Large Scale Sequence Search using Exact Indices (k-mer sets as de Bruijn Graphs)

UNIVERSITY OF MARYLAND

NOTE: This lecture is being recorded

Different kind of graph

"tomorrow and tomorrow and tomorrow"

Different kind of graph

"tomorrow and tomorrow and tomorrow"

tomorrow

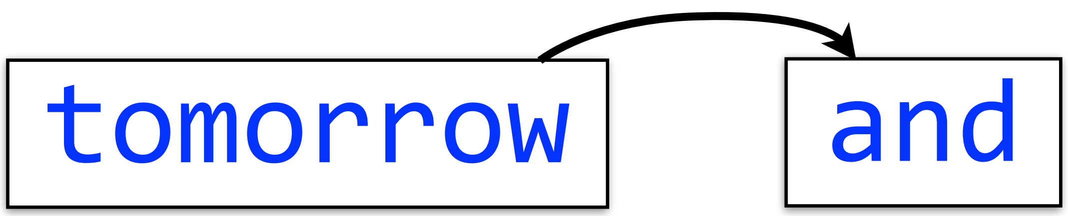
and

Lecture slides adapted from the dBG lecture slides of Ben Langmead.
All slides in this lecture marked with "*" courtesy of Ben Langmead.

*

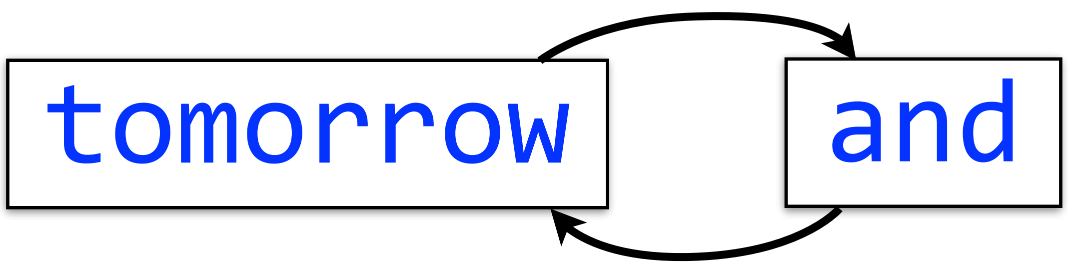
Different kind of graph

"tomorrow and tomorrow and tomorrow"



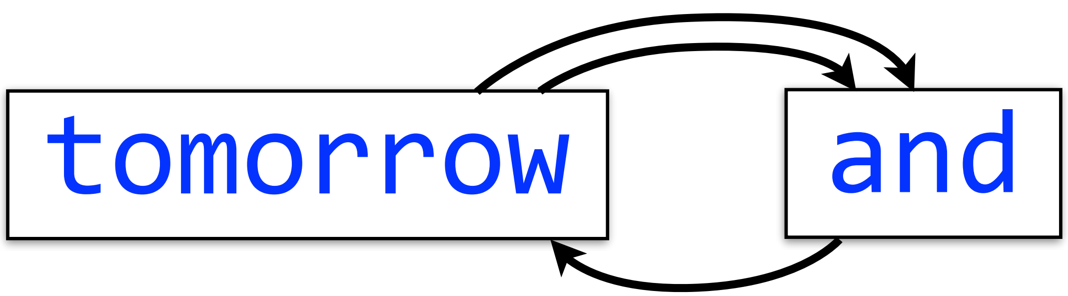
Different kind of graph

“tomorrow and tomorrow and tomorrow and tomorrow”



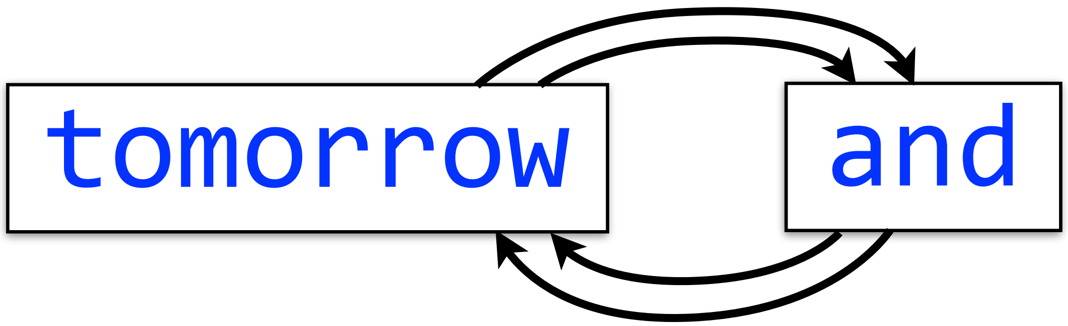
Different kind of graph

“tomorrow and tomorrow and tomorrow”



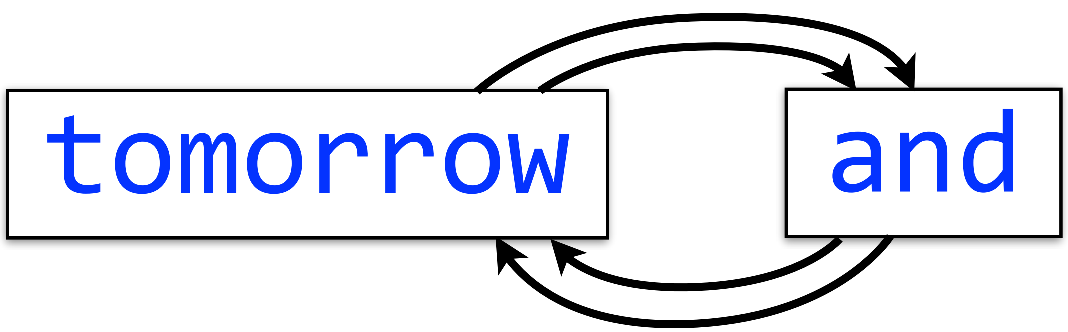
Different kind of graph

“tomorrow and tomorrow and tomorrow”



Different kind of graph

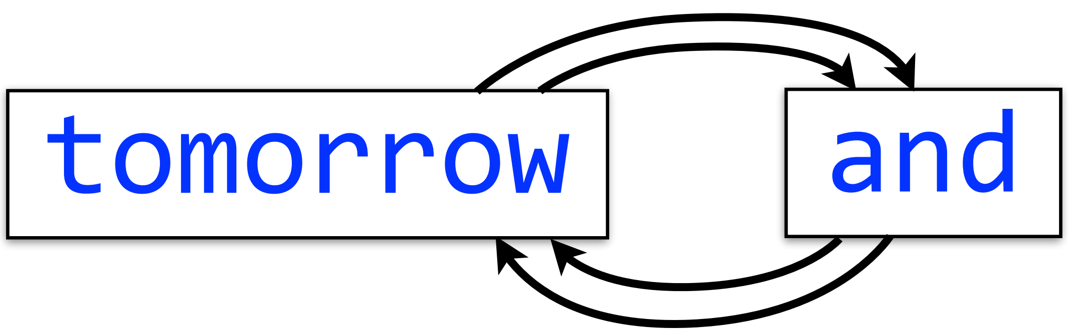
“tomorrow and tomorrow and tomorrow”



An edge represents an ordered pair of adjacent words in the input

Different kind of graph

"tomorrow and tomorrow and tomorrow"



An edge represents an ordered pair of adjacent words in the input

Multigraph: there can be more than one edge from node A to node B

De Bruijn graph

genome: AAABBBBA

*

De Bruijn graph

genome: AAABBBBA

3-mers: AAA, AAB, ABB, BBB, BBB, BBA

*

De Bruijn graph

genome: AAABBBBA

3-mers: AAA, AAB, ABB, BBB, BBB, BBA

L/R 2-mers: AA, AA

*

De Bruijn graph

genome: AAABBBBA

3-mers: AAA, AAB, ABB, BBB, BBB, BBA

L/R 2-mers: AA, AA

AA

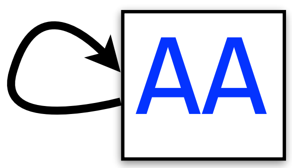
*

De Bruijn graph

genome: AAABBBBA

3-mers: AAA, AAB, ABB, BBB, BBB, BBA

L/R 2-mers: AA, AA

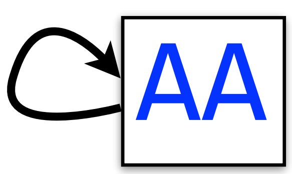


De Bruijn graph

genome: AAABBBBA

3-mers: AAA, AAB, ABB, BBB, BBB, BBA

L/R 2-mers: AA, AA AA, AB

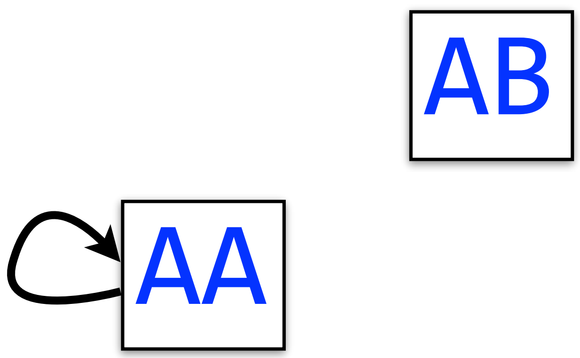


De Bruijn graph

genome: AAABBBBA

3-mers: AAA, AAB, ABB, BBB, BBB, BBA

L/R 2-mers: AA, AA AA, AB



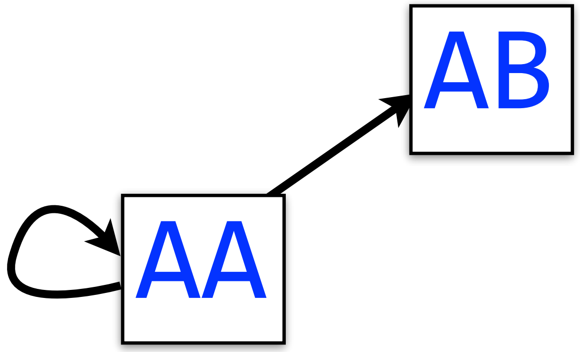
*

De Bruijn graph

genome: AAABBBBA

3-mers: AAA, AAB, ABB, BBB, BBB, BBA

L/R 2-mers: AA, AA AA, AB



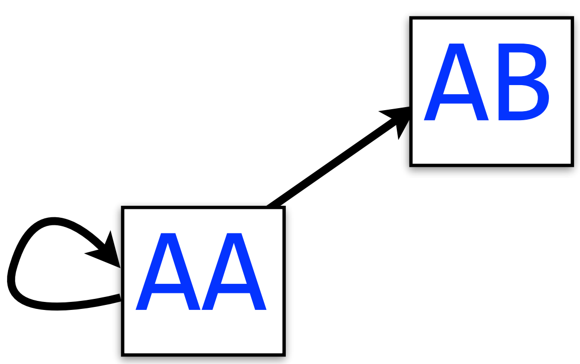
*

De Bruijn graph

genome: AAABBBBA

3-mers: AAA, AAB, ABB, BBB, BBB, BBA

L/R 2-mers: AA, AA   AA, AB   AB, BB



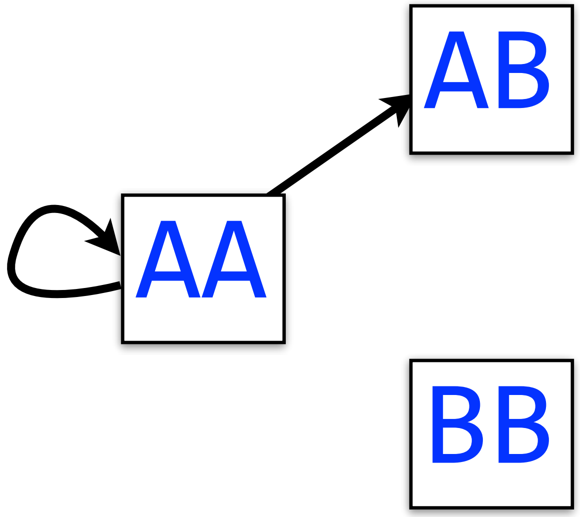
*

De Bruijn graph

genome: AAABBBBA

3-mers: AAA, AAB, ABB, BBB, BBB, BBA

L/R 2-mers: AA, AA AA, AB AB, BB



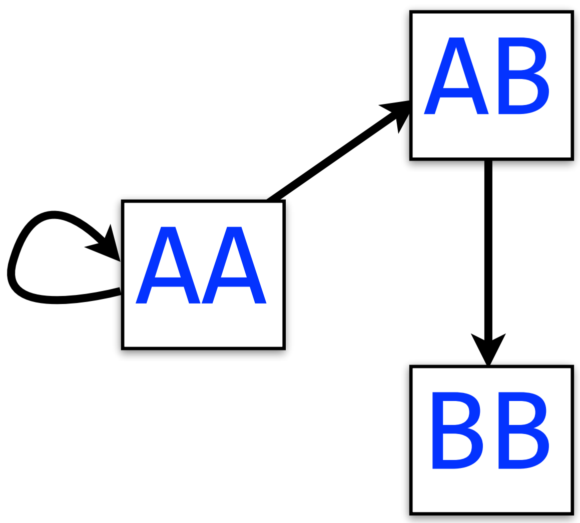
*

De Bruijn graph

genome: AAABBBBA

3-mers: AAA, AAB, ABB, BBB, BBB, BBA

L/R 2-mers: AA, AA   AA, AB   AB, BB



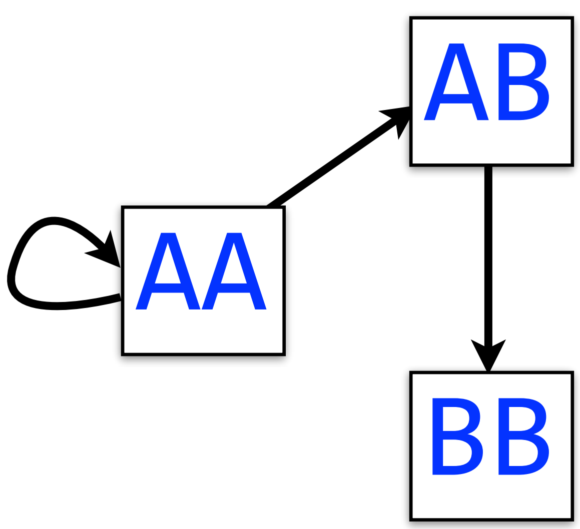
*

De Bruijn graph

genome: AAABBBBA

3-mers: AAA, AAB, ABB, BBB, BBB, BBA

L/R 2-mers: AA, AA AA, AB AB, BB BB, BB



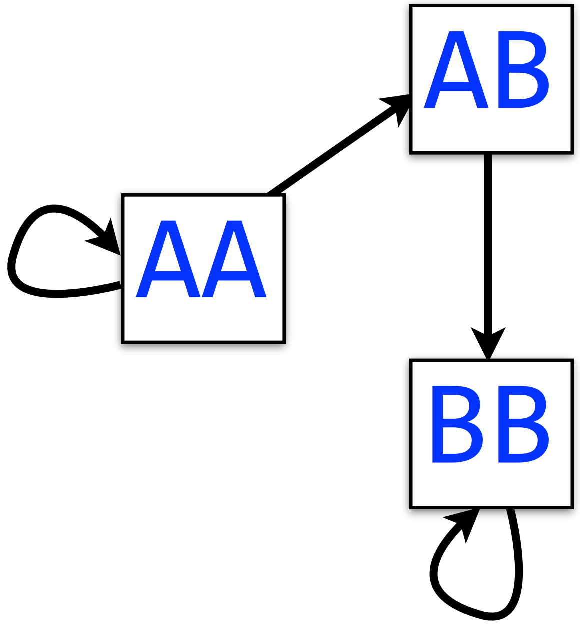
*

De Bruijn graph

genome: AAABBBBA

3-mers: AAA, AAB, ABB, BBB, BBB, BBA

L/R 2-mers: AA, AA AA, AB AB, BB BB, BB



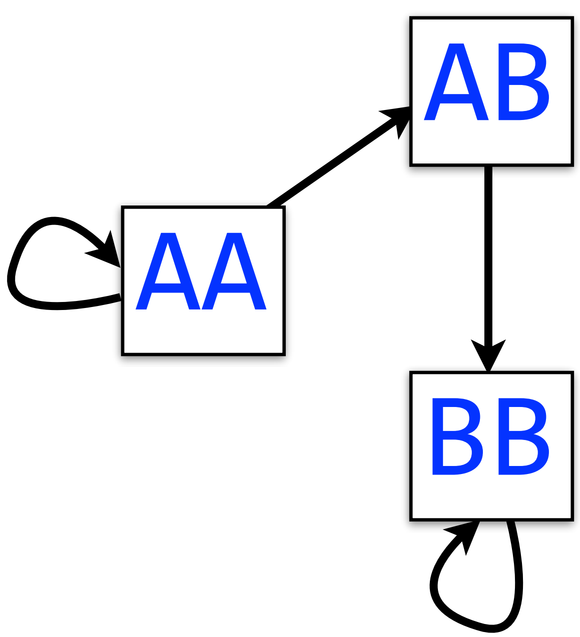
*

De Bruijn graph

genome: AAABBBBA

3-mers: AAA, AAB, ABB, BBB, BBB, BBA

L/R 2-mers: AA, AA   AA, AB   AB, BB   BB, BB   BB, BB



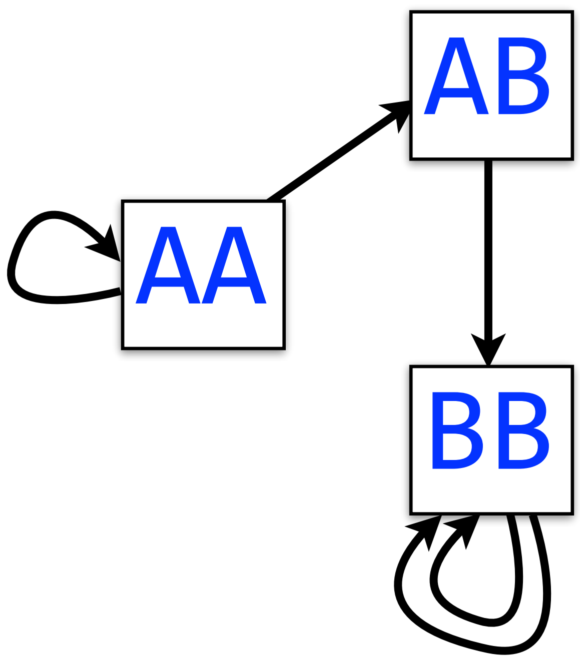
*

De Bruijn graph

genome: AAABBBBA

3-mers: AAA, AAB, ABB, BBB, BBB, BBA

L/R 2-mers: AA, AA   AA, AB   AB, BB   BB, BB   BB, BB



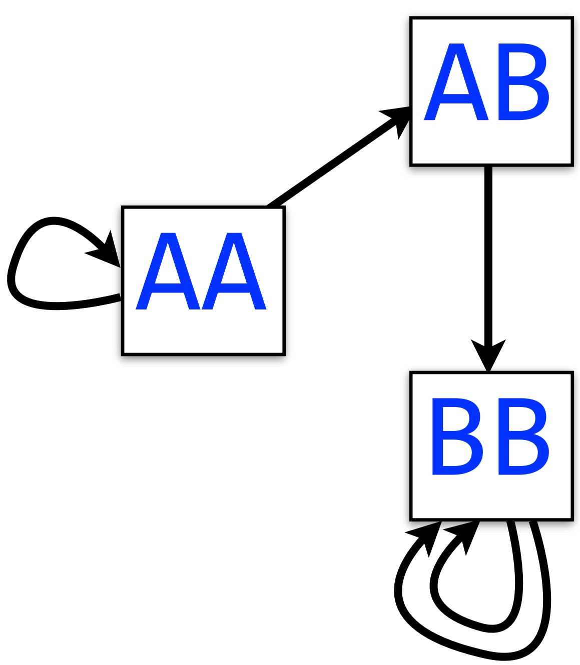
*

De Bruijn graph

genome: AAABBBBA

3-mers: AAA, AAB, ABB, BBB, BBB, BBA

L/R 2-mers: AA, AA AA, AB AB, BB BB, BB BB, BB BB, BA



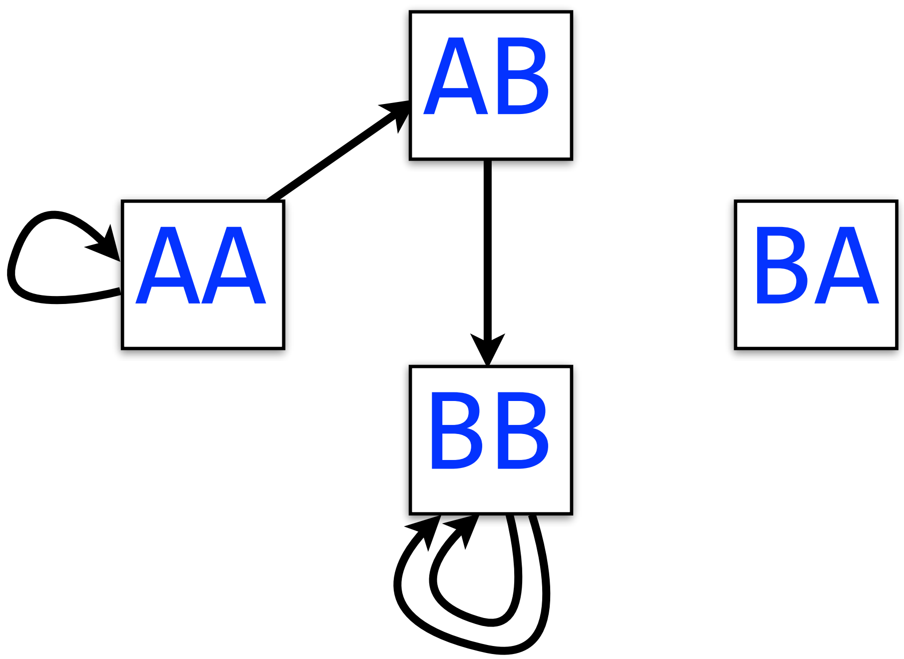
*

De Bruijn graph

genome: AAABBBBA

3-mers: AAA, AAB, ABB, BBB, BBB, BBA

L/R 2-mers: AA, AA   AA, AB   AB, BB   BB, BB   BB, BB   BB, BA



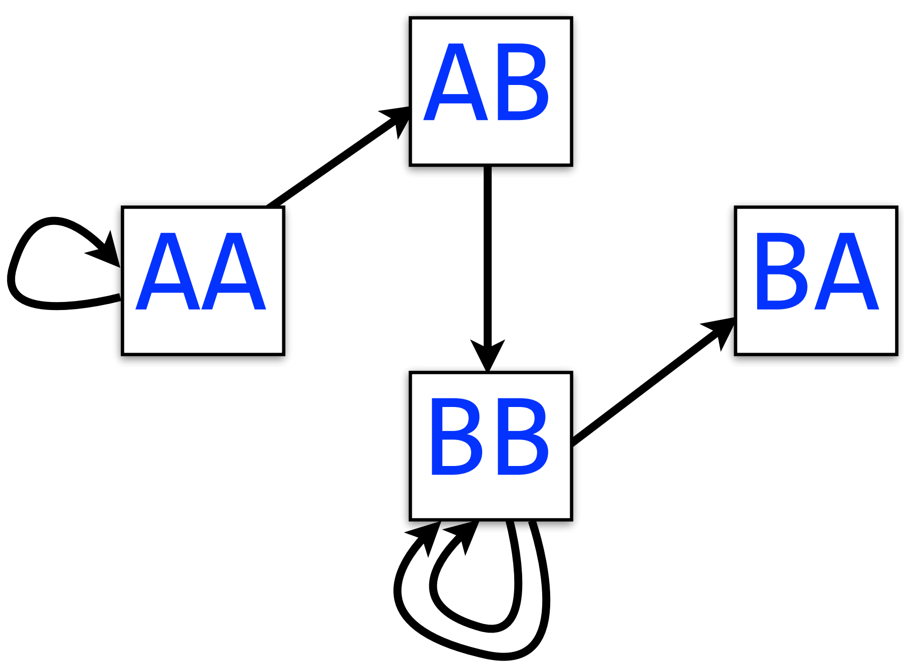
*

De Bruijn graph

genome: AAABBBBA

3-mers: AAA, AAB, ABB, BBB, BBB, BBA

L/R 2-mers: AA, AA   AA, AB   AB, BB   BB, BB   BB, BB   BB, BA

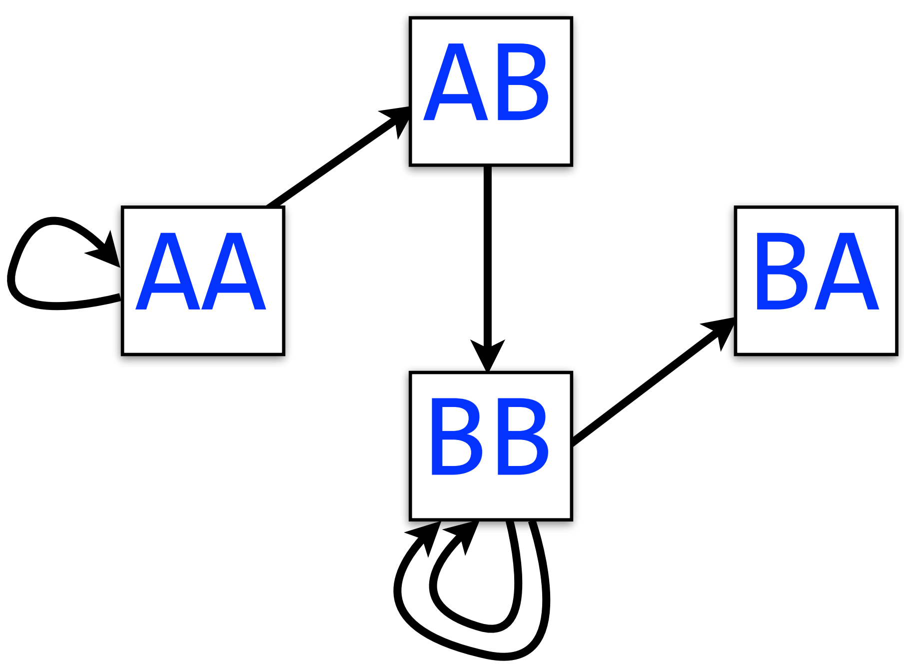


De Bruijn graph

genome: AAABBBBA

3-mers: AAA, AAB, ABB, BBB, BBB, BBA

L/R 2-mers: AA, AA   AA, AB   AB, BB   BB, BB   BB, BB   BB, BA



One edge per k-mer

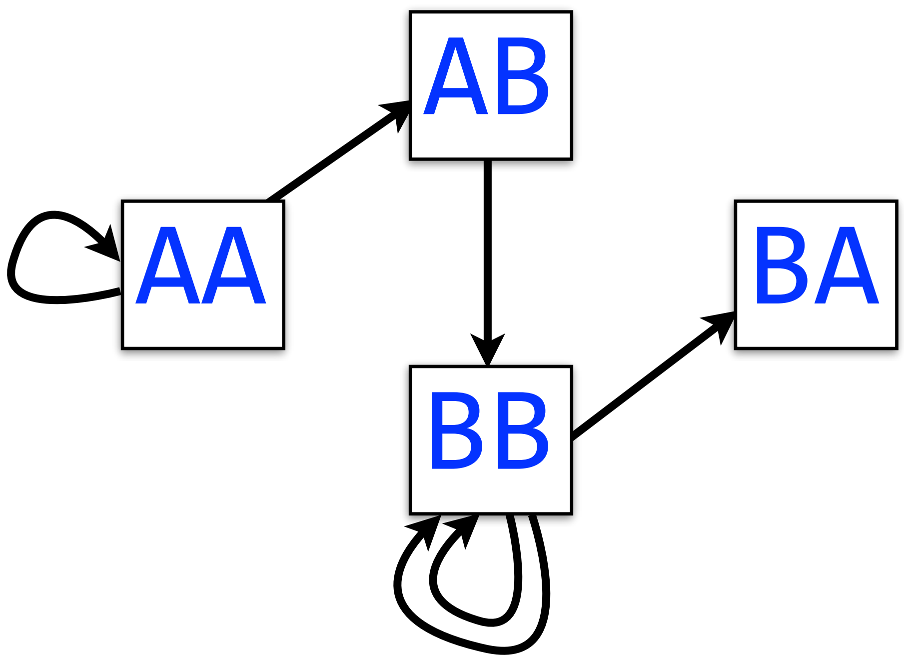
*

De Bruijn graph

genome: AAABBBBA

3-mers: AAA, AAB, ABB, BBB, BBB, BBA

L/R 2-mers: AA, AA AA, AB AB, BB BB, BB BB, BB BB, BA

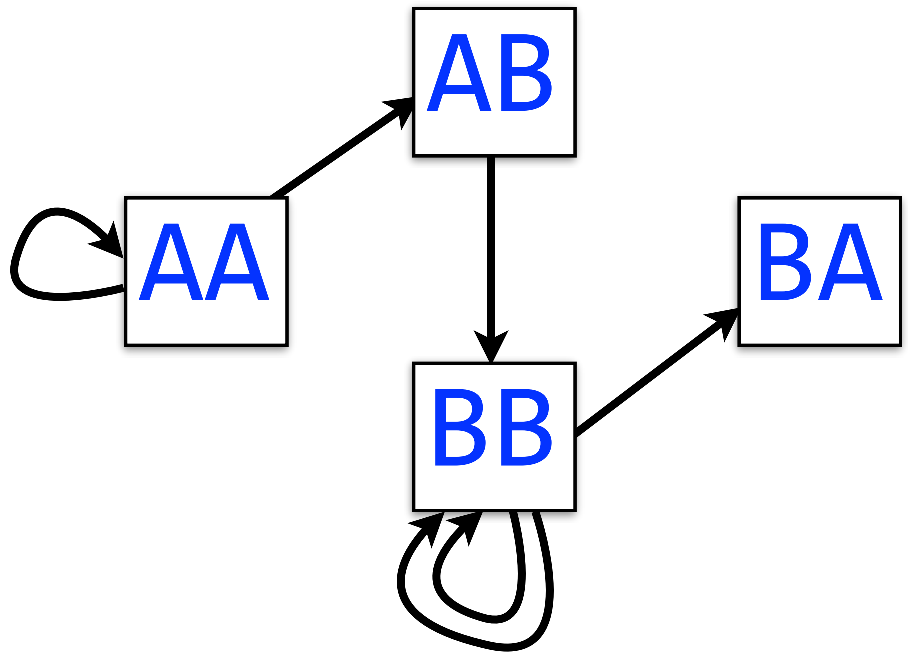


One edge per k-mer

One node per distinct k-1-mer

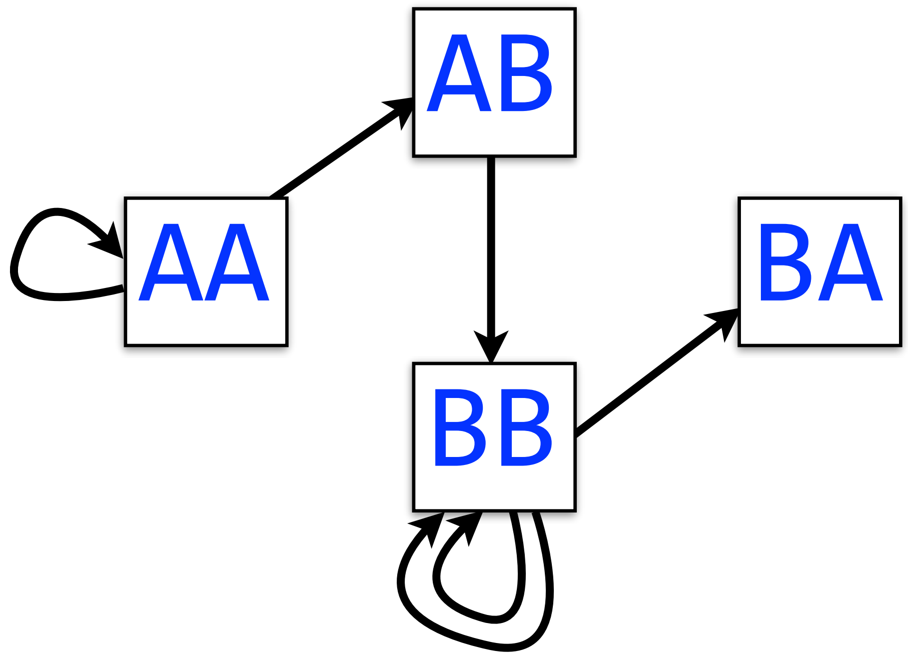
*

De Bruijn graph



*

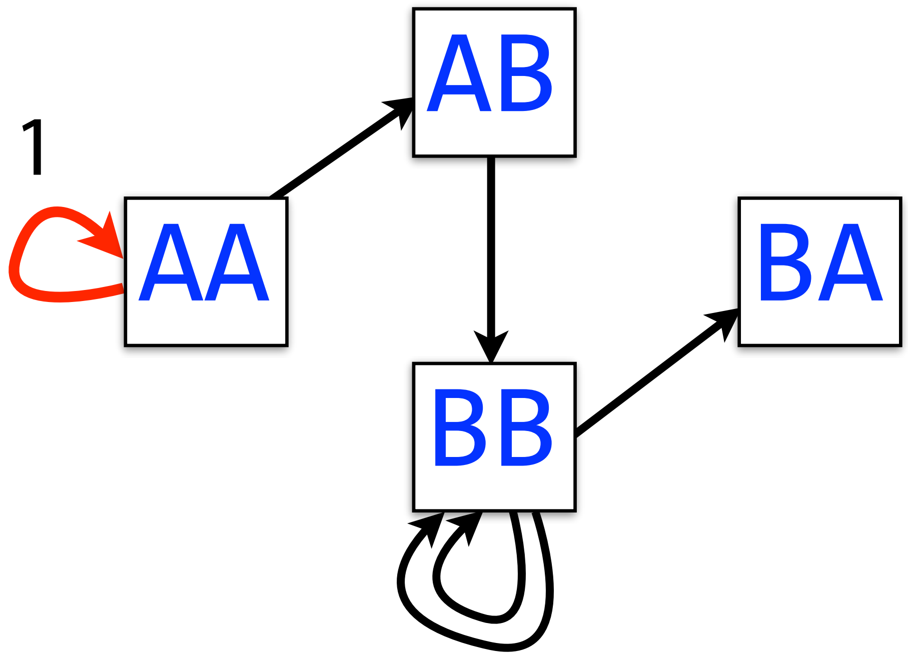
De Bruijn graph



Walk crossing each edge exactly once gives a reconstruction of the genome

*

De Bruijn graph

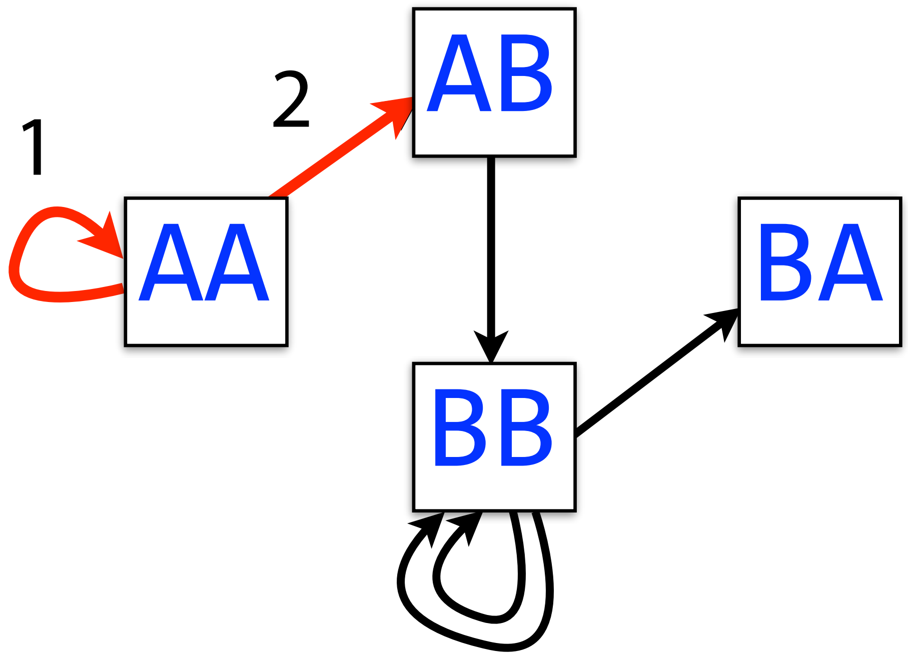


AAA

Walk crossing each edge exactly once gives a reconstruction of the genome

*

De Bruijn graph

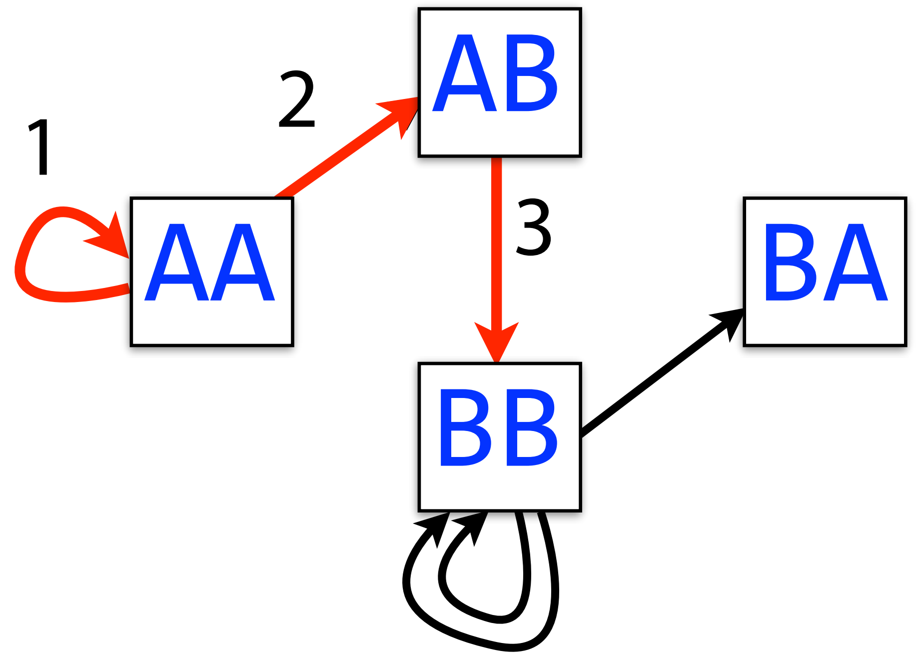


AAA B

Walk crossing each edge exactly once gives a reconstruction of the genome

*

De Bruijn graph

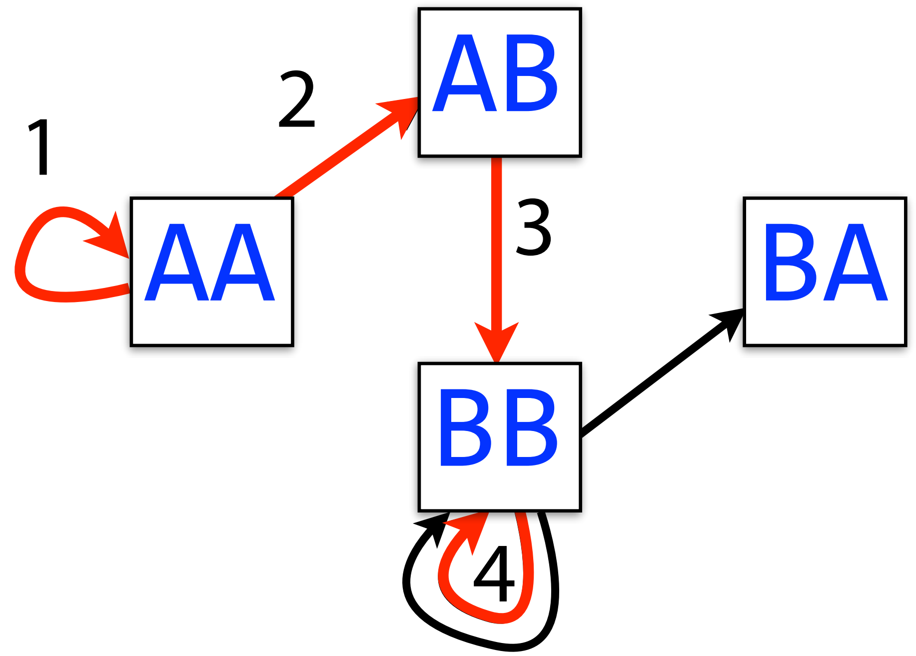


AAA BB

Walk crossing each edge exactly once gives a reconstruction of the genome

*

De Bruijn graph

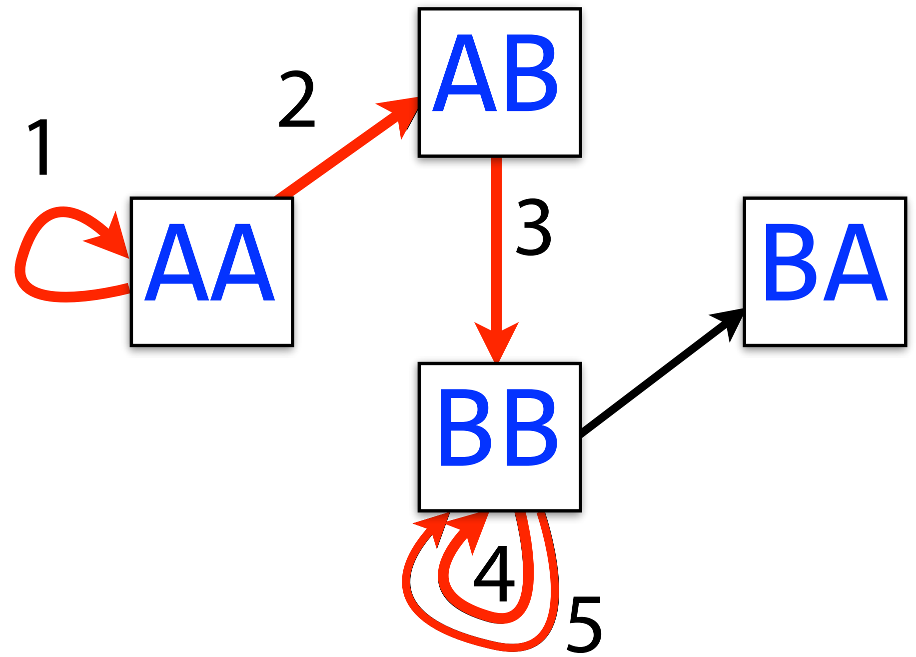


AAA BBB

Walk crossing each edge exactly once gives a reconstruction of the genome

*

De Bruijn graph

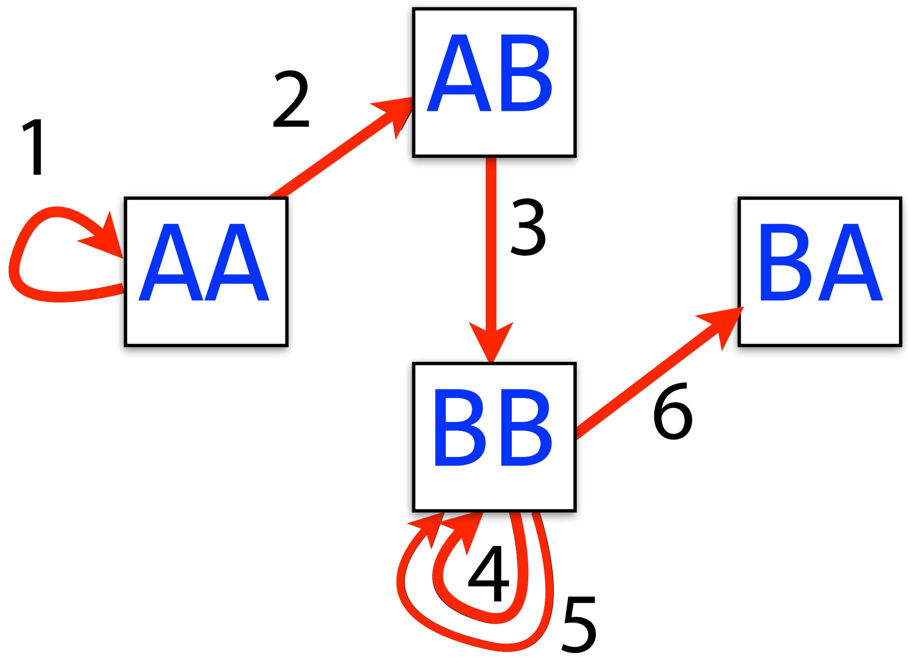


AAA BBBB

Walk crossing each edge exactly once gives a reconstruction of the genome

*

De Bruijn graph

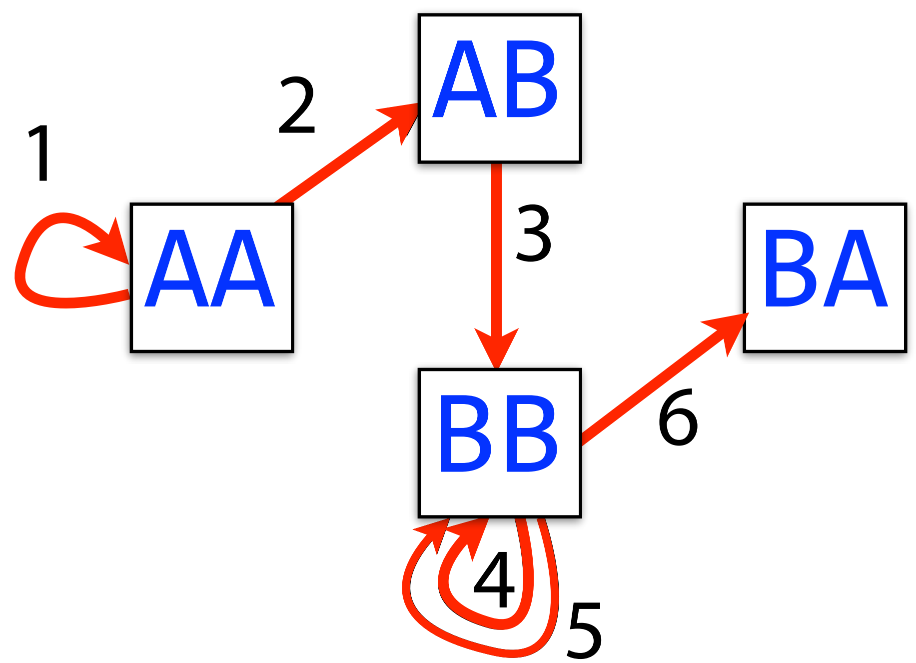


AAA BBBBA

Walk crossing each edge exactly once gives a reconstruction of the genome

*

De Bruijn graph



AAA BBBBA

Walk crossing each edge exactly once gives a reconstruction of the genome . This is an Eulerian walk.

*

De Bruijn graph

Aside: how do you pronounce "De Bruijn"?

There is debate:

https://www.biostars.org/p/7186/

Nicolaas Govert
de Bruijn
1918 -- 2012

*

The (vertex-centric) dBG is *implicit* in the k-mer set

How can a membership structure be used to navigate the dBG?

A given (k-1)-mer can only have 2*|Σ| neighbors; |Σ| incoming and |Σ| outgoing neighbors — for genomes |Σ| = 4

To navigate in the De Bruijn graph, we can simply query all possible successors, and see which are actually present.

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
Stony Brook University
Stony Brook, NY, USA
{ppandey, bender, rob, rob.patro}@cs.stonybrook.edu

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]

1 Stony Brook University, Stony Brook, NY, USA
falmodaresit@cs.stonybrook.edu
2 Stony Brook University, Stony Brook, NY, USA
ppandey@cs.stonybrook.edu
3 Stony Brook University, Stony Brook, NY, USA
rob.patro@cs.stonybrook.edu

WABI 2017

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

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

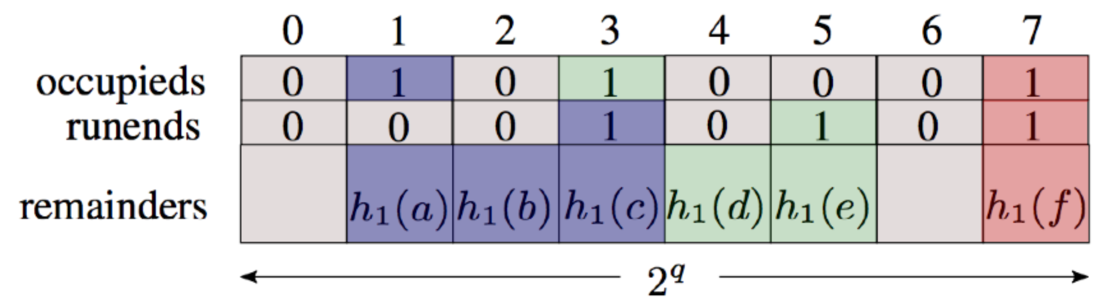
Prashant Pandey[1], Fatemeh Almodaresi[1], Michael A. Bender[1], Michael Ferdman[1], Rob Johnson[2,1], and Rob Patro[1]

[1] Computer Science Dept., Stony Brook University
{ppandey,falmodaresit,bender,mferdman,rob.patro}@cs.stonybrook.edu
[2] VMware Research
robj@vmware.com

RECOMB 2018 & Cell Systems (https://doi.org/10.1016/j.cels.2018.05.021)

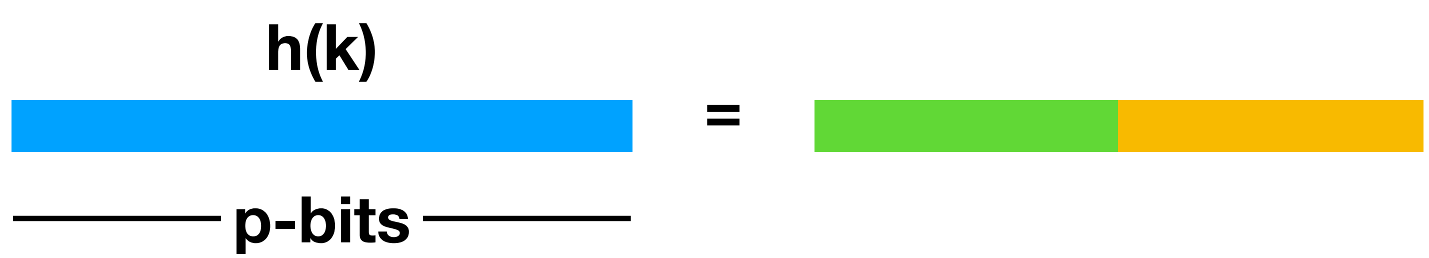
The CQF

Approximate *Multiset* Representation



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 ⇒ 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)

The CQF

Approximate *Multiset* Representation

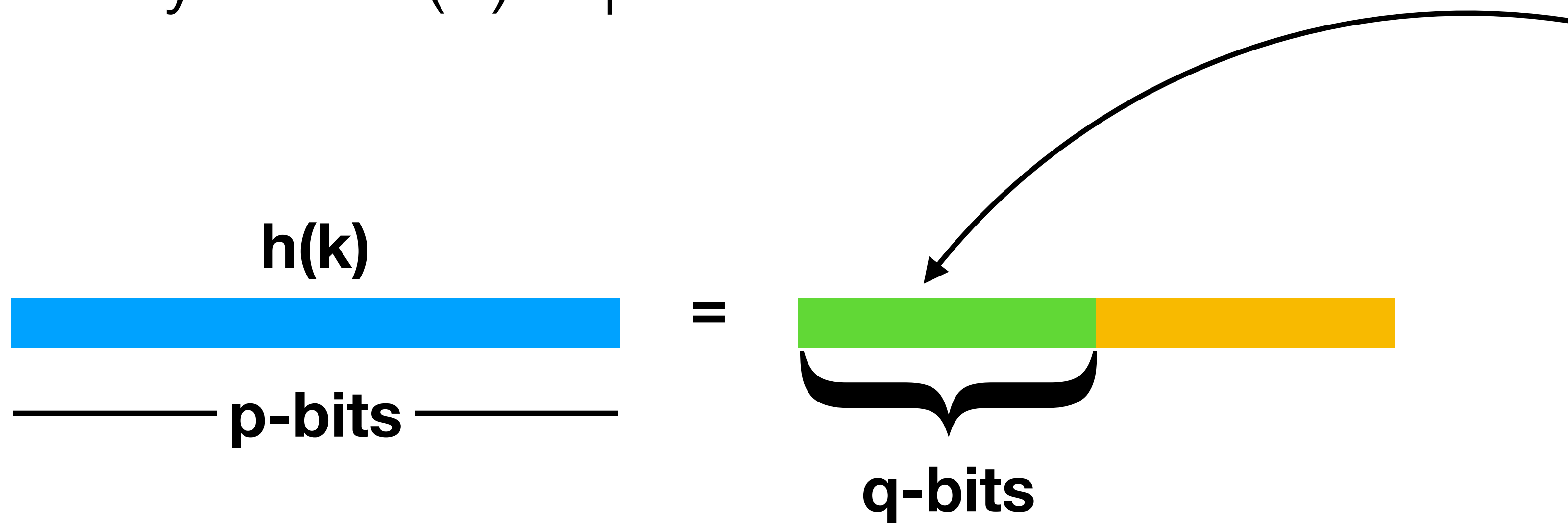
	0	1	2	3	4	5	6	7
occupieds	0	1	0	1	0	0	0	1
runends	0	0	0	1	0	1	0	1
remainders		$h_1(a)$	$h_1(b)$	$h_1(c)$	$h_1(d)$	$h_1(e)$		$h_1(f)$

$\longleftarrow 2^q \longrightarrow$

Works based on quotienting* & fingerprinting keys

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

Determines position in array of size 2^q r-bit slots



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

The CQF

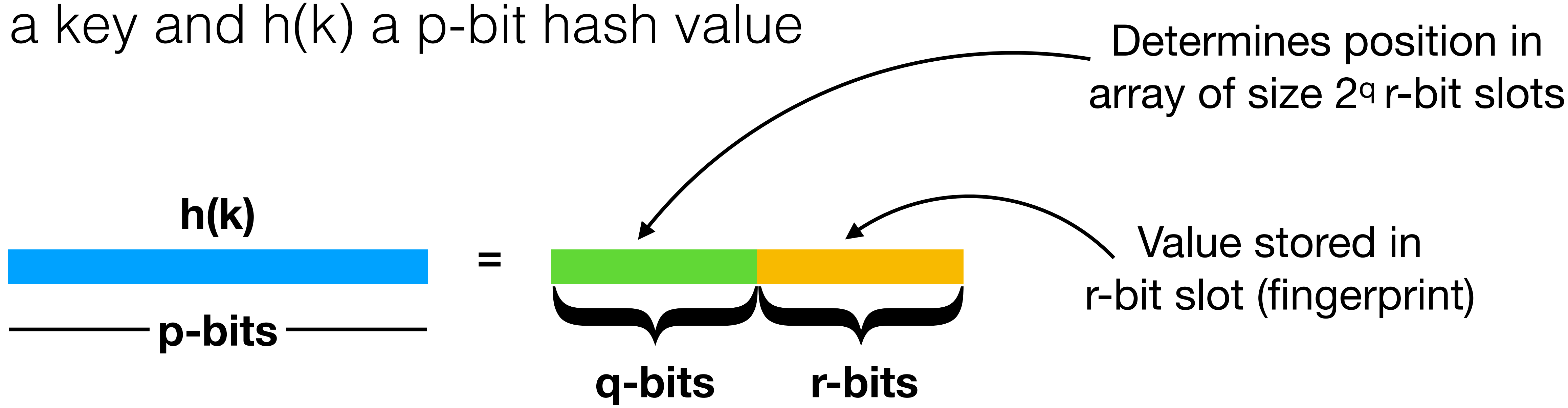
Approximate *Multiset* Representation

	0	1	2	3	4	5	6	7
occupieds	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 ⇒ 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 $\leq \min\left(N, 2^{|E|}\right)$ 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?

https://github.com/splatlab/mantis

Mantis

Observation 1 : If I want to index N k-mers over E experiments, there are $\leq \min\left(N, 2^{|E|}\right)$ 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 it's k-mers*.

https://github.com/splatlab/mantis

Mantis

Observation 1 : If I want to index N k-mers over E experiments, there are $\leq \min\left(N, 2^{|E|}\right)$ 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.*

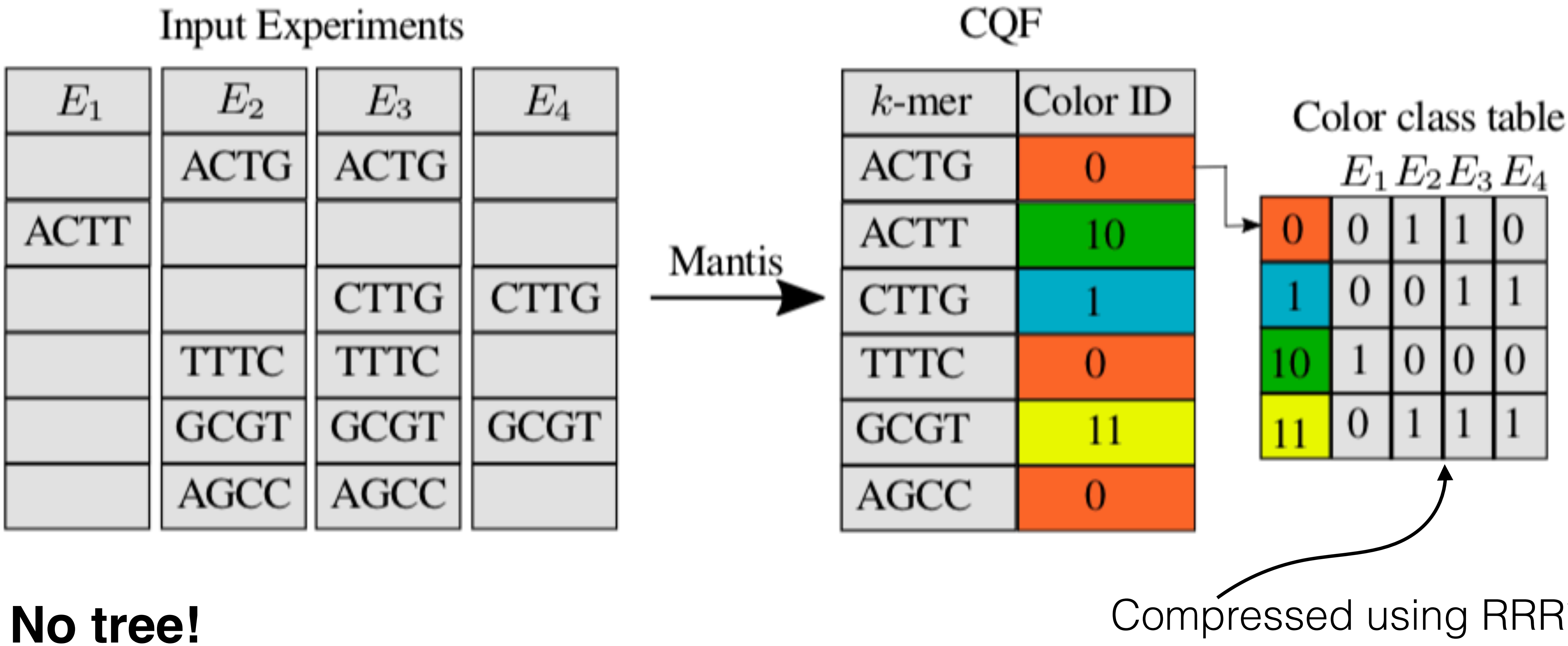
Why? Consider e.g. a gene G (~1000 k-mers). If it is present in an experiment at moderate to high abundance, we will likely observe *all of it’s k-mers*.

What if we add a layer of indirection: Store each distinct pattern (color class) only once. *label* each pattern with 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



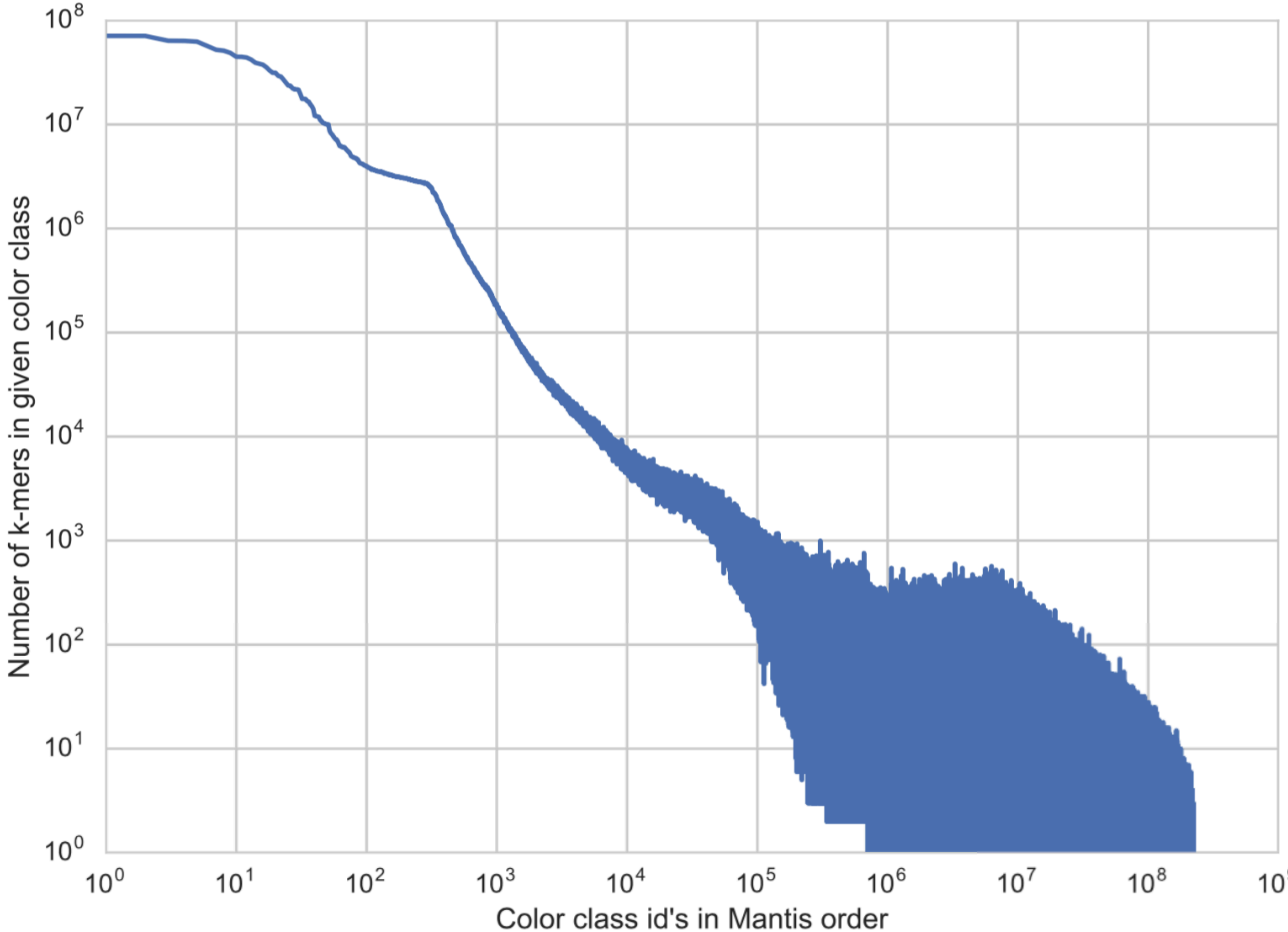
No tree!

- 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

*Raman, et al. (2002). Succinct indexable dictionaries with applications to encoding k-ary trees and multisets. In Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms, pages 233–242.

Why does this work?

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

Mantis is compact enough that we can *exactly* rather than *approximately* index the k-mers in our experiment set.

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 (GZip compressed) of data

Table 1. Time and Space Measurement for Mantis and SSBT

	Mantis	SSBT
Build time	16 hr 35 min	97 hr
Representation size	32 GB	39.7 GB

- Mantis can be constructed ~6x 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.

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

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

A Mantis query returns, for each experiment containing at least one query k-mer, the *fraction* (true θ) of query k-mers contained in the experiment.

Mantis : Query Quality

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

"Both" means the number of those experiments that are reported by both Mantis and SSBT. "Only Mantis" and "Only SSBT" mean the number of experiments reported by only Mantis and only SSBT. All three query benchmarks are taken from Table 2 for $\theta = 0.8$.

- Recall : Mantis is exact! Returns *only* experiments having ≥ θ fraction of the query k-mers.

Mantis : Query Quality

Querying for the presence of randomly selected genes across all 2,652 experiments. SSBT θ = 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 ≥ θ 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.

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:

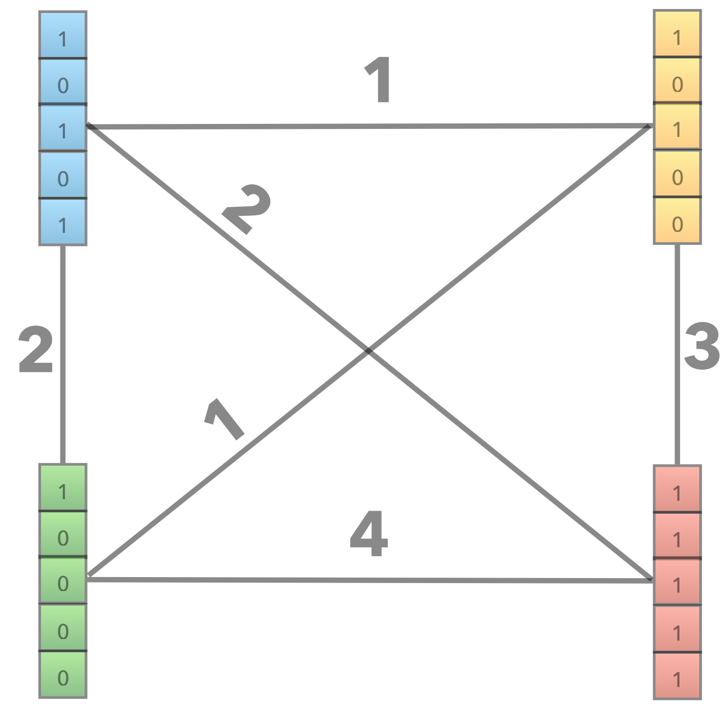
- K-mers grow at worst linearly
- Color classes increase super-linearly

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

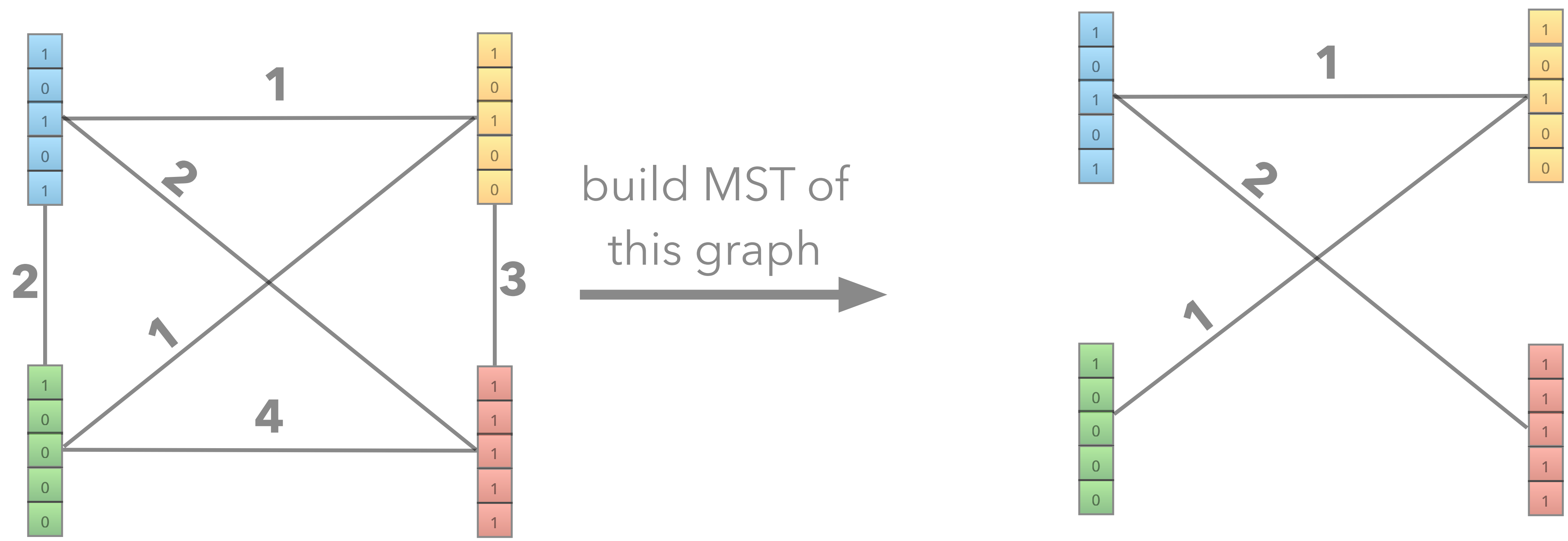


Bookstein, Abraham, and Shmuel T. Klein. "Compression of correlated bit-vectors." *Inf. Syst.* 16.4 (1991): 387-400.

Consider the following color class graph

Each color class is a vertex

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

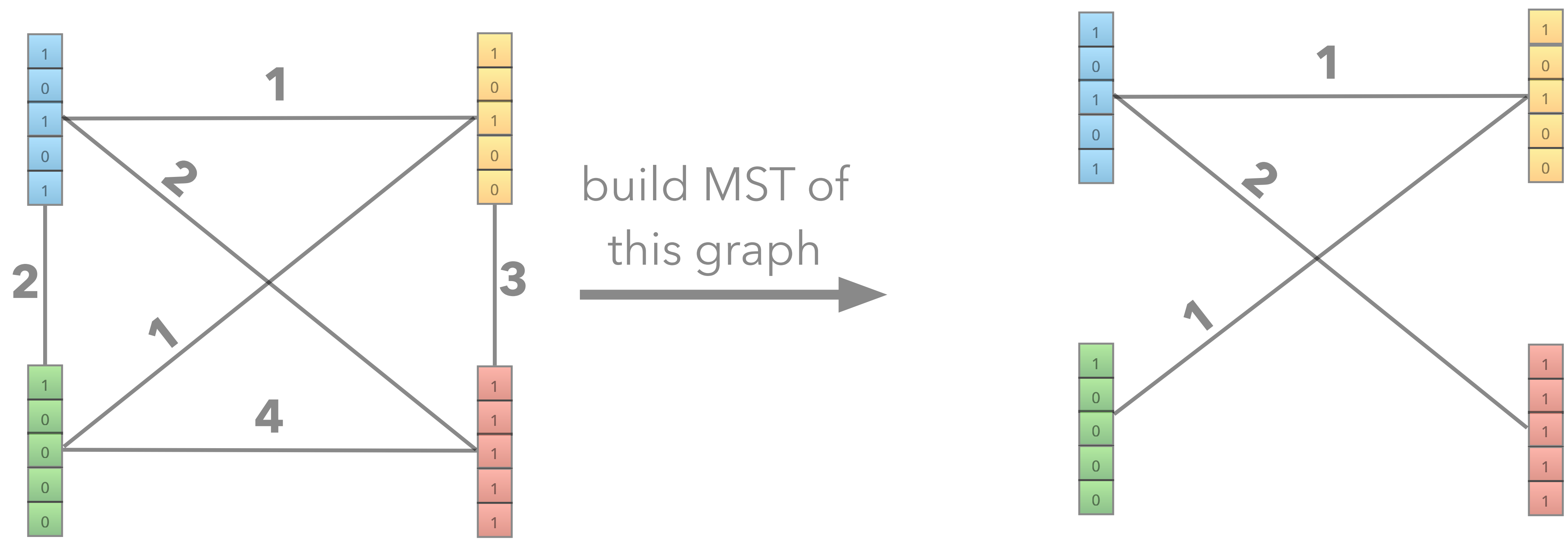


Bookstein, Abraham, and Shmuel T. Klein. "Compression of correlated bit-vectors." *Inf. Syst.* 16.4 (1991): 387-400.

Consider the following color class graph

Each color class is a vertex

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



Unfortunately:

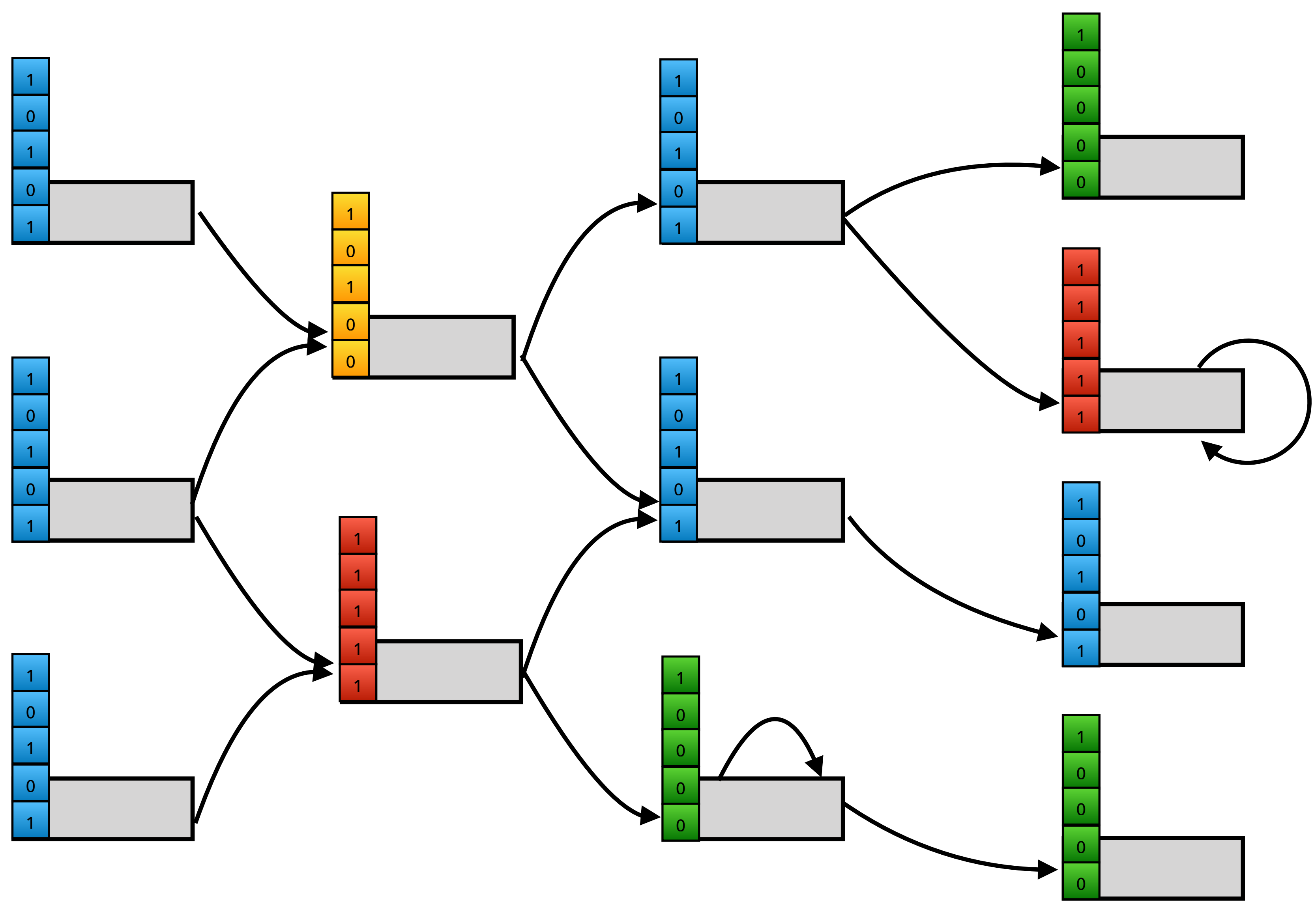
1) There are *many* color classes (full graph too big)
2) They are high-dimensional (# of experiments), neighbor search is very hard (LSH scheme seem to work poorly)

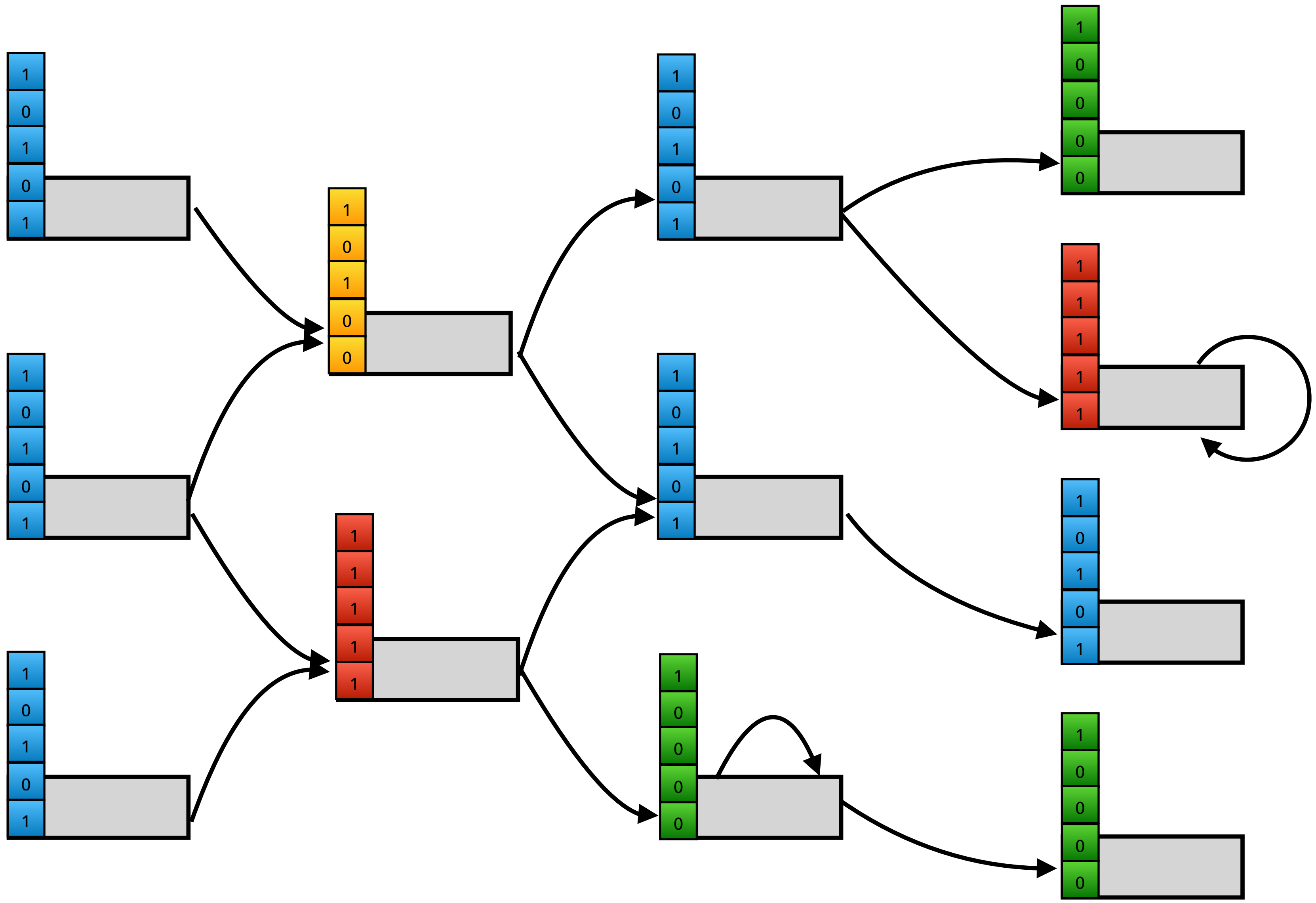
Bookstein, Abraham, and Shmuel T. Klein. "Compression of correlated bit-vectors." *Inf. Syst.* 16.4 (1991): 387-400.

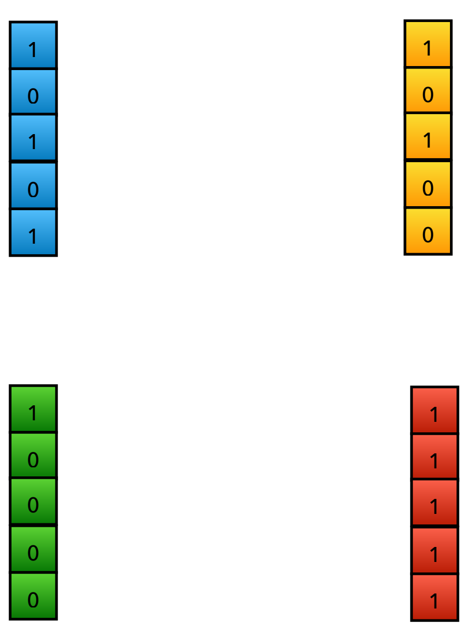
Mantis implicitly represents a colored dBG

Each CQF key represents a kmer → can explicitly query neighbors

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

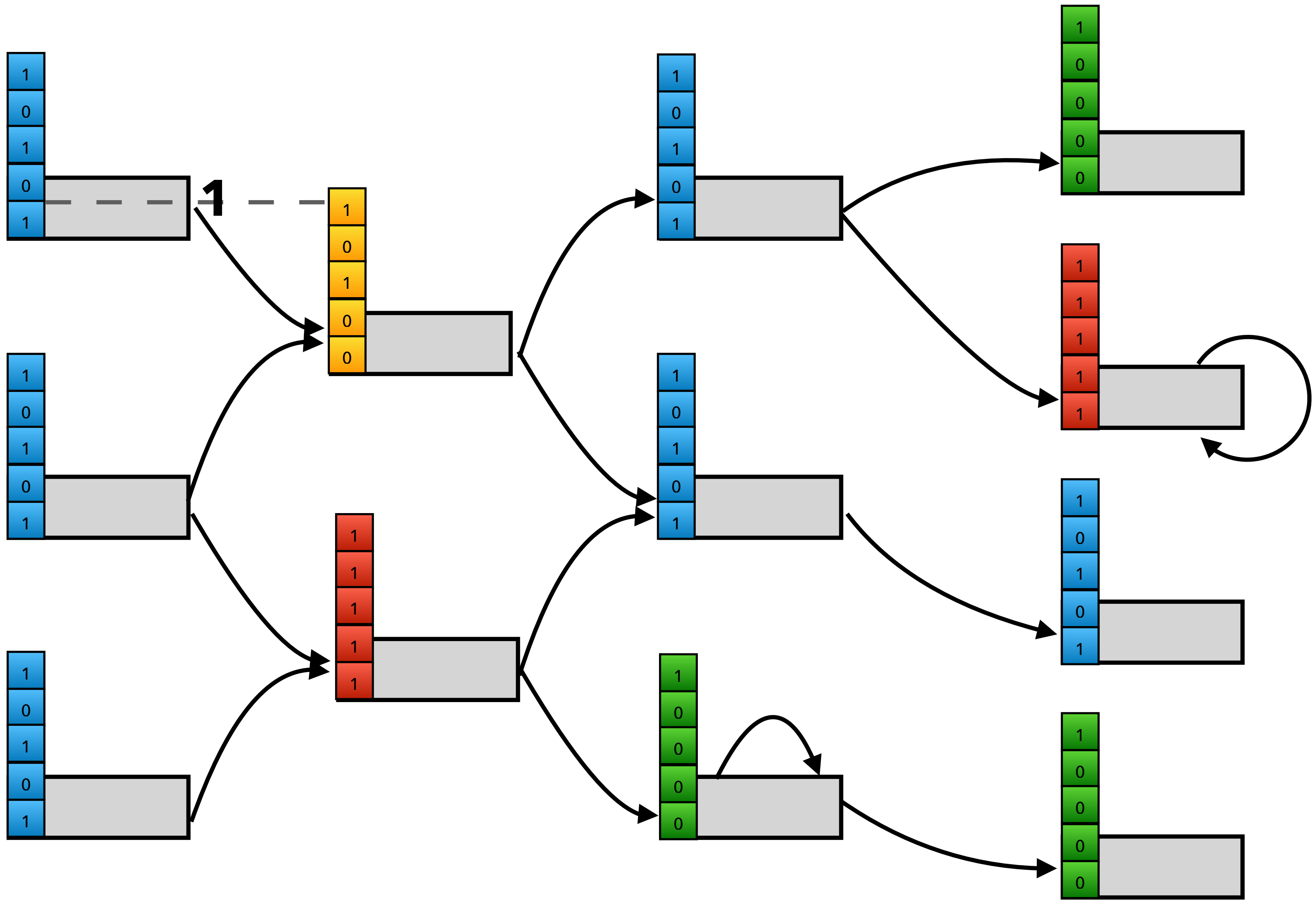


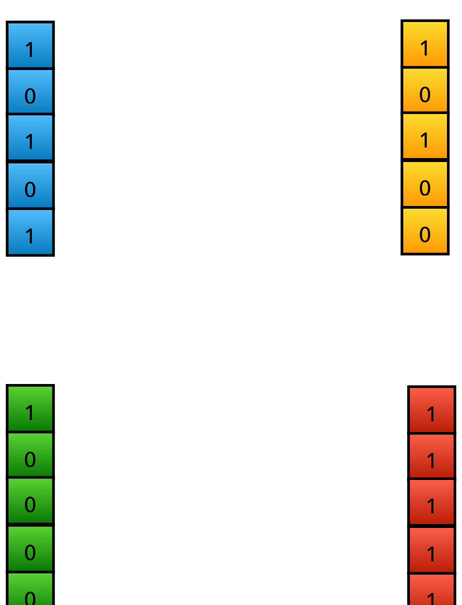




Use the **de Bruin graph** (dBG) as an efficient guide for near-neighbor search in the space of color classes!

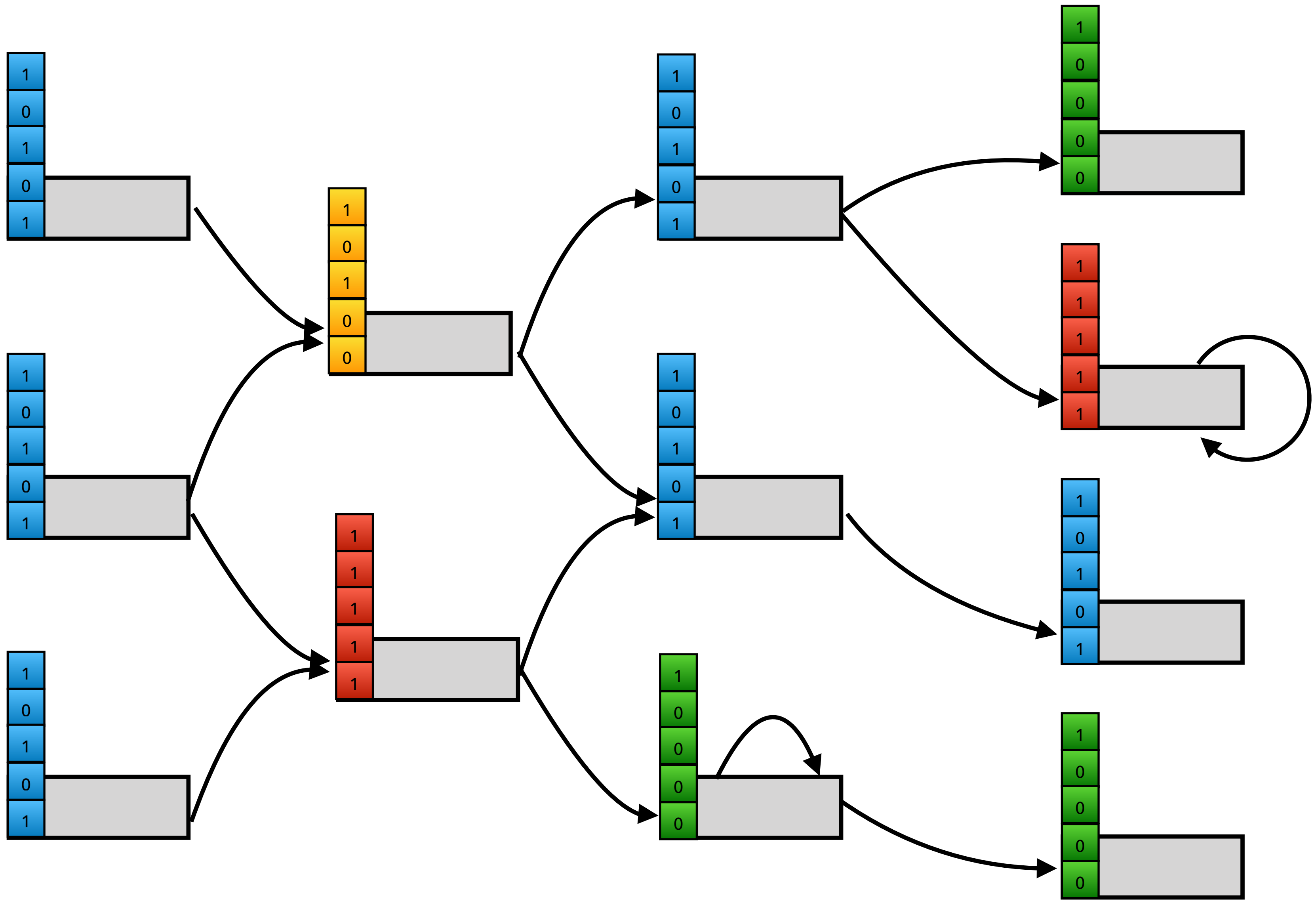
dBG common in genomics. Nodes u,v are k-mers & are *adjacent* if k-1 suffix of u is the same as k-1 prefix of v

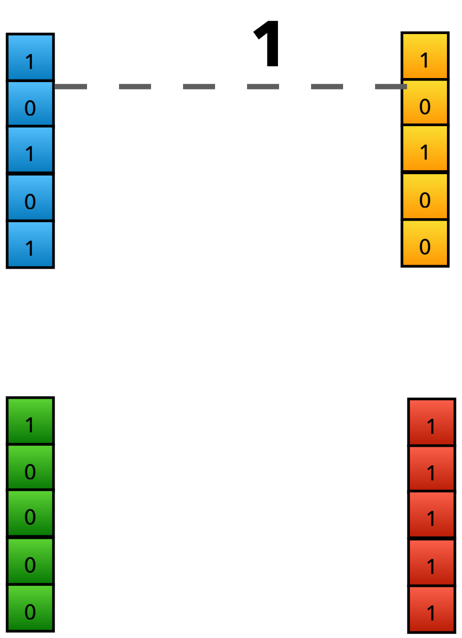




Use the **de Bruin graph** (dBG) as an efficient guide for near-neighbor search in the space of color classes!

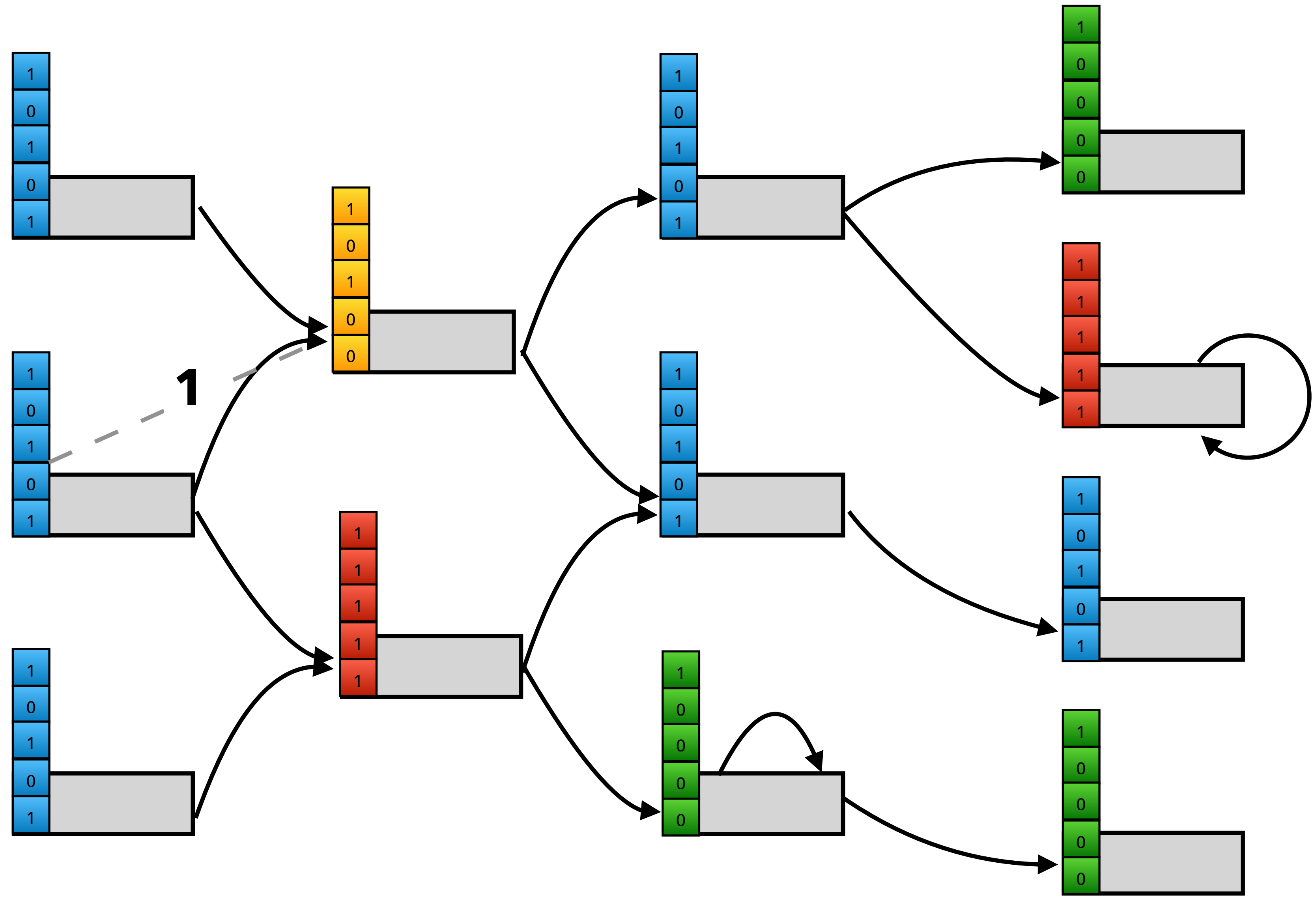
dBG common in genomics. Nodes u,v are k-mers & are *adjacent* if k-1 suffix of u is the same as k-1 prefix of v

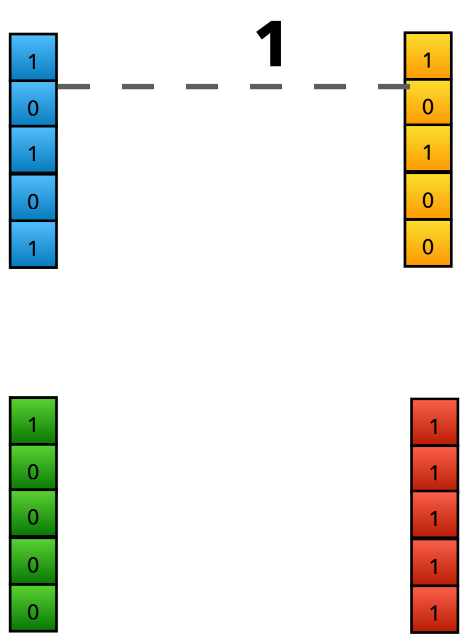




Use the **de Bruin graph** (dBG) as an efficient guide for near-neighbor search in the space of color classes!

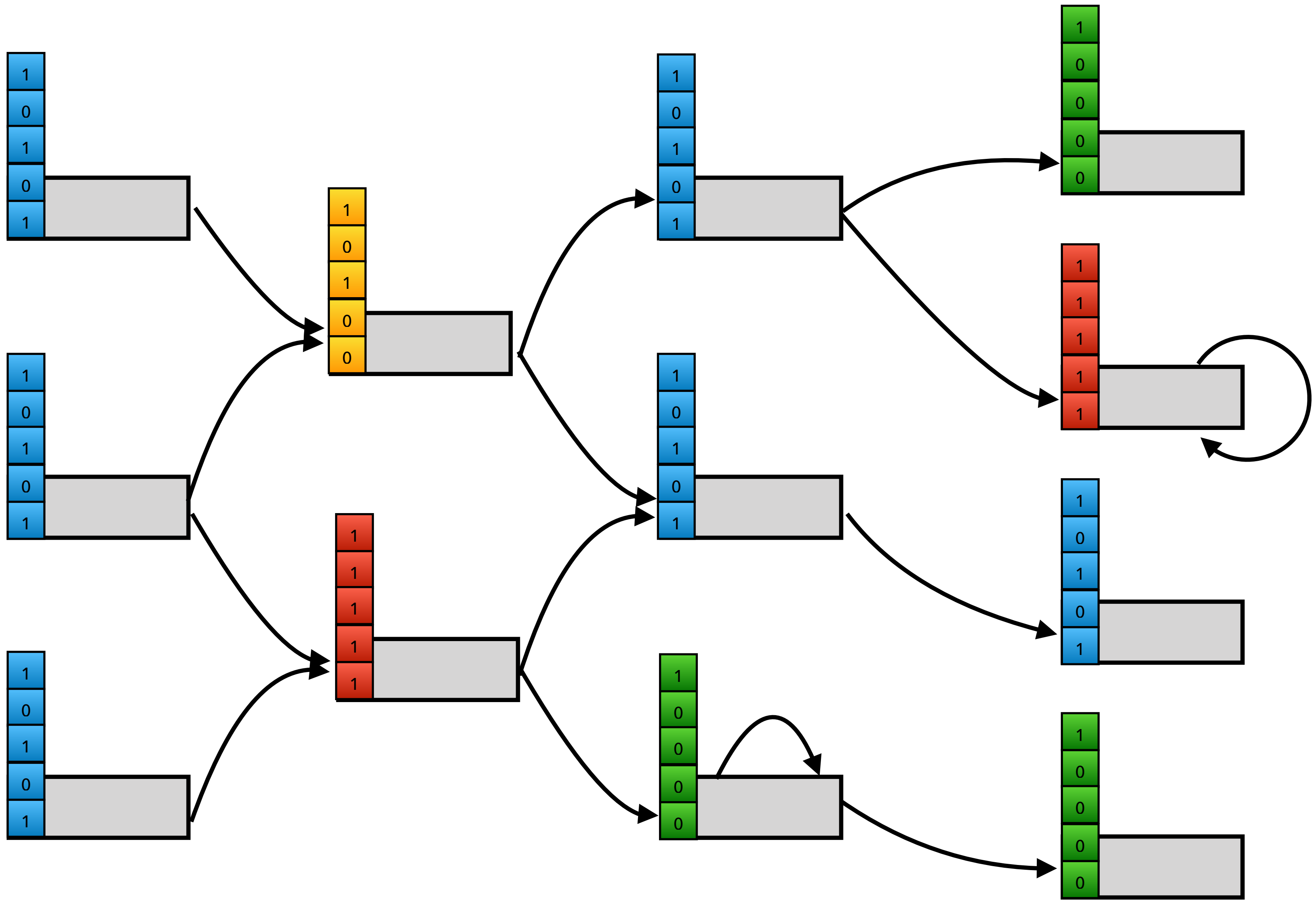
dBG common in genomics. Nodes u,v are k-mers & are *adjacent* if k-1 suffix of u is the same as k-1 prefix of v

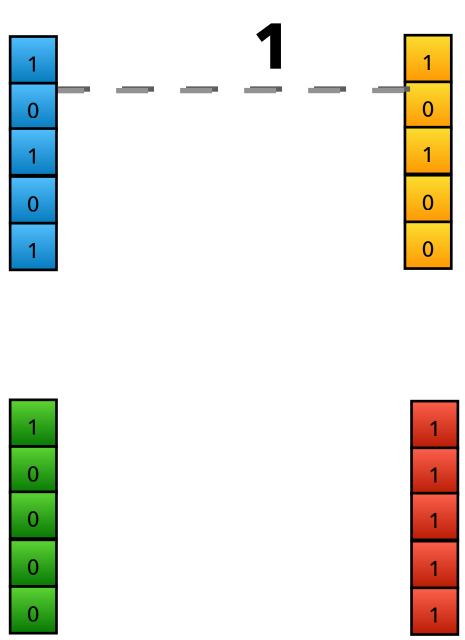




Use the **de Bruin graph** (dBG) as an efficient guide for near-neighbor search in the space of color classes!

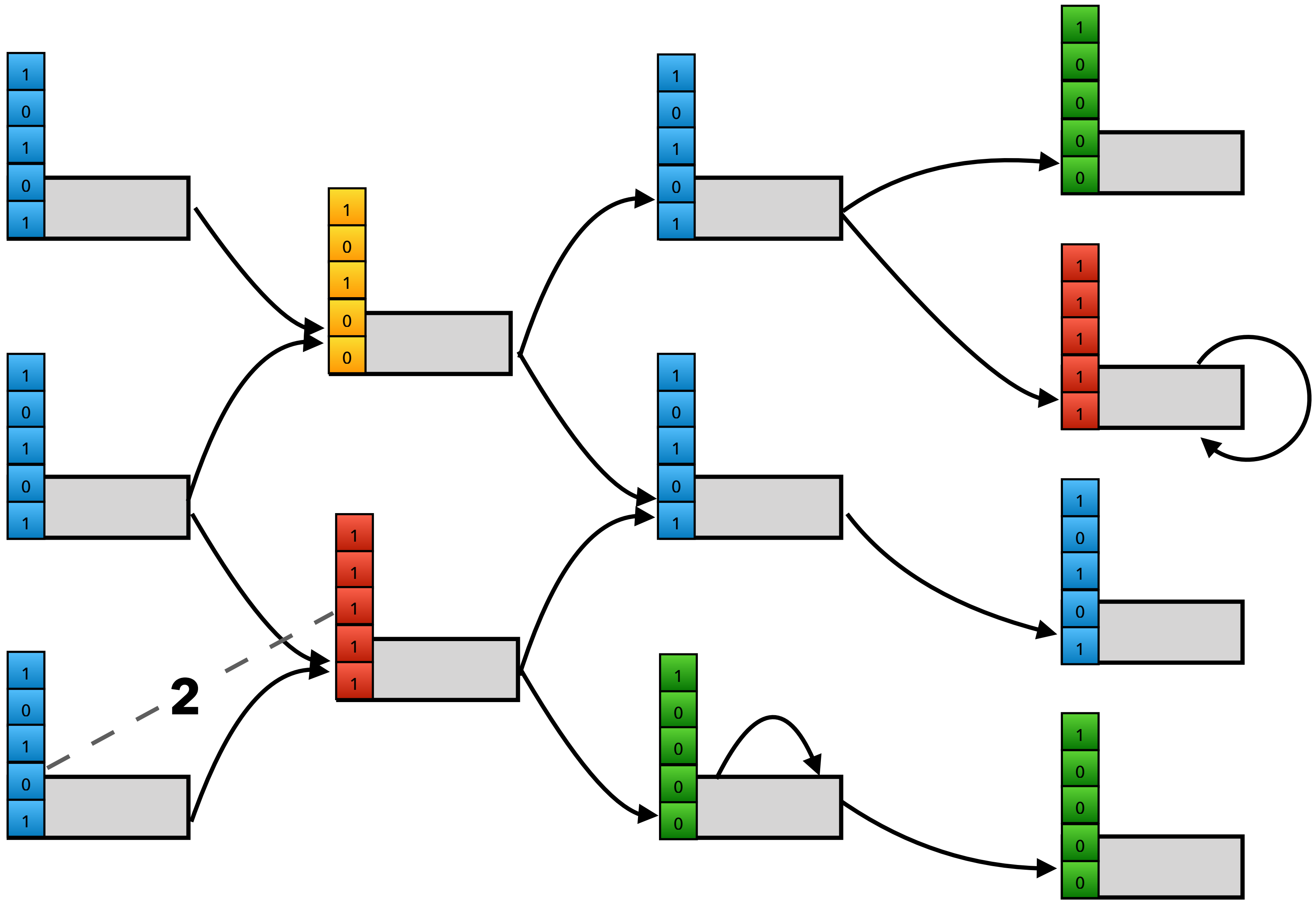
dBG common in genomics. Nodes u,v are k-mers & are *adjacent* if k-1 suffix of u is the same as k-1 prefix of v

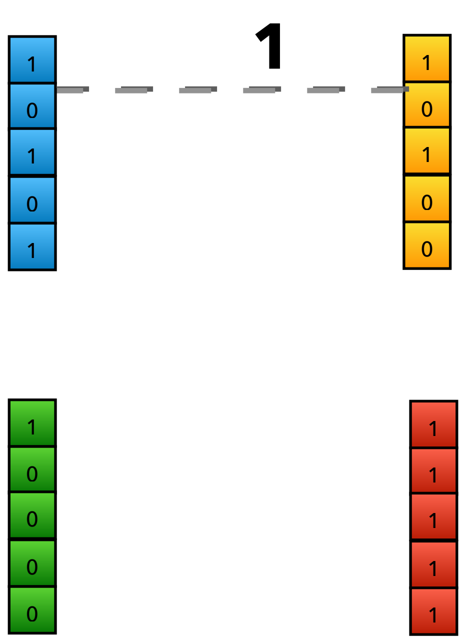




Use the **de Bruin graph** (dBG) as an efficient guide for near-neighbor search in the space of color classes!

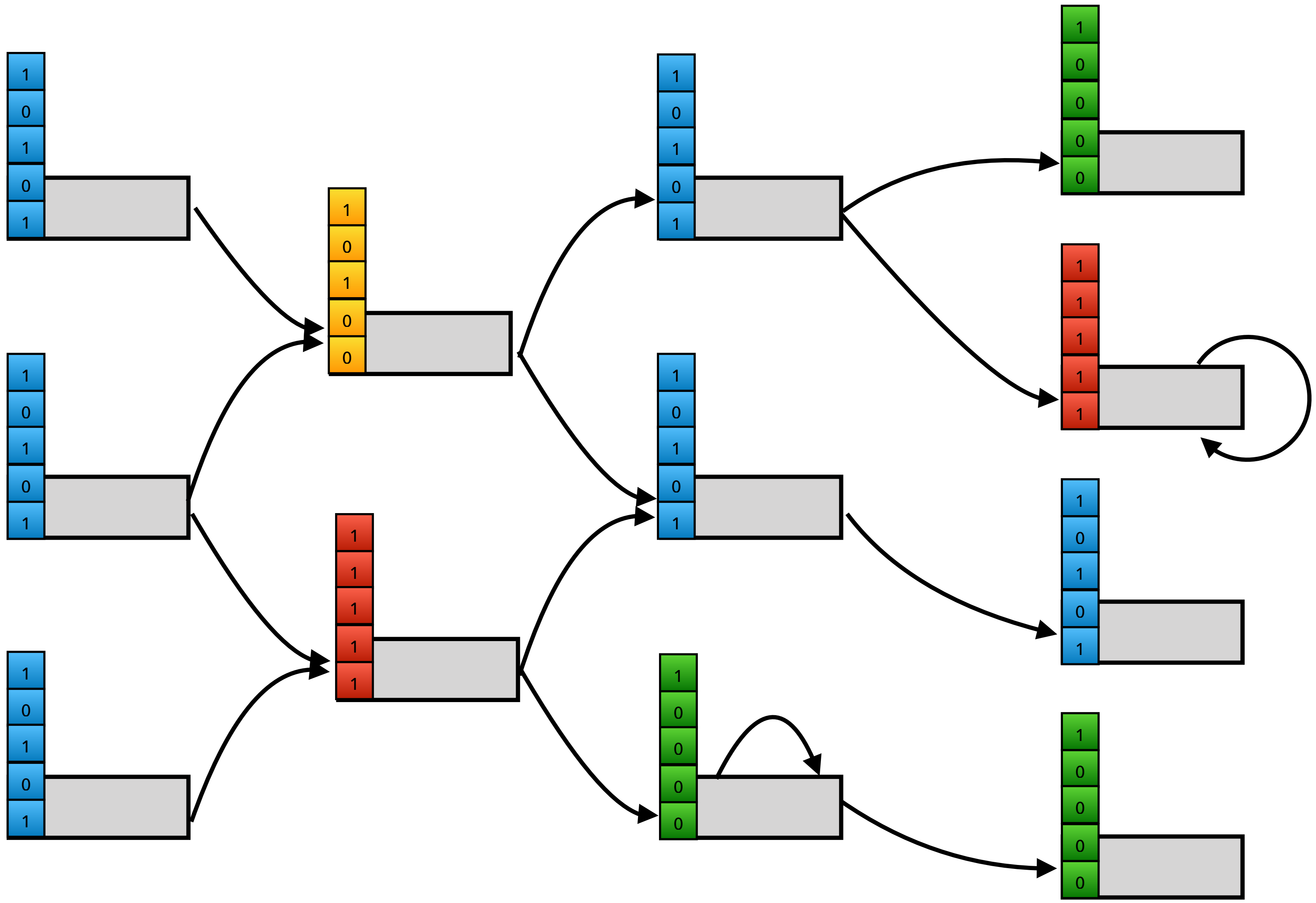
dBG common in genomics. Nodes u,v are k-mers & are *adjacent* if k-1 suffix of u is the same as k-1 prefix of v

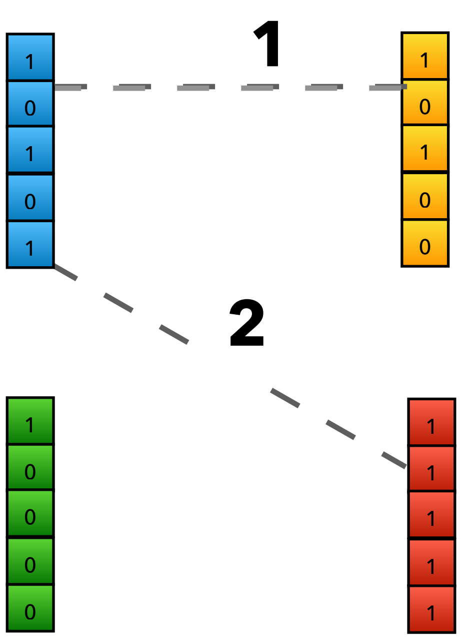




Use the **de Bruin graph** (dBG) as an efficient guide for near-neighbor search in the space of color classes!

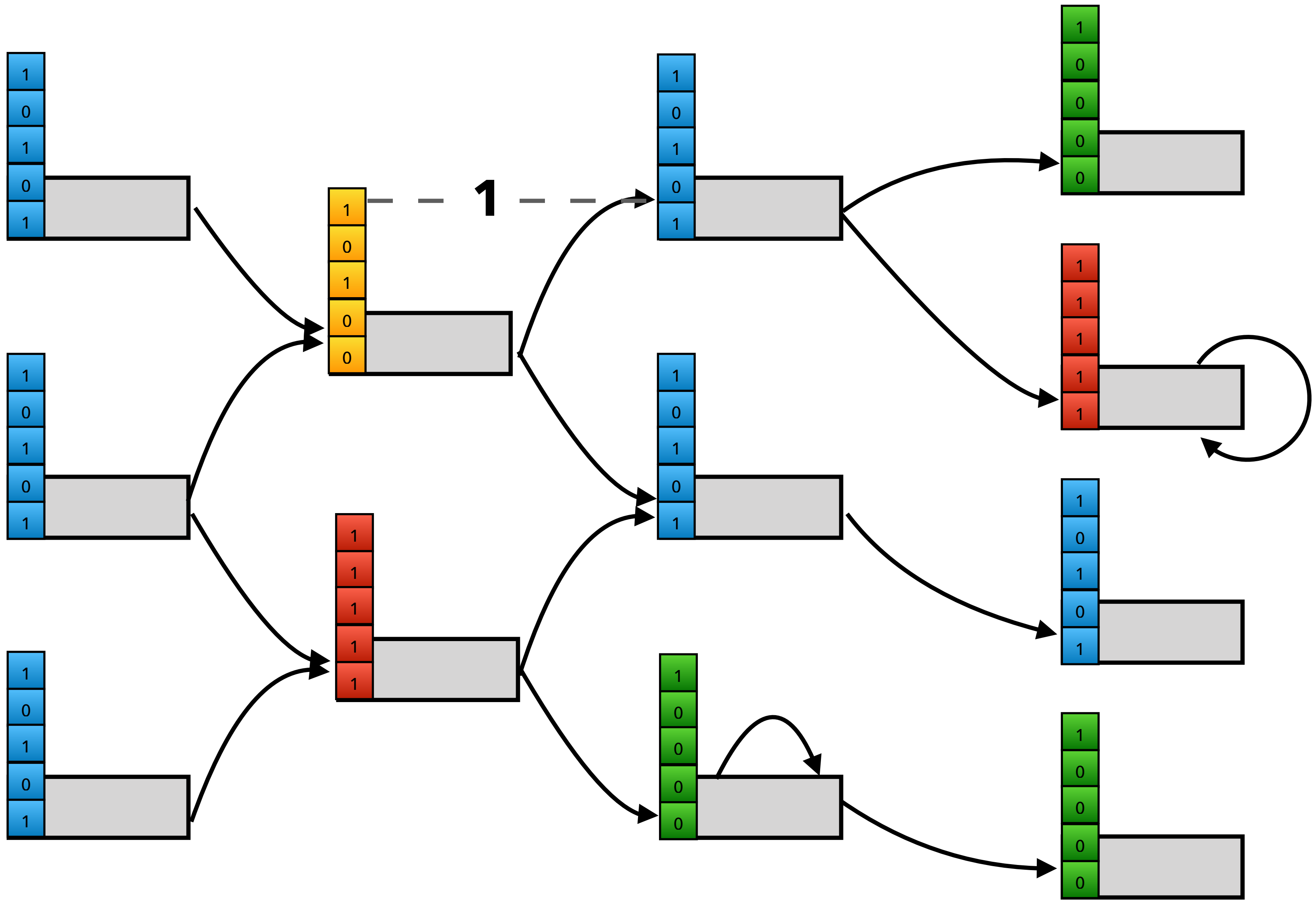
dBG common in genomics. Nodes u,v are k-mers & are *adjacent* if k-1 suffix of u is the same as k-1 prefix of v

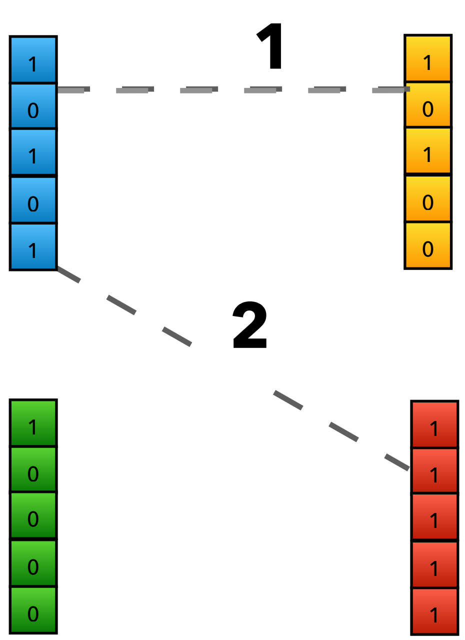




Use the **de Bruin graph** (dBG) as an efficient guide for near-neighbor search in the space of color classes!

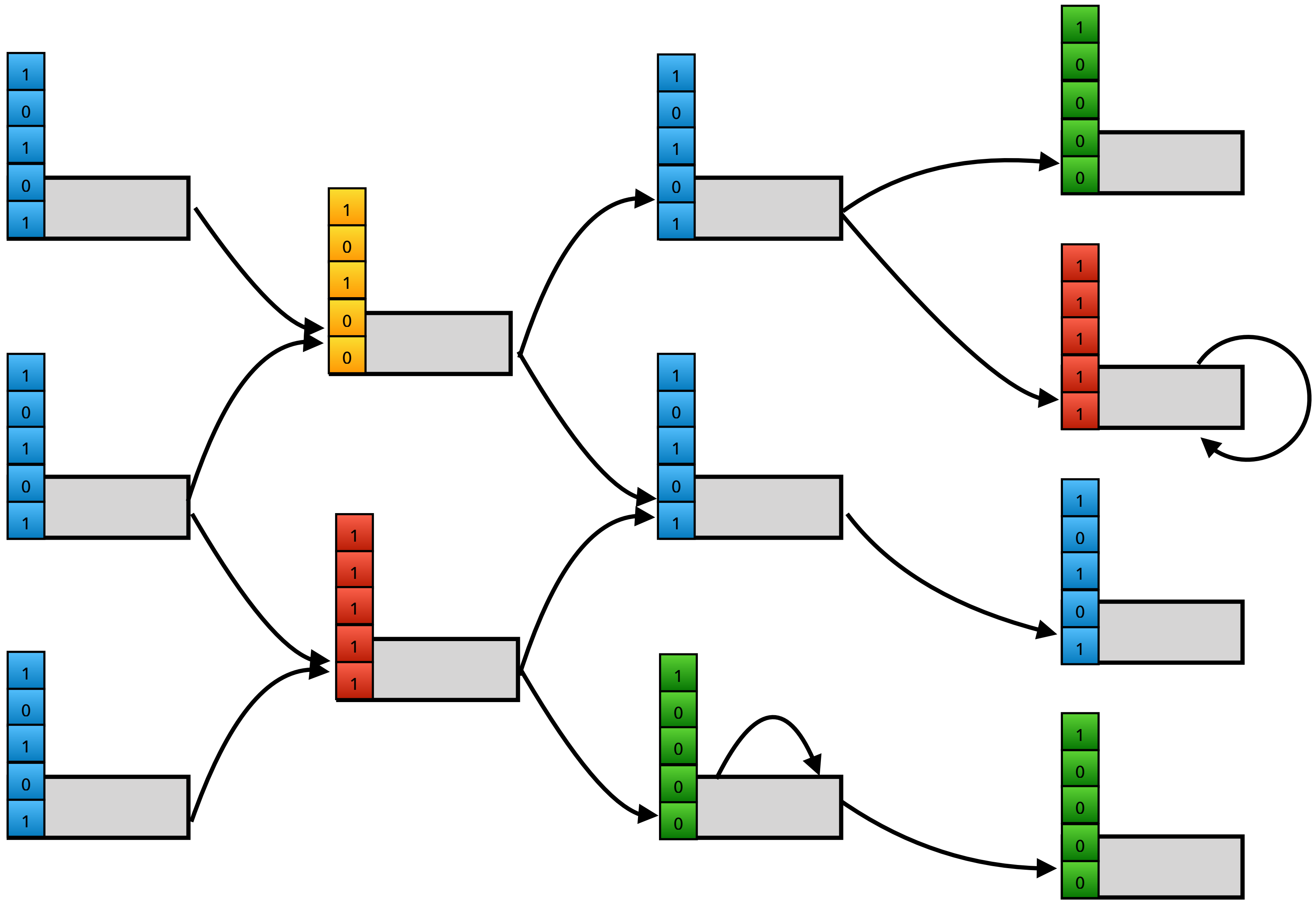
dBG common in genomics. Nodes u,v are k-mers & are *adjacent* if k-1 suffix of u is the same as k-1 prefix of v

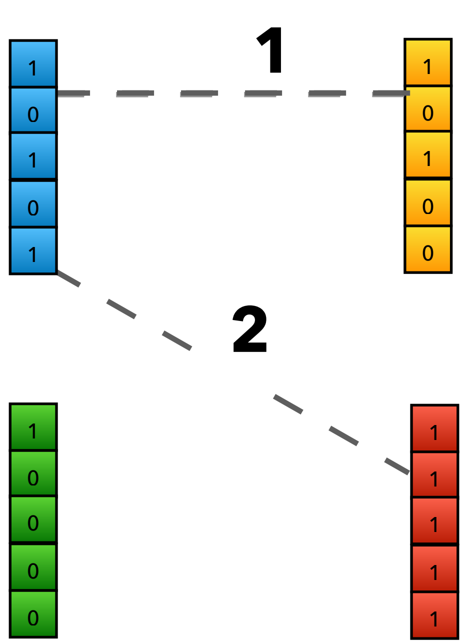




Use the **de Bruin graph** (dBG) as an efficient guide for near-neighbor search in the space of color classes!

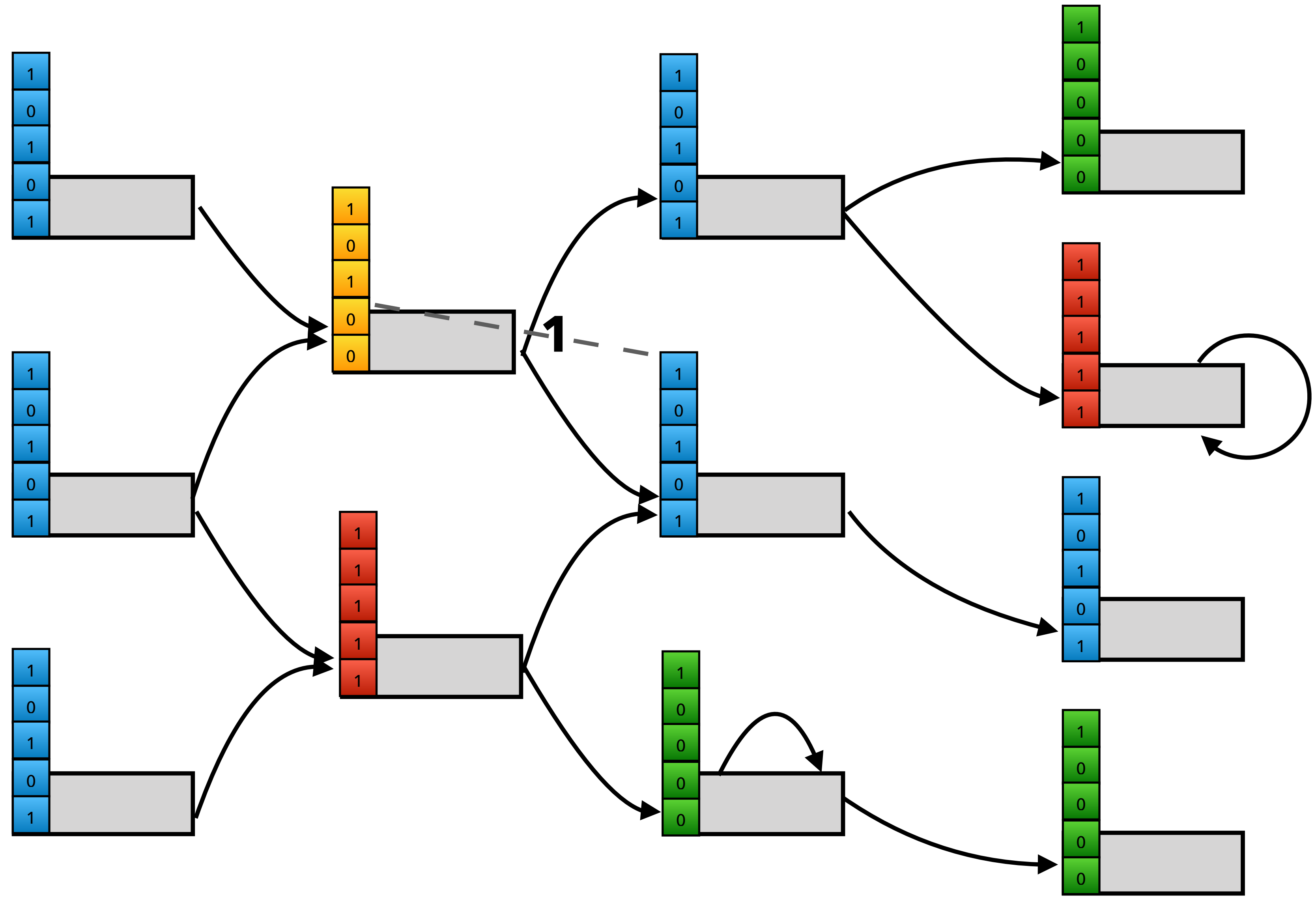
dBG common in genomics. Nodes u,v are k-mers & are *adjacent* if k-1 suffix of u is the same as k-1 prefix of v

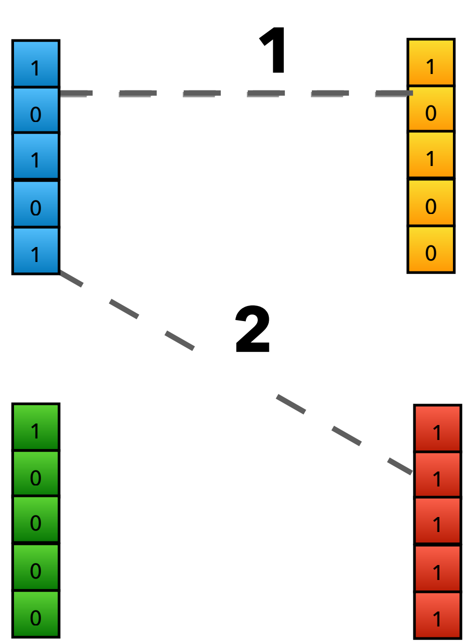




Use the **de Bruin graph** (dBG) as an efficient guide for near-neighbor search in the space of color classes!

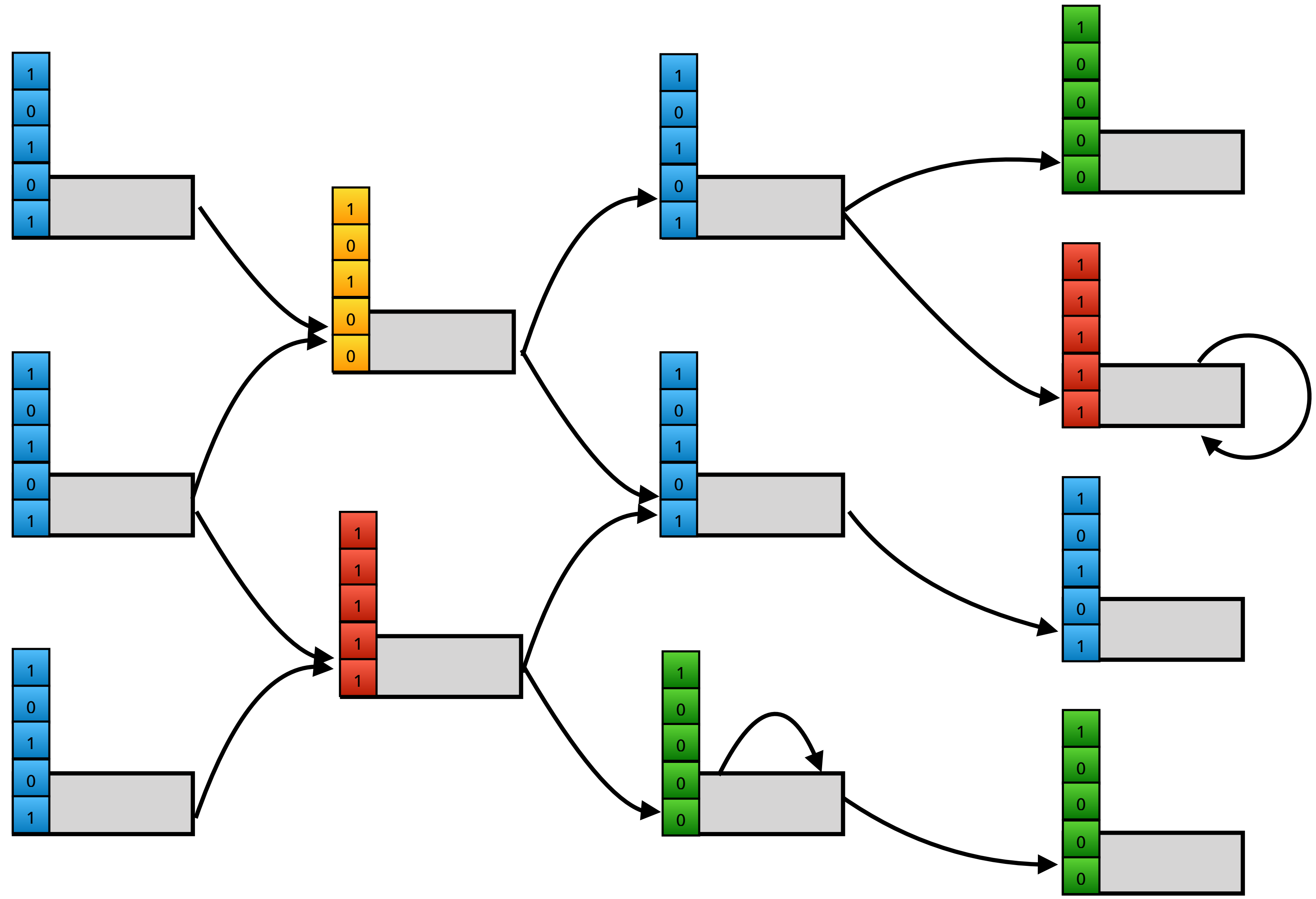
dBG common in genomics. Nodes u,v are k-mers & are *adjacent* if k-1 suffix of u is the same as k-1 prefix of v

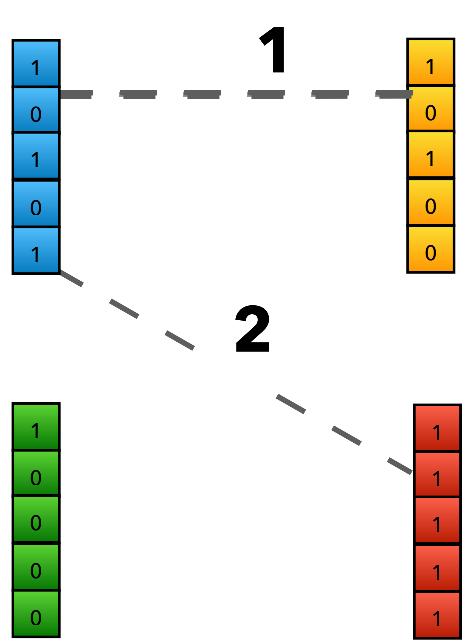




Use the **de Bruin graph** (dBG) as an efficient guide for near-neighbor search in the space of color classes!

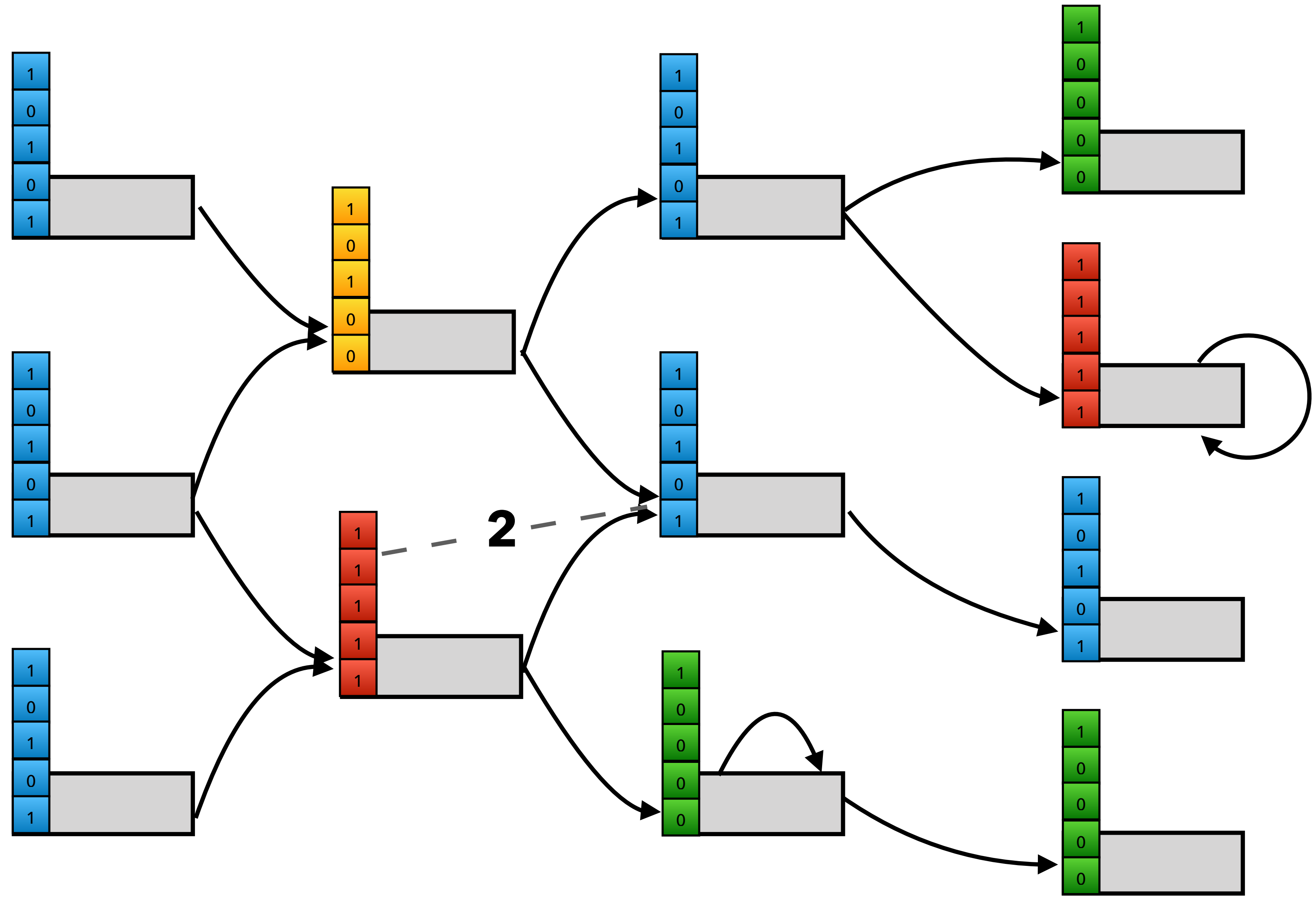
dBG common in genomics. Nodes u,v are k-mers & are *adjacent* if k-1 suffix of u is the same as k-1 prefix of v

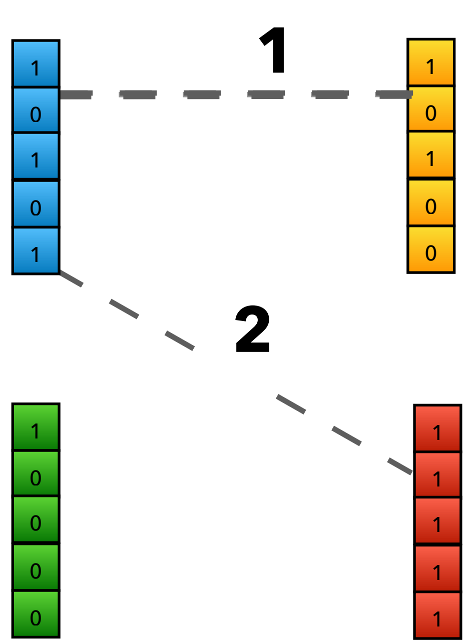




Use the **de Bruin graph** (dBG) as an efficient guide for near-neighbor search in the space of color classes!

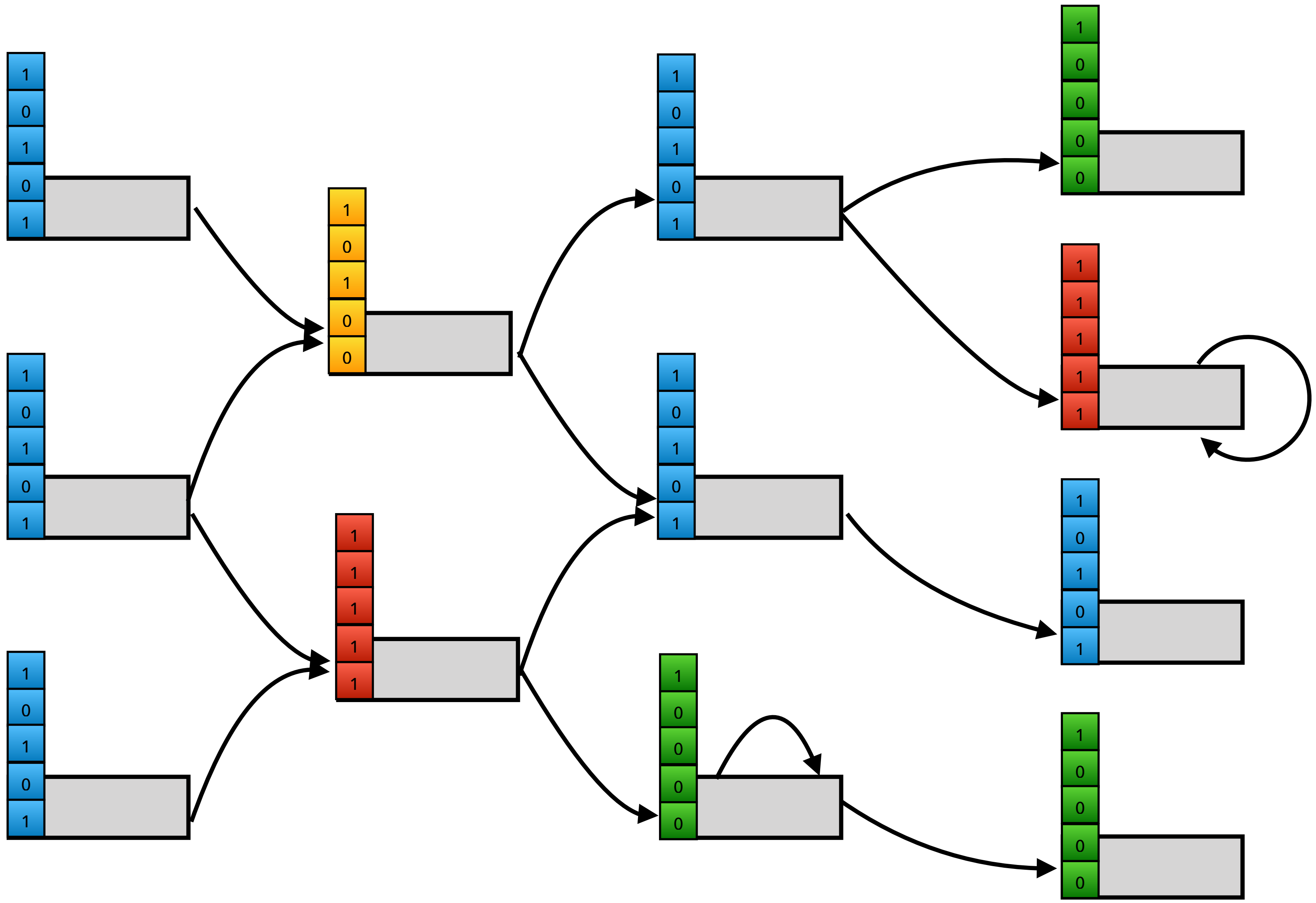
dBG common in genomics. Nodes u,v are k-mers & are *adjacent* if k-1 suffix of u is the same as k-1 prefix of v

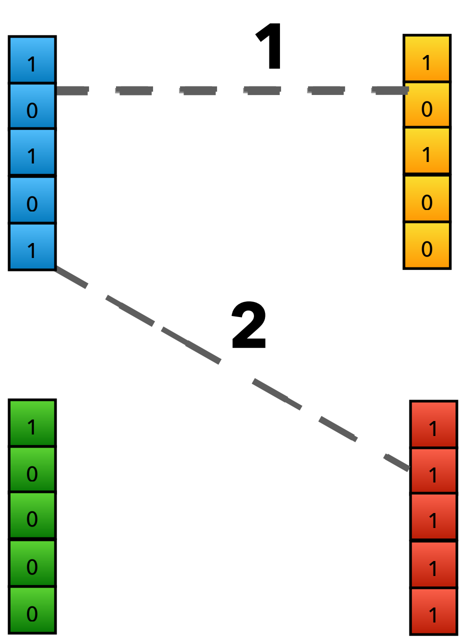




Use the **de Bruin graph** (dBG) as an efficient guide for near-neighbor search in the space of color classes!

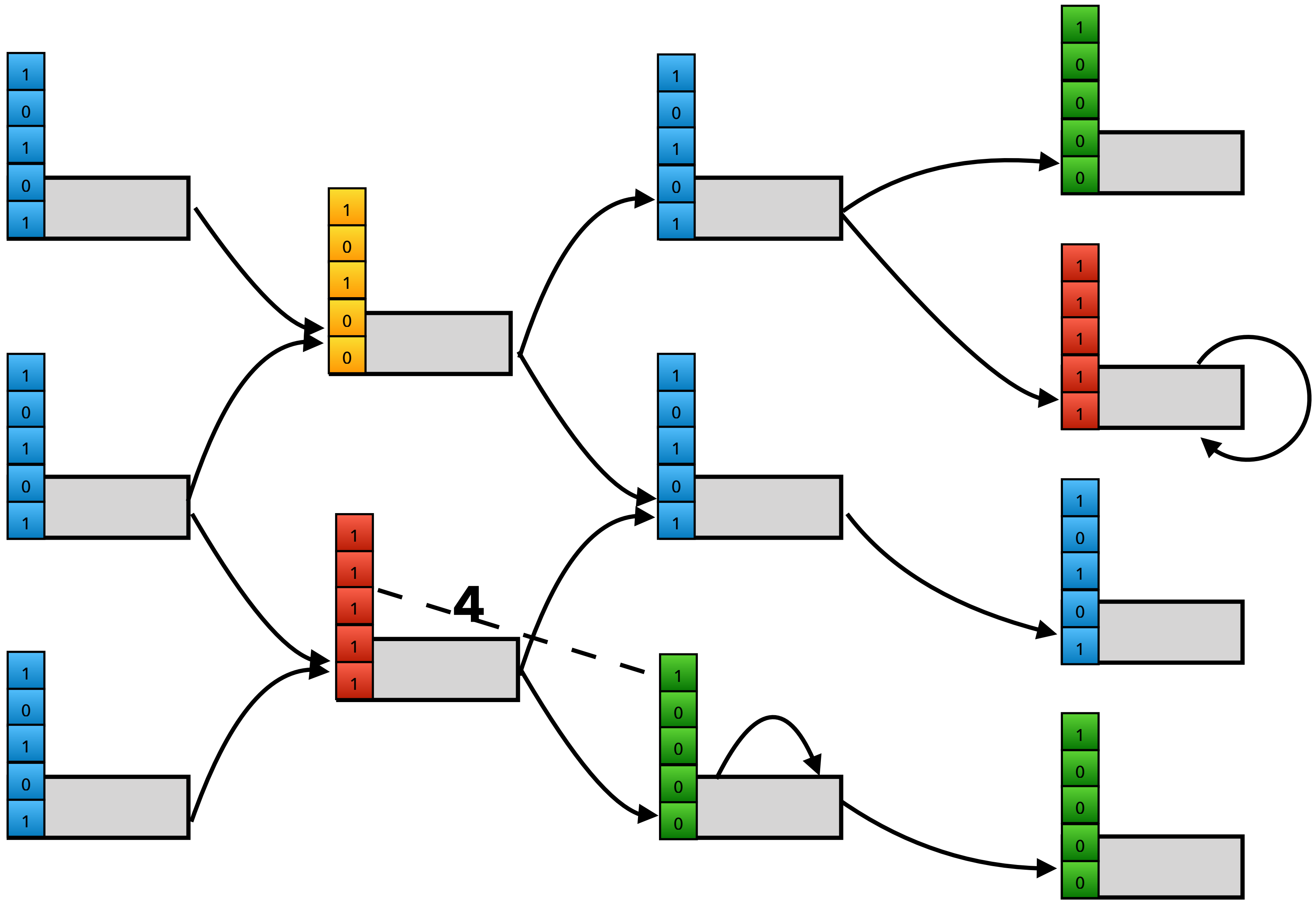
dBG common in genomics. Nodes u,v are k-mers & are *adjacent* if k-1 suffix of u is the same as k-1 prefix of v

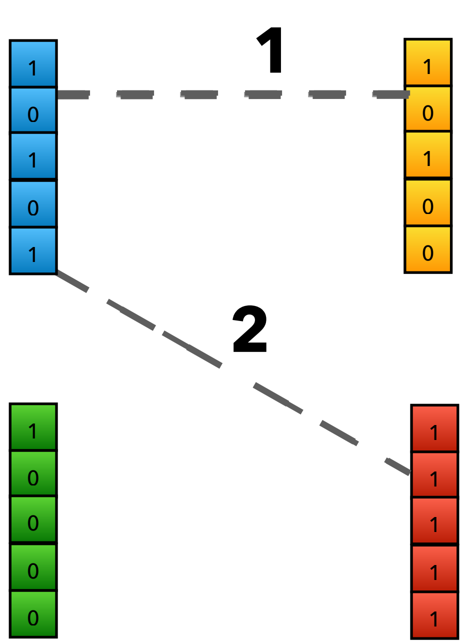




Use the **de Bruin graph** (dBG) as an efficient guide for near-neighbor search in the space of color classes!

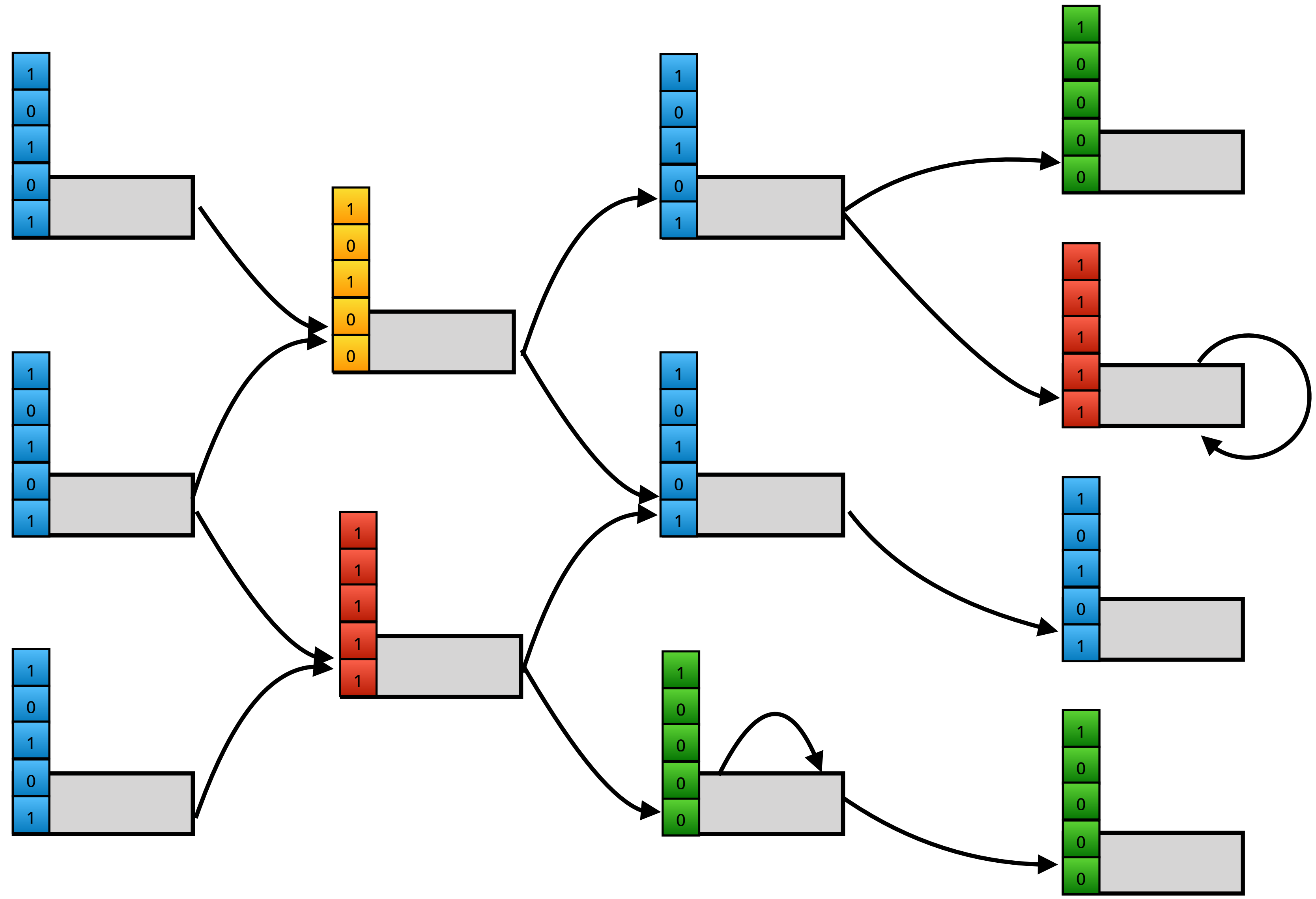
dBG common in genomics. Nodes u,v are k-mers & are *adjacent* if k-1 suffix of u is the same as k-1 prefix of v

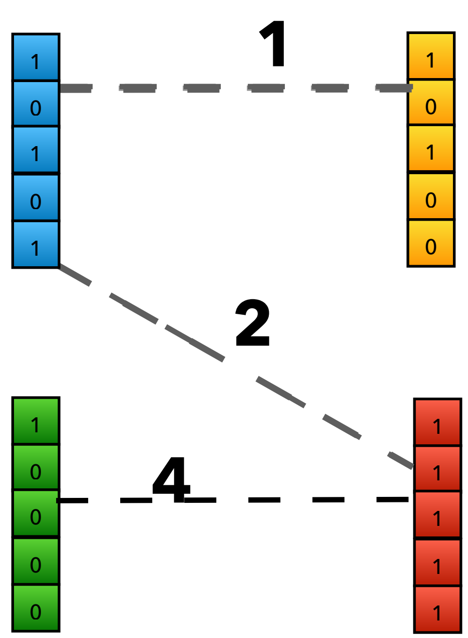




Use the **de Bruin graph** (dBG) as an efficient guide for near-neighbor search in the space of color classes!

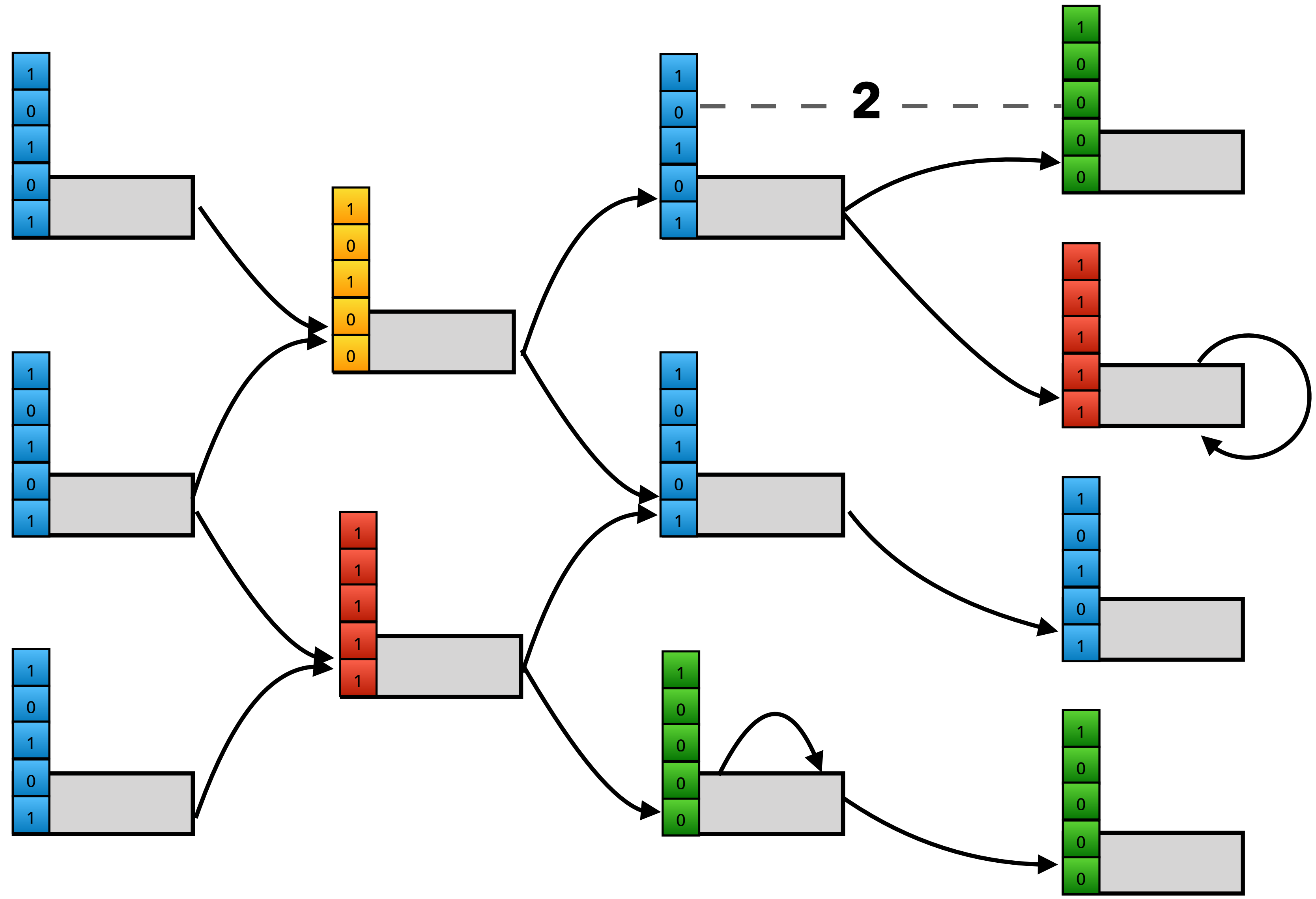
dBG common in genomics. Nodes u,v are k-mers & are *adjacent* if k-1 suffix of u is the same as k-1 prefix of v

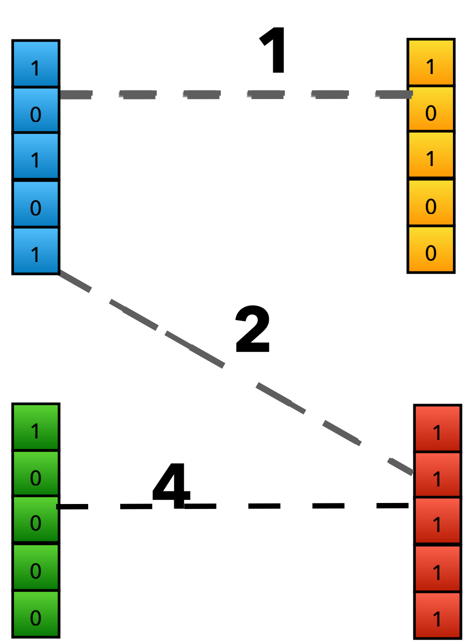




Use the **de Bruin graph** (dBG) as an efficient guide for near-neighbor search in the space of color classes!

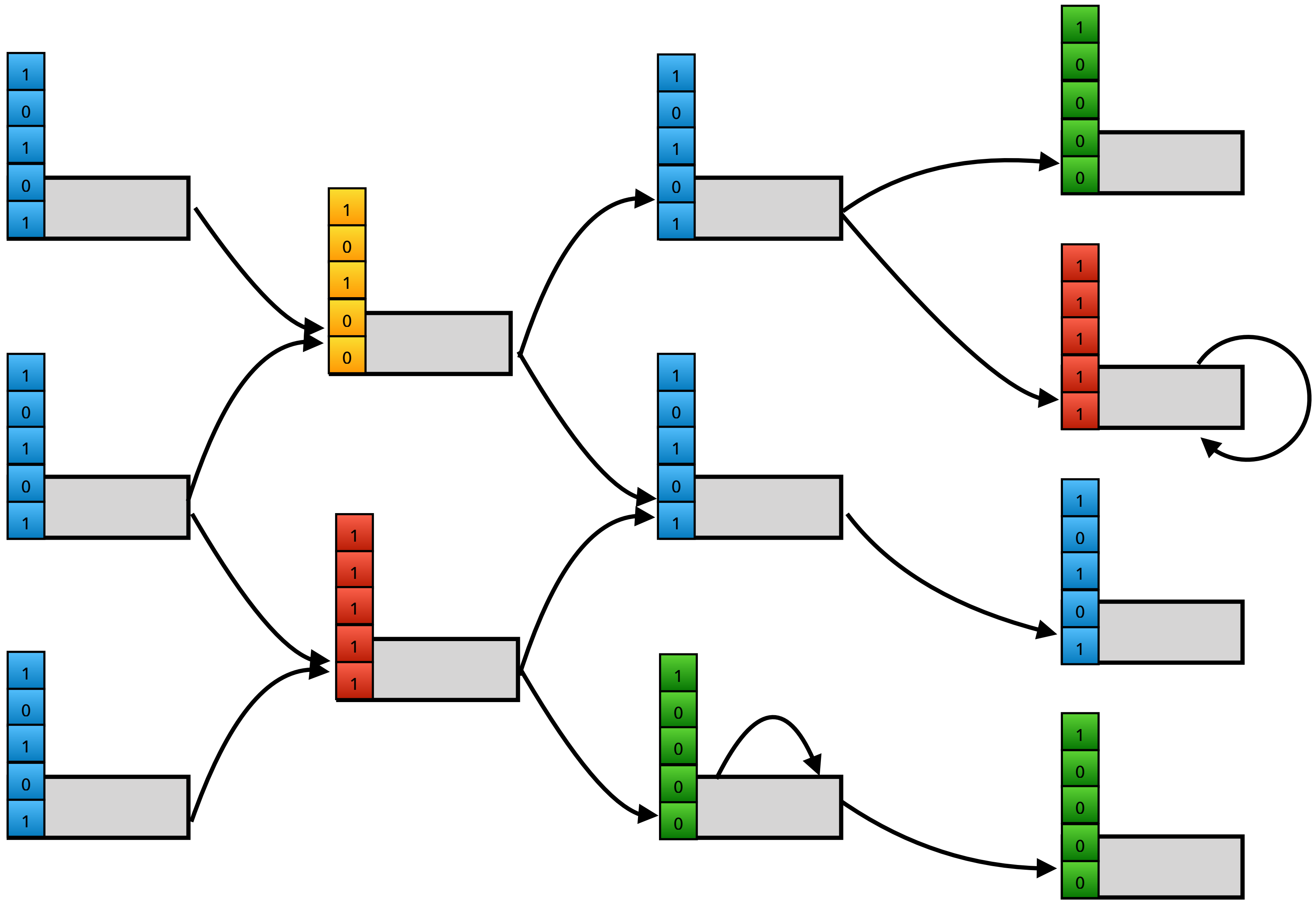
dBG common in genomics. Nodes u,v are k-mers & are *adjacent* if k-1 suffix of u is the same as k-1 prefix of v

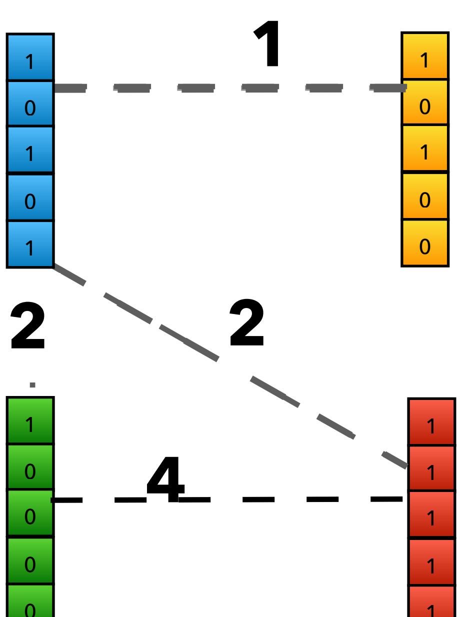




Use the **de Bruin graph** (dBG) as an efficient guide for near-neighbor search in the space of color classes!

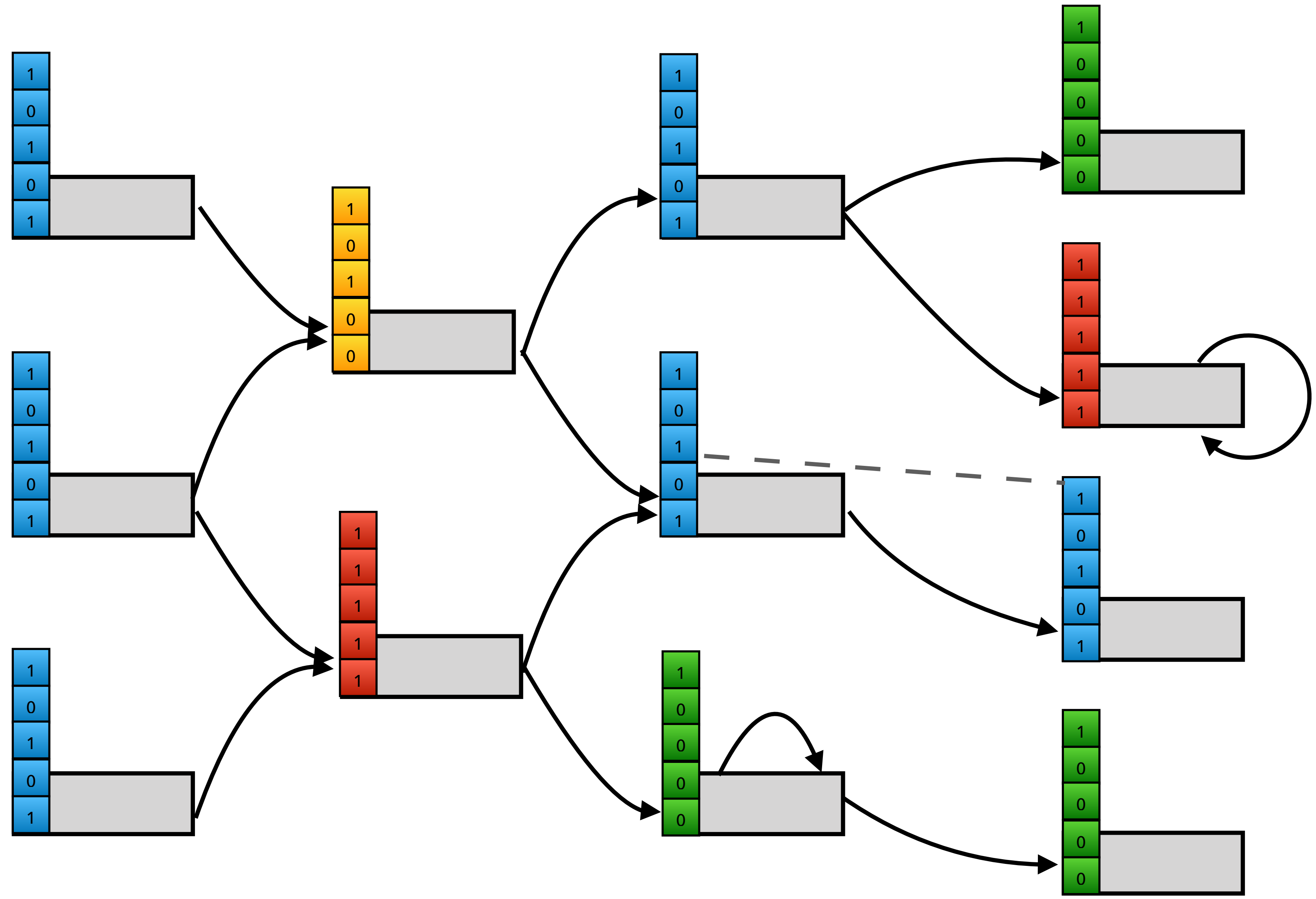
dBG common in genomics. Nodes u,v are k-mers & are *adjacent* if k-1 suffix of u is the same as k-1 prefix of v

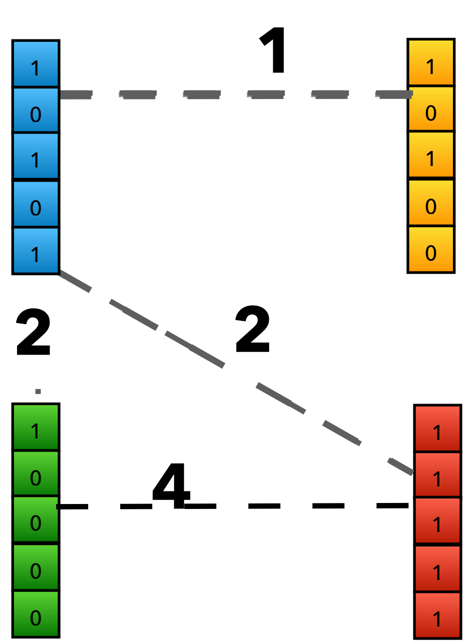




Use the **de Bruin graph** (dBG) as an efficient guide for near-neighbor search in the space of color classes!

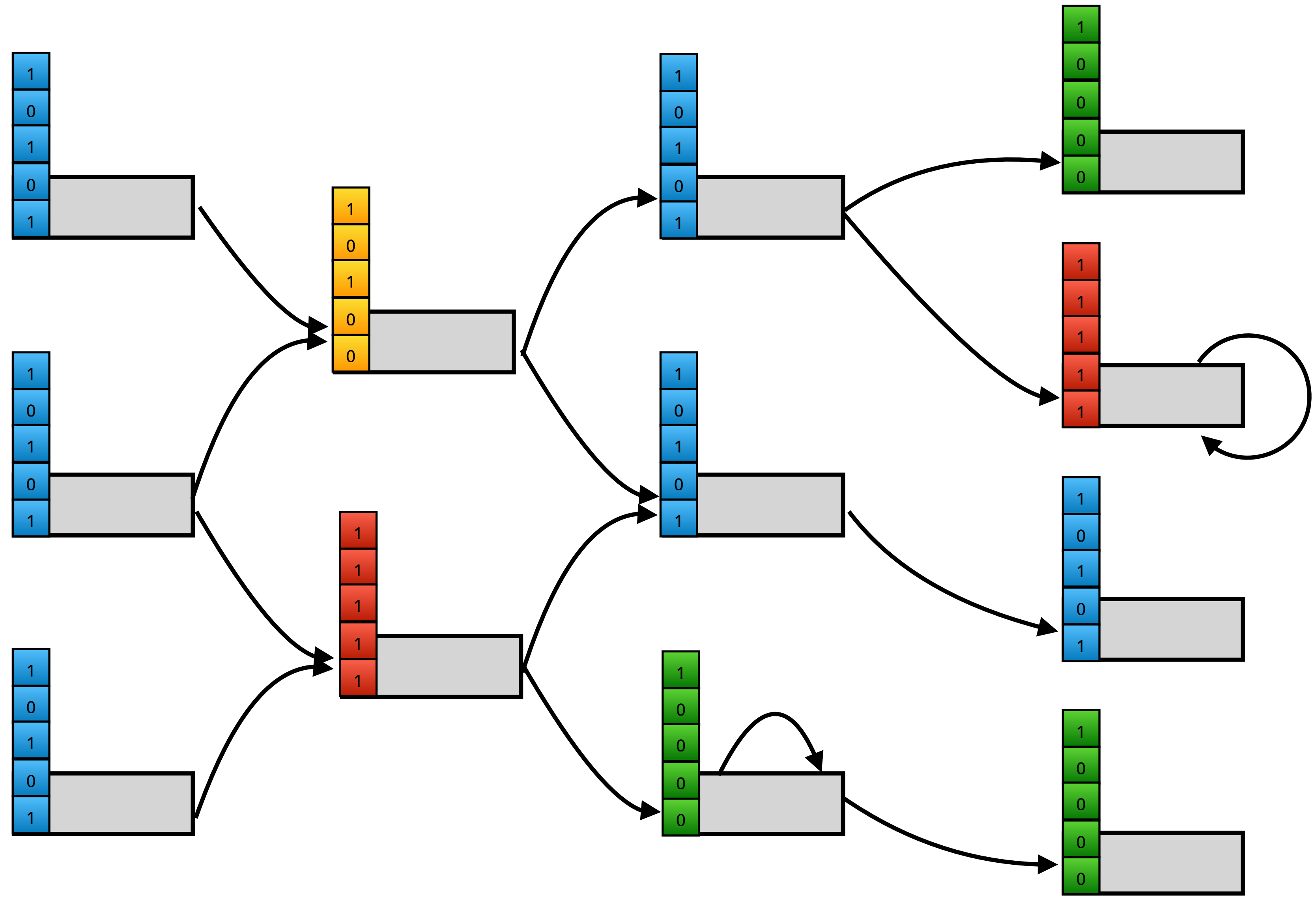
dBG common in genomics. Nodes u,v are k-mers & are *adjacent* if k-1 suffix of u is the same as k-1 prefix of v

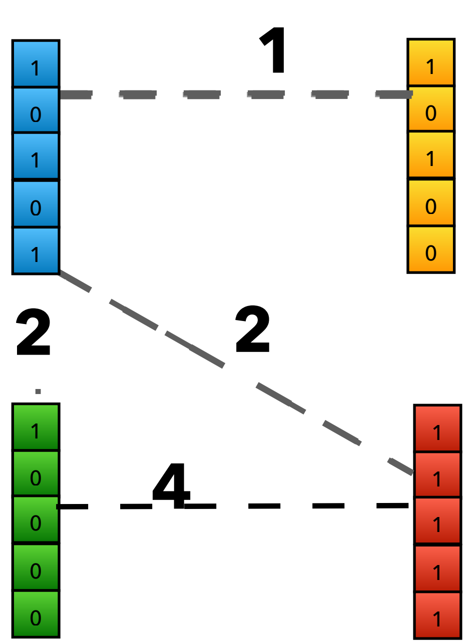




Use the **de Bruin graph** (dBG) as an efficient guide for near-neighbor search in the space of color classes!

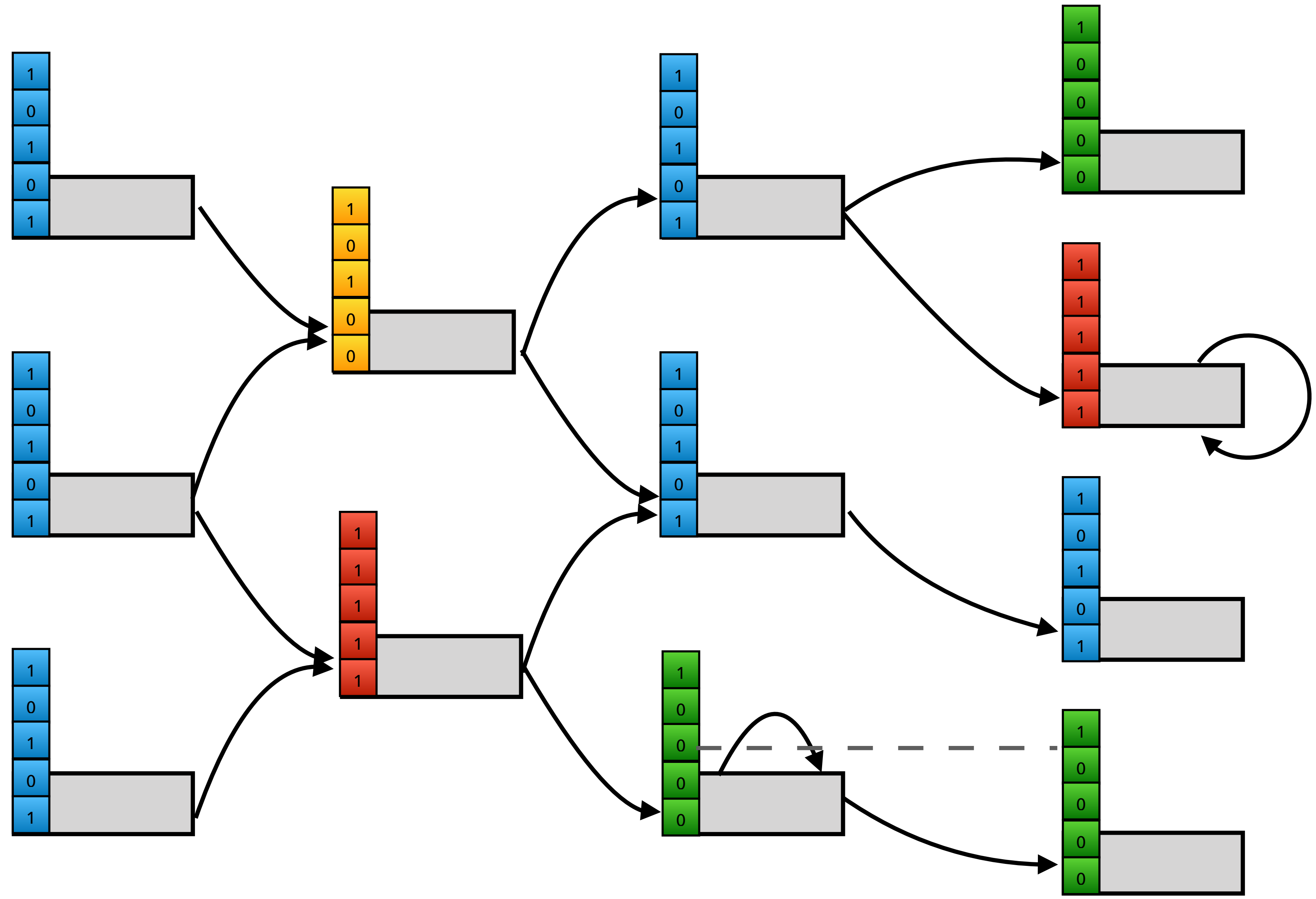
dBG common in genomics. Nodes u,v are k-mers & are *adjacent* if k-1 suffix of u is the same as k-1 prefix of v

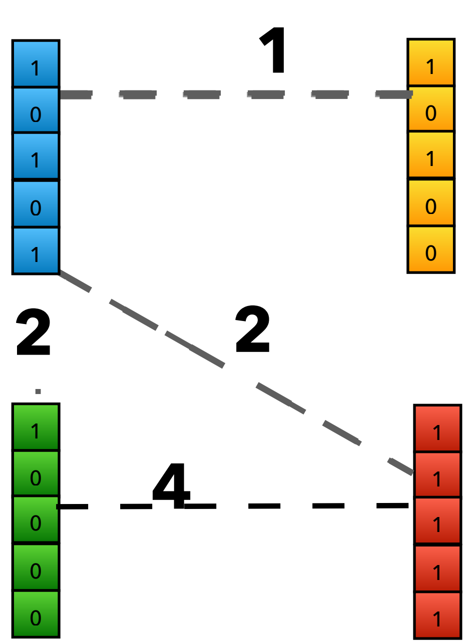




Use the **de Bruin graph** (dBG) as an efficient guide for near-neighbor search in the space of color classes!

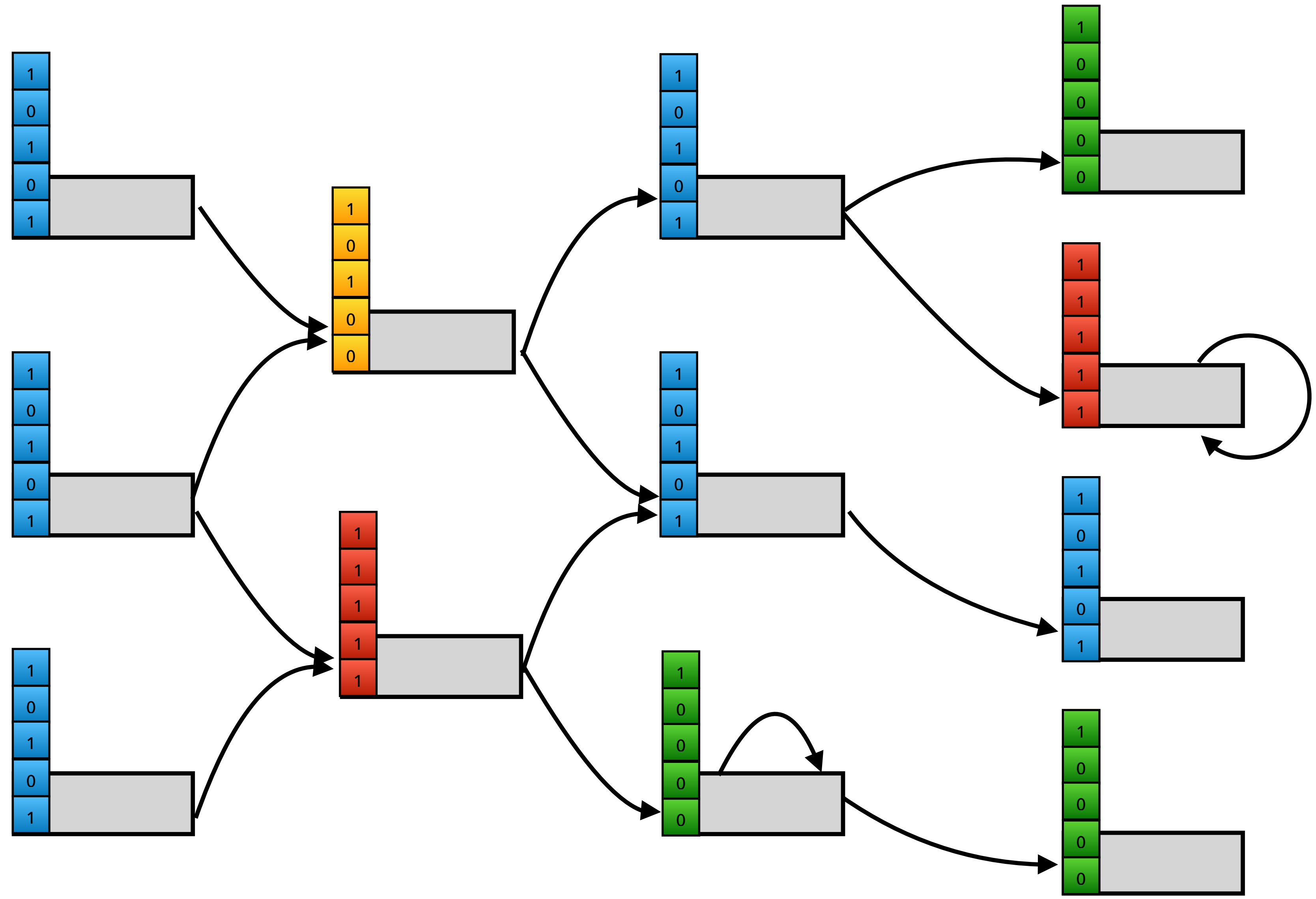
dBG common in genomics. Nodes u,v are k-mers & are *adjacent* if k-1 suffix of u is the same as k-1 prefix of v

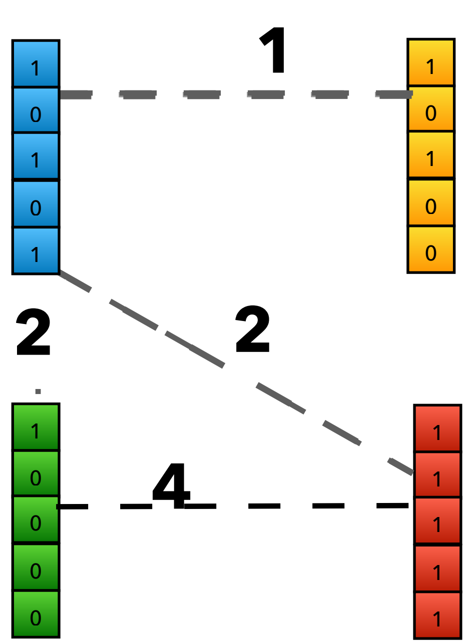




Use the **de Bruin graph** (dBG) as an efficient guide for near-neighbor search in the space of color classes!

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



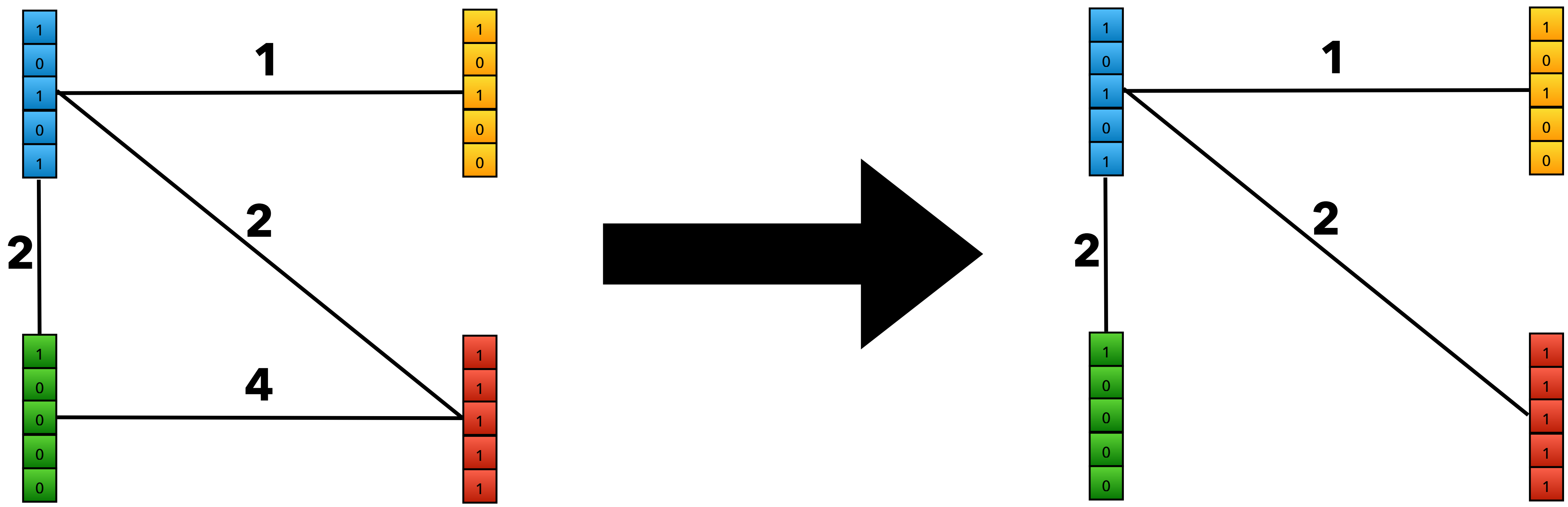


Use the **de Bruin graph** (dBG) as an efficient guide for near-neighbor search in the space of color classes!

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

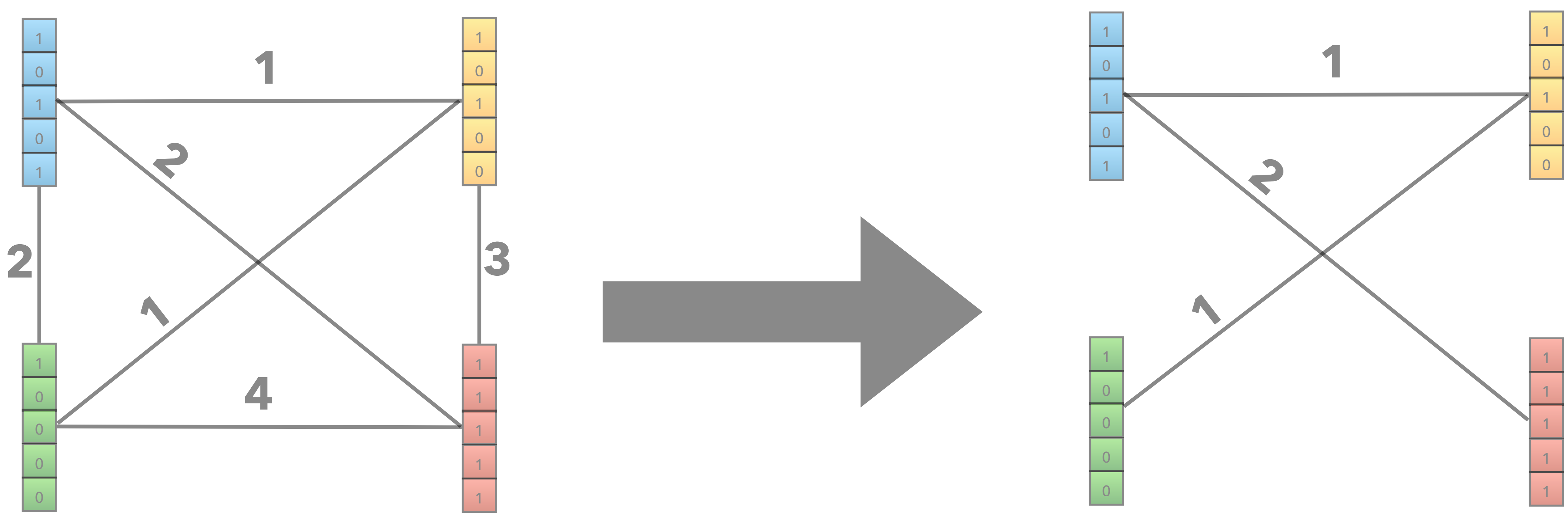
CCG derived from dbG

MST on our Graph

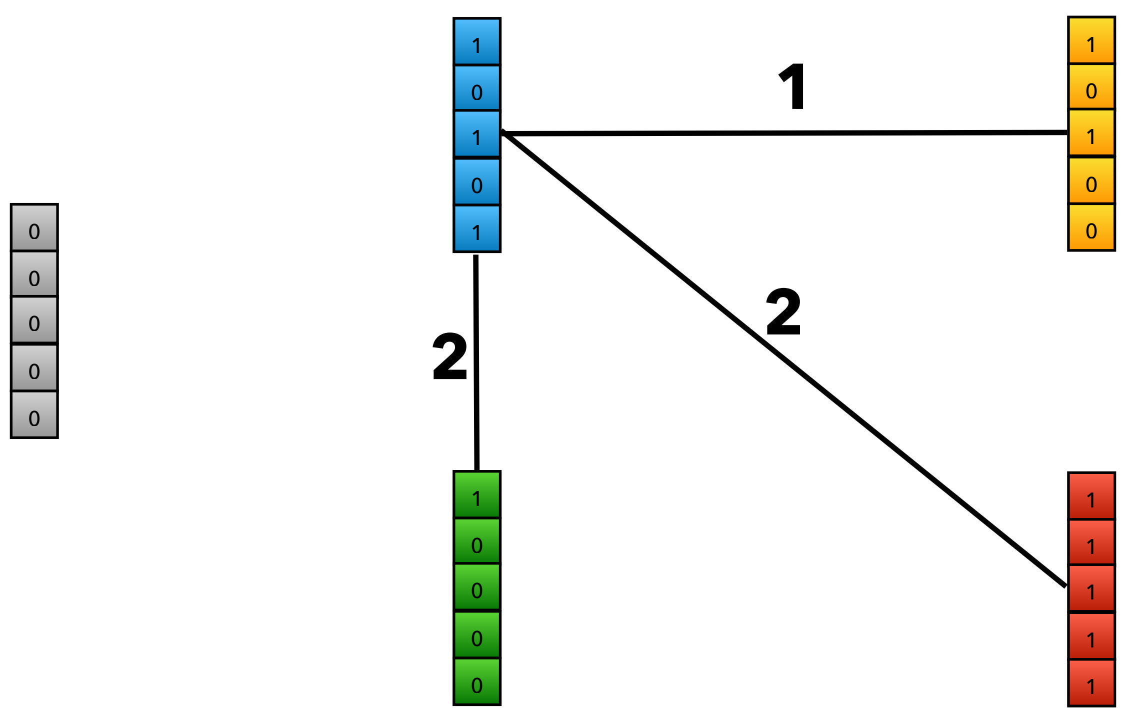


Complete CCG

Optimal MST

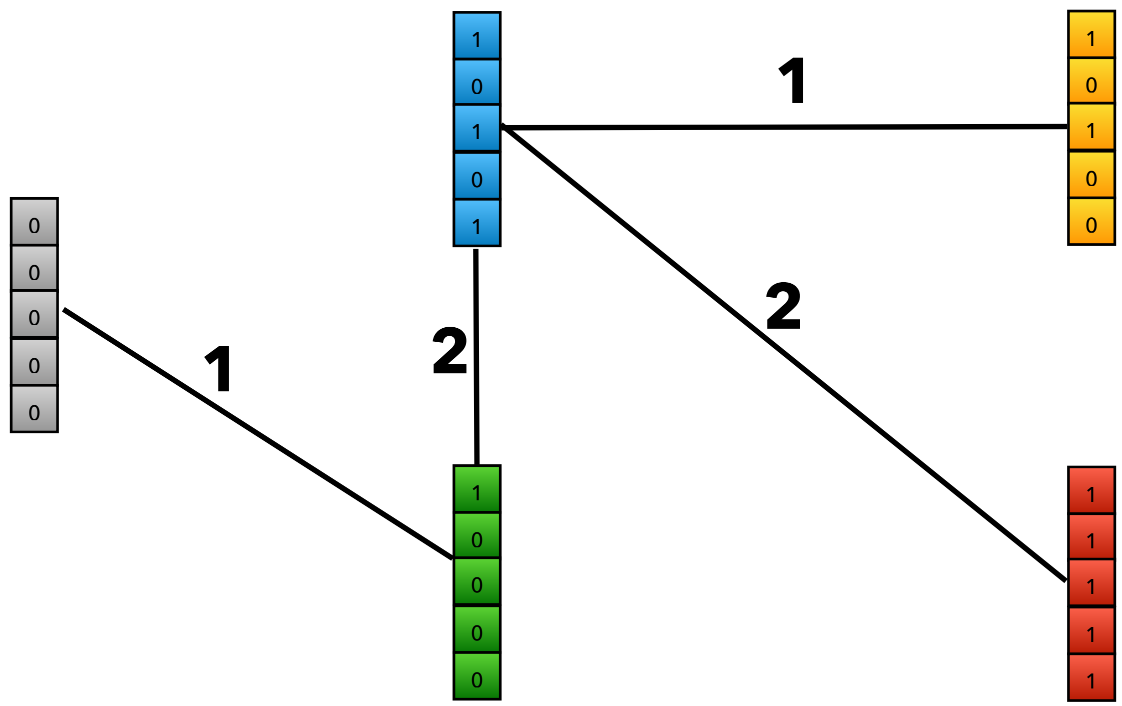


The MST efficiently encodes related color classes



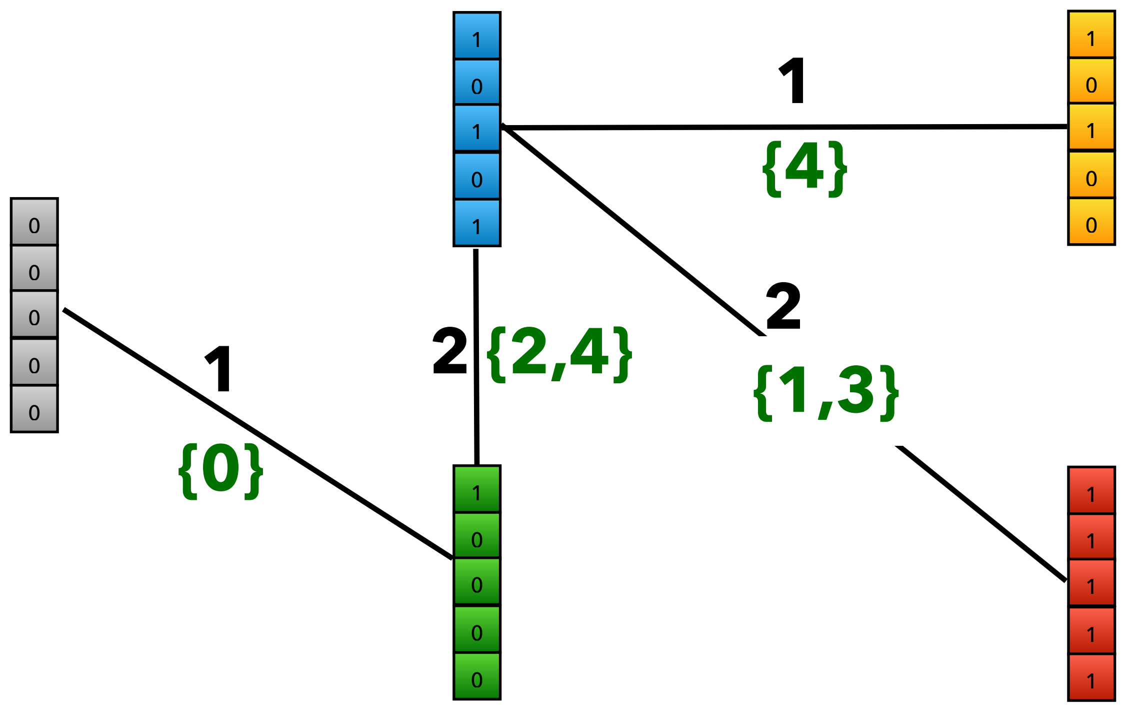
The MST efficiently encodes related color classes

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



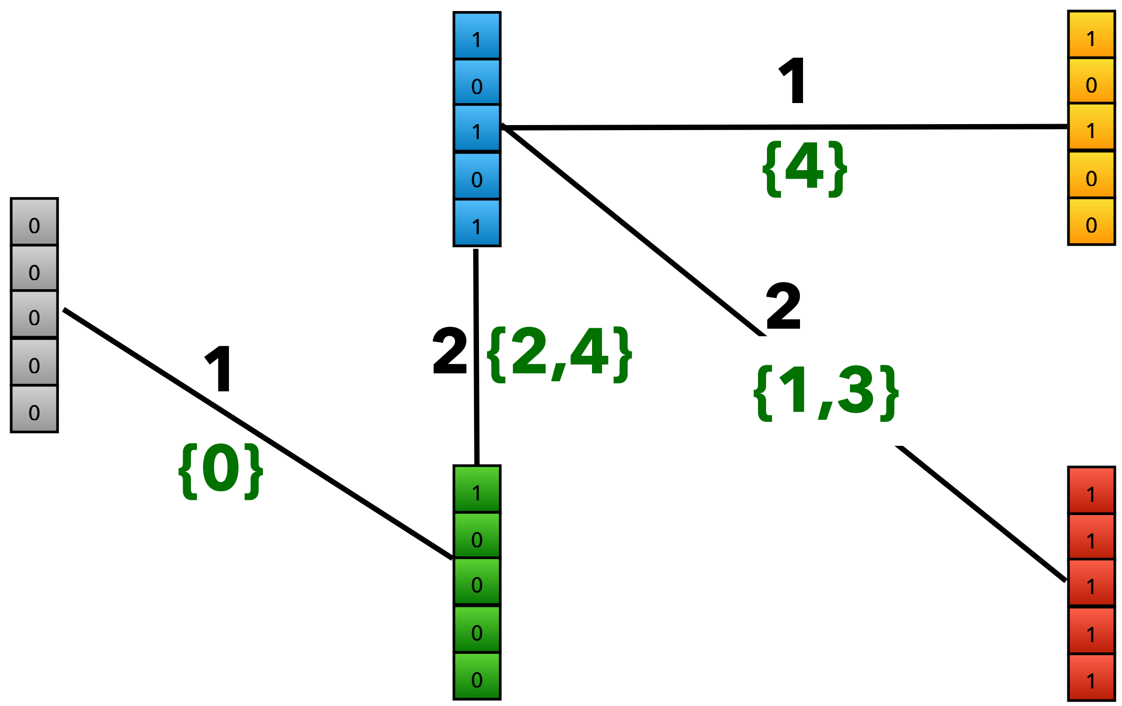
The MST efficiently encodes related color classes

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



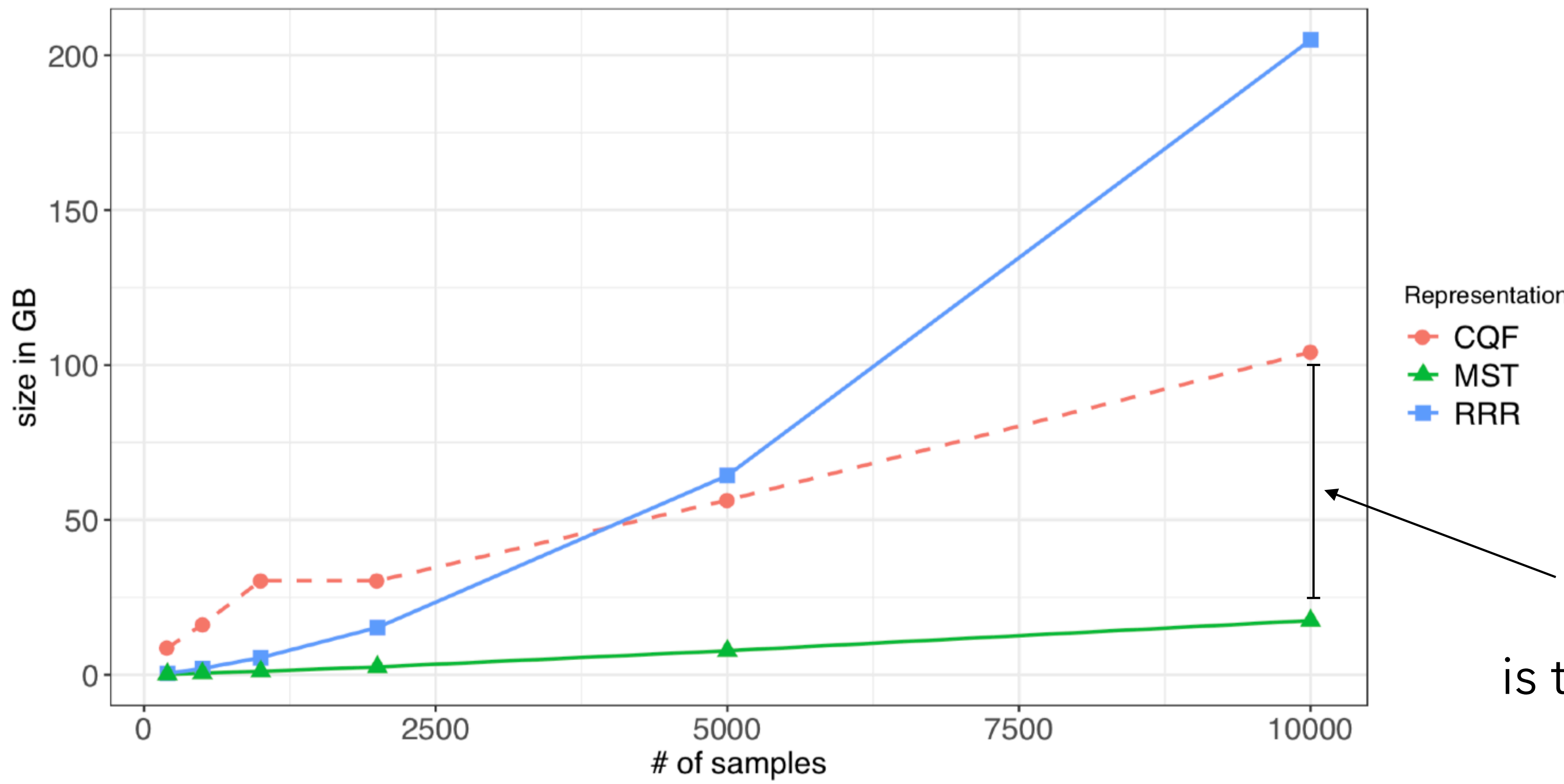
The MST efficiently encodes related color classes

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



now the
k-mer table
is the bottleneck

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

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?

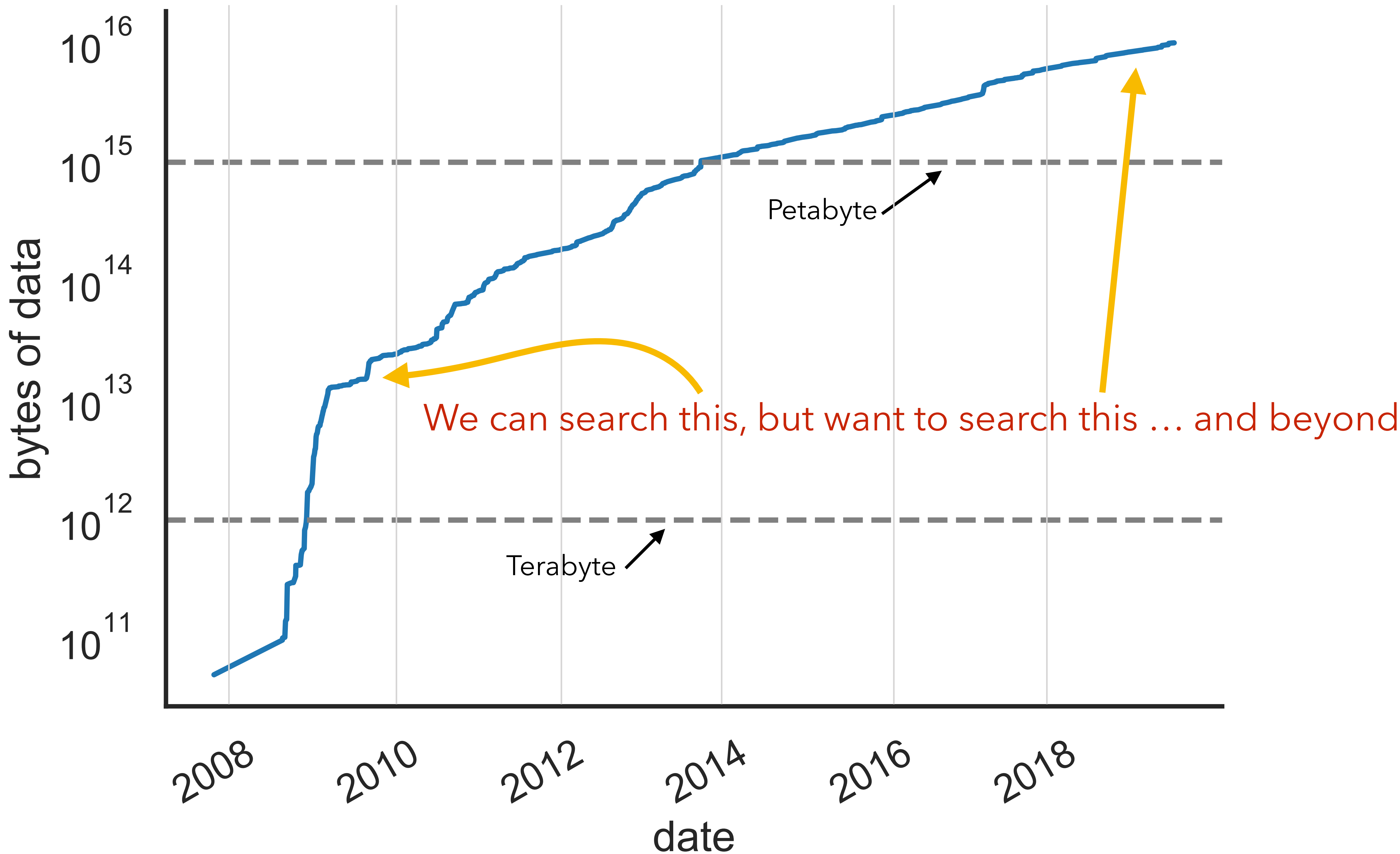
How does MST approach affect query time?

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

A Call To Arms



> "It seems that some essentially new … ideas are here needed"
> – Paul Adrien Maurice Dirac*

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

Data from: https://www.ncbi.nlm.nih.gov/Traces/sra/sra_stat.cgi