RECOMB 2018 & Cell Systems

“I bet we can make
it even smaller”

A fundamentally different approach

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

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 Almodaresi[1], Prashant Pandey[2], and Rob Patro[3]

WABI 2017

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

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

Prashant Pandey[1], Fatemeh Almodaresi[1], Michael A. Bender[1], Michael Ferdman[1], Rob Johnson[2,1], and Rob Patro[1]

RECOMB 2018 & Cell Systems

K-mer index

Squeakr: an exact an... counting system

Prashant Pandey[1,*], Michael A...
[1]Department of Computer Science, Stony Brook...
Palo Alto, CA 94304, USA

"I bet we can make
it even smaller"

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

Fatemeh Almodaresi[1], Prashant Pandey[1], Michael Ferdman[1], Rob Johnson[2,1], and Rob Patro[1]

RECOMB 2019 & JCB 2020

"I bet we can make
it scale and updatable"

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

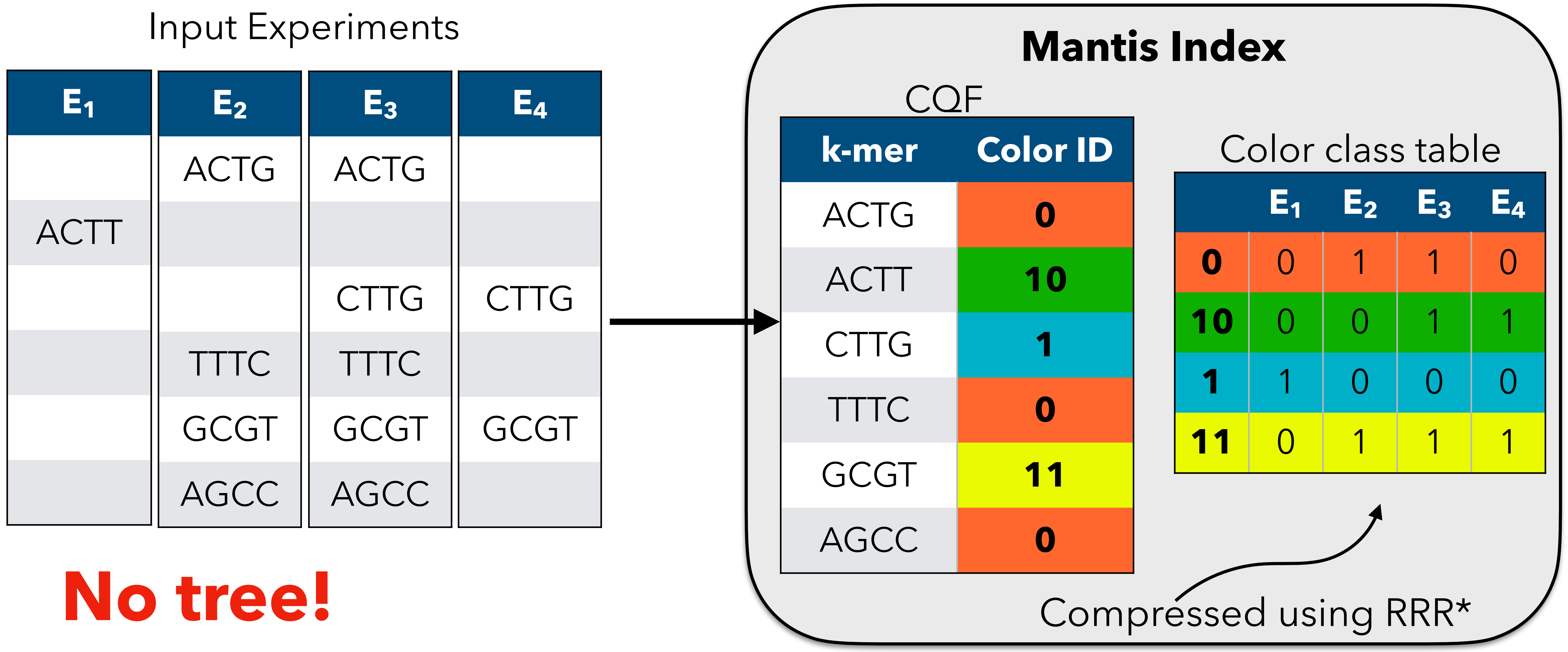
Fatemeh Almodaresi [1], Jamshed Khan [1], Sergey Madaminov[2], Michael Ferdman[2], Rob Johnson[3], Prashant Pandey[3] and Rob Patro [1,*]

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

Bioinformatics 2022

Color aggregative methods: Mantis [Pandey et al. 2018]

- Build a counting quotient filter for each input sample (can be different sizes based on the number of *k*-mers)
- CQF: key=*k*-mer value=color class ID
- Combine them via multi-way merge
- Estimate a good ordering of color class IDs from first few million *k*-mers



No tree!

*Raman, et al. (2002). Succinct indexable dictionaries with applications to encoding k-ary trees and multisets. SODA

Mantis is faster, smaller, and accurate than SBT

Indexed 2,652 human BBB RNA-seq (gene expression) samples ~**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

Query **includes index loading**

θ threshold for SSBT query

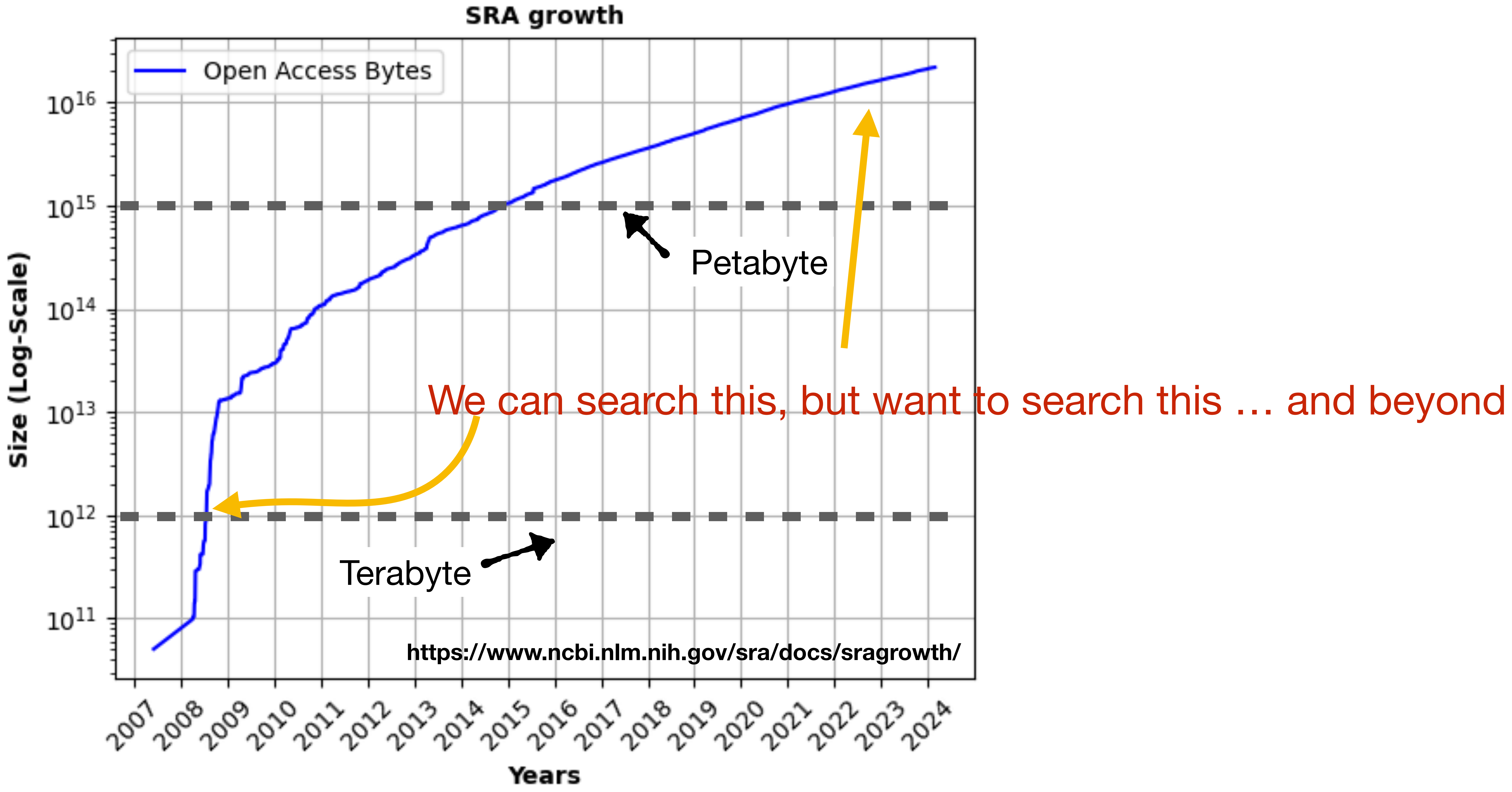
	Mantis	SSBT (0.7)	SSBT (0.8)	SSBT (0.9)
10 Transcripts	25 s	3 min 8 s	2 min 25 s	2 min 7 s
100 Transcripts	28 s	14 min 55 s	10 min 56 s	7 min 57 s
1000 Transcripts	1 min 3 s	2 hr 22 min	1 hr 54 min	1 hr 20 min

Mantis can be constructed ~24x faster than a comparable SSBT* [Solomon & Kingsford 2017]

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

The final Mantis representation is ~20% smaller than the comparable SSBT representation.

Where we are now

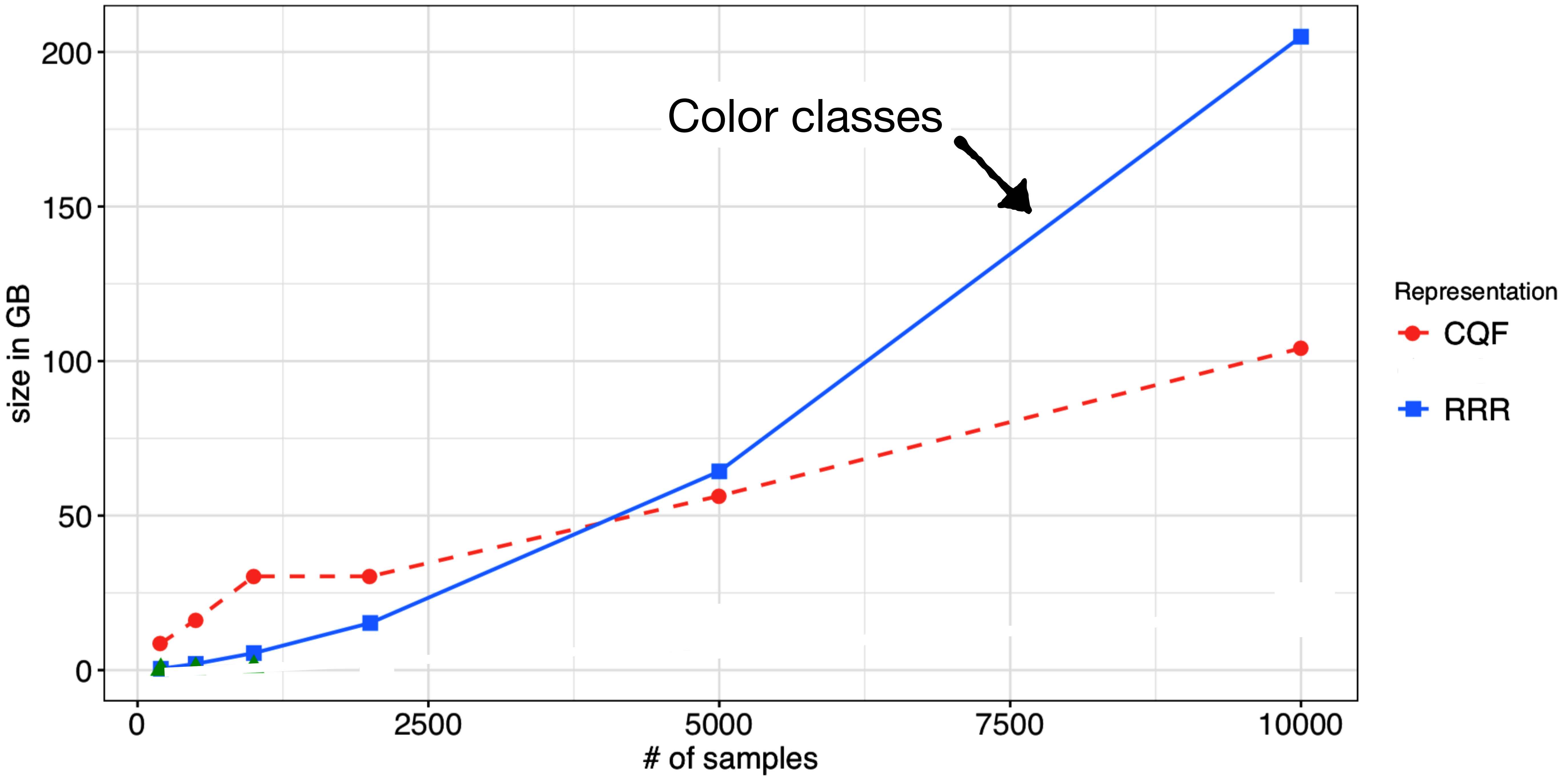


Some remaining challenges

We demonstrate indexing on the order of 10^3 samples, we really want to index on the order of 10^6 samples

Key observations:

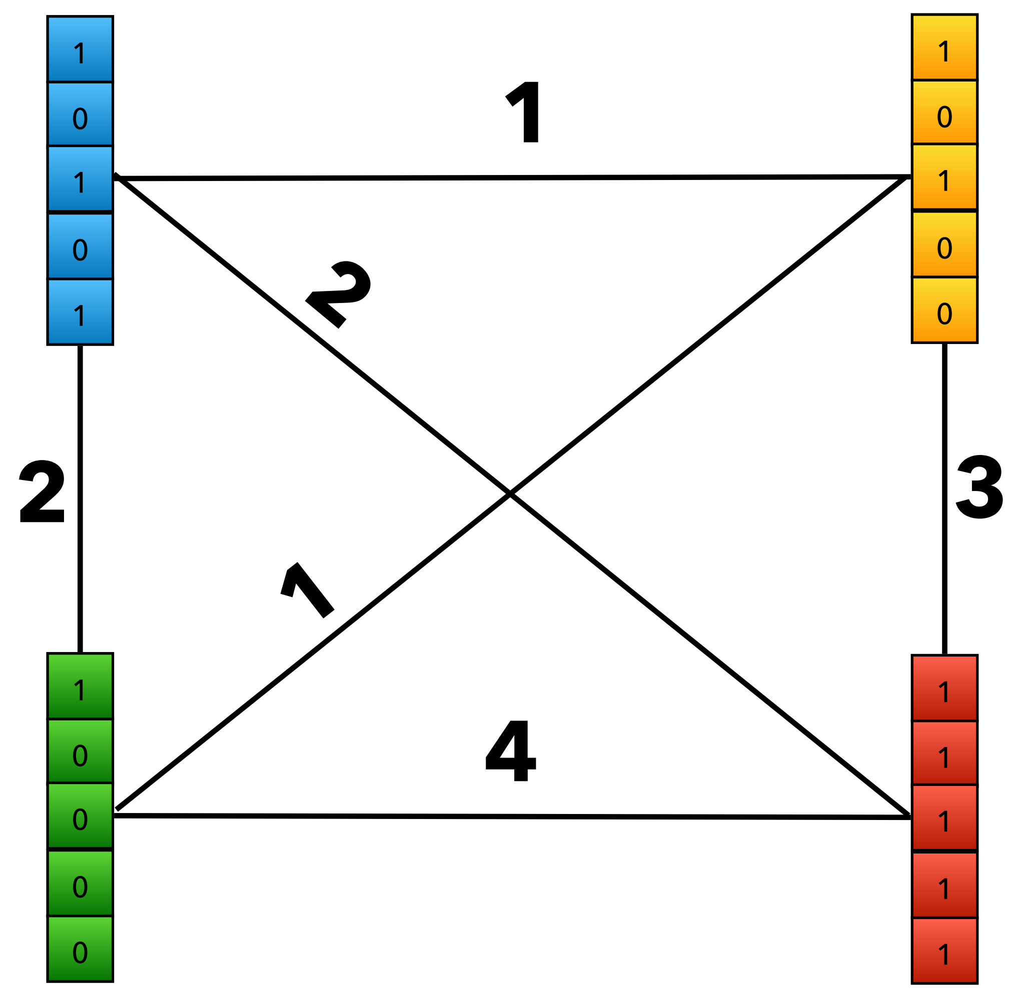
- *K*-mers grow at worst linearly
- Color classes increase super-linearly



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

Consider the following color class graph

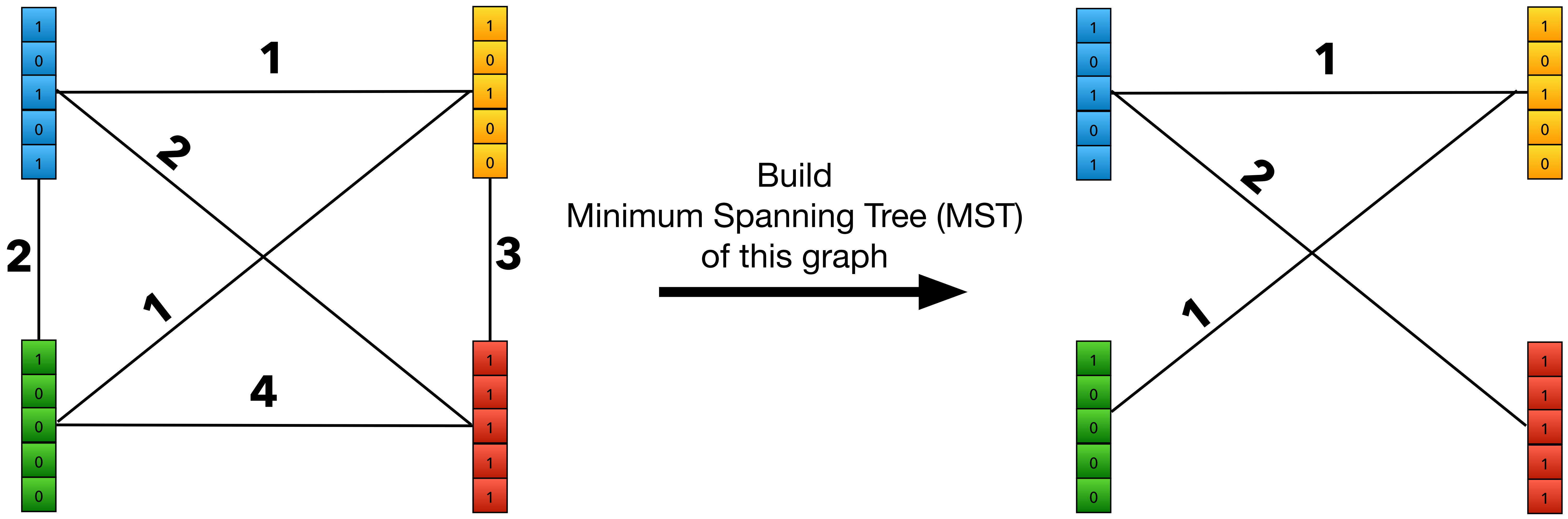
Each color class is a vertex in the graph
Every pair of color classes is connected by an edge whose weight is the hamming distance between the color class vectors



Bookstein, Abraham, and Shmuel T. Klein. "Compression of correlated bit-vectors." *Inf. Syst.* 16.4 (1991): 387-400.

Consider the following color class graph

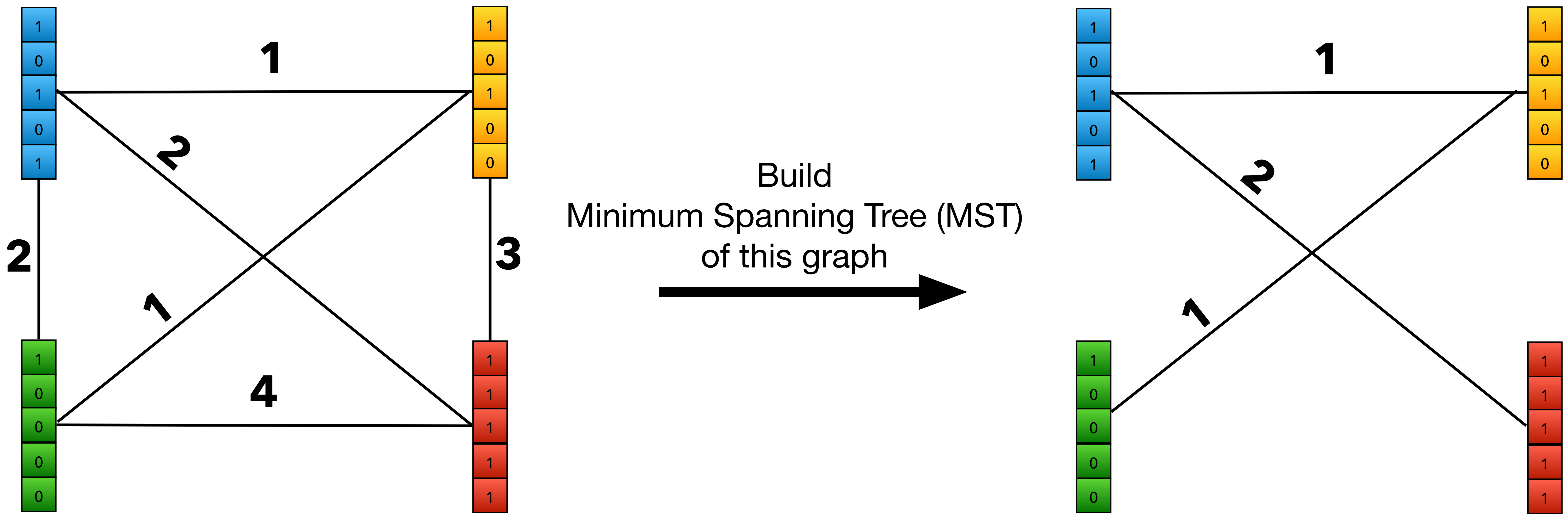
Each color class is a vertex in the graph
Every pair of color classes is connected by an edge whose weight is the hamming distance between the color class vectors



Bookstein, Abraham, and Shmuel T. Klein. "Compression of correlated bit-vectors." *Inf. Syst.* 16.4 (1991): 387-400.

Consider the following color class graph

Each color class is a vertex in the graph
Every pair of color classes is connected by an edge whose weight is the hamming distance between the color class vectors



Unfortunately:

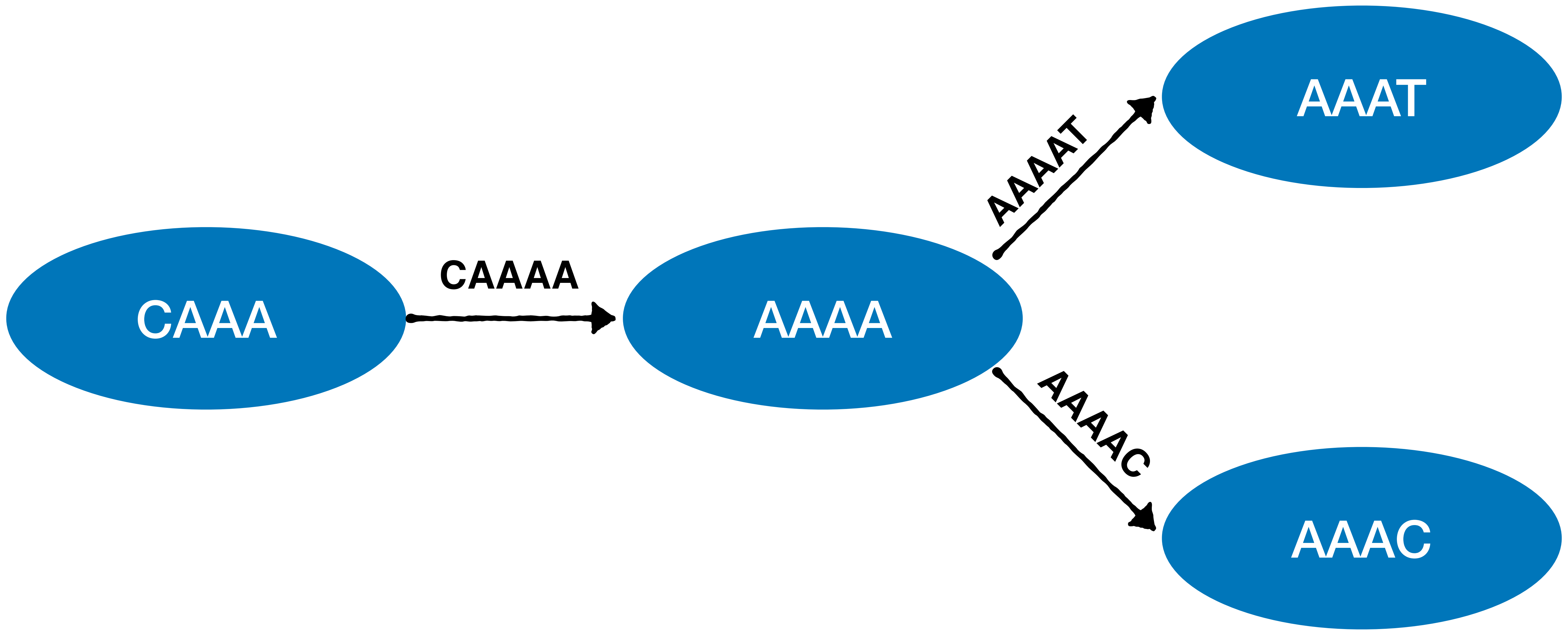
1) There are many color classes (> 1 Billion, full graph too big)
2) They are high-dimensional (# of samples), neighbor search is hard (LSH scheme seem to work poorly)

Bookstein, Abraham, and Shmuel T. Klein. "Compression of correlated bit-vectors." *Inf. Syst.* 16.4 (1991): 387-400.

De Bruijn graph (dBG) over *k*-mers

Read 1: ….CAAAAT….

Read 2: ….CAAAAC….

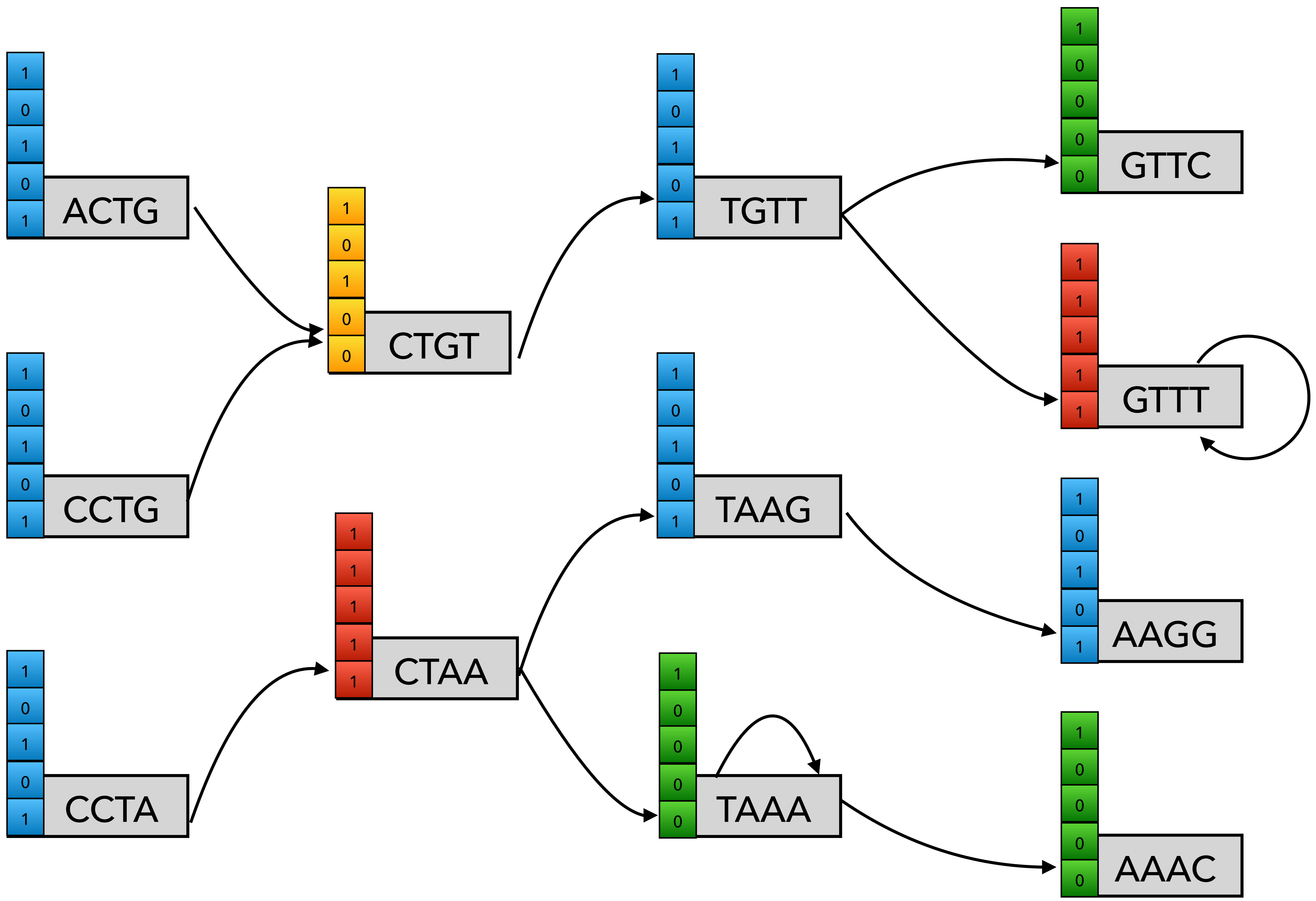


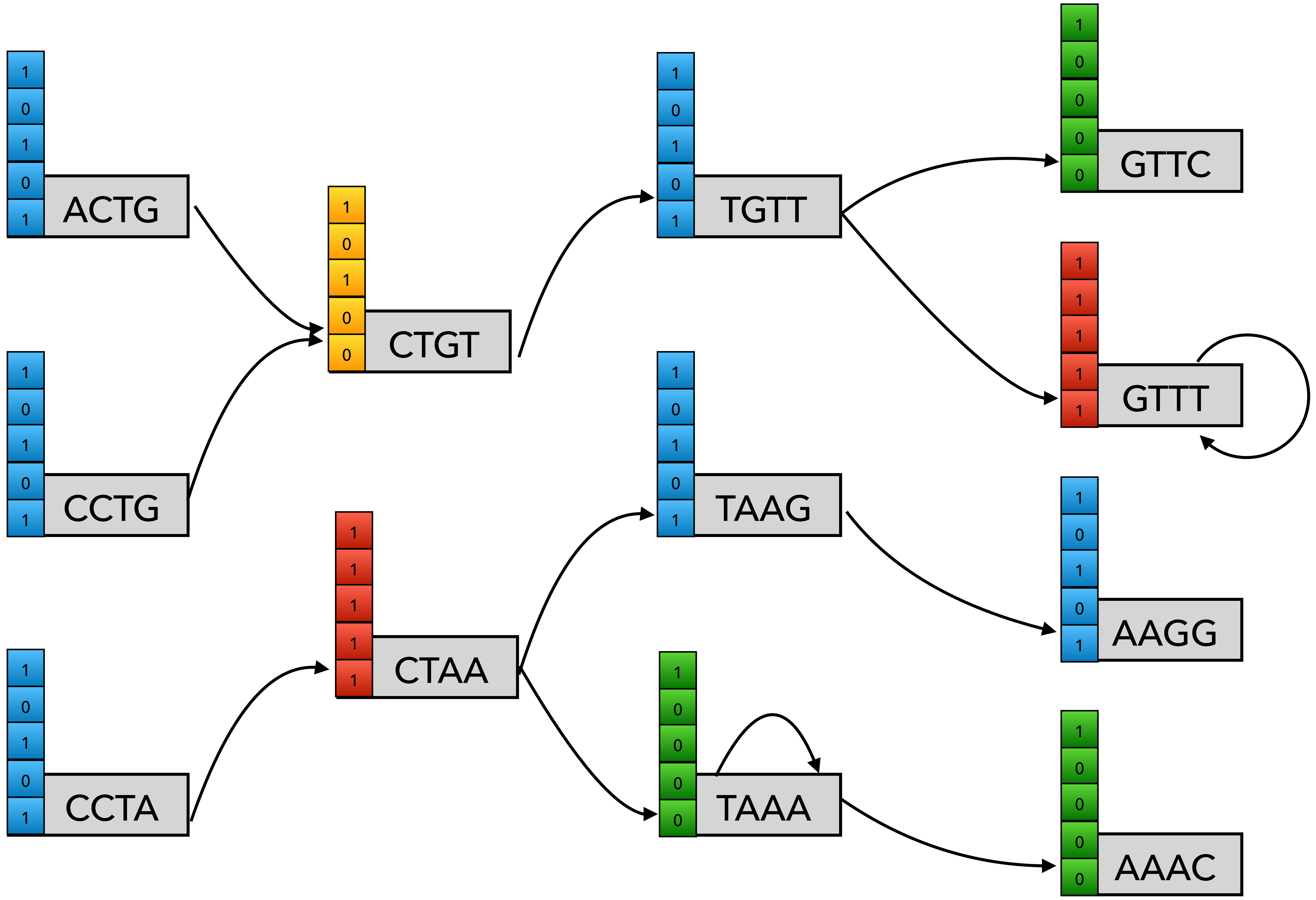
Use the de Bruin graph (dBG) as an efficient guide for near-neighbor search in the space of color classes!

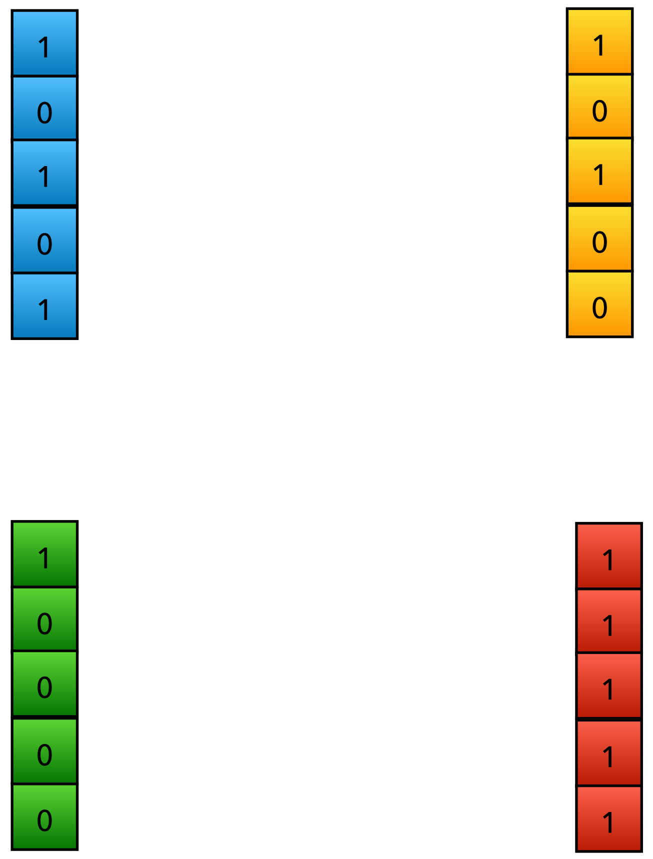
dBG common in genomics. Nodes u, v are *k*-mers & are adjacent if *k*-1 suffix of u is the *k*-1 prefix of v

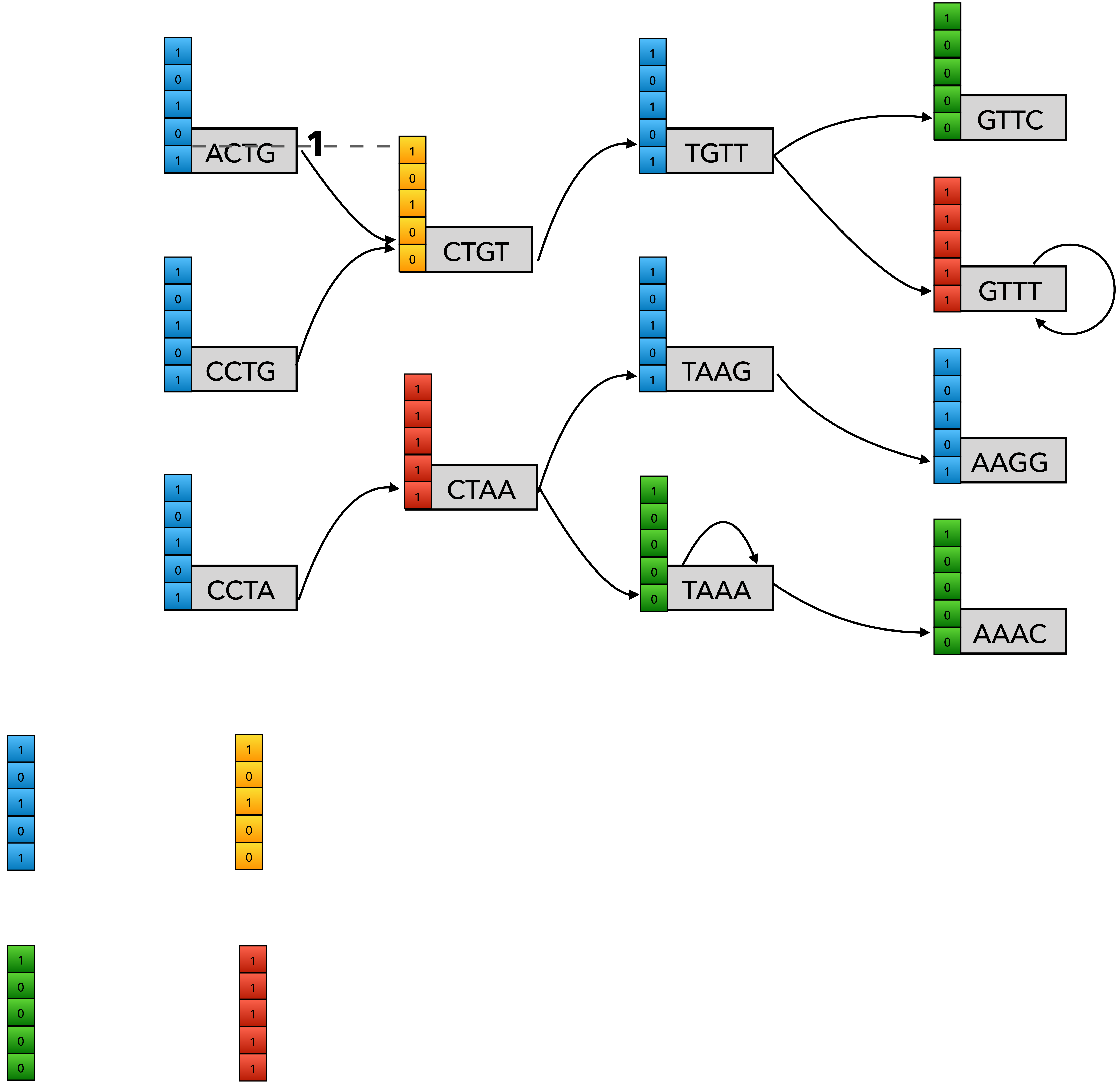
Mantis implicitly represents a colored dBG

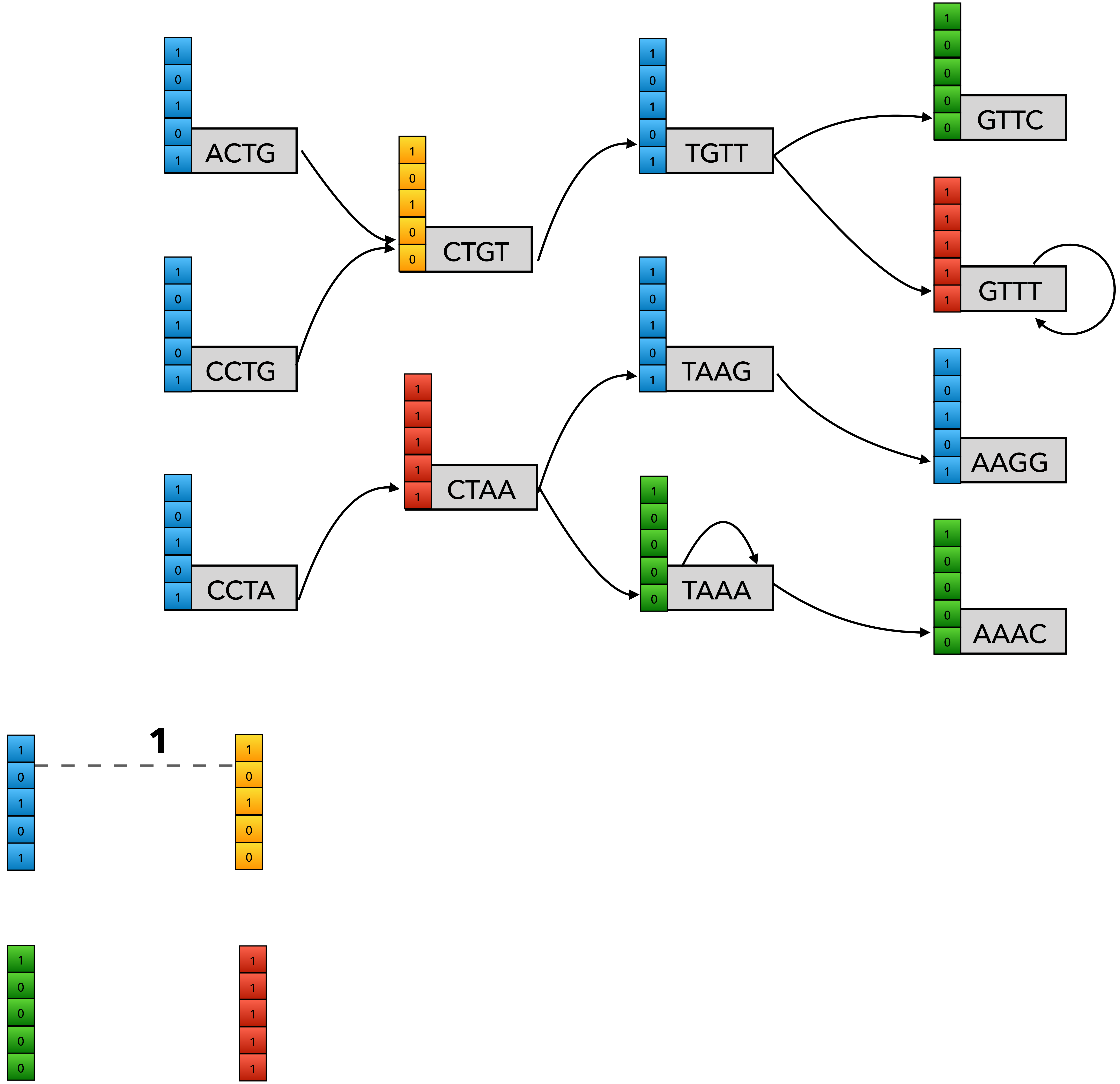
Each CQF key represents a *k*-mer → can explicitly query neighbors
Each *k*-mer associated with color class id → vector of occurrences

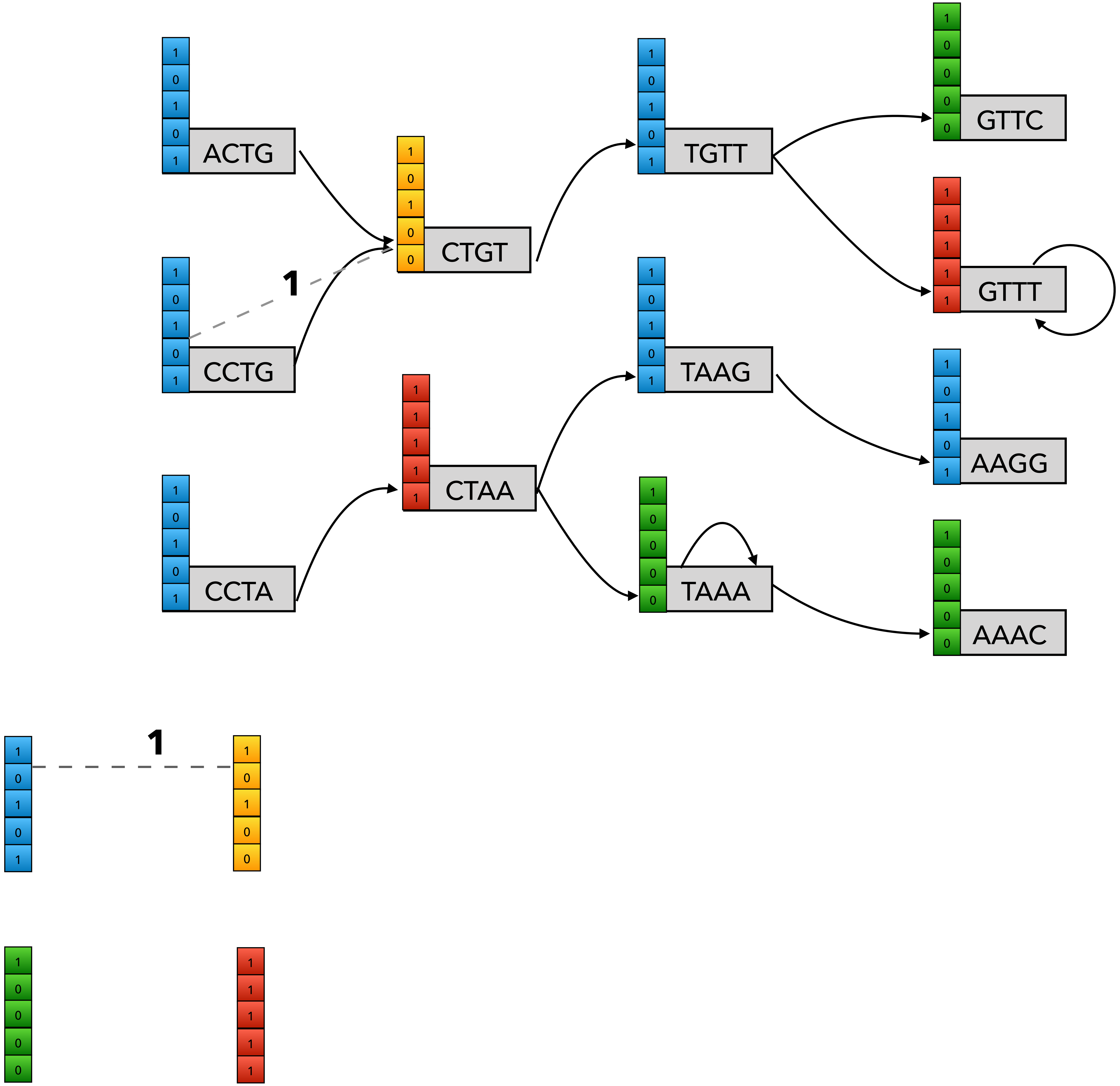


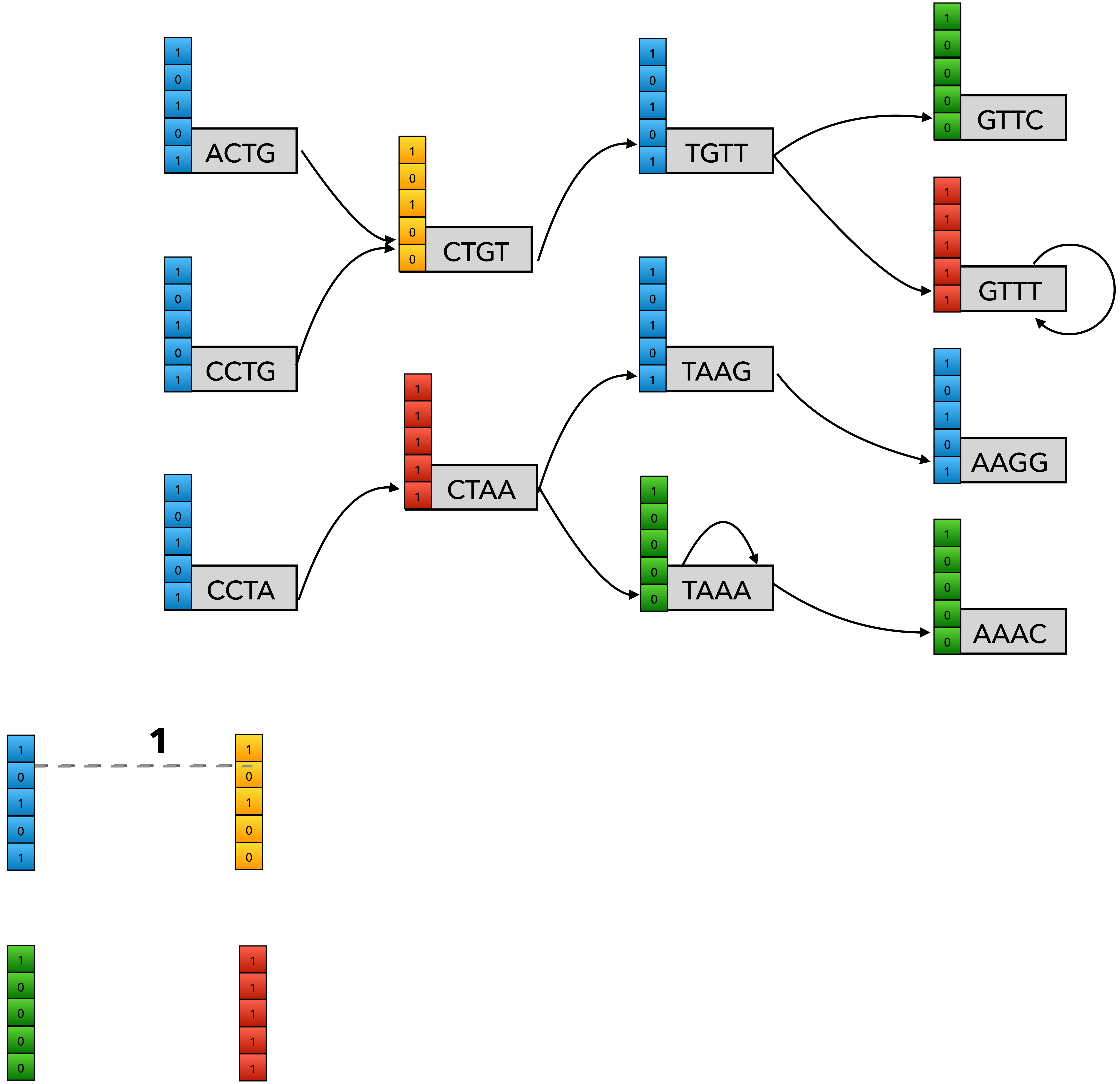

ACTG
CCTG
CCTA
CTGT
CTAA
TGTT
TAAG
TAAA
GTTC
GTTT
AAGG
AAAC

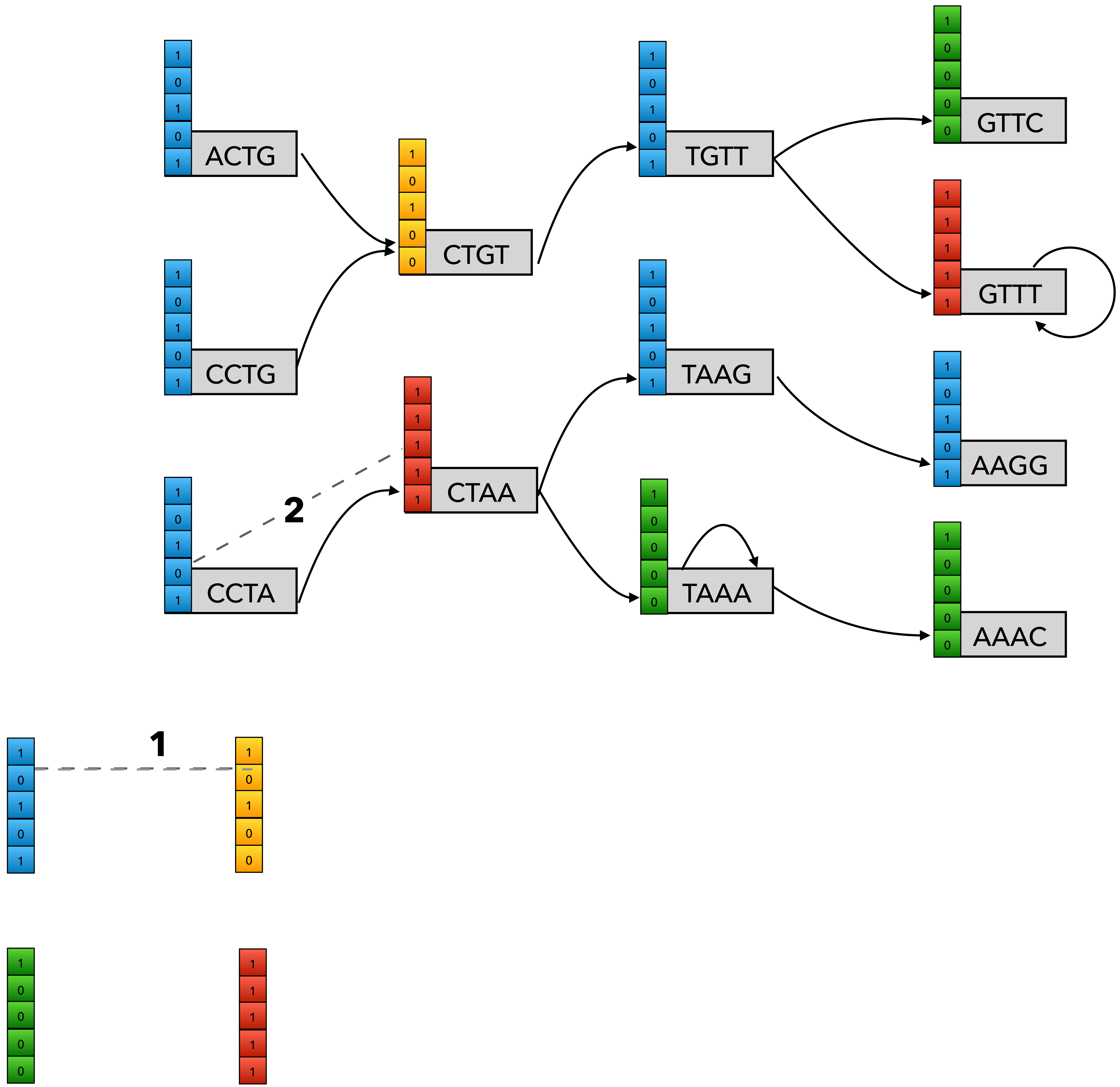


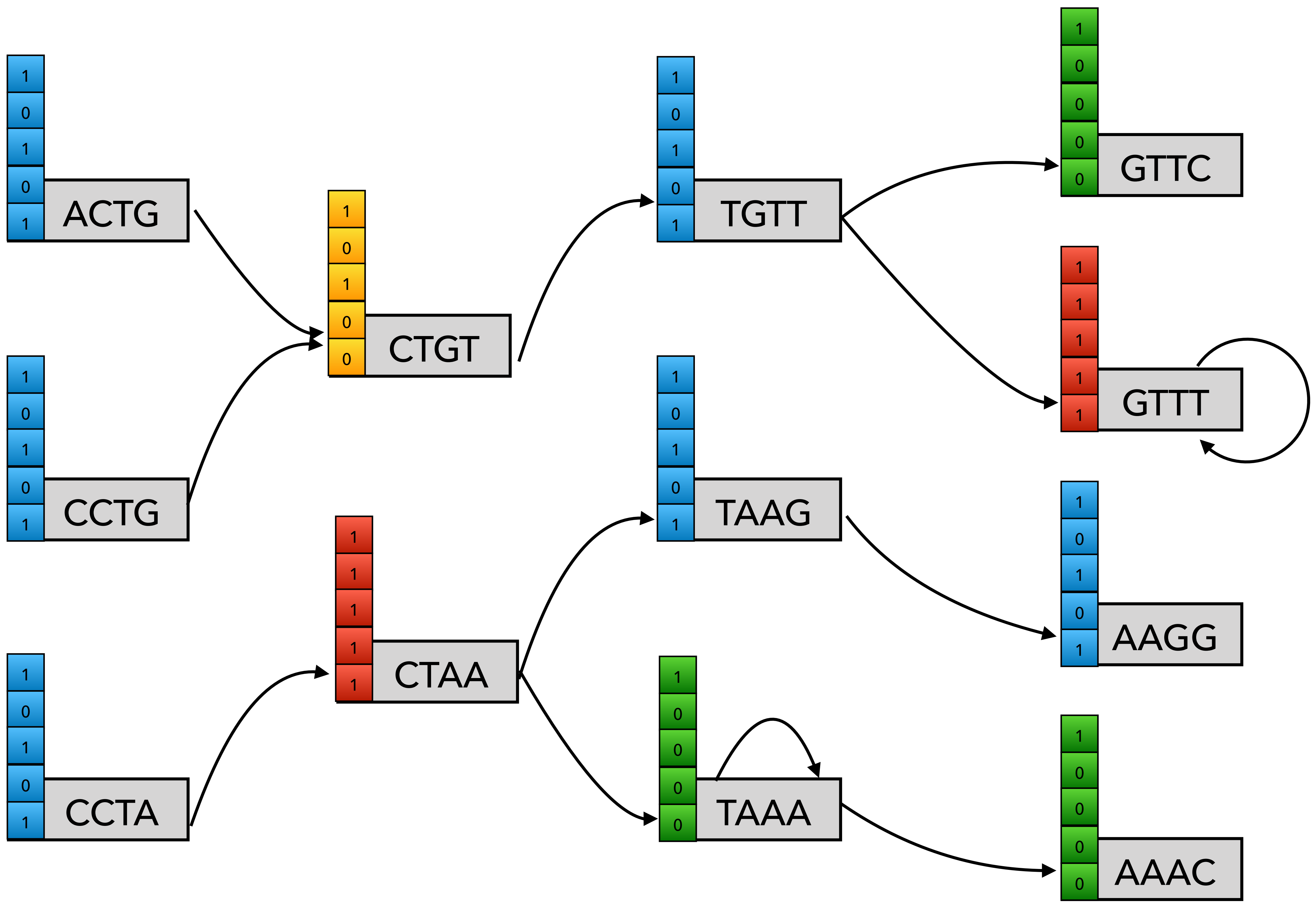

ACTG
1
CTGT
CCTG
CCTA
CTAA
TGTT
TAAG
TAAA
GTTC
GTTT
AAGG
AAAC

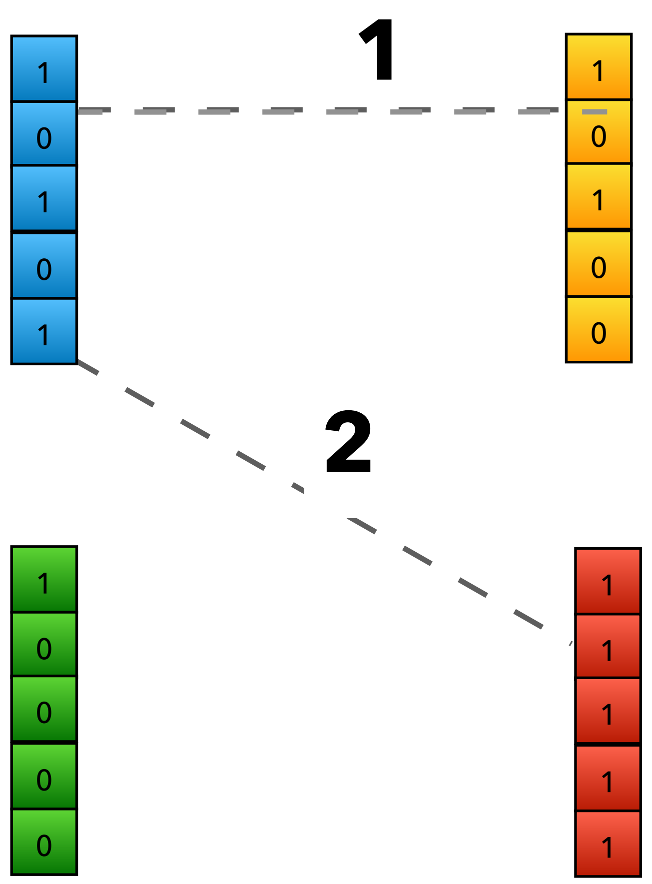

ACTG
CCTG
CCTA
CTGT
CTAA
TGTT
TAAG
TAAA
GTTC
GTTT
AAGG
AAAC
1

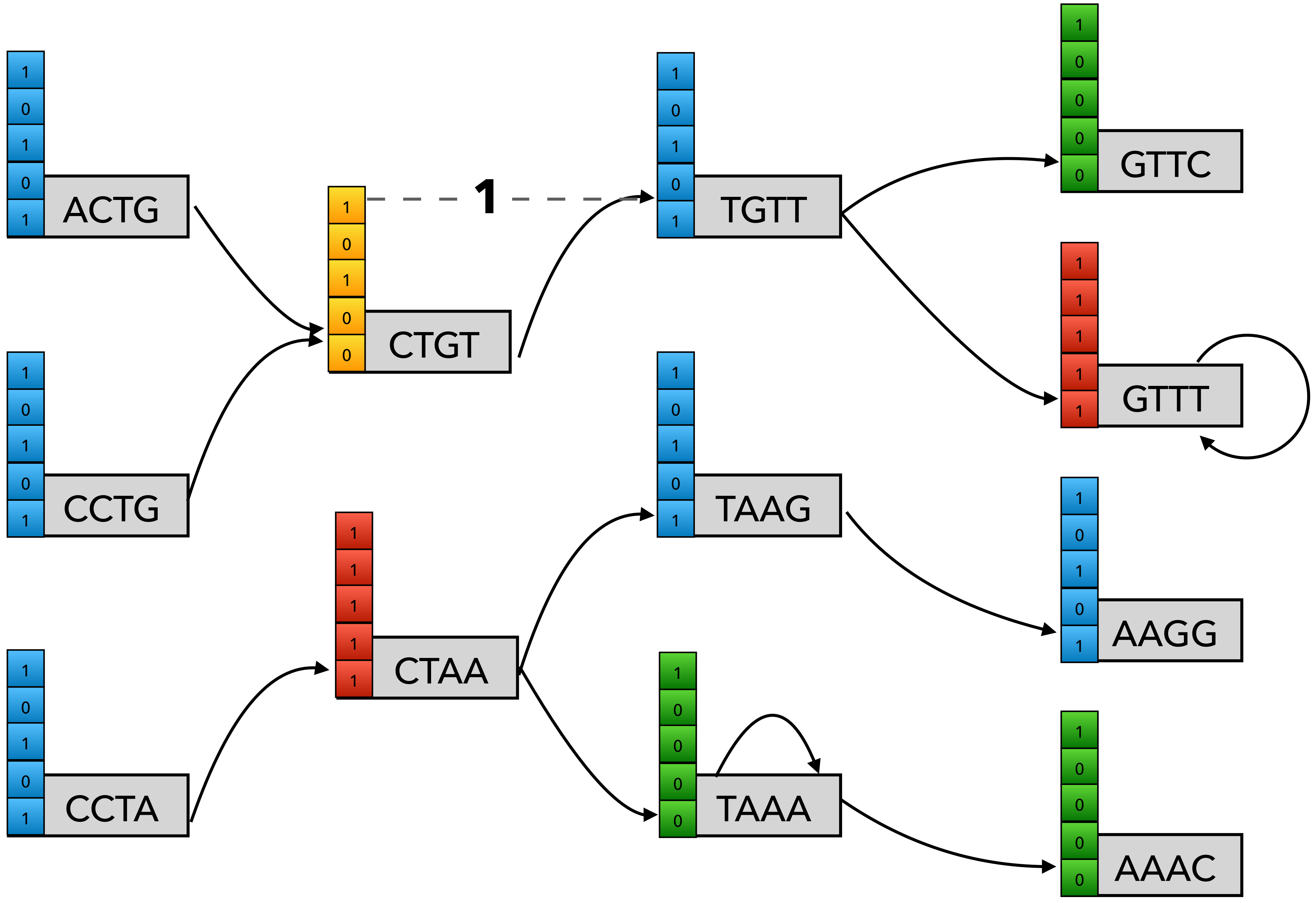

ACTG
CCTG
CCTA
1
CTGT
CTAA
TGTT
TAAG
TAAA
GTTC
GTTT
AAGG
AAAC
1

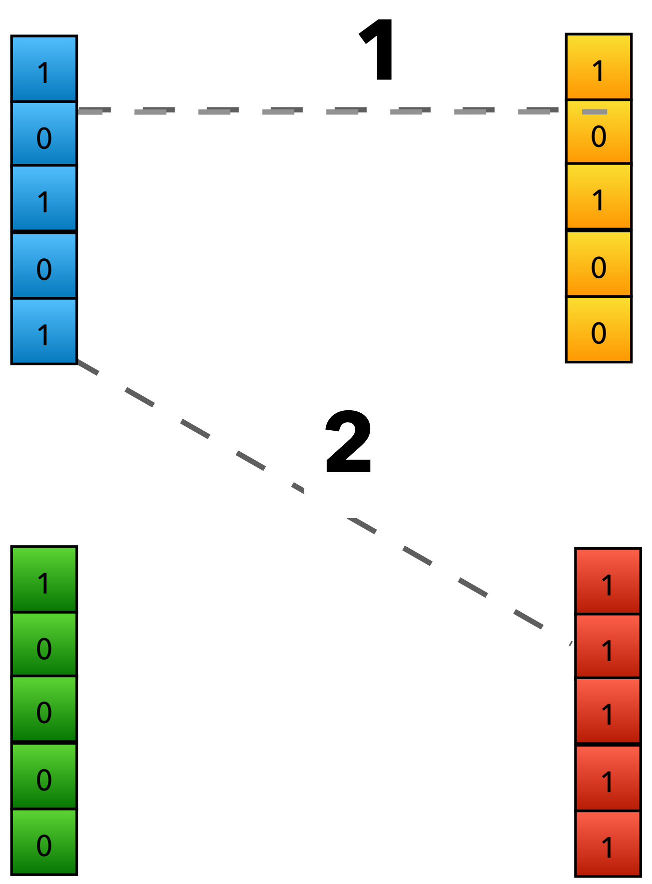

ACTG
CCTG
CCTA
CTGT
CTAA
TGTT
TAAG
TAAA
GTTC
GTTT
AAGG
AAAC
1

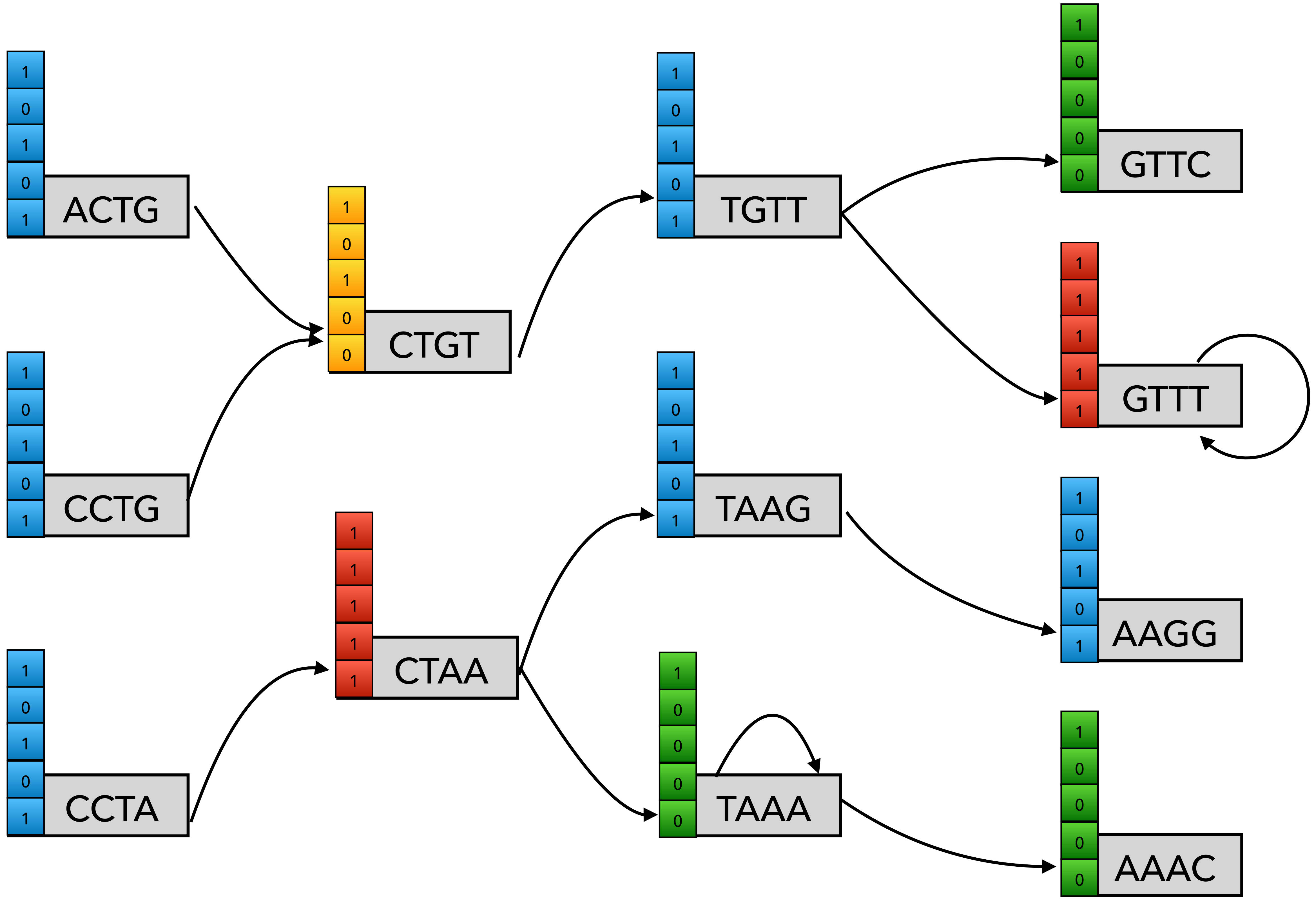

ACTG
CCTG
CCTA
CTGT
CTAA
TGTT
TAAG
TAAA
GTTC
GTTT
AAGG
AAAC
2
1

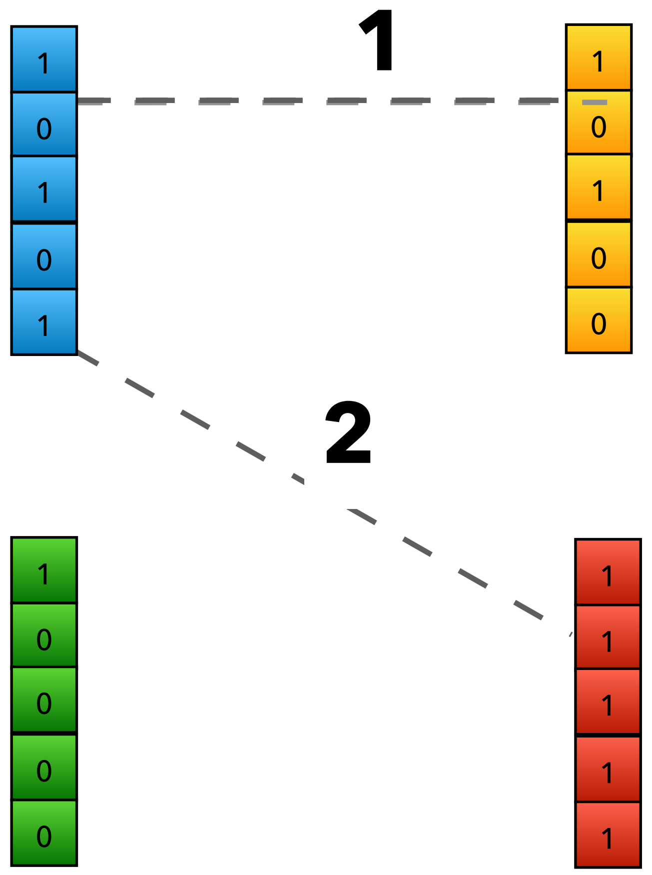

ACTG
CCTG
CCTA
CTGT
CTAA
TGTT
TAAG
TAAA
GTTC
GTTT
AAGG
AAAC

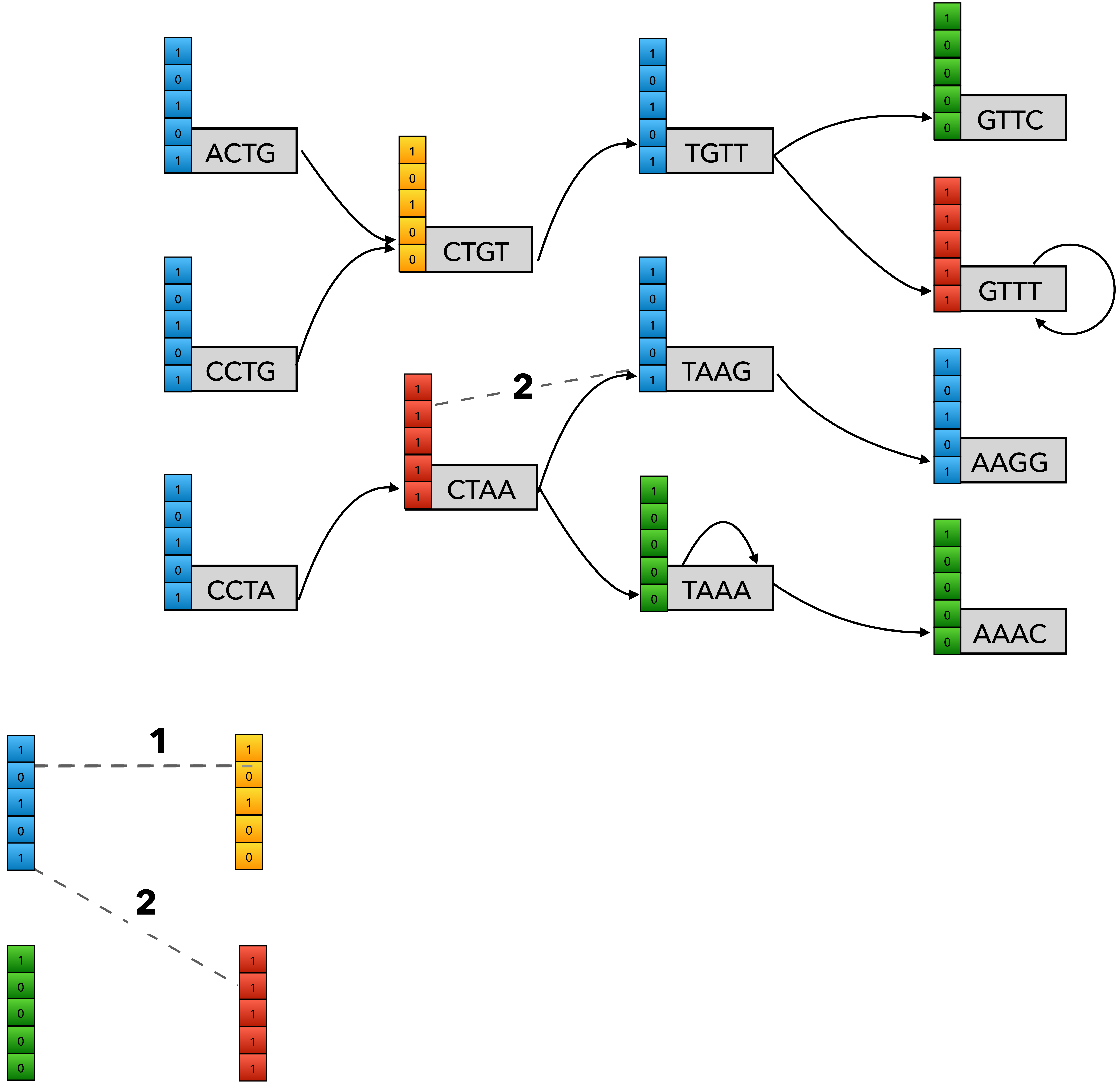

1
2

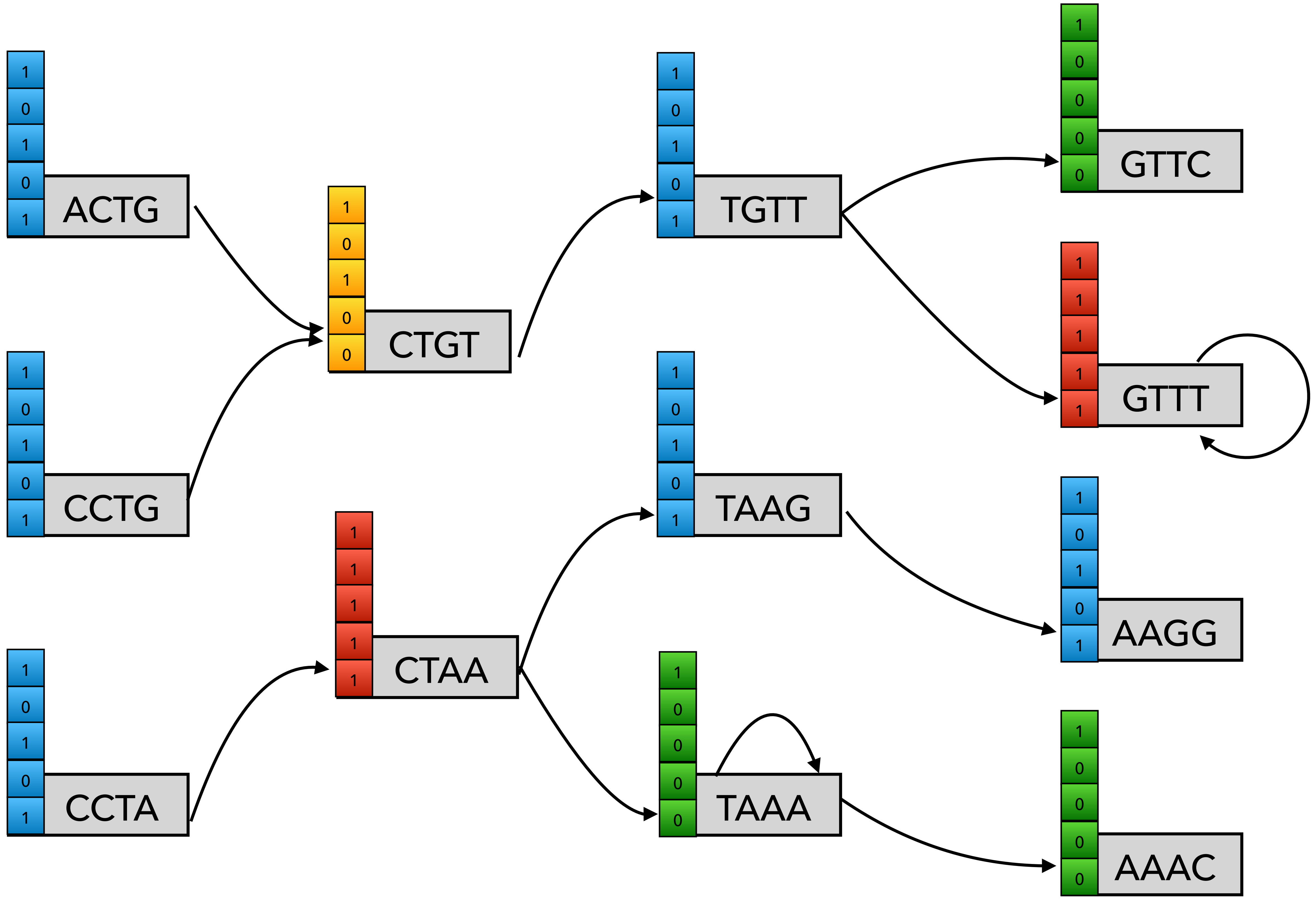

ACTG
CCTG
CCTA
CTGT
CTAA
1
TGTT
TAAG
TAAA
GTTC
GTTT
AAGG
AAAC

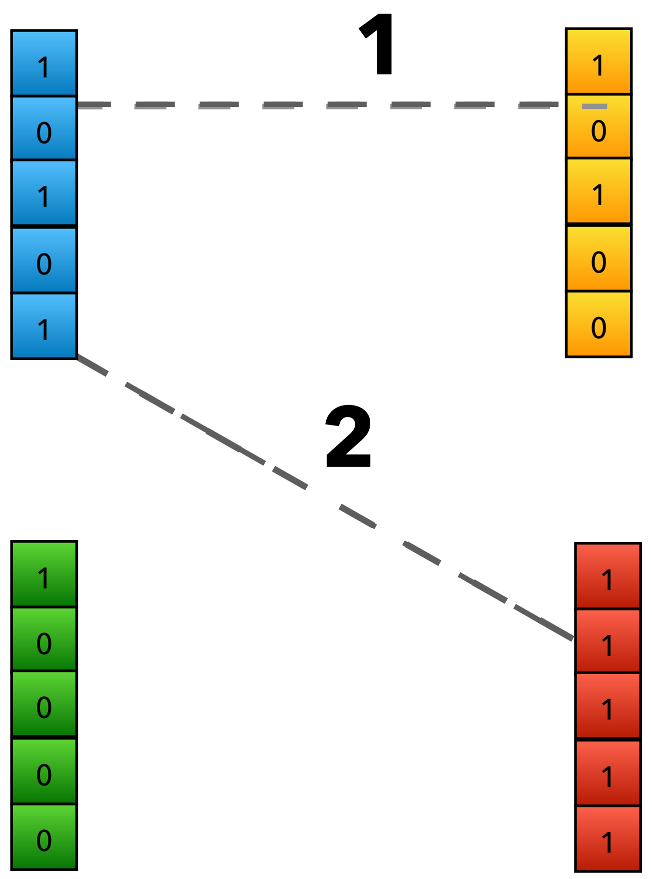

1
2

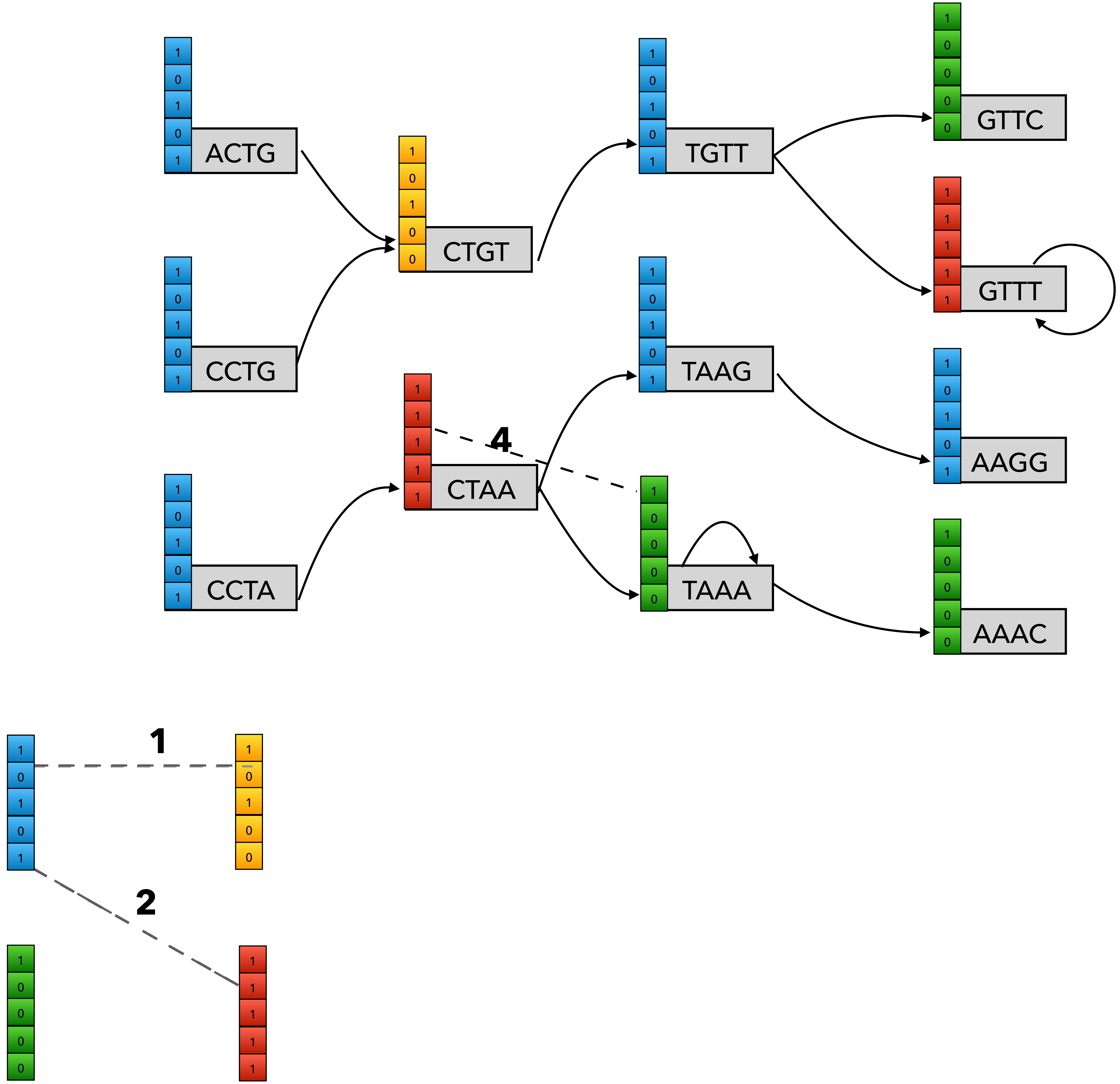

ACTG
CCTG
CCTA
CTGT
CTAA
TGTT
TAAG
TAAA
GTTC
GTTT
AAGG
AAAC

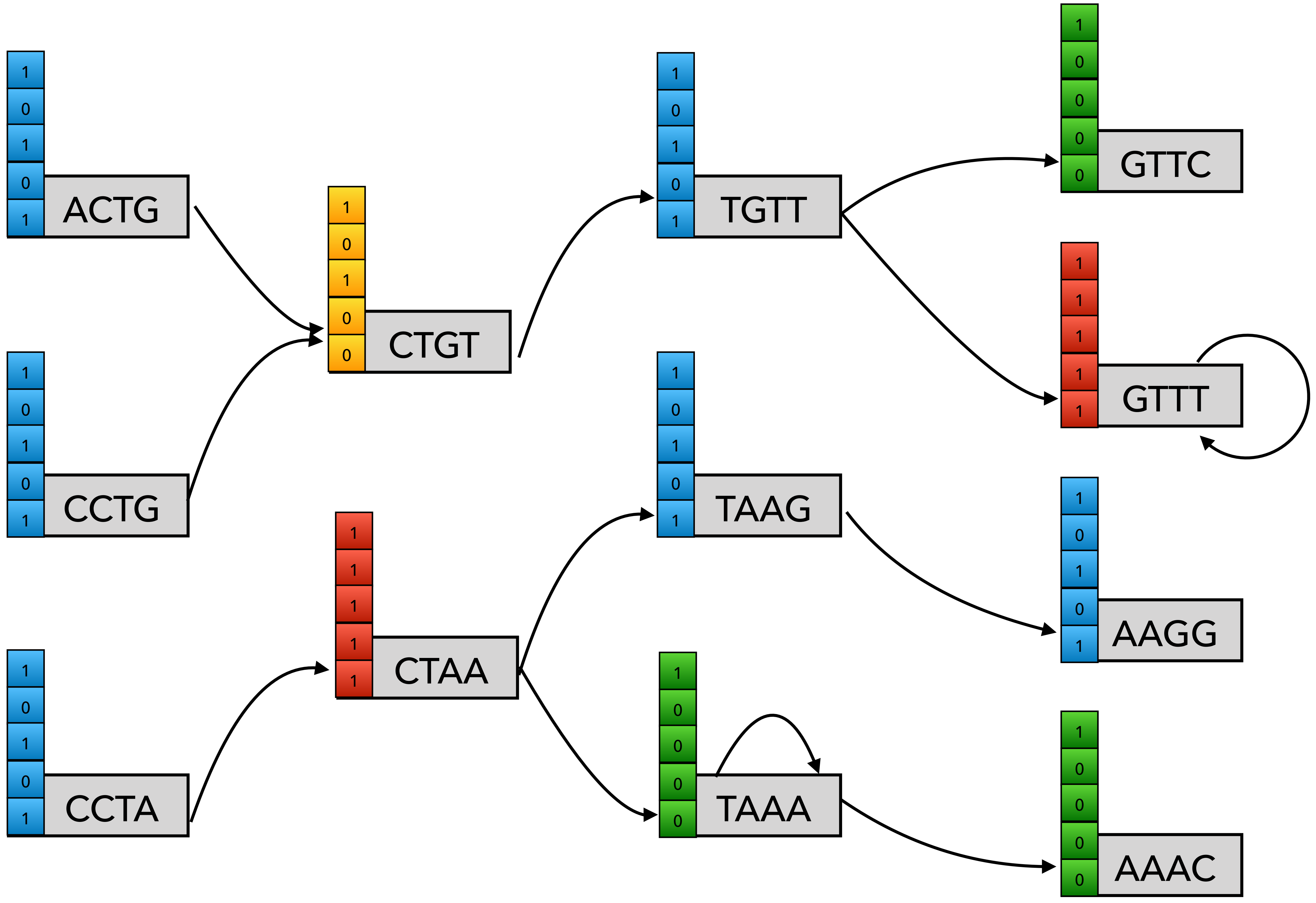

1
2

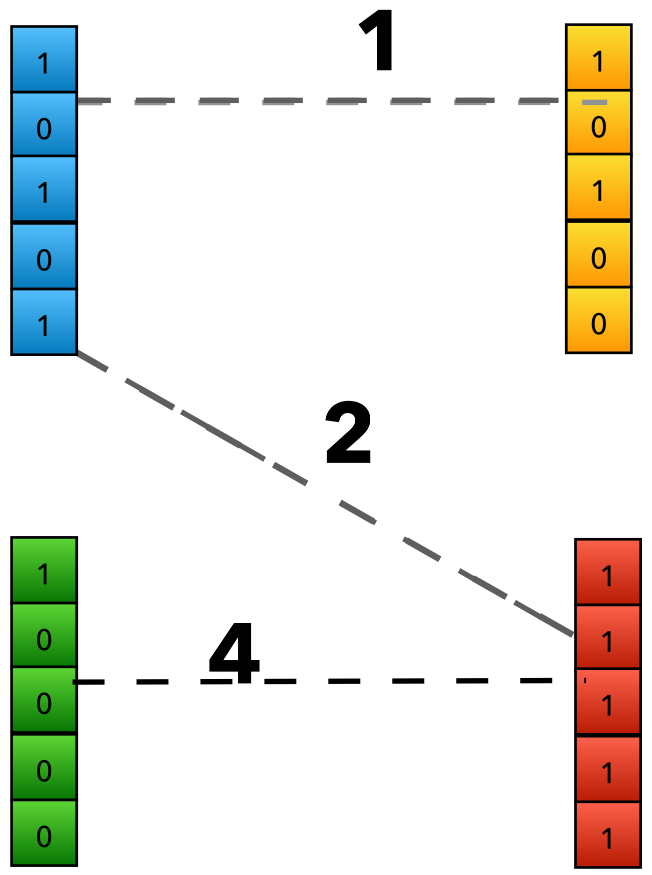

ACTG
CCTG
CCTA
CTGT
CTAA
TGTT
TAAG
TAAA
GTTC
GTTT
AAGG
AAAC
1
2
2

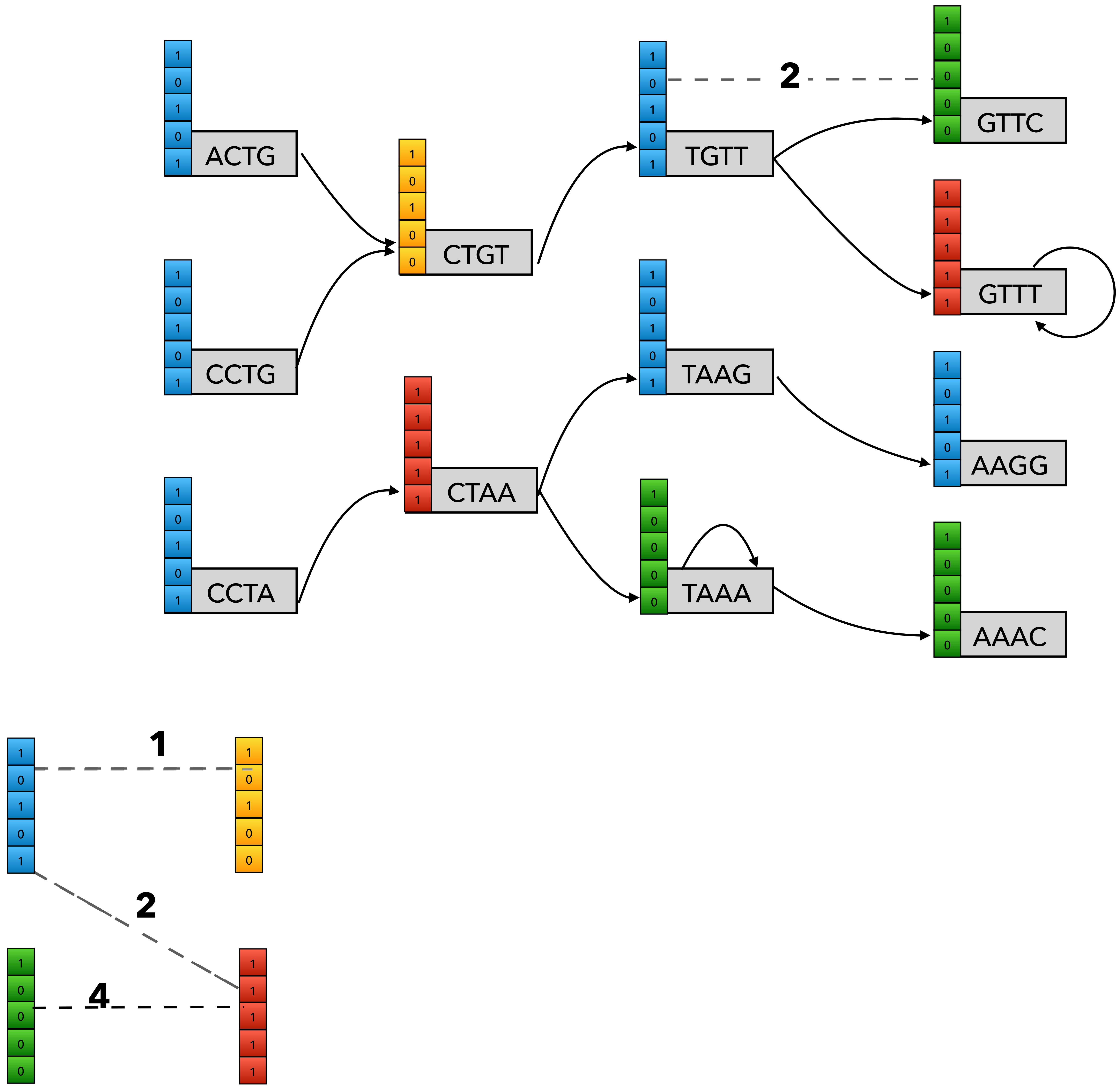

ACTG
CCTG
CCTA
CTGT
CTAA
TGTT
TAAG
TAAA
GTTC
GTTT
AAGG
AAAC


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

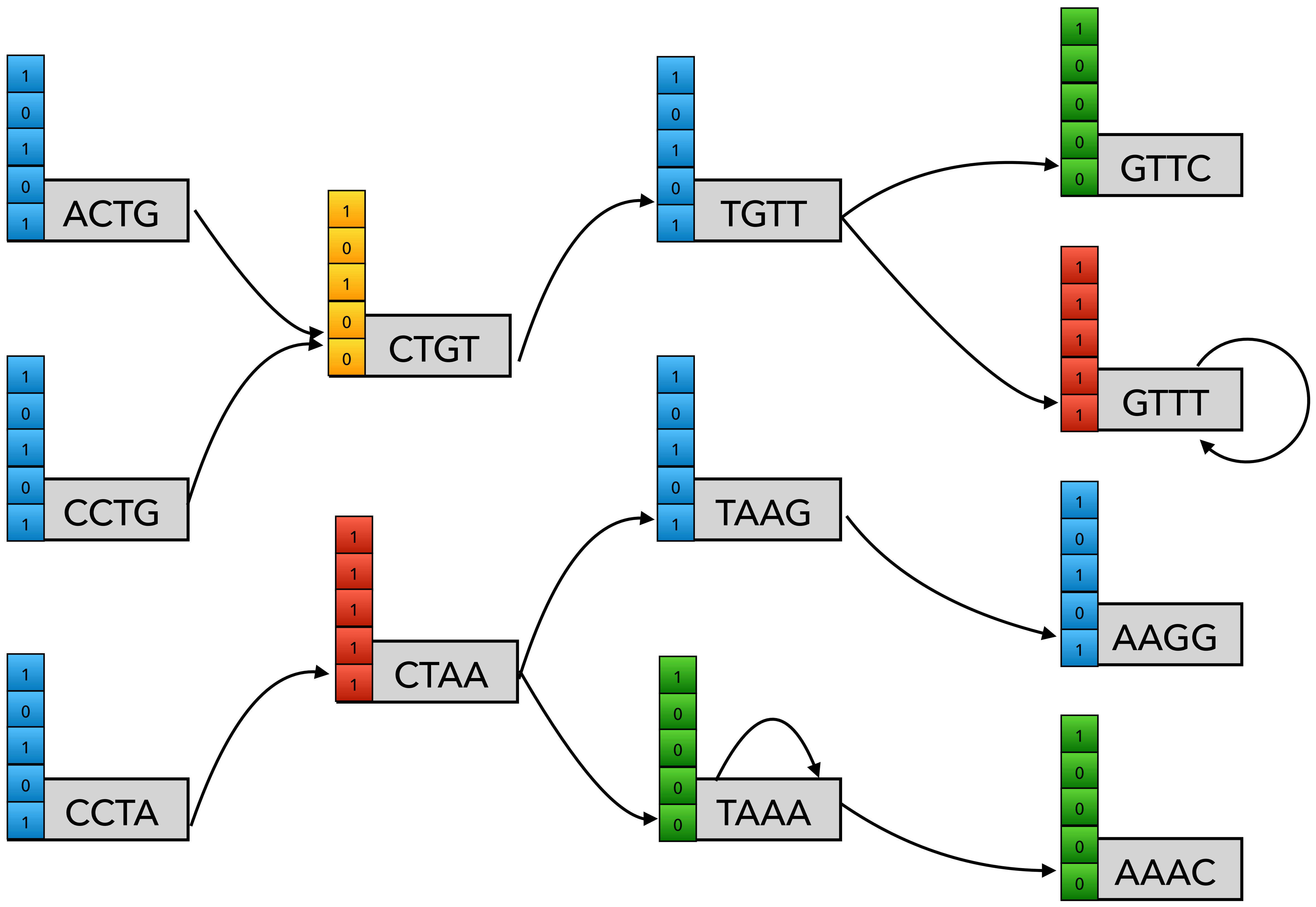

ACTG
CCTG
CCTA
CTGT
CTAA
TGTT
TAAG
TAAA
GTTC
GTTT
AAGG
AAAC
4
1
2

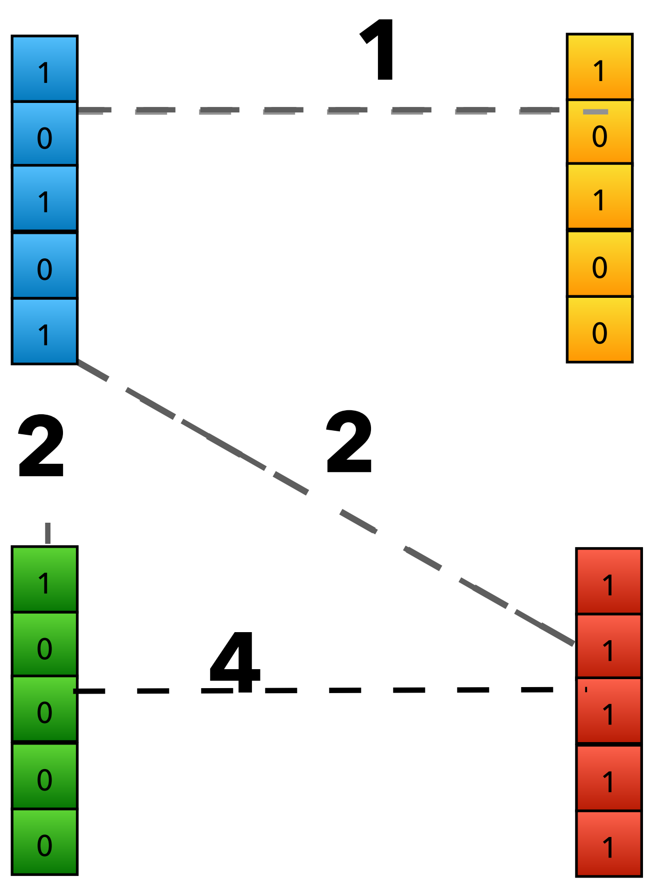

ACTG
CCTG
CCTA
CTGT
CTAA
TGTT
TAAG
TAAA
GTTC
GTTT
AAGG
AAAC

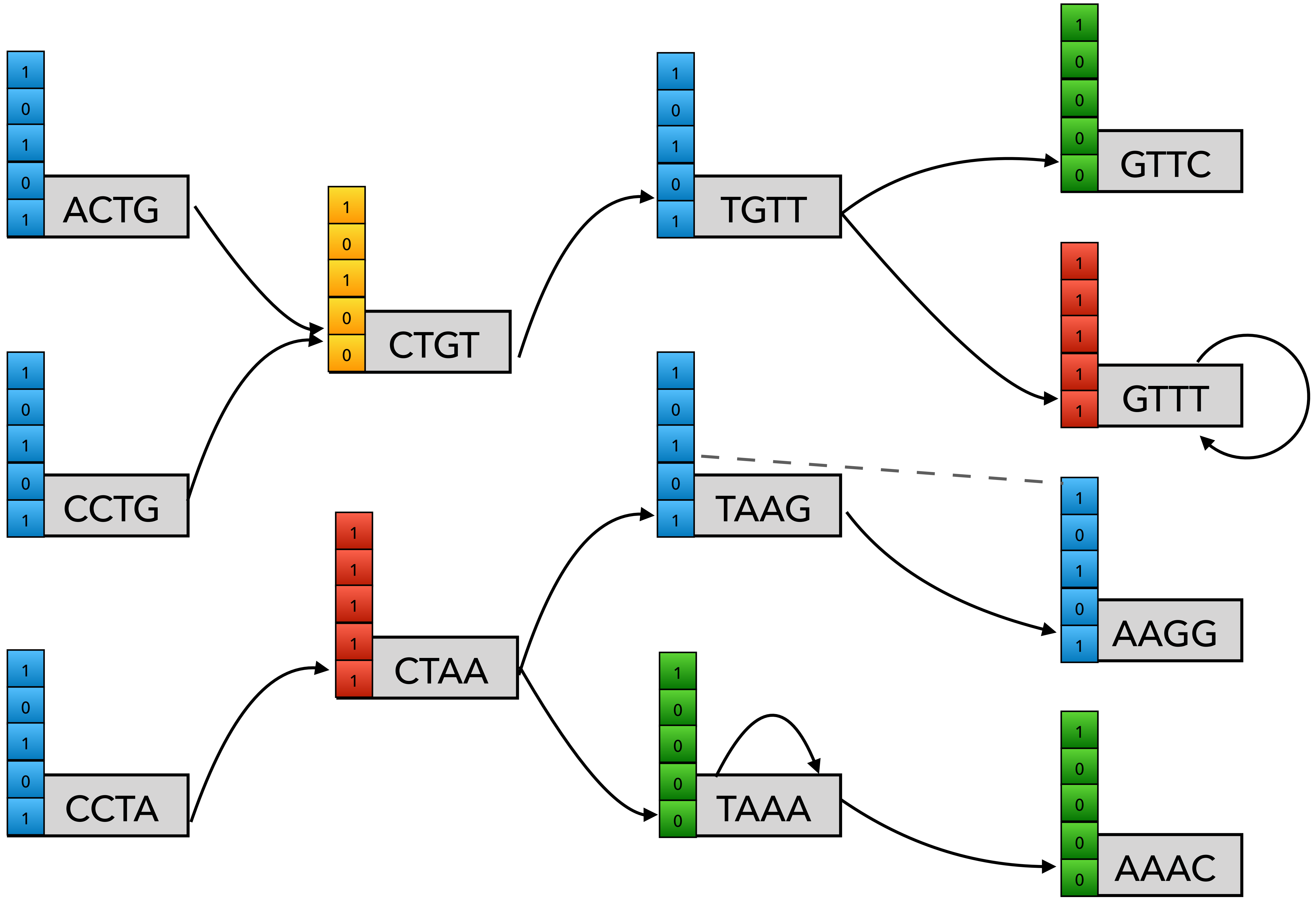

1
2
4

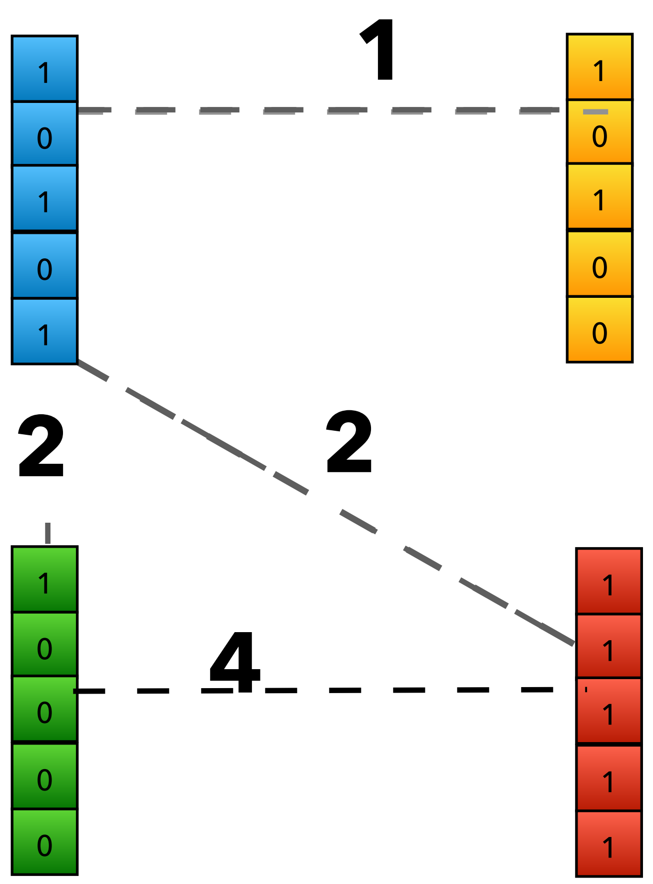


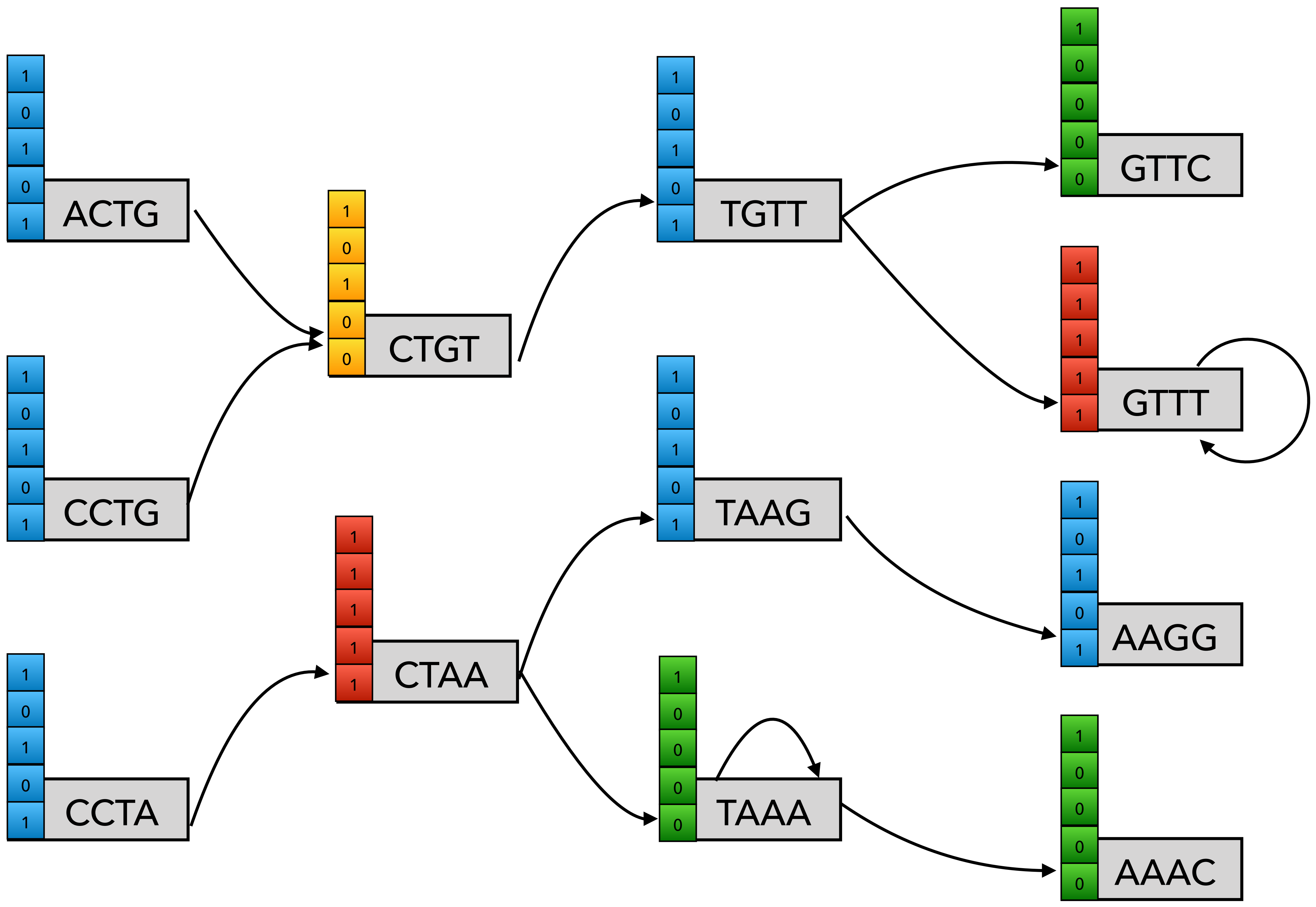
ACTG
CCTG
CCTA
CTGT
CTAA
TGTT
TAAG
TAAA
GTTC
GTTT
AAGG
AAAC
2
1
2
4

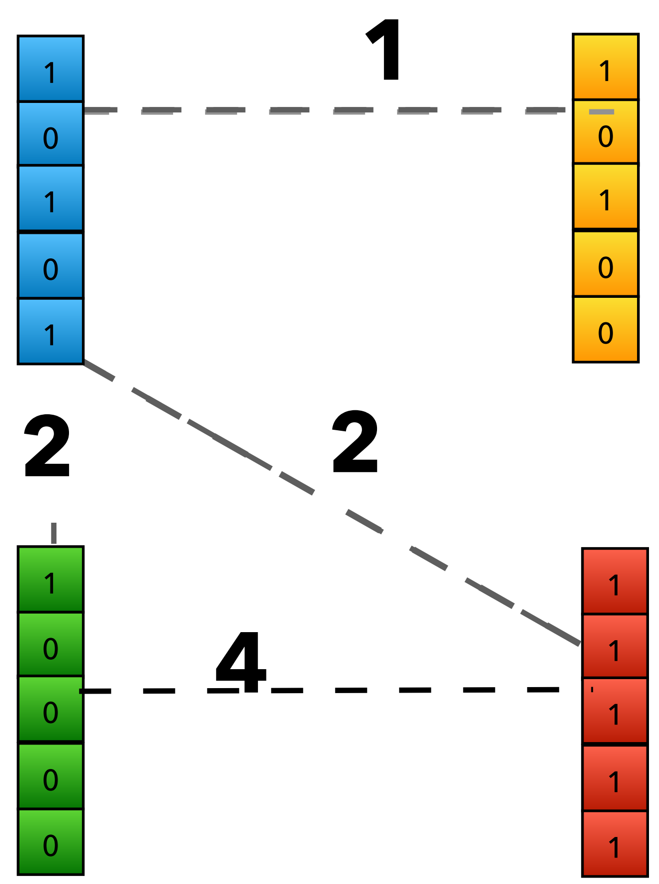

ACTG
CCTG
CCTA
CTGT
CTAA
TGTT
TAAG
TAAA
GTTC
GTTT
AAGG
AAAC

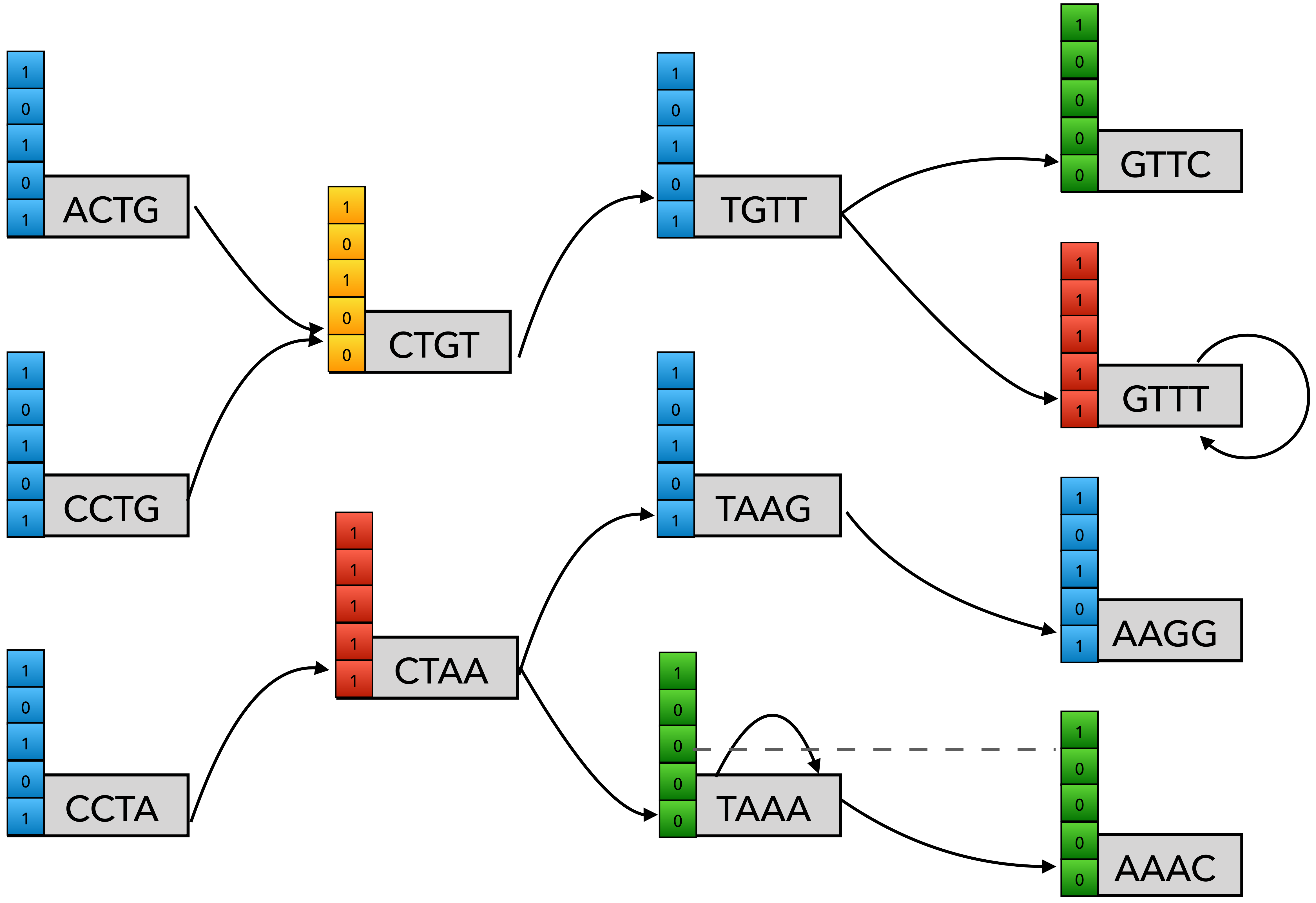

1
2
2
4

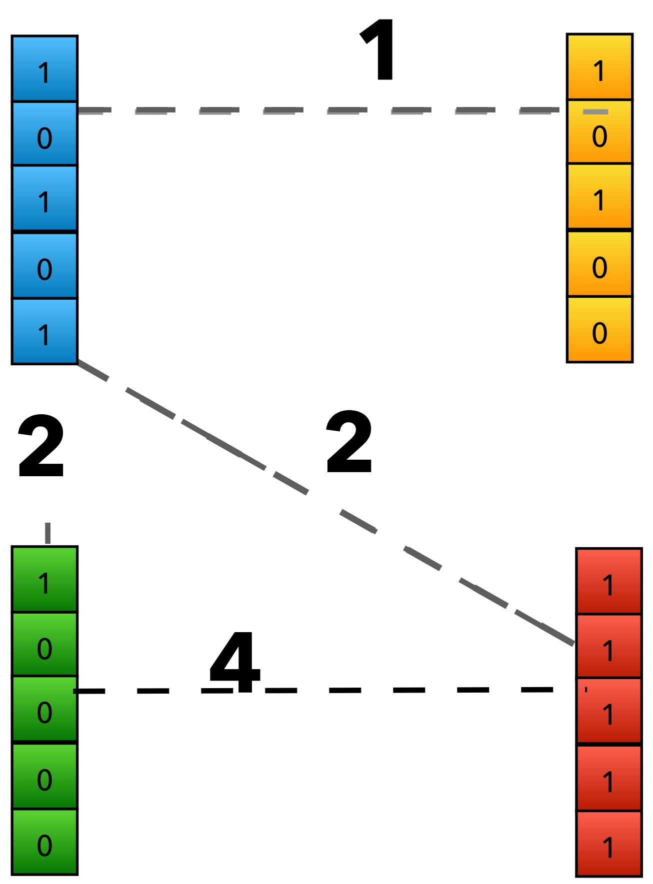

ACTG
CCTG
CCTA
CTGT
CTAA
TGTT
TAAG
TAAA
GTTC
GTTT
AAGG
AAAC

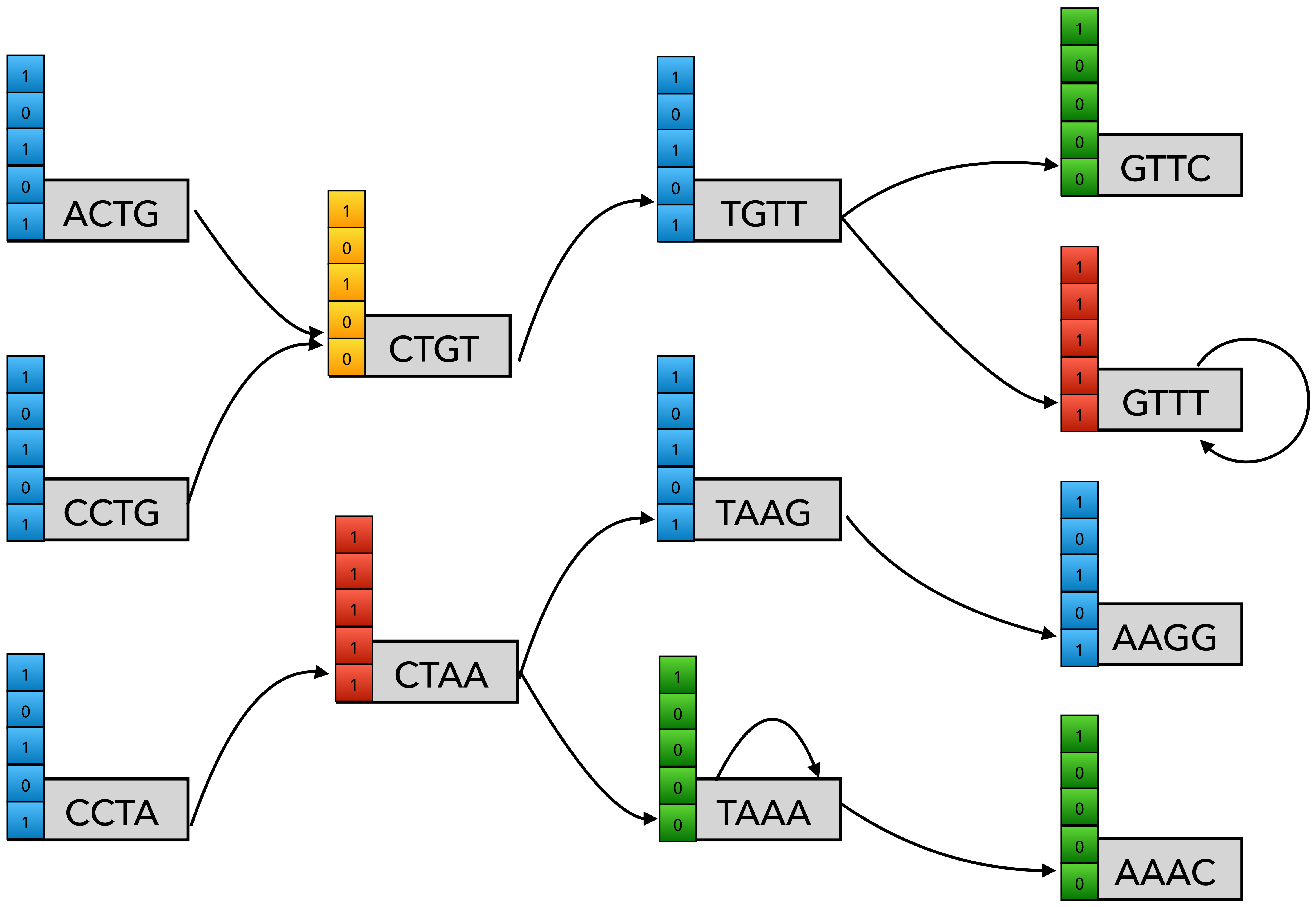

1
2
2
4


ACTG
CCTG
CCTA
CTGT
CTAA
TGTT
TAAG
TAAA
GTTC
GTTT
AAGG
AAAC

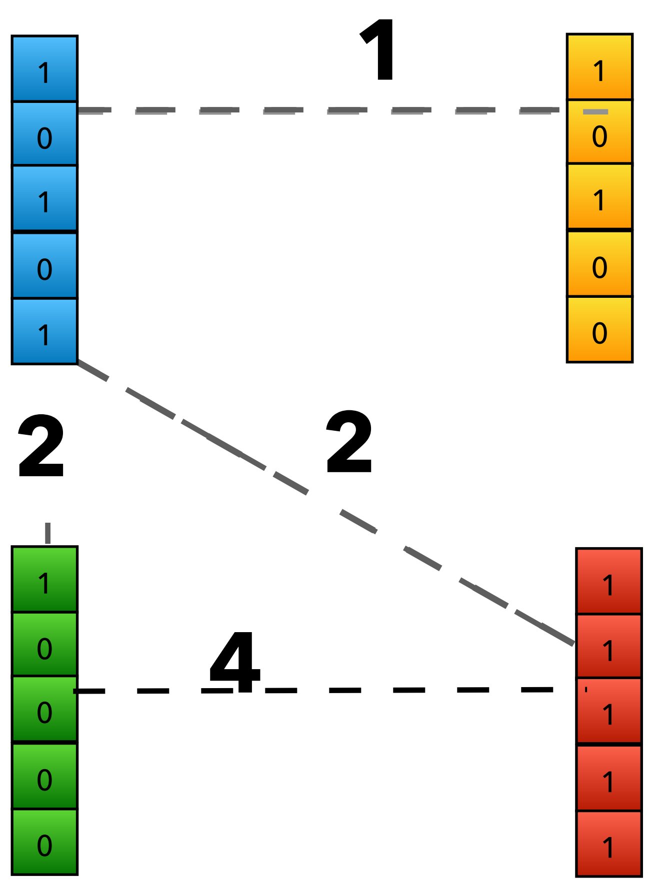

1
2
2
4


ACTG
CCTG
CCTA
CTGT
CTAA
TGTT
TAAG
TAAA
GTTC
GTTT
AAGG
AAAC

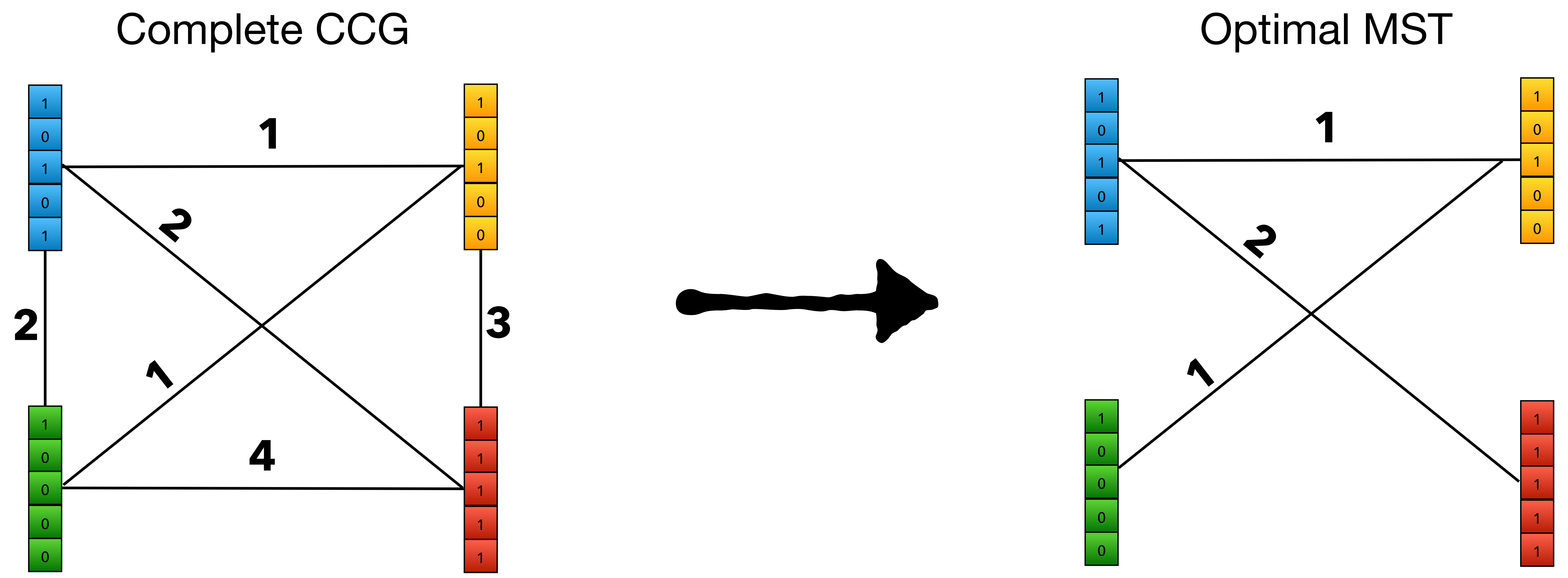

1
2
2
4

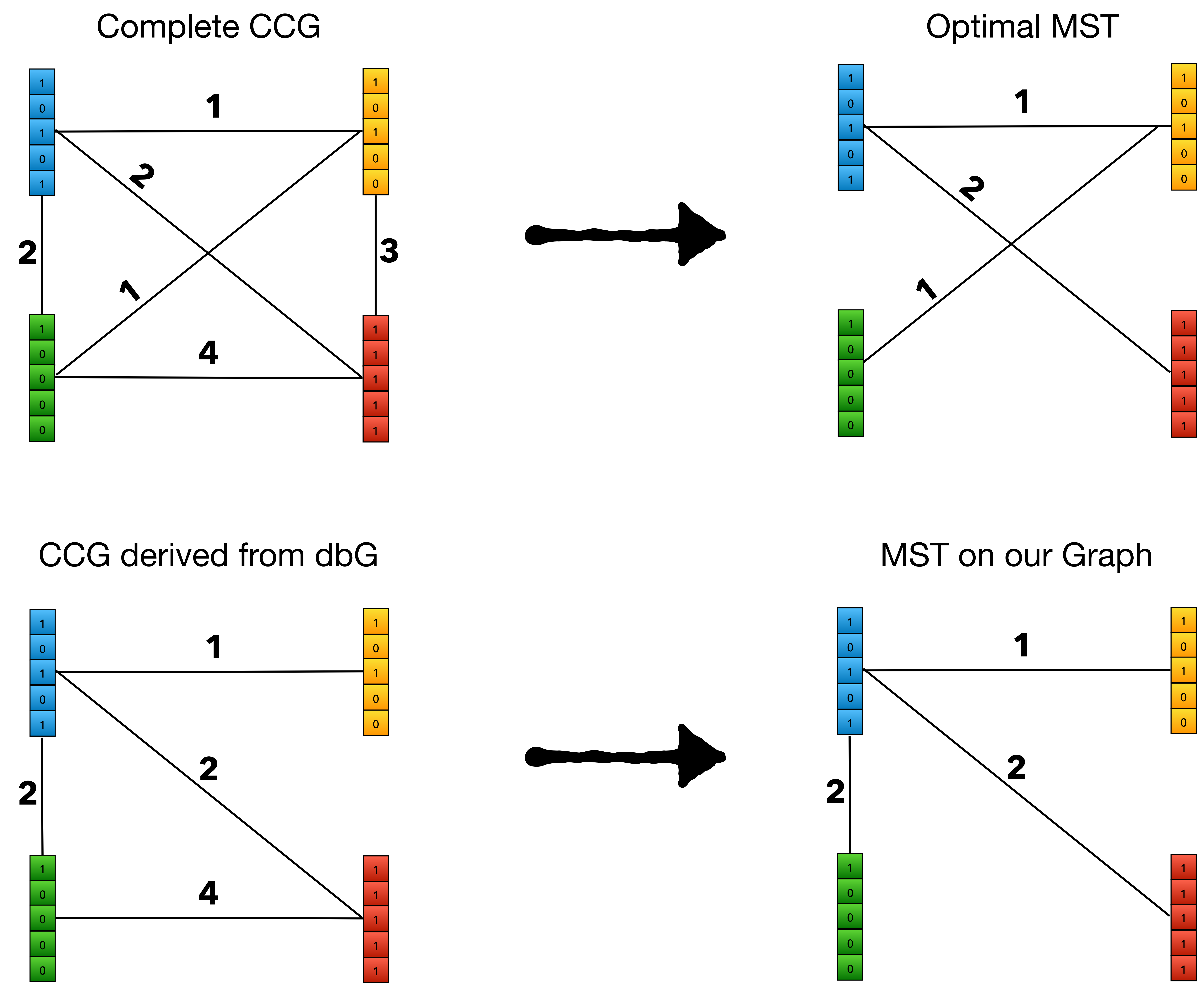


ACTG
CCTG
CCTA
CTGT
CTAA
TGTT
TAAG
TAAA
GTTC
GTTT
AAGG
AAAC

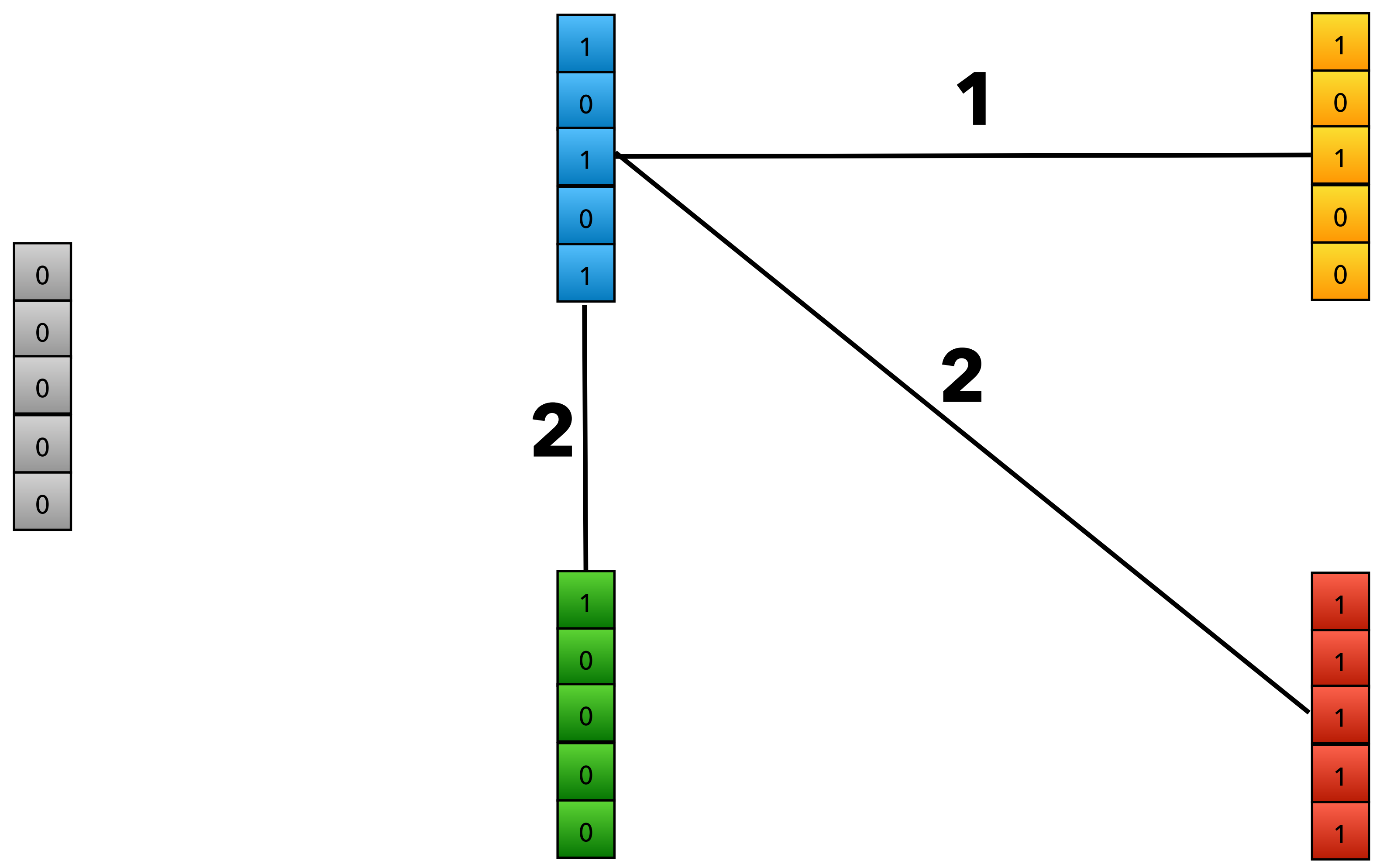


1
2
2
4


Complete CCG
Optimal MST
1
2
2
3
1
4
1
2
1

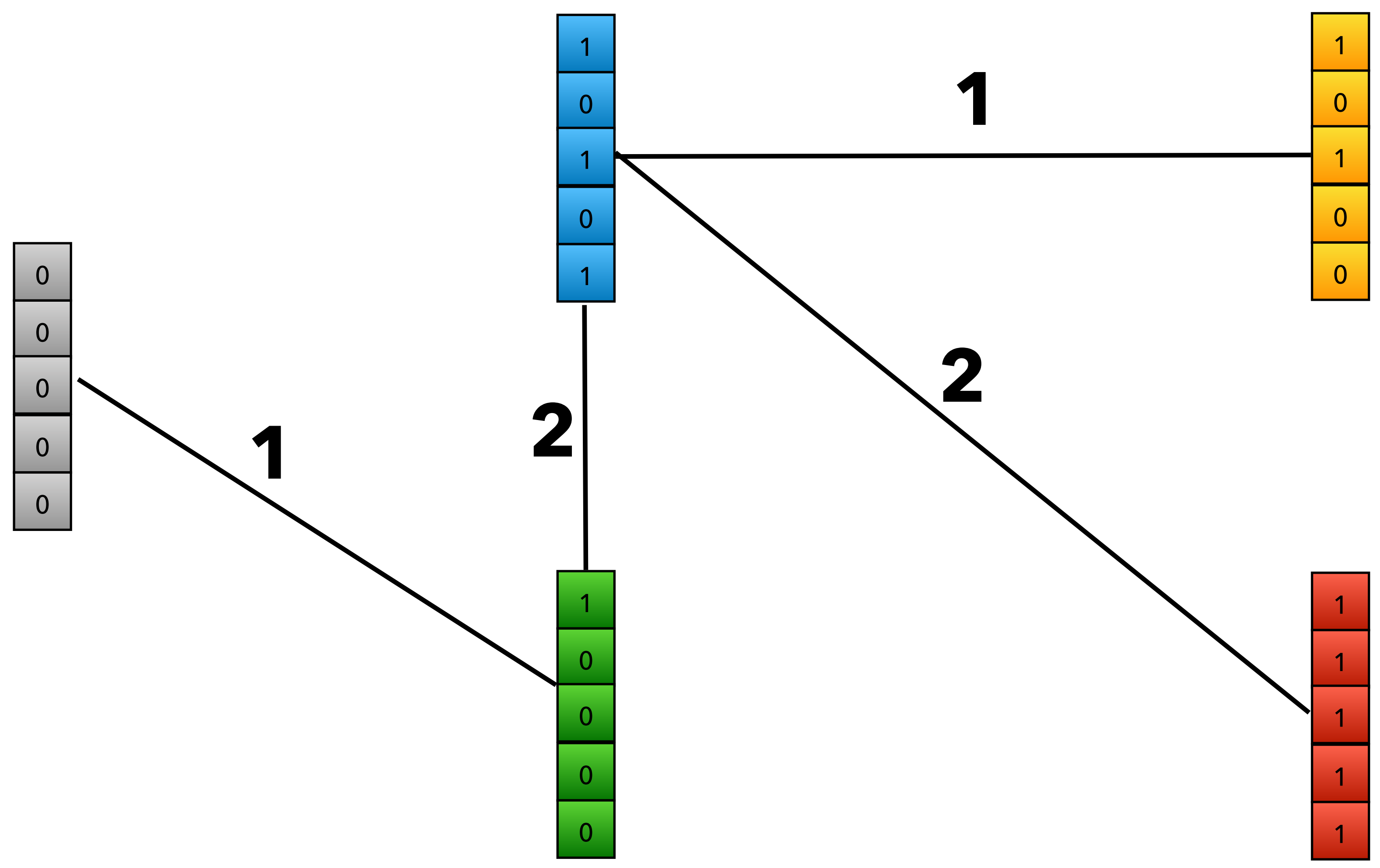

Complete CCG
1
2
2
3
1
4
Optimal MST
1
2
1
CCG derived from dbG
1
2
2
4
MST on our Graph
1
2
2

The MST efficiently encodes related color classes



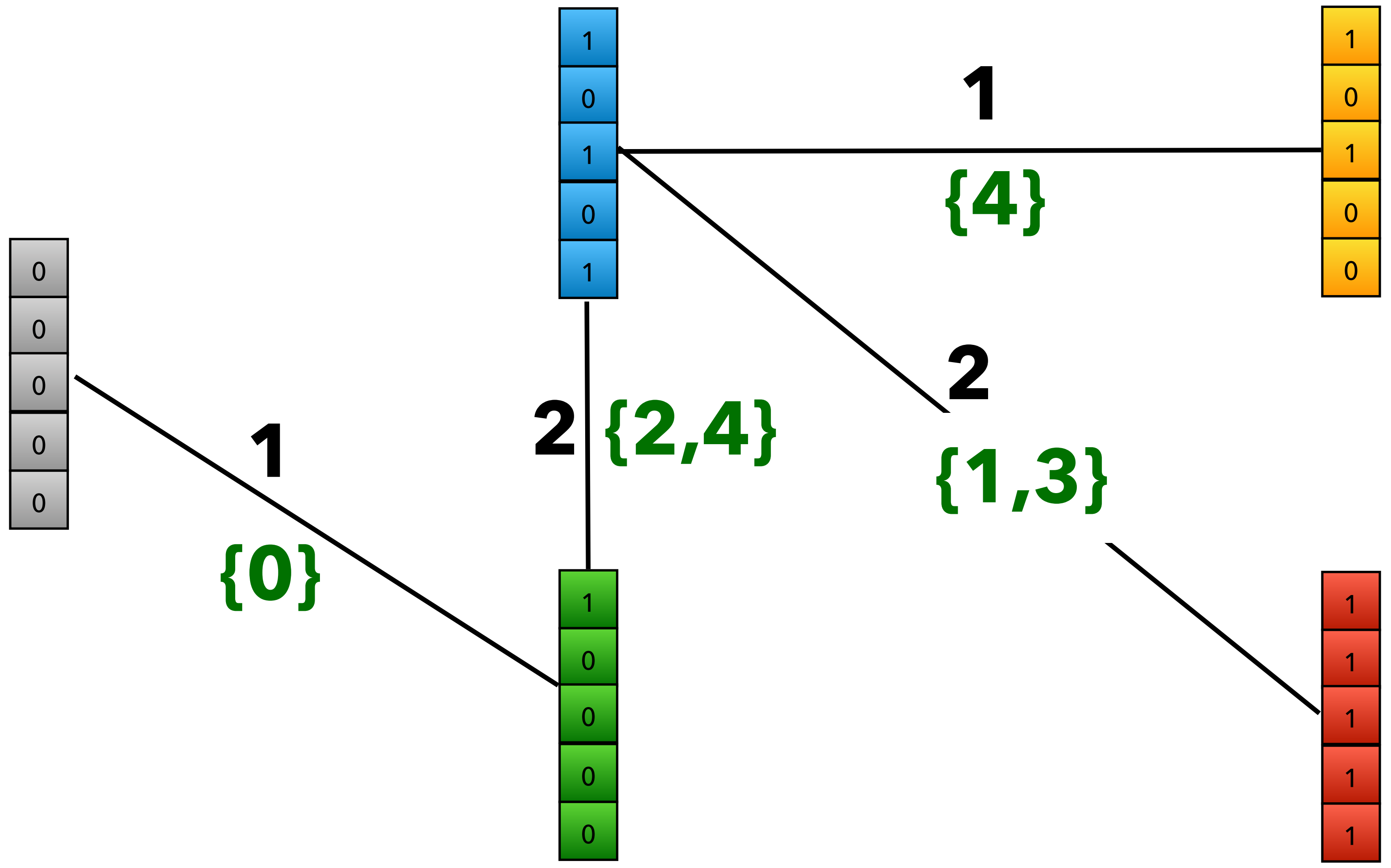
The MST efficiently encodes related color classes

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



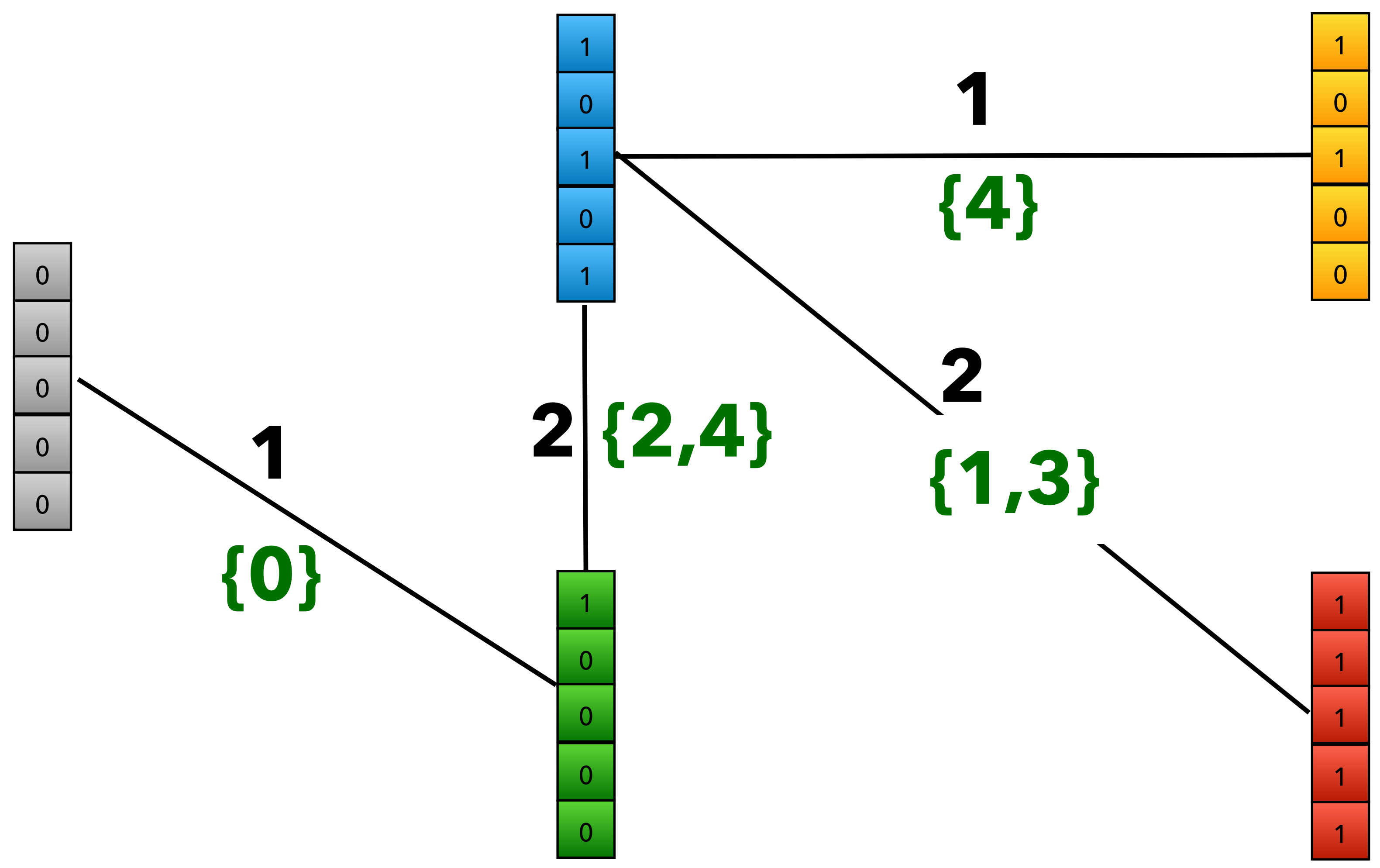
The MST efficiently encodes related color classes

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



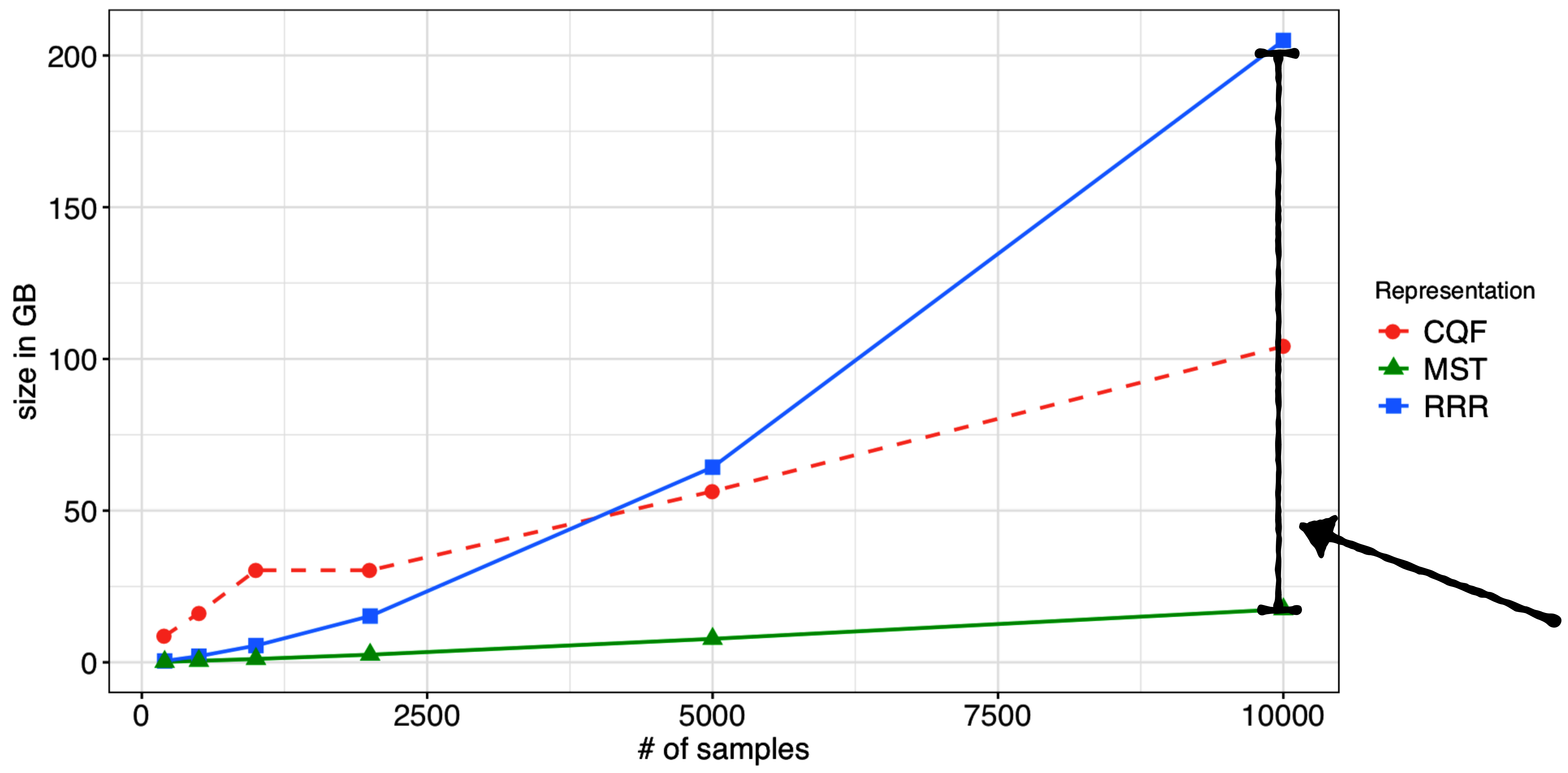
The MST efficiently encodes related color classes

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



MST-based compression of bit vectors

Dataset	# samples	RRR matrix	MST Total space	MST Parent vector	MST Delta vector	MST Boundary bit-vector	$\frac{\text{size}(MST)}{\text{size(RRR)}}$
H. sapiens 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

dataset from SBT / SSBT / Mantis paper

Improvement over RRR improves with # of samples

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?

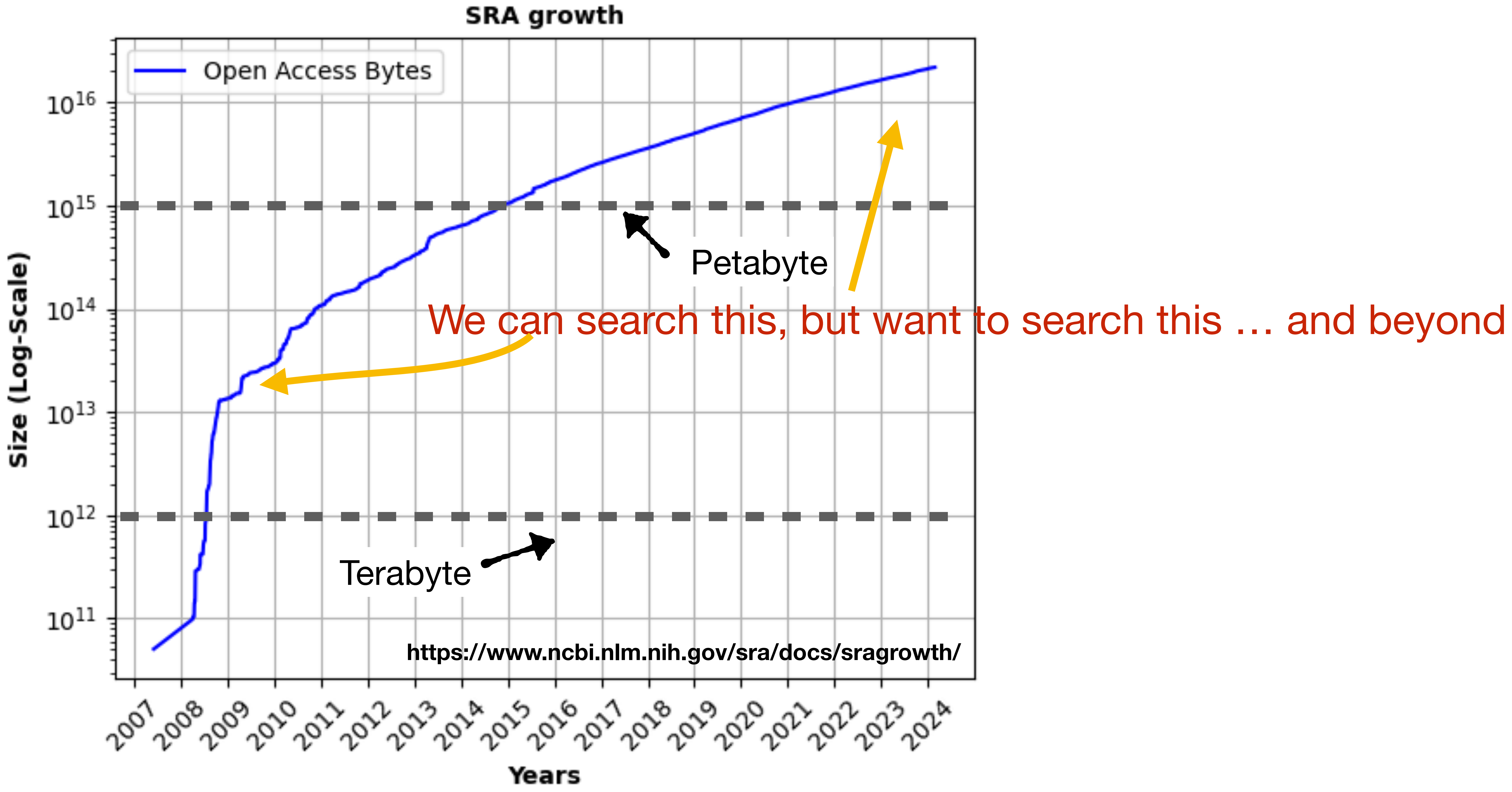
How does MST approach affect query time?

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



Some remaining challenges

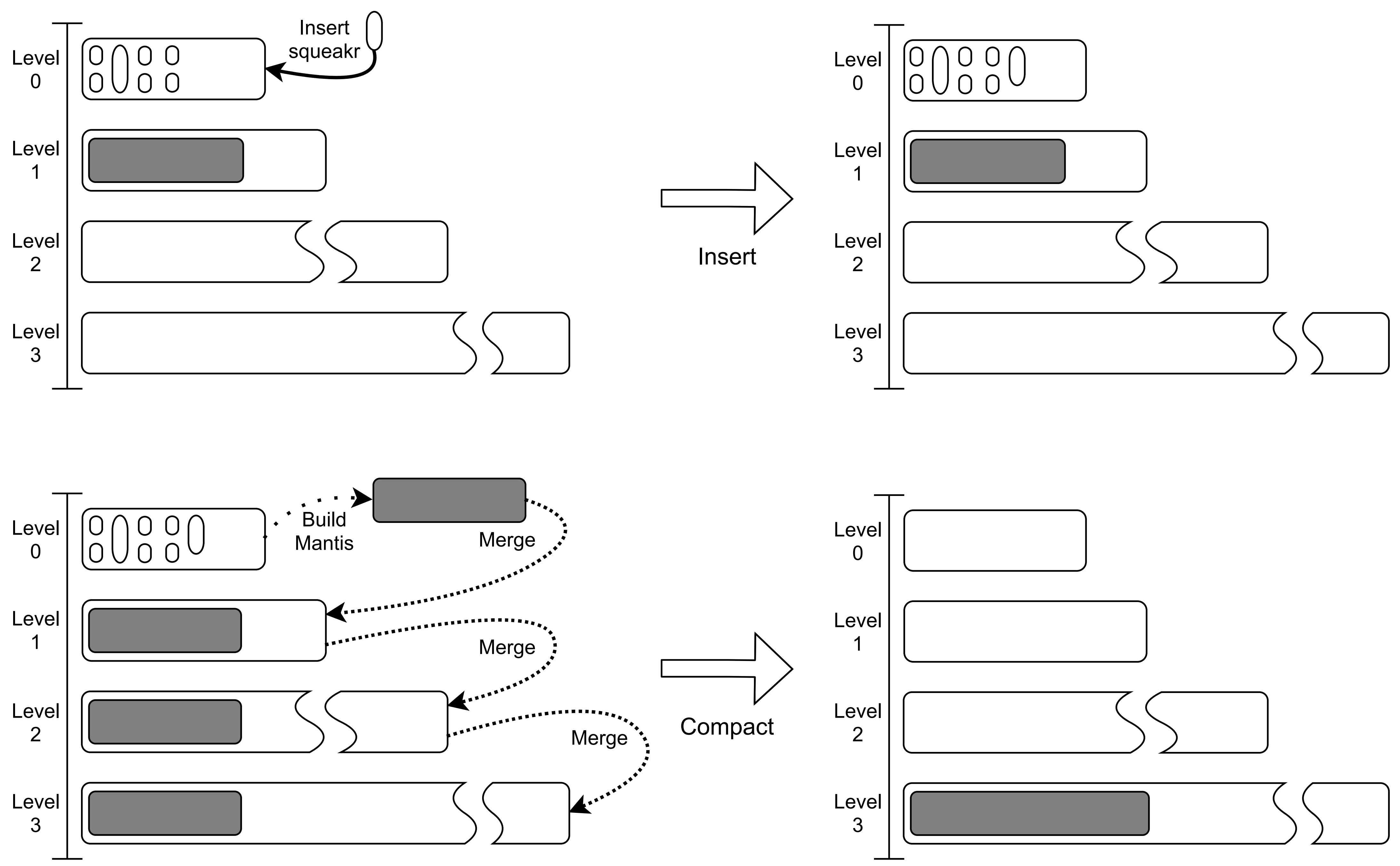
- We can scale to even larger datasets by compressing color class representation
- We demonstrate indexing on the order of 10^3 experiments, we really want to index on the order of 10^5 - 10^6
- We need to scale out of RAM and also support adding new experiments

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 samples and can scale out of RAM to disk.

Mantis-LSM design



- Level 0 resides in RAM
- $L_1 \ldots L_n$ remain on disk
- Levels grow in size exponentially

- ***Minimizers*** to partition the *k*-mer index on disk
- Helps to minimize RAM usage during merging and queries.

Mantis-LSM performance

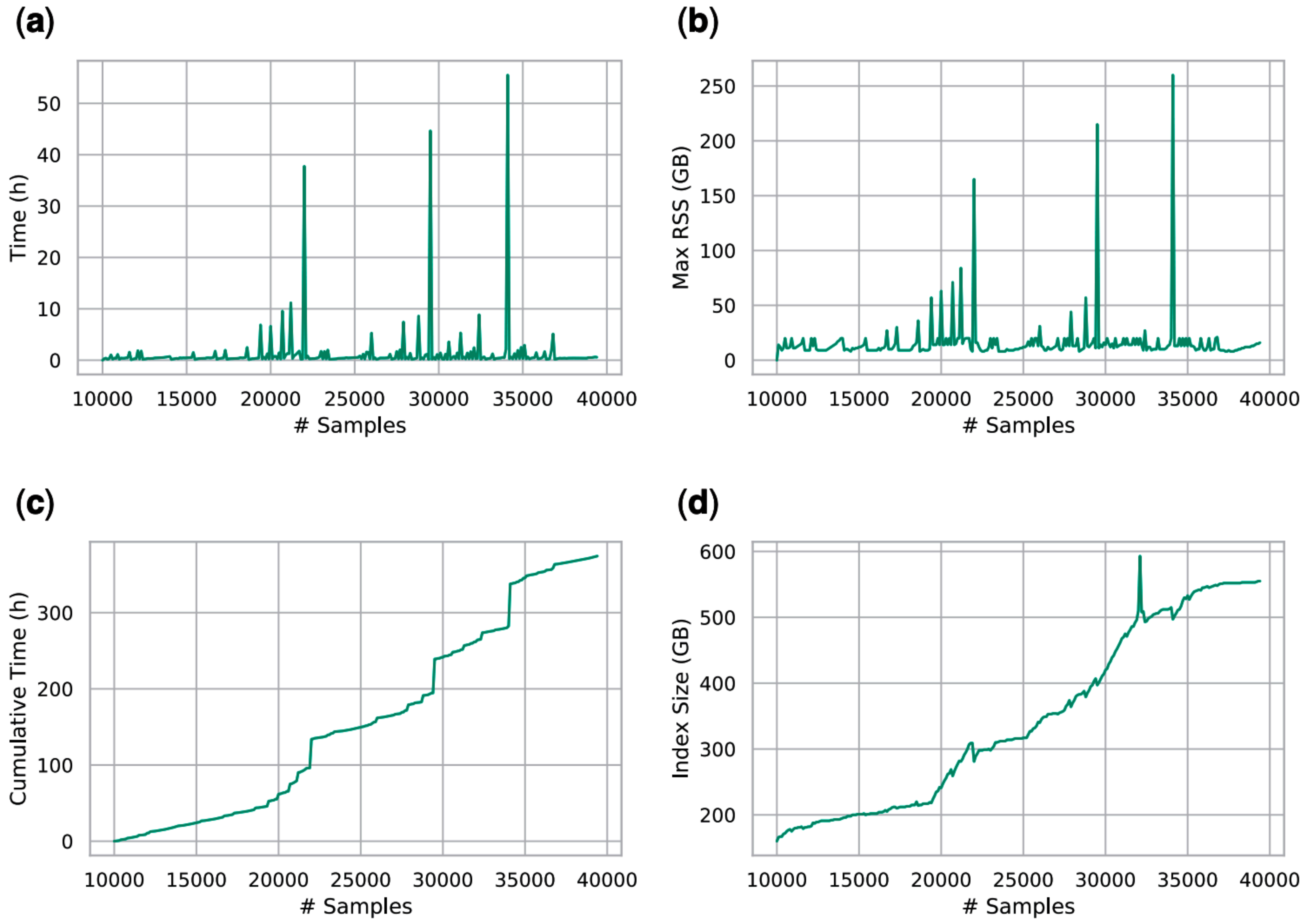


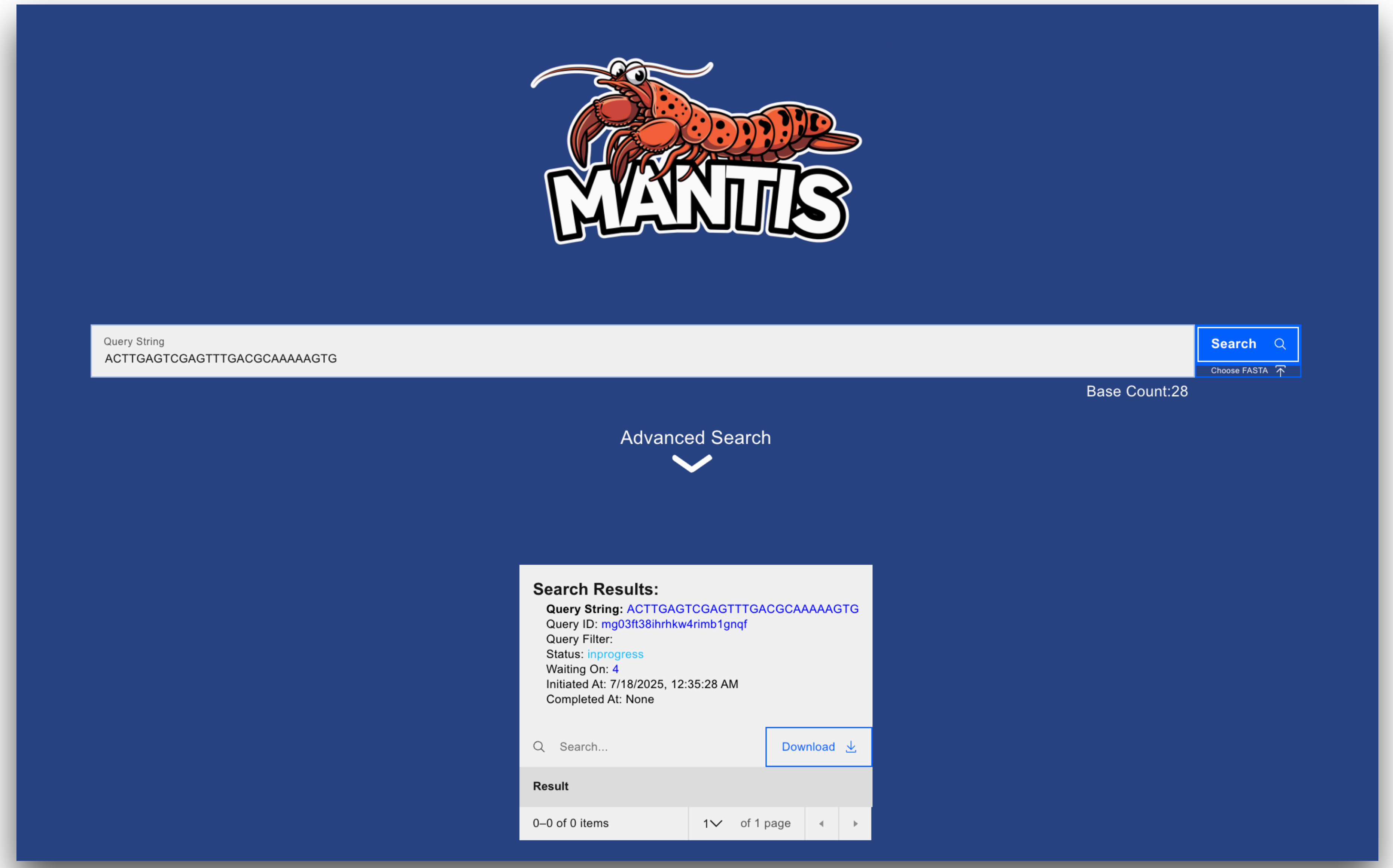
Fig. 4. Performance of the Dynamic Mantis update process. The spikes in time (Fig. a) and memory (Fig. b) happen when the cascading merge happens with deeper and thus larger indexes. Cumulative Time (Fig. c) shows the total time required to addd all the samples up to thae current one. and index size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

LSM-Mantis can index up to 40K samples on a single machine

Mantis Distributed

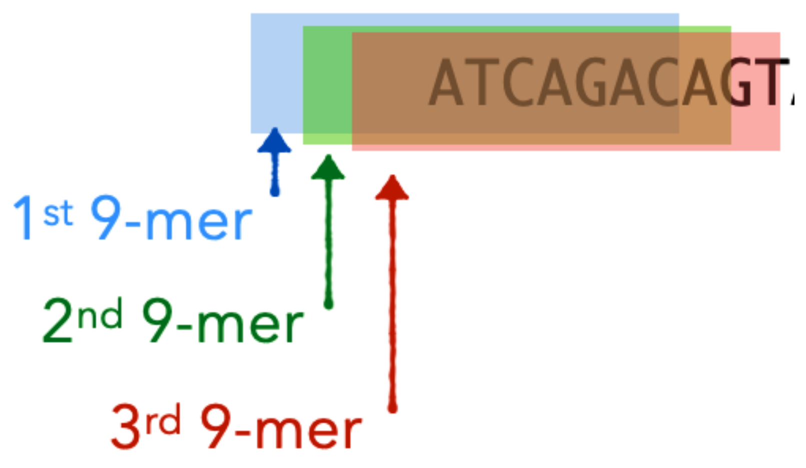
- We now have a distributed Mantis (publicly-hosted search service over SRA)
- Each individual machine runs Mantis-LSM (written in RUST)
- Employ *minimizers* to efficiently distribute samples across machines
- Indexes 100K samples from SRA (~200TB compressed data)
- 47 machines each with 32GB RAM
- Support real-time queries (< 1 sec)

https://query.bio/

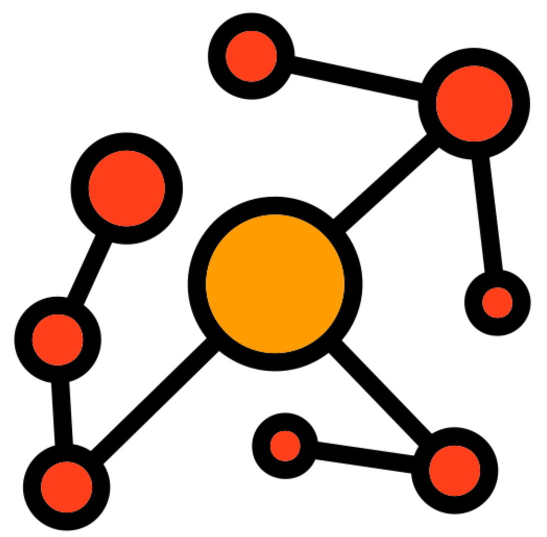


Major insights from sample discovery problem

Domain specific knowledge enables to develop efficient solutions



K-mer embeddings appropriately capture sequence similarity



K-mer de Bruijn graph is a good proxy for the neighborhood graph

However, scale of the SRA is still the biggest challenge!

K-mer set representation ignores relative order

x = CCCCACCAACACAAAACCC ⟶ AAAA, AAAC, AACA, AACC, ACAA, ACAC, ACCA, ACCC, CAAA, CAAC, CACA, CACC, CCAA, CCAC, CCCA, CCCC

y = AAAACACAACCCCACCAAA ⟶ AAAA, AAAC, AACA, AACC, ACAA, ACAC, ACCA, ACCC, CAAA, CAAC, CACA, CACC, CCAA, CCAC, CCCA, CCCC

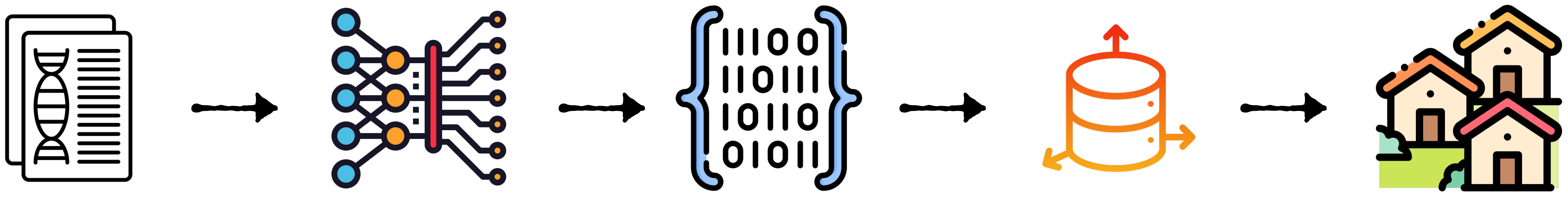
x and *y* have the same *k*-mer representation but low sequence similarity

K-mer representation might not be the right proxy*. Can introduce a large number of false positives.

Need to evaluate the end-to-end accuracy of *k*-mer-based methods.

*Guillaume Marcais, Dan DeBlasio, Prashant Pandey, Carl Kingsford "Locality Sensitive Hashing for the Edit Distance" ISMB 2019

Employing ML/vector databases for genomic discovery



Samples from sequence read archive

ML model optimizing for edit distance

Embeddings produced by the ML model

Vector database indexing the embeddings

Neighborhood search using VecDB

A ton of publicly-available data from SRA

Major takeaways

- Sequence search over SRA promises to unlock deep biological insights
- Scalability is the biggest challenge; there's still a long way to go to index/search all of SRA
- Exploiting domain knowledge enables simpler, more scalable solutions
- Time to employ ML and vector databases to solve sequence search

A special thanks to my collaborators

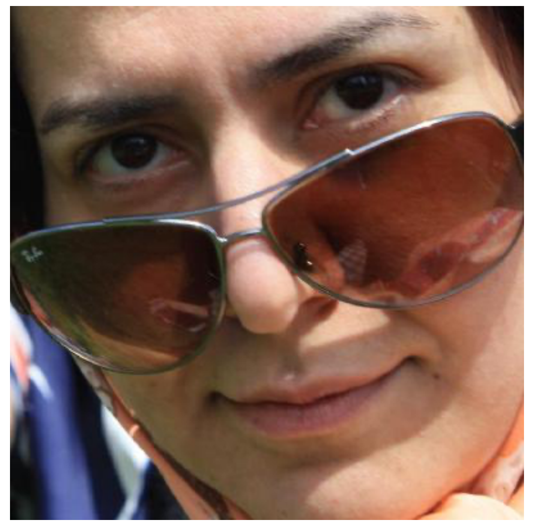
Funding:

National Science Foundation

U.S. DEPARTMENT OF ENERGY

Jamshed Khan (UMD)

Fatemeh Almodaresi (OICR)



Rob Patro (UMD)

Mike Ferdman (Stony Brook)

Rob Johnson (VMware Research)

Michael Bender (Stony Brook)

https://prashantpandey.github.io/