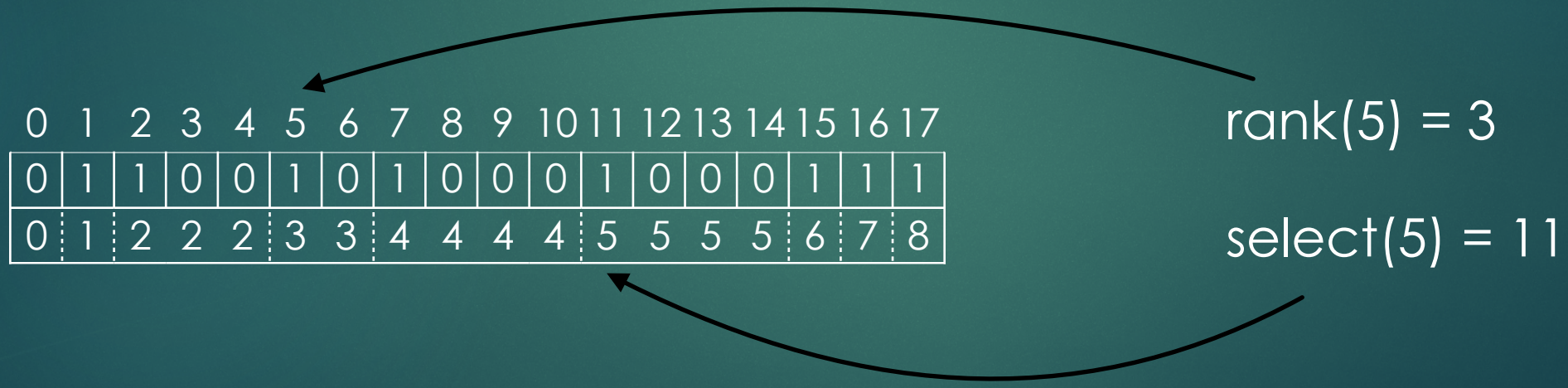
The colored de Bruijn Graph as a Sequence Index

Unless otherwise noted, slides courtesy of Fatemeh Almodaresi

Succinct Data Structures & Operations[8]

- Represent an object in a space close to the theoretic lower bound
- Provide specific constant time operations



Main Topics

- Rainbowfish
  - Succinct colored de Bruijn graph representation
- Pufferfish
  - Efficient Compacted de Bruijn graph indexing representation
  - Dense and Sparse

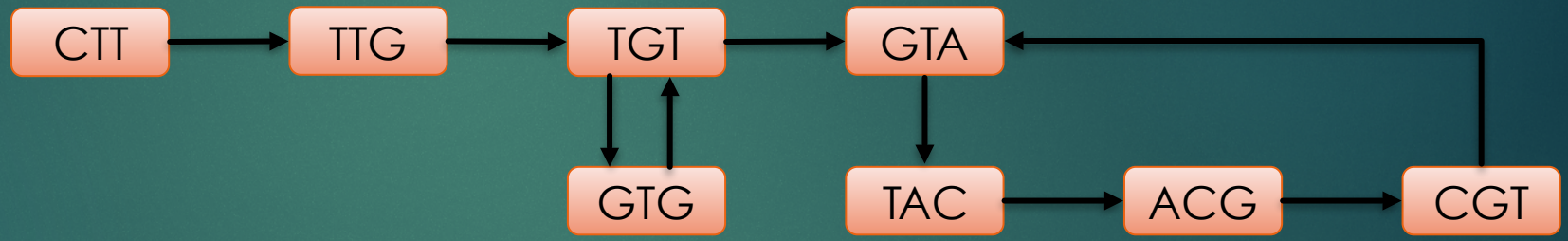
de Bruijn Graph[1-3]

Edge-centric Variant

K = 4

Sample:

CTTGTGTACGTA



A directed graph to represent a (set of) sequence(s) in terms of their k-mer components.

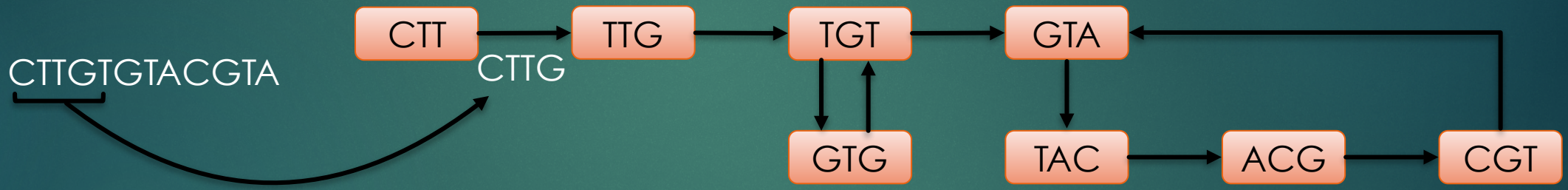
K-mer : unique substring of size k

de Bruijn Graph[1-3]

Edge-centric Variant

K = 4

Sample:

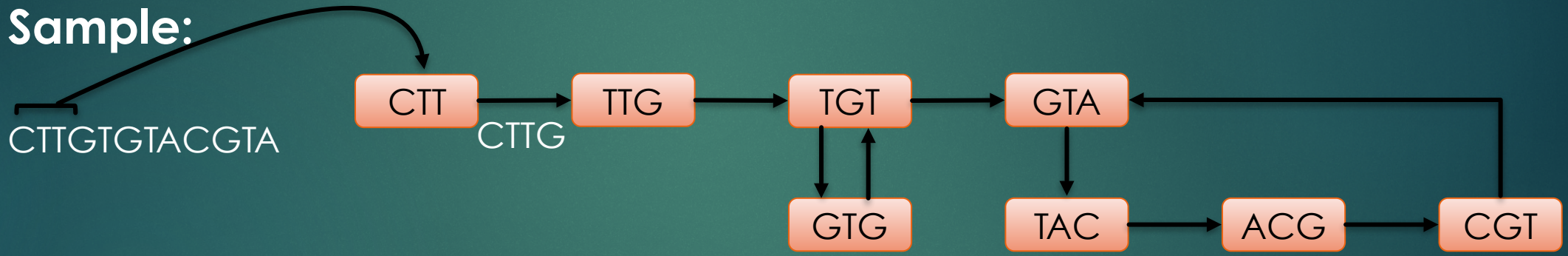


de Bruijn Graph[1-3]

Edge-centric Variant

K = 4

Sample:



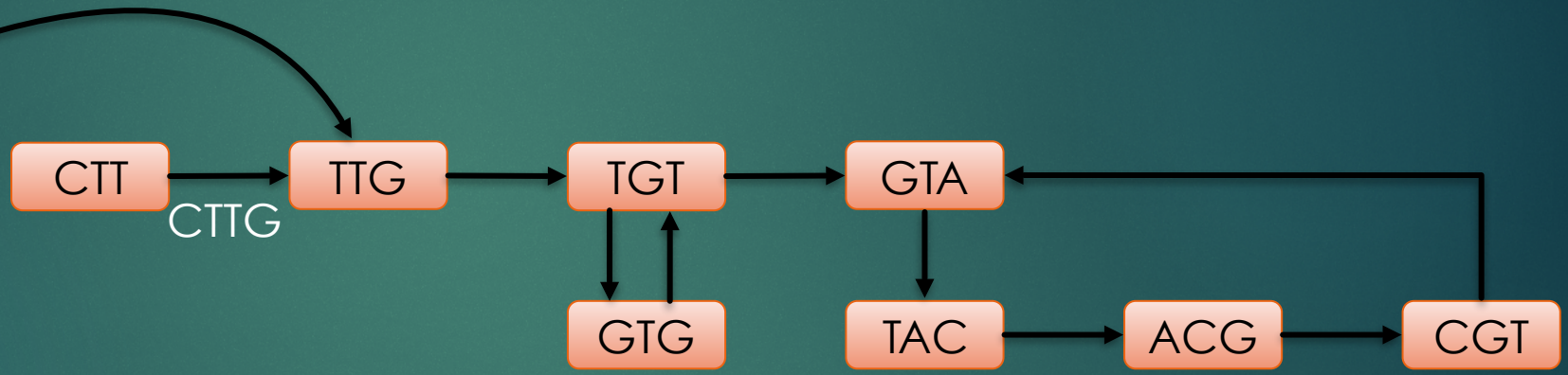
de Bruijn Graph[1-3]

Edge-centric Variant

K = 4

Sample:

CTTGTGTACGTA



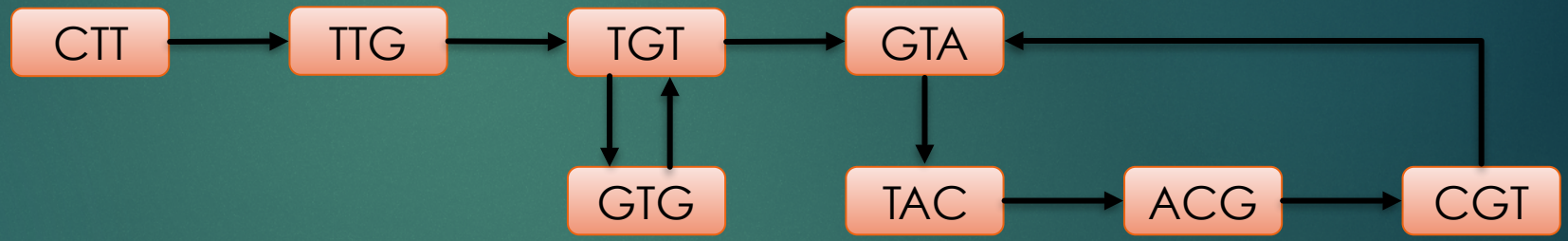
de Bruijn Graph[1-3]

Edge-centric Variant

K = 4

Sample:

CTTGTGTACGTA



- Genome/Transcriptome Assembly
- Sequence Indexing

Rainbowfish

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

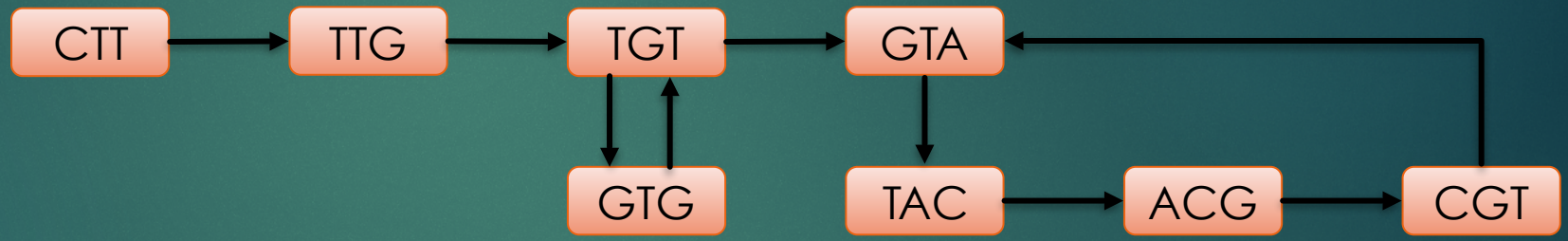
WABI 2017

Colored de Bruijn Graph

K = 4

Sample:

CTTGTGTACGTA

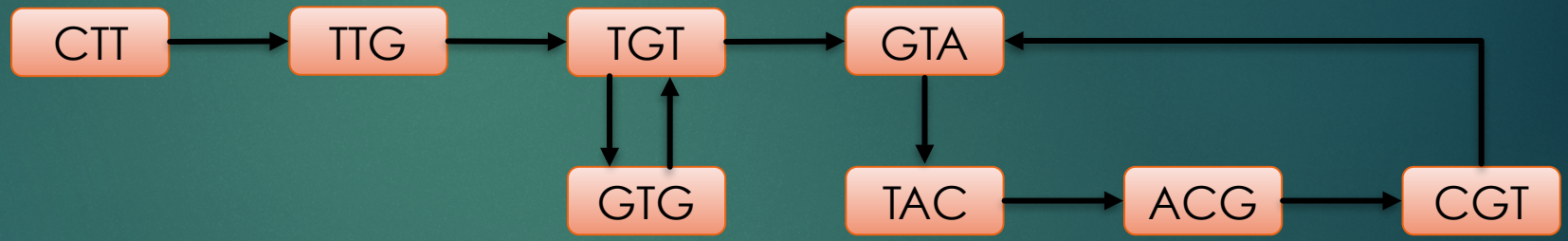


Colored de Bruijn Graph

K = 4

Sample:

- CTTGTGTACGTA

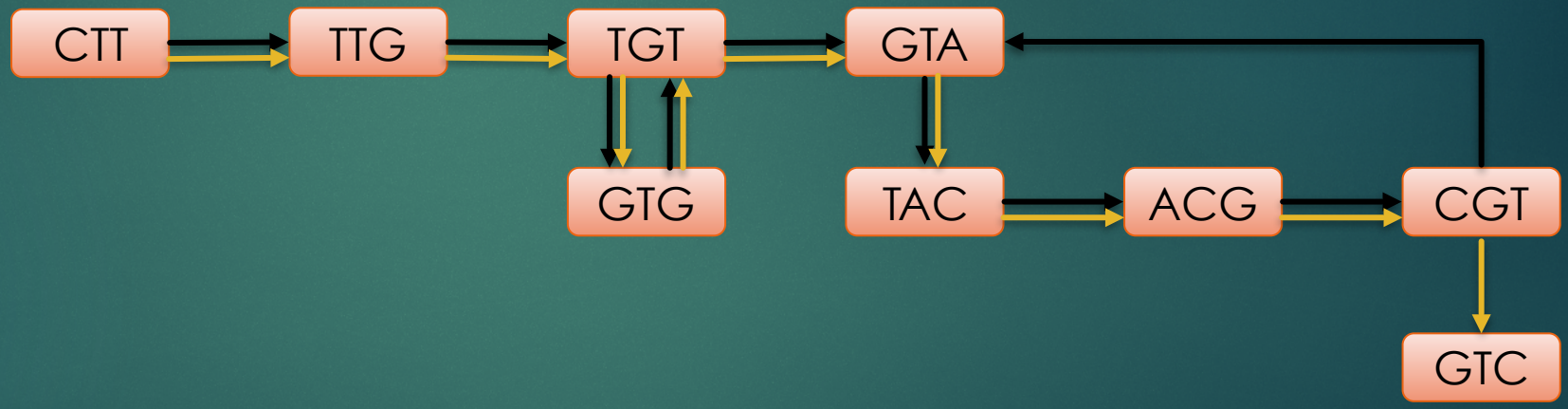


Colored de Bruijn Graph

K = 4

Samples:

- CTTGTGTACGTA
- CTTGTGTACGTC

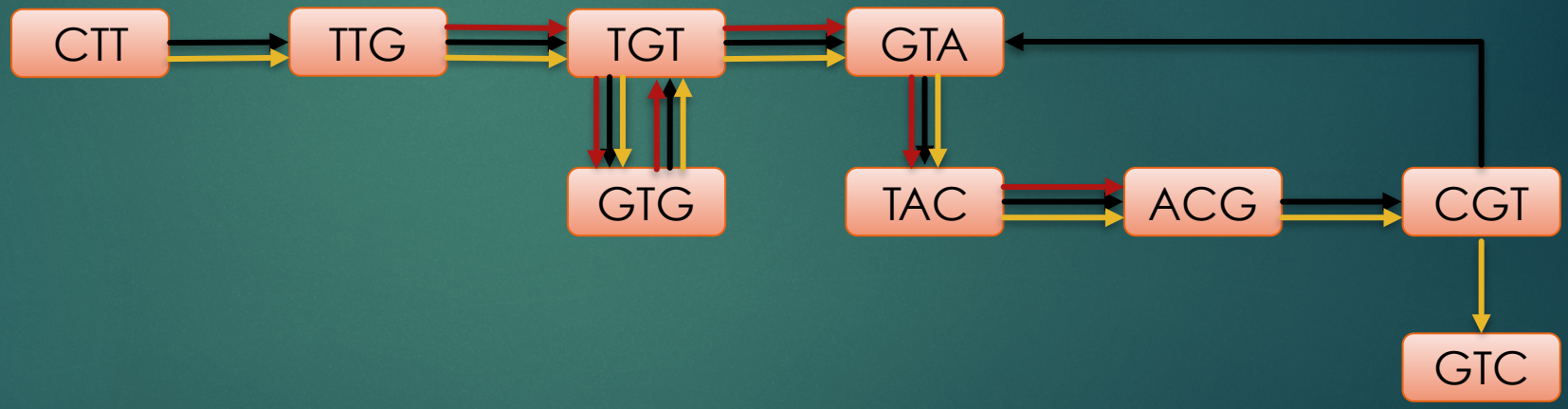


Colored de Bruijn Graph

K = 4

Samples:

- CTTGTGTACGTA
- CTTGTGTACGTC
- TTGTGTACG



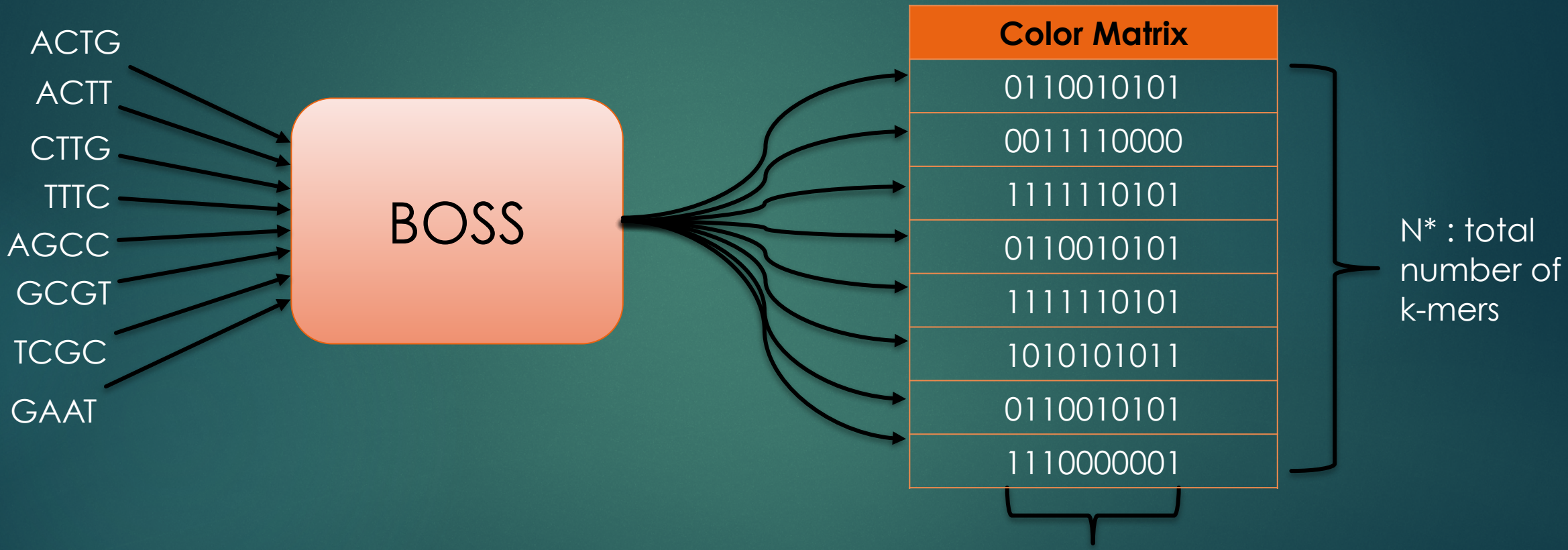
Colored de Bruijn Graphs (cdBg)

- Cortex[4]
  - Introduces cdBgs
  - Hash-table based representation for dBg
  - K-mer frequency for each color
  - 5 bytes per <color,k-mer>
  - Optimized for speed
  - Considerably large in space

Colored de Bruijn Graphs

- VARI[7]
  - In conjunction with BOSS[5]. BOSS is:
    - an efficient representation of dBgs
    - uses succinct data structures and rank and select operations to navigate through dBg
    - Returns an index for each k-mer
  - Represent Colors in a Color Matrix
  - Compress Color Matrix using bit-encoding schemes (Elias-fano)

BOSS → VARI



* Details of the BOSS representation require additional dummy k-mers

VARI

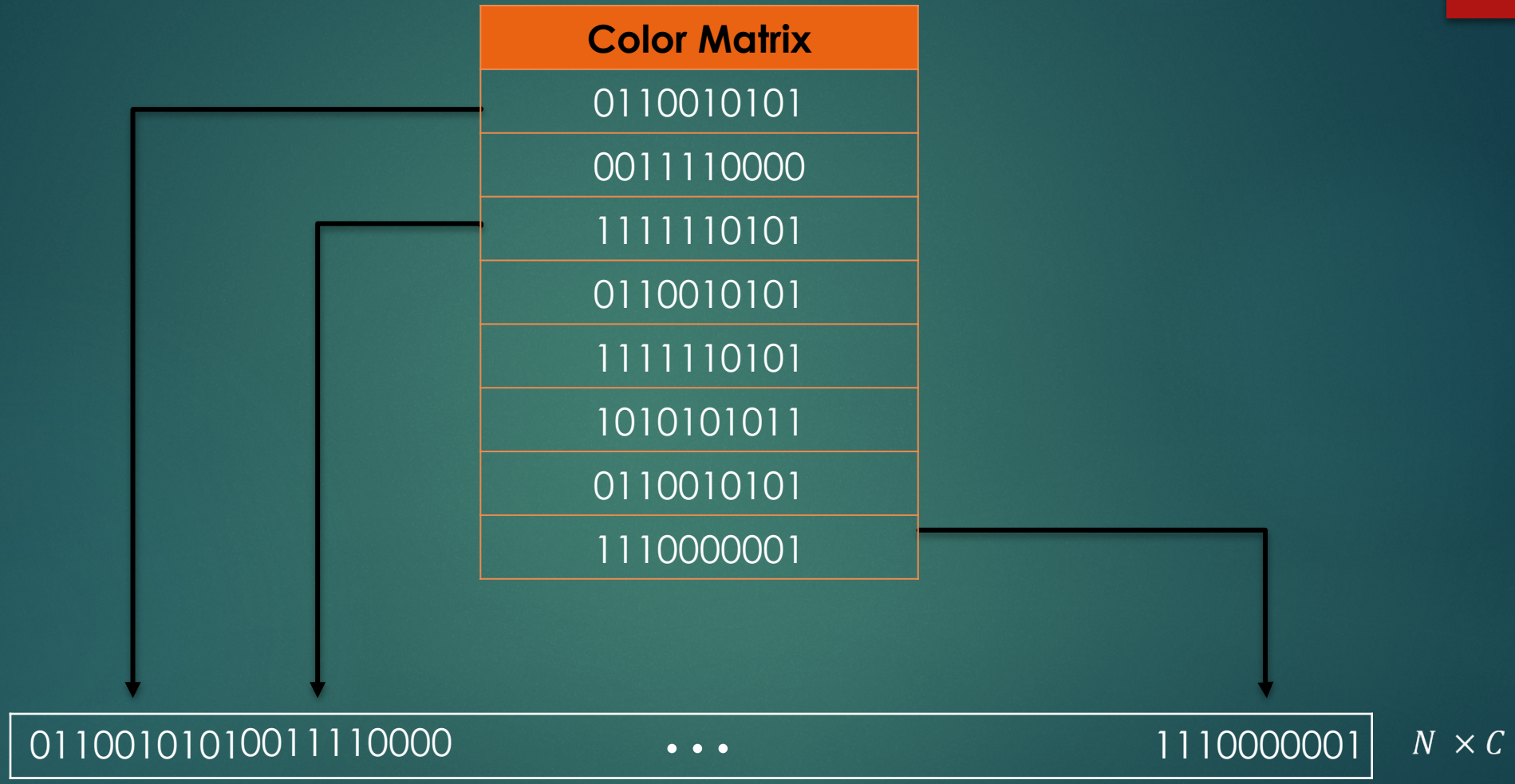
Color Matrix
0110010101
0011110000
1111110101
0110010101
1111110101
1010101011
0110010101
1110000001

VARI

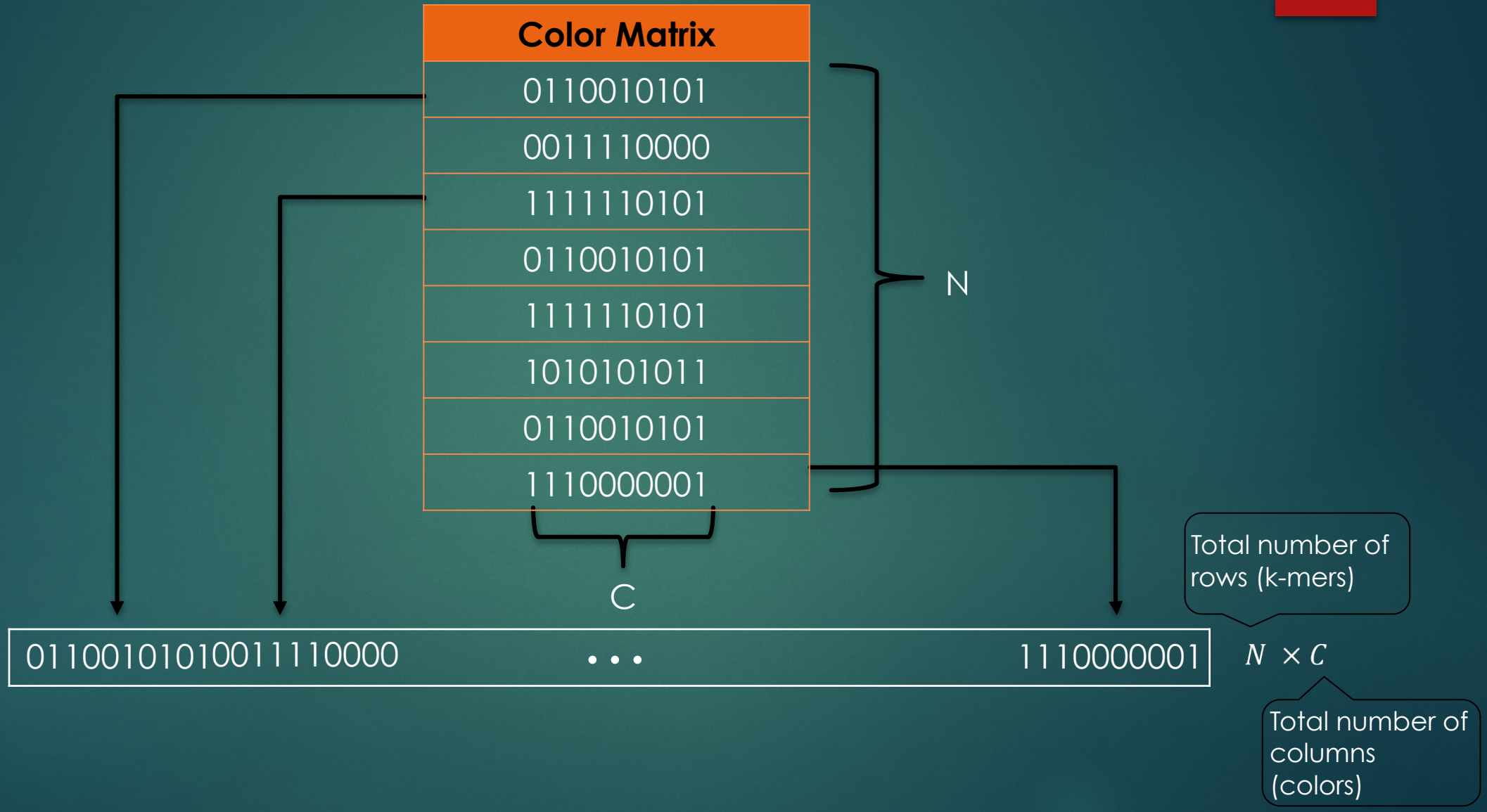
Color Matrix
0110010101
0011110000
1111110101
0110010101
1111110101
1010101011
0110010101
1110000001

0110010101 0011110000 • • • 1110000001

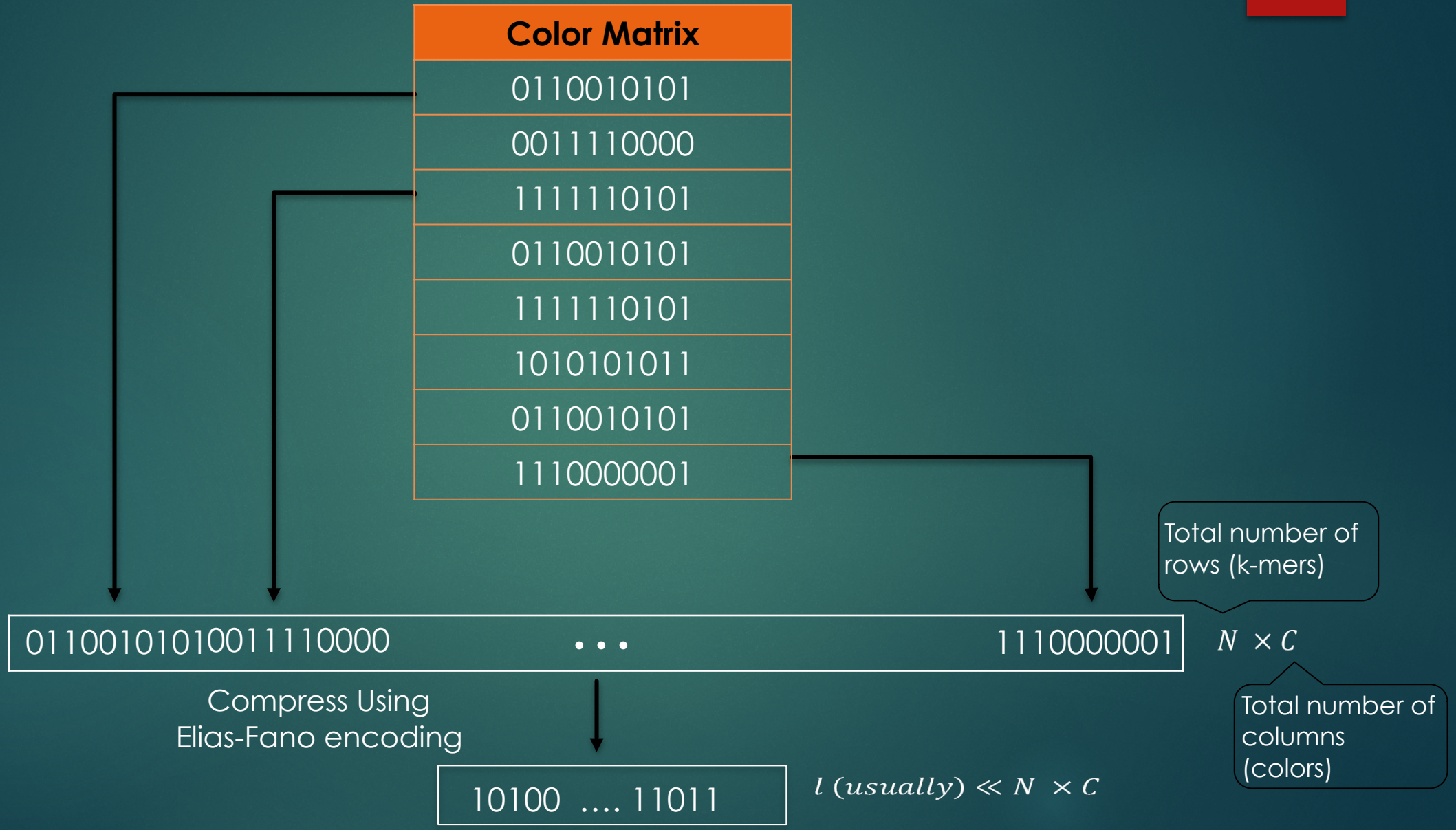
VARI


Color Matrix
0110010101
0011110000
1111110101
0110010101
1111110101
1010101011
0110010101
1110000001
0110010101001111000 0
• • •
1110000001
N × C

VARI


Color Matrix
0110010101
0011110000
1111110101
0110010101
1111110101
1010101011
0110010101
1110000001
N
C
Total number of rows (k-mers)
0110010101001111000 • • • 1110000001
$N \times C$
Total number of columns (colors)

VARI


Color Matrix
0110010101
0011110000
1111110101
0110010101
1111110101
1010101011
0110010101
1110000001
Total number of rows (k-mers)
01100101010011110000
• • •
1110000001
$N \times C$
Total number of columns (colors)
Compress Using Elias-Fano encoding
10100 11011
$l\ (usually) \ll N \times C$

VARI results compared to Cortex

Datasets	No. of k-mers	Colors	Cortex	VARI
Plant (k=32)	1,709,427,823	4	100.93 GB	3.53 GB (sdBG=0.89 GB, sC=1.95 GB)
E. Coli (k=32)	158,501,209	3,765	NA	42.17 GB (sdBG=0.09 GB, sC=38.35 GB)
Beef Safety (k=32)	40,995,794,366	88	NA	245.54 GB (sdBG=27.08 GB, sC=200.34 GB)

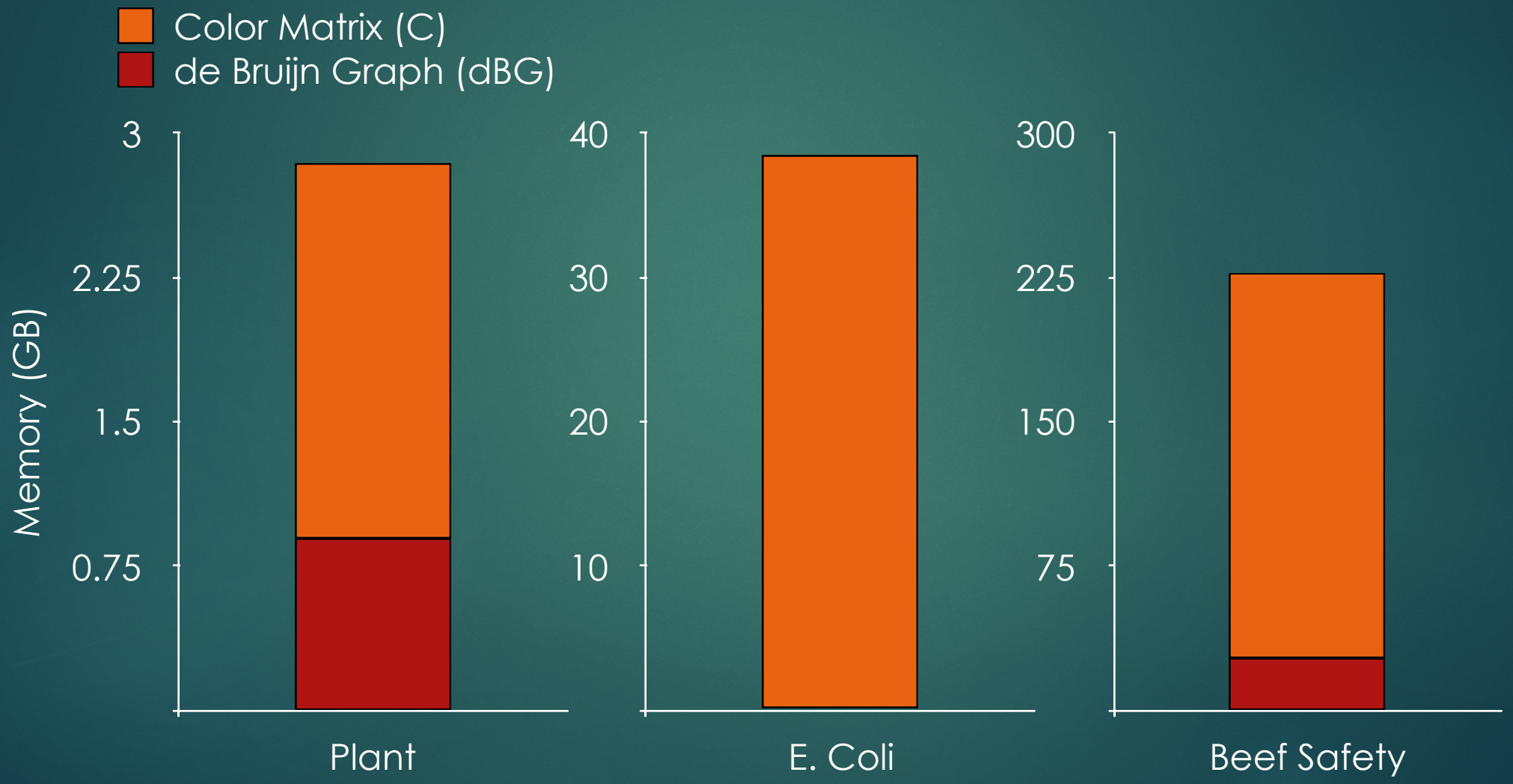
VARI results compared to Cortex

Datasets	No. of k-mers	Colors	Cortex	VARI
Plant (k=32)	1,709,427,823	4	100.93 GB	3.53 GB (sdBG=0.89 GB, sC=1.95 GB)
E. Coli (k=32)	158,501,209	3,765	NA	42.17 GB (sdBG=0.09 GB, sC=38.35 GB)
Beef Safety (k=32)	40,995,794,366	88	NA	245.54 GB (sdBG=27.08 GB, sC=200.34 GB)

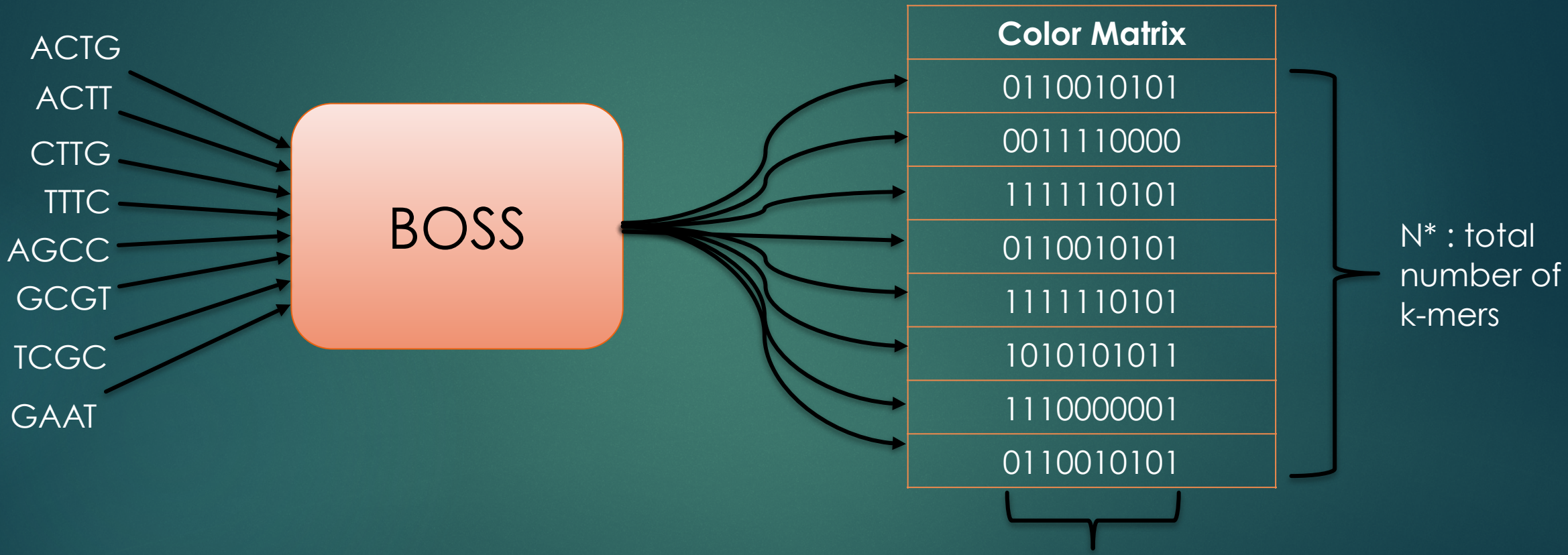
VARI results compared to Cortex

Datasets	No. of k-mers	Colors	Cortex	VARI
Plant (k=32)	1,709,427,823	4	100.93 GB	3.53 GB (sdBG=0.89 GB, sC=1.95 GB)
E. Coli (k=32)	158,501,209	3,765	NA	42.17 GB (sdBG=0.09 GB, sC=38.35 GB)
Beef Safety (k=32)	40,995,794,366	88	NA	245.54 GB (sdBG=27.08 GB, sC=200.34 GB)

VARI results compared to Cortex

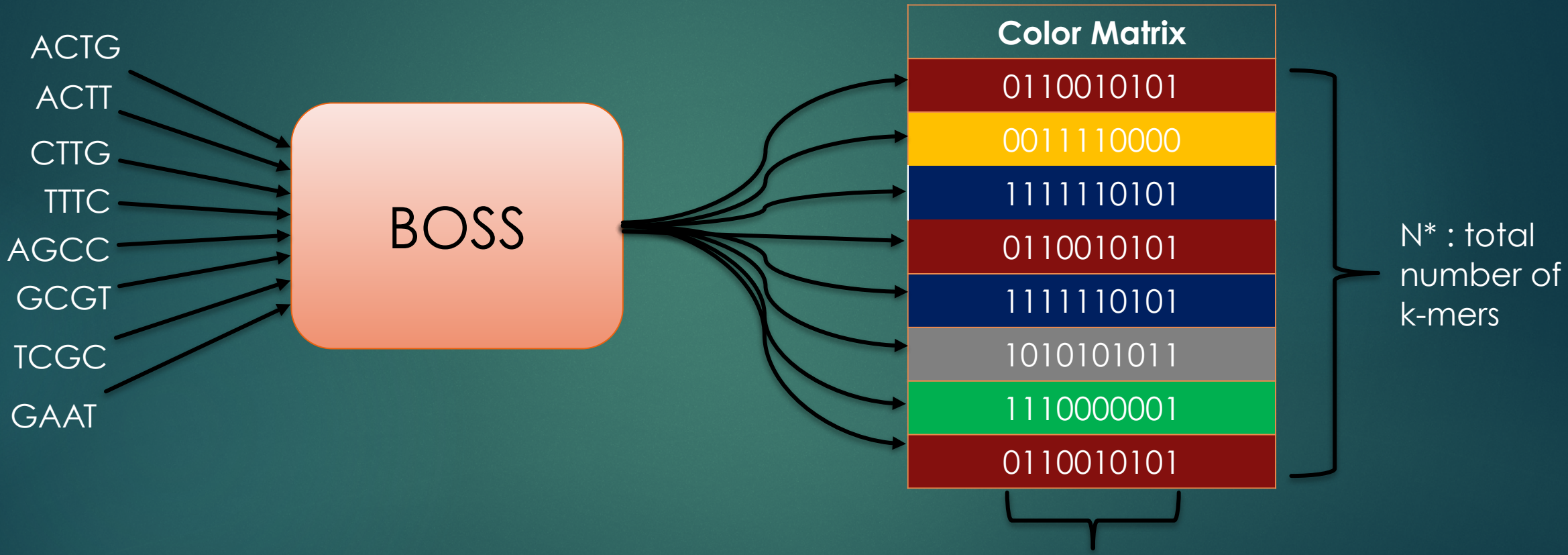


BOSS + VARI



* Details of the BOSS representation require additional dummy k-mers

BOSS + VARI

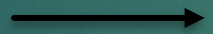


* Details of the BOSS representation require additional dummy k-mers

VARI to Rainbowfish

Equivalence Table

Color Matrix
0110010101
0011110000
1111110101
0110010101
1111110101
1010101011
1110000001
0110010101



Color Matrix
0
1
2
0
2
3
4
0

*

Label	Equivalence Class
0	0110010101
1	0011110000
2	1111110101
3	1010101011
4	1110000001

* This idea was briefly discussed in BFT[6] paper

Rainbowfish

Equivalence Table

Label	Equivalence Class
0	0110010101
1	0011110000
2	1111110101
3	1010101011
4	1110000001

+

Color Matrix
0
1
2
0
2
3
4
0

Rainbowfish

Equivalence Table

Label	Equivalence Class
0	0110010101
1	0011110000
2	1111110101
3	1010101011
4	1110000001

Color Matrix
0
1
2
0
2
3
4
0

Equivalence Bitvector

Label Bitvector

Rainbowfish

Equivalence Table

Label	Equivalence Class
0	0110010101
1	0011110000
2	1111110101
3	1010101011
4	1110000001

E = 5

C : total number of colors

Color Matrix
0
1
2
0
2
3
4
0

N : total number of k-mers

$l = E \times C$

0110010101	0011110000	1111110101	1010101011	1110000001

Equivalence Bitvector

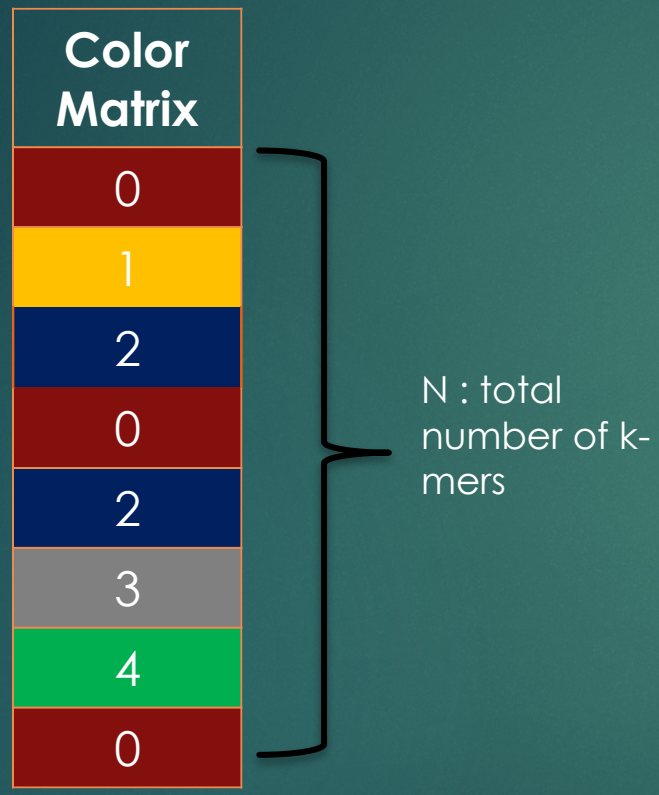
$l = N \times \log_2 E \leq N \times C$

000	001	010	000	010	011	100	000

Label Bitvector

Majority of the space

Rainbowfish
Equivalence Table



Datasets	$\frac{l}{N \times C}$
E. Coli (10)	0.9
E. Coli (1000)	0.02
E. Coli (5598)	0.004
Plant (4)	1
Beef Safety (88)	< 0.34
Human Txme (95,146)	< 0.0002

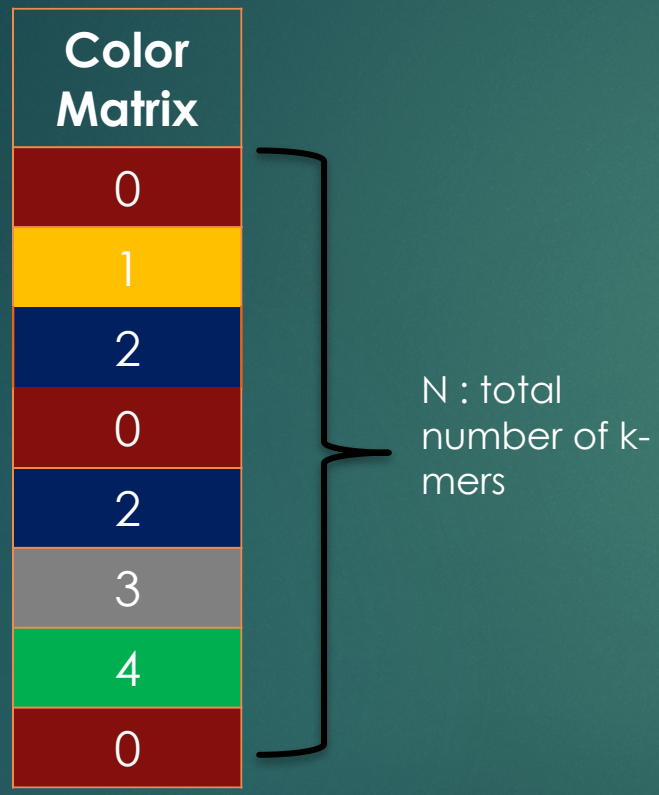
000 001 010 000 010 011 100 000

Label Bitvector

$$l = N \times \log_2 E \leq N \times C$$

Rainbowfish

Equivalence Table



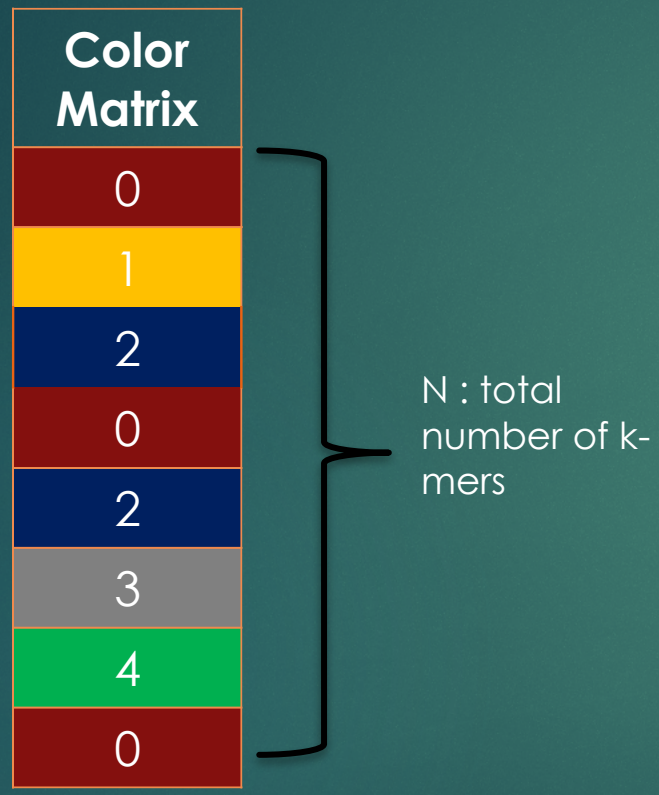
Datasets	$\frac{l}{N \times C}$
E. Coli (10)	0.9
E. Coli (1000)	0.02
E. Coli (5598)	0.004
Plant (4)	1
Beef Safety (88)	< 0.34
Human Txme (95,146)	< 0.0002

000 001 010 000 010 011 100 000

Label Bitvector

$$l = N \times \log_2 E \leq N \times C$$

Rainbowfish
Equivalence Table



Datasets	$\frac{l}{N \times C}$
E. Coli (10)	0.9
E. Coli (1000)	0.02
E. Coli (5598)	0.004
Plant (4)	1
Beef Safety (88)	< 0.34
Human Txme (95,146)	< 0.0002

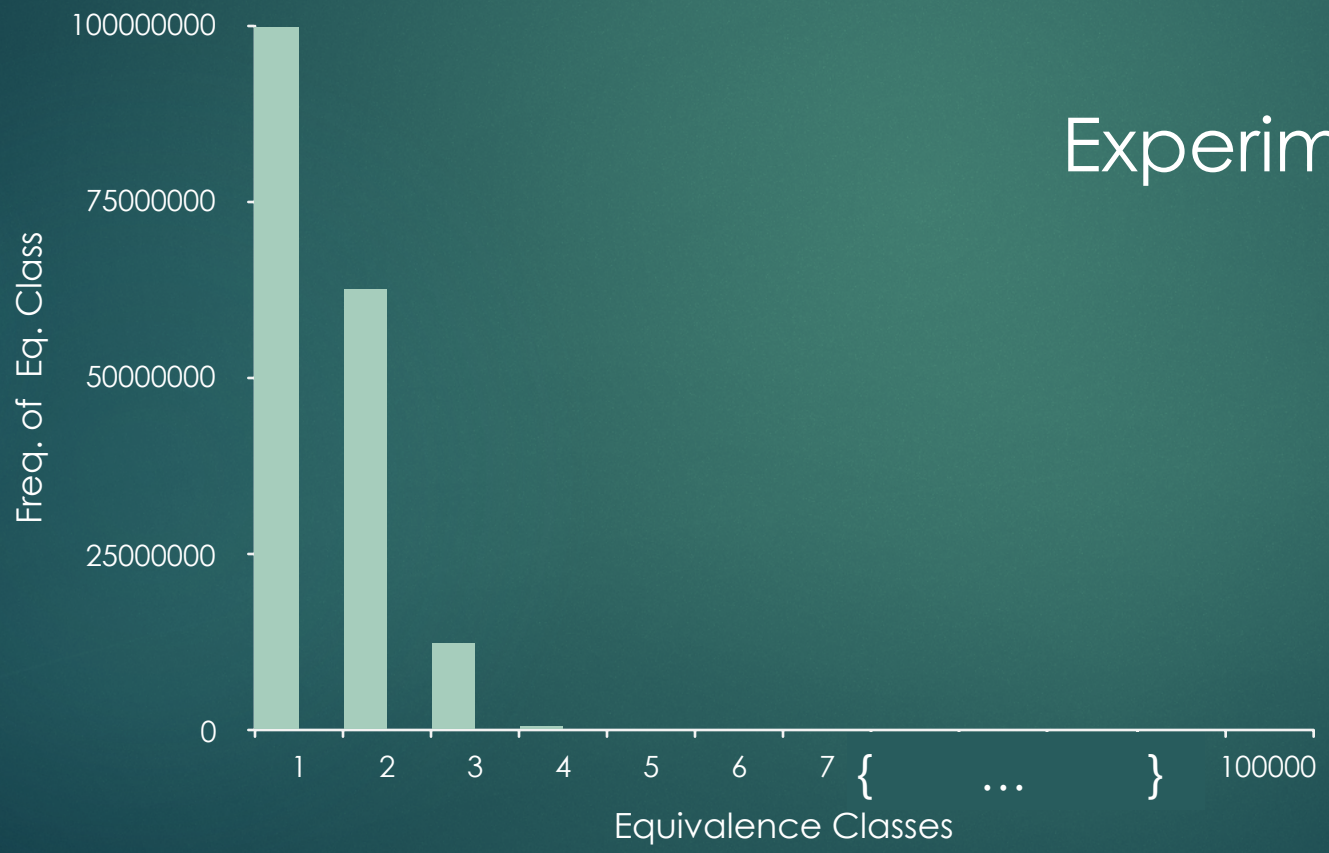
000 001 010 000 010 011 100 000

Label Bitvector

$$l = N \times \log_2 E \leq N \times C$$

Rainbowfish

Label & Boundary Bitvectors



Experimental Observation

Rainbowfish

Label & Boundary Bitvectors

Label	Equivalence Class	Freq.
0	0110010101	3
1	0011110000	1
2	1111110101	2
3	1010101011	1
4	1110000001	1

*

Equivalence Bitvector

Color Matrix
0
1
2
0
2
3
4
0

Label Bitvector

* This idea is briefly discussed in BFT paper

Rainbowfish

Label & Boundary Bitvectors

Label	Equivalence Class	Freq. *
0	0110010101	3
1	1111110101	2
2	0011110000	1
3	1010101011	1
4	1110000001	1

Equivalence Bitvector

Color Matrix
0
2
1
0
1
3
4
0

Label Bitvector

* This idea is briefly discussed in the BFT paper

Rainbowfish

Label & Boundary Bitvectors

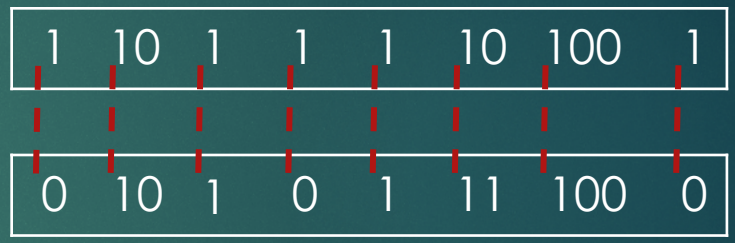
Label	Equivalence Class
0	0110010101
1	1111110101
2	0011110000
3	1010101011
4	1110000001

*

Equivalence Bitvector

Color Matrix
0
2
1
0
1
3
4
0

Boundary Bitvector



Label Bitvector

Label Bitvector

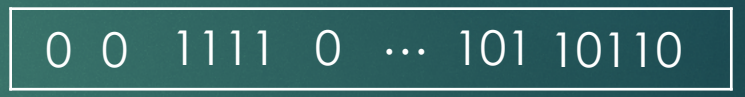
* This idea is briefly discussed in BFT paper

Rainbowfish

Label & Boundary Bitvectors

Label Bitvector

Boundary Bitvector



Label Bitvector

$$N \times \log_2(E) \quad vs \quad 2\sum_{i=1}^{E} \log_2(i) \times freq_{ec_i}$$

Why is Rainbowfish Succinct

- Uses amount of space close to information-theoretic optimum
  - Z: Information-theoretic Optimal
  - $Z + o(Z)$: Space by data structure
- Common Operations for navigation:
  - Rank
  - Select

Why Is Rainbowfish Succinct

- ***Lemma 1:***

  The size of each color class label is bounded by $\log_2 M$ bits, where M is the total number of distinct color classes. For a dataset with N distinct k-mers coming from C input samples (i.e., colors), we have that $M \leq \min(N, 2^C)$.

- ***Theorem 1:***

  Given an ordering of edges (or k-mers) in a de Bruijn Graph, the space needed by rainbowfish to represent a set of colors attached to each edge is bounded by $O(MC + NH(X_e))$.

  - M: Number of distinct color classes
  - C : Number of colors
  - N : Number of distinct k-mers
  - $H(X_e) : -\sum_{i=1}^{M} P(X_i) \log_2 P(X_i)$ is the entropy over random variable X_i which shows the frequency distribution of the color classes

Why Is Rainbowfish Succinct

- ***Theorem 2***

  The lower bound to represent a mapping from an ordered list of k-mers in a de Bruijn graph to a set of color classes is $log_2(MN - M.M!)$ bits, where M is the number of distinct color classes, N is the number of edges, and for a dataset with N distinct k-mers coming from C input samples (i.e., colors), we have that $M \leq \min(N, 2^C)$.

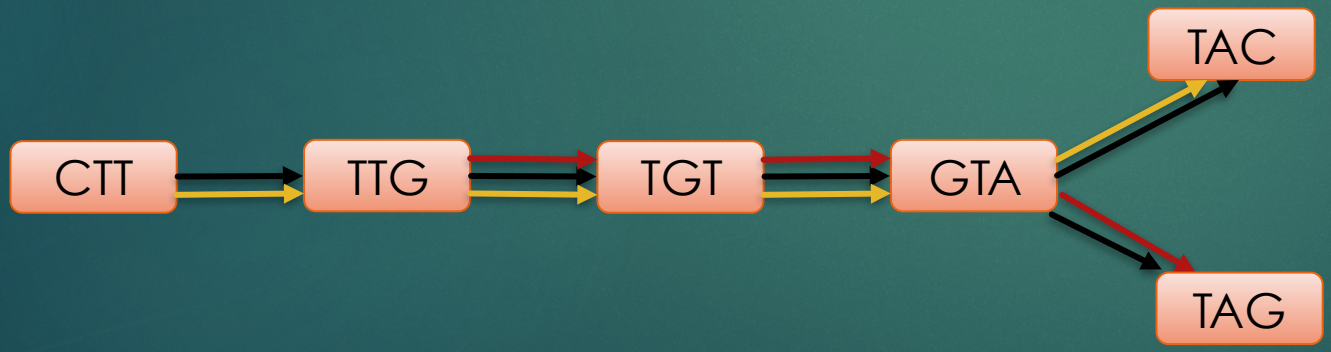
- Counting argument (Lower bound)
  - Count number of ways to map M distinct color class to N set of edges
  - Assign set of M edges a distinct color class : $M!$
  - Assign colors to $N - M$ in any possible manner : $M^{N-M}.M!$
  - To represent any of these we need at least $\log_2(M^{N-M}.M!)$
  - $= (N - M)\, log_2\, M + Mlog_2(M) - 0.44M + O(\log_2 M) \geq Nlog_2 M - 0.44M$
  - Given $1 \leq M \leq N$, $Nlog_2 M$ dominates lower bound

Why Is Rainbowfish Succinct

- Succinct Definition + Lemma 1 + Theorem 1 + Theorem 2
- $\mathrm{S} = Z + o(Z)$
- $Z \geq N log_2 M$
- $\mathrm{s} = O\big(MC + NH(X_e)\big) \leq 2N log_2 M$
- $\mathrm{S} = \mathrm{s} + \mathrm{o}(\mathrm{N})$ for overhead of the metadata to perform select
- $\mathrm{S} = Z + o(Z)$

Rainbowfish Query

- Basic Query: search(k-mer, color)
  - Returns true if the **k-mer** is assigned to the **color** in dbg or false otherwise
- Use search(k-mer, color) in "Bubble Calling"[4]



Navigation Through de Bruijn Graph

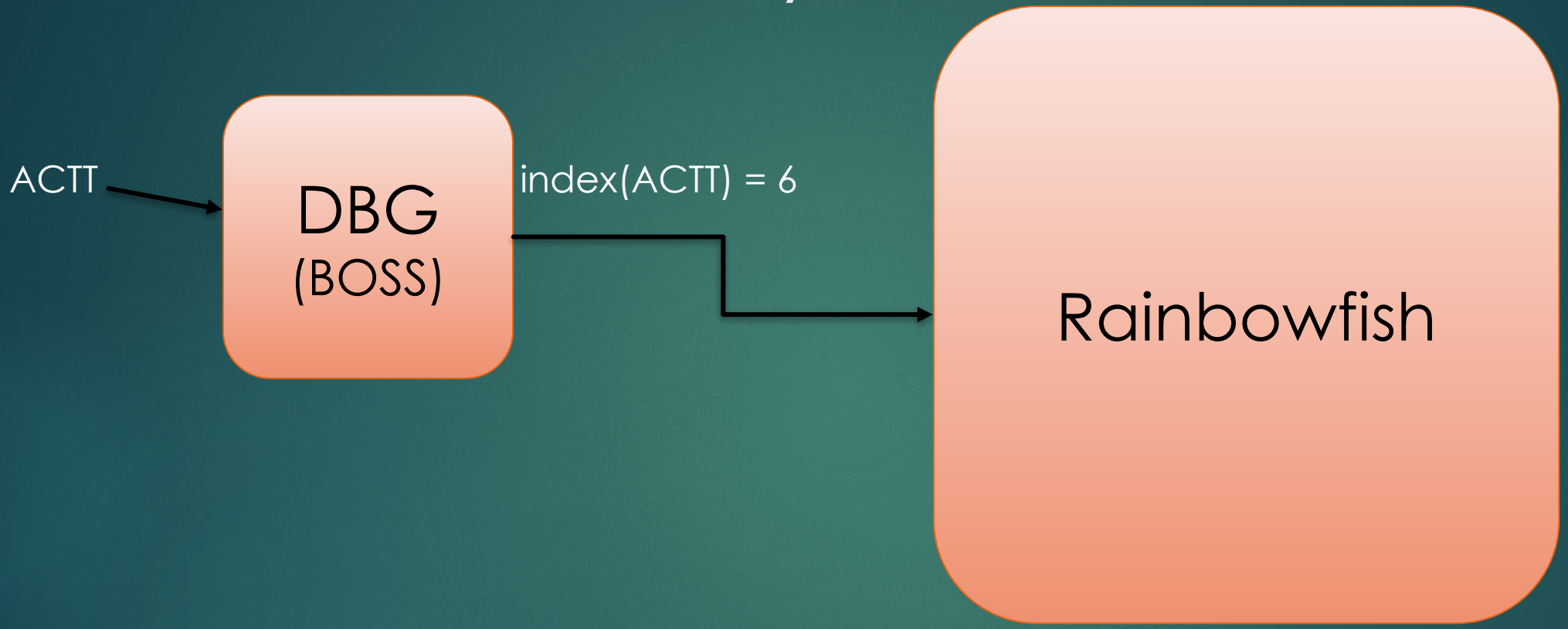
For each k-mer

search(k-mer, color1)

search(k-mer, color2)

Until one of these are false

Rainbowfish Query



Rainbowfish Query

Boundary Bitvector

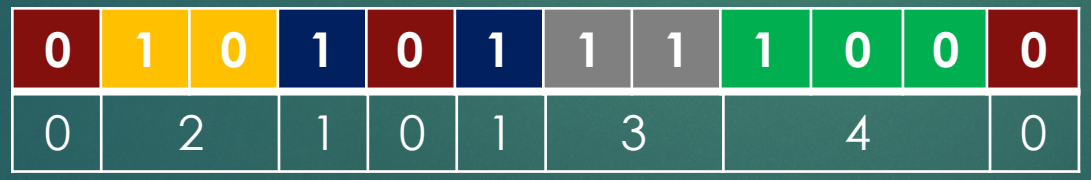
Label Bitvector

Equivalence Bitvector

Rainbowfish Query

index(ACTT) = 6

Boundary Bitvector



Label Bitvector

Equivalence Bitvector

Rainbowfish Query

index(ACTT) = 6

Select(i=6)=8

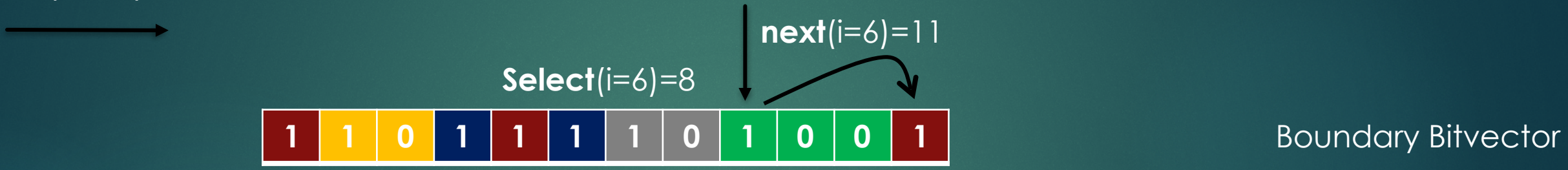
Boundary Bitvector

Label Bitvector

Equivalence Bitvector

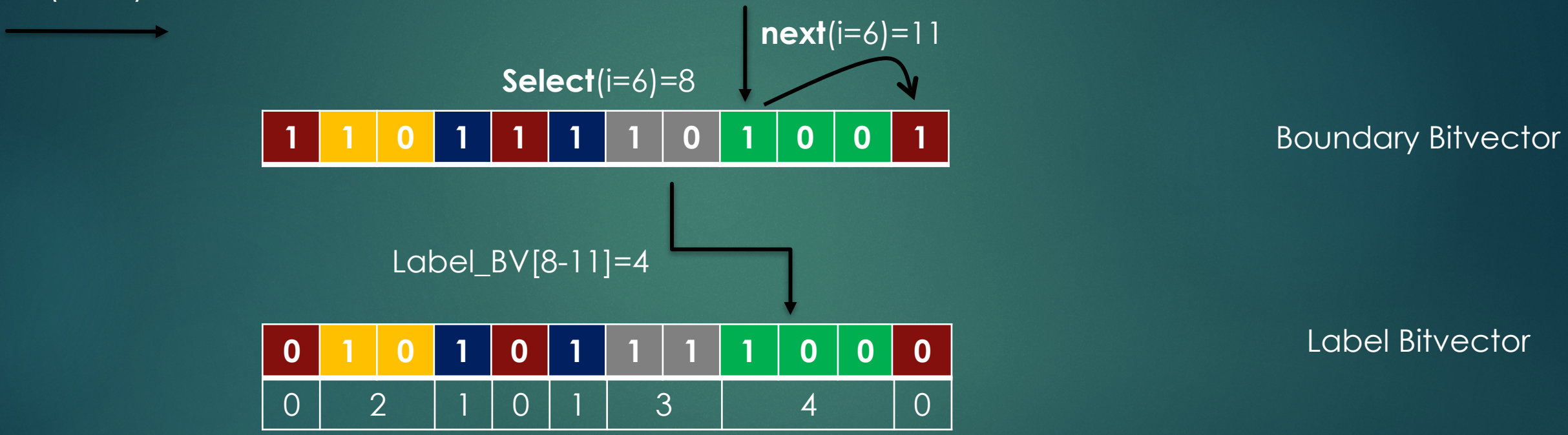
Rainbowfish Query

index(ACTT) = 6



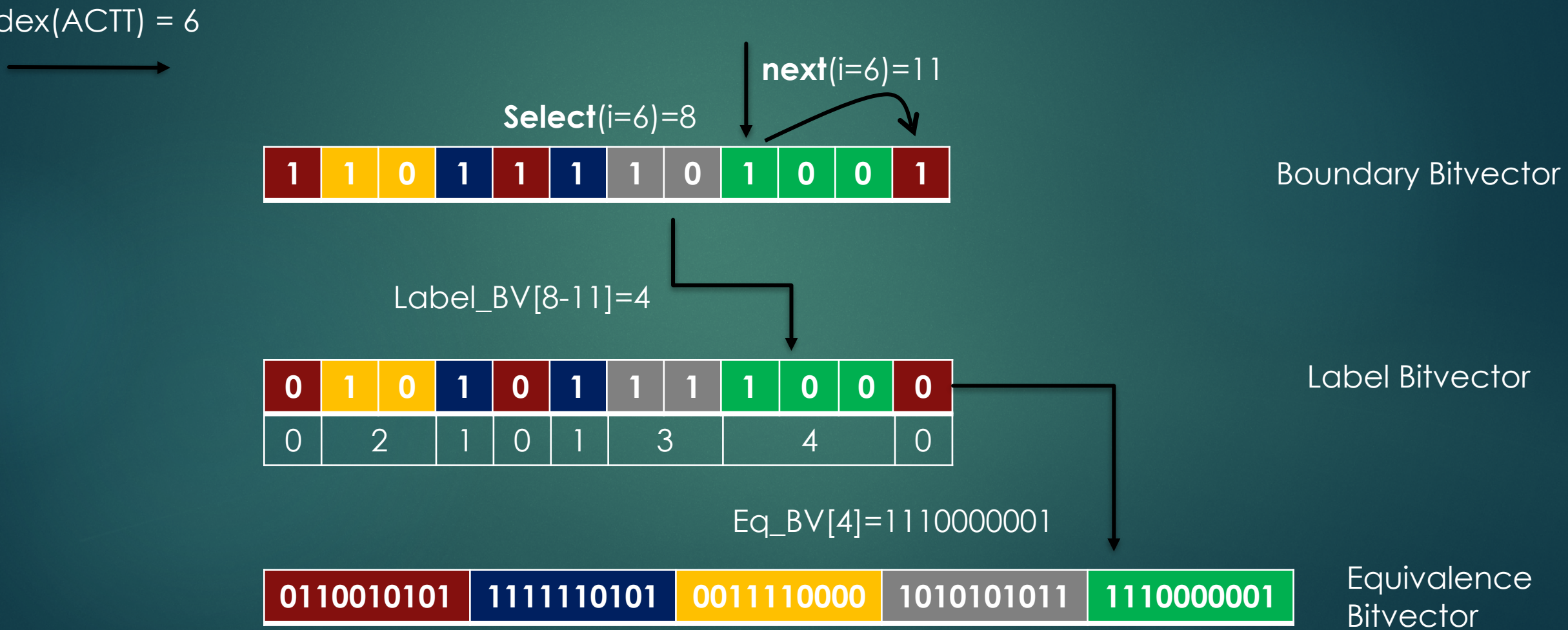
Rainbowfish Query

index(ACTT) = 6



Rainbowfish Query

index(ACTT) = 6



Characteristics of Datasets

Datasets	# of Colors (C)	# k-mers (N)	# Color Eq. Cls (E)
E. Coli 10	10	28,273,951	479
E. Coli 1000	1000	157,737,064	2,669,157
E. Coli 5598	5598	435,705,390	7,000,715
Plant	4	2,520,140,426	16
Beef Safety	88	> 97,096,576,010	623,022,532
Human Txme	95,146	> 159,441,804	340,762

Disk Space*(MB)

Datasets	Uncompressed Color Matrix	VARI	Rainbowfish
E. Coli 10	34	58	20
E. Coli 1000	18,804	8,848	475
E. Coli 5598	290,761	58,718	2,938
E. Coli 1000 (k=63)	185,669	8,872	637
Plant (4)	1,202	1,603	497
Beef Safety (88)	1,007,009	210,998	144,564
Human Txme (95,146)	1,808,435	841	817

* For rainbowfish Memory and disk space is almost the same

Disk Space*(MB)

Datasets	Uncompressed Color Matrix	VARI	Rainbowfish
E. Coli 10	34	58	20
E. Coli 1000	18,804	8,848	475
E. Coli 5598	290,761	58,718	2,938
E. Coli 1000 (k=63)	185,669	8,872	637
Plant (4)	1,202	1,603	497
Beef Safety (88)	1,007,009	210,998	144,564
Human Txme (95,146)	1,808,435	841	817

* For rainbowfish Memory and disk space is almost the same

Construction & Query Time

	Construction Time (secs)		Bubble Calling Time (secs)	
Datasets	VARI	Rainbowfish	VARI	Rainbowfish
E. Coli 10	44	31	344	366
E. Coli 1000	340	270	2,610	2,356
E. Coli 5598	3,141	4,021	8,796	8,201
Plant	108	339	47,040	48,537
Beef Safety	15,375	30,478	NA	NA
Human Txme	13,961	30,804	NA	NA

Construction & Query Time

	Construction Time (secs)		Bubble Calling Time (secs)	
Datasets	VARI	Rainbowfish	VARI	Rainbowfish
E. Coli 10	44	31	344	366
E. Coli 1000	340	270	2,610	2,356
E. Coli 5598	3,141	4,021	8,796	8,201
Plant	108	339	47,040	48,537
Beef Safety	15,375	30,478	NA	NA
Human Txme	13,961	30,804	NA	NA

Construction & Query Time

	Construction Time (secs)		Bubble Calling Time (secs)	
Datasets	VARI	Rainbowfish	VARI	Rainbowfish
E. Coli 10	44	31	344	366
E. Coli 1000	340	270	2,610	2,356
E. Coli 5598	3,141	4,021	8,796	8,201
Plant	108	339	47,040	48,537
Beef Safety	15,375	30,478	NA	NA
Human Txme	13,961	30,804	NA	NA

Pufferfish

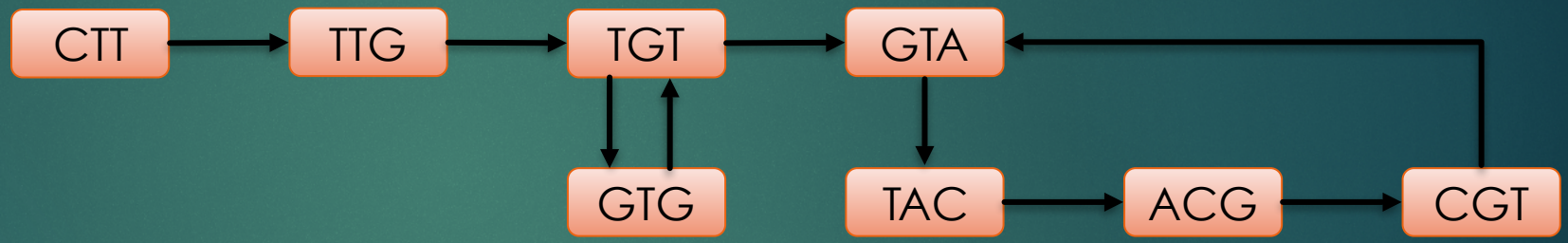

New Results
Previous
A space and time-efficient index for the compacted colored de Bruijn graph
Posted September 21, 2017.
Fatemeh Almodaresi, Hirak Sarkar, Rob Patro
Download PDF
doi: https://doi.org/10.1101/191874
Email
This article is a preprint and has not been peer-reviewed [what does this mean?].

Indexing de Bruijn Graph

K = 4

Samples:

CTTGTGTACGTA

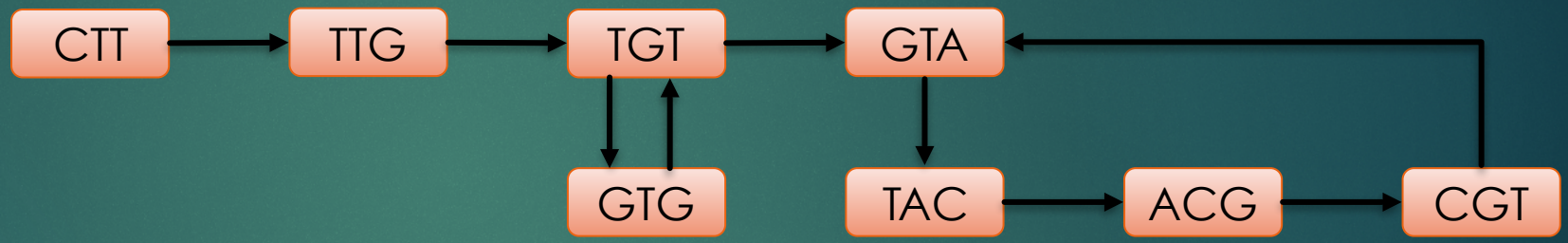


Indexing de Bruijn Graph

K = 4

References:

CTTGTGTACGTA



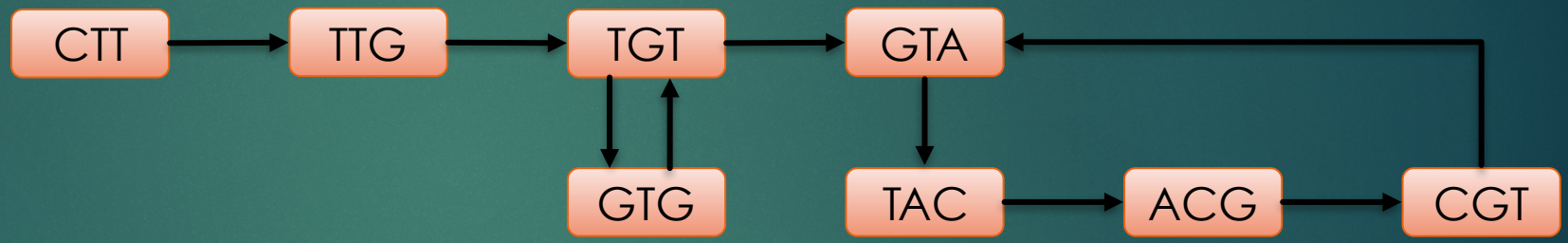
Reference: known genome or transcriptome

Indexing de Bruijn Graph

K = 4

References:

CTTGTGTACGTA



K-mer Info Table

k-mer ID	K-mer Info
0	{ref_y, p_2^y, o_2^y}, {ref_r, p_2^r, o_2^r}, {ref_r, p_2^r, o_2^r}
1	{ref_r, p_3^r, o_3^r}, {ref_g, p_3^g, o_3^g}
⋮	⋮

Reference Indexing

- Mapping and Alignment of reads to a (set of) reference
  - starting point for downstream analysis on genes and transcripts
    - Transcript and gene quantification
    - Metagenomic abundance estimation
    - …
- Time and space bottleneck
- Indexing the reference sequences
  - Efficiency in time and space

Reference Indexing

Existing indexing data structures

- linear self-indexing
  - Indexes all substrings of the text
- Hash-based indexing
  - Inverted index from words to reference positions

Reference Indexing

- **linear self-indexing**
  - Space-efficient
  - Slower search
  - variable length pattern
  - BWT and FM-index

- Bowtie2[9]
- BWA-MEM[10]

- **Hash-based indexing**
  - Take more space than linear but still Space-efficient
  - Fast search
  - Fixed patterns of length k (k-mer)
  - Recently used with the de bruin graph

- Genome Mapper[11]
- BGreat[12]
- deBGA[13]
- Kallisto[14]

Reference Indexing

- **linear self-indexing**
  - Space-efficient
  - Slower search
  - variable length pattern
  - BWT and FM-index
- Bowtie2[9]
- BWA-MEM[10]

- **Hash-based indexing**
  - Take more space than linear but still Space-efficient
  - Fast search
  - Fixed patterns of length k (k-mer)
  - Recently used with the de bruin graph
- Genome Mapper[11]
- BGreat[12]
- deBGA[13]
- Kallisto[14]

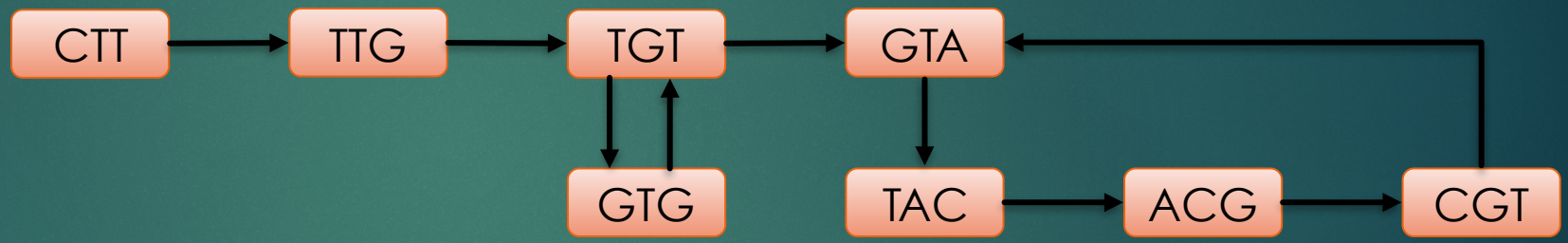
→ Compacted de Bruijn Graph

Compacted de Bruijn Graph

K = 4

References:

CTTGTGTACGTA

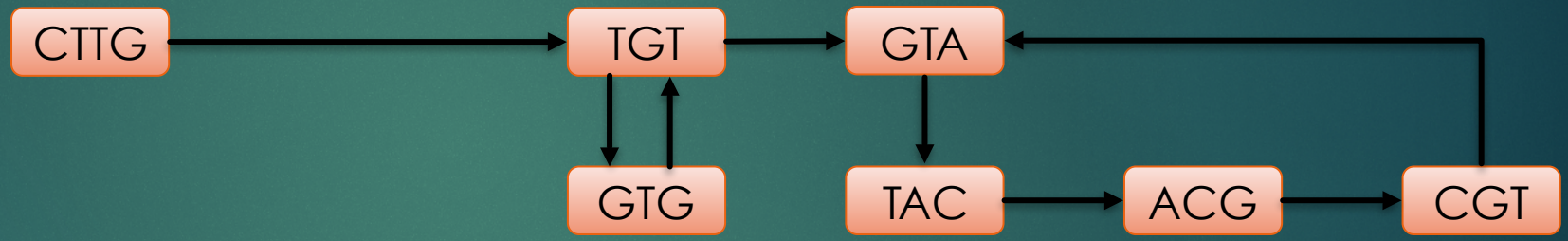


Compacted de Bruijn Graph

K = 4

References:

CTTGTGTACGTA

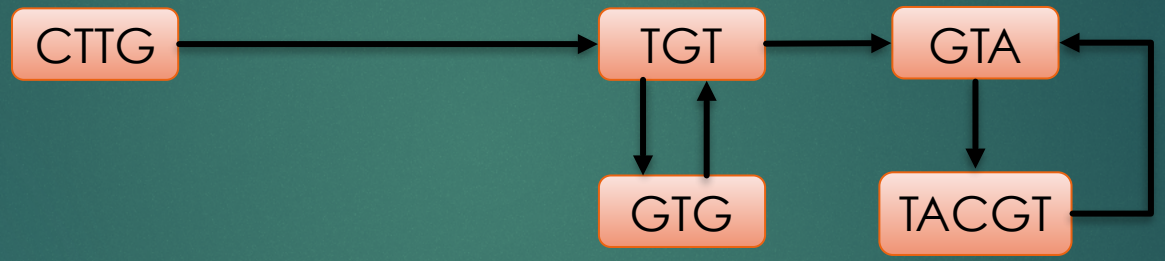


Compacted de Bruijn Graph

K = 4

References:

CTTGTGTACGTA

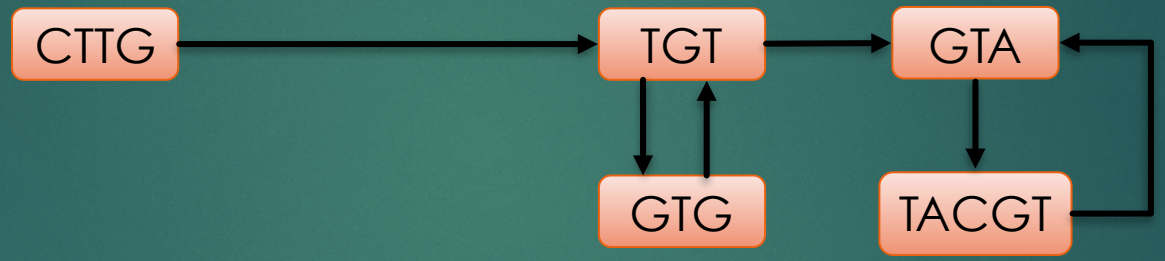


Compacted de Bruijn Graph

K = 4

References:

CTTGTGTACGTA



Uniting Info Table

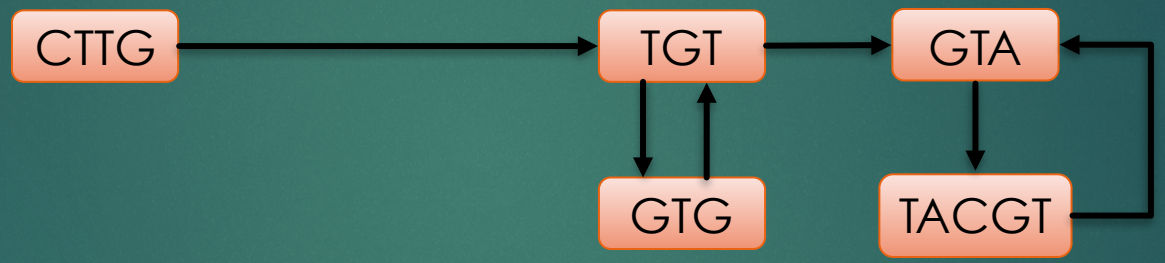
Unitig ID	Unitig Info
0	{ref_y, p_2^y, o_2^y}, {ref_r, p_2^r, o_2^r}, {ref_r, p_2^r, o_2^r}
1	{ref_r, p_3^r, o_3^r}, {ref_g, p_3^g, o_3^g}
⋮	⋮

Compacted de Bruijn Graph

K = 4

References:

CTTGTGTACGTA



Uniting Info Table

Unitig ID	Unitig Info
0	{ref_y, p_2^y, o_2^y}, {ref_r, p_2^r, o_2^r}, {ref_r, p_2^r, o_2^r}
1	{ref_r, p_3^r, o_3^r}, {ref_g, p_3^g, o_3^g}
⋮	⋮

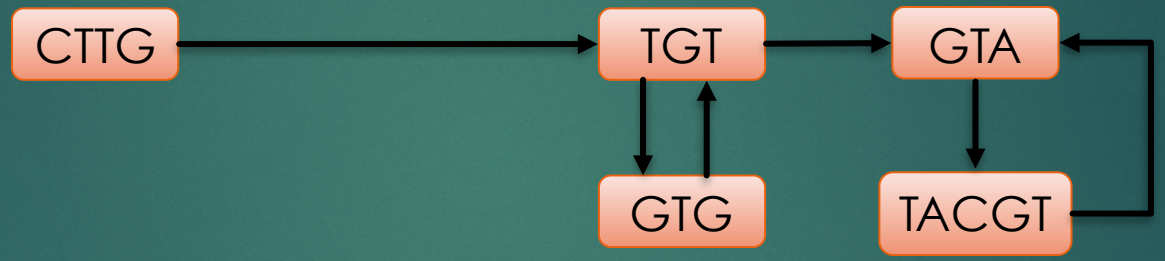
Reduce size of the info table

Compacted de Bruijn Graph

K = 4

References:

CTTGTGTACGTA

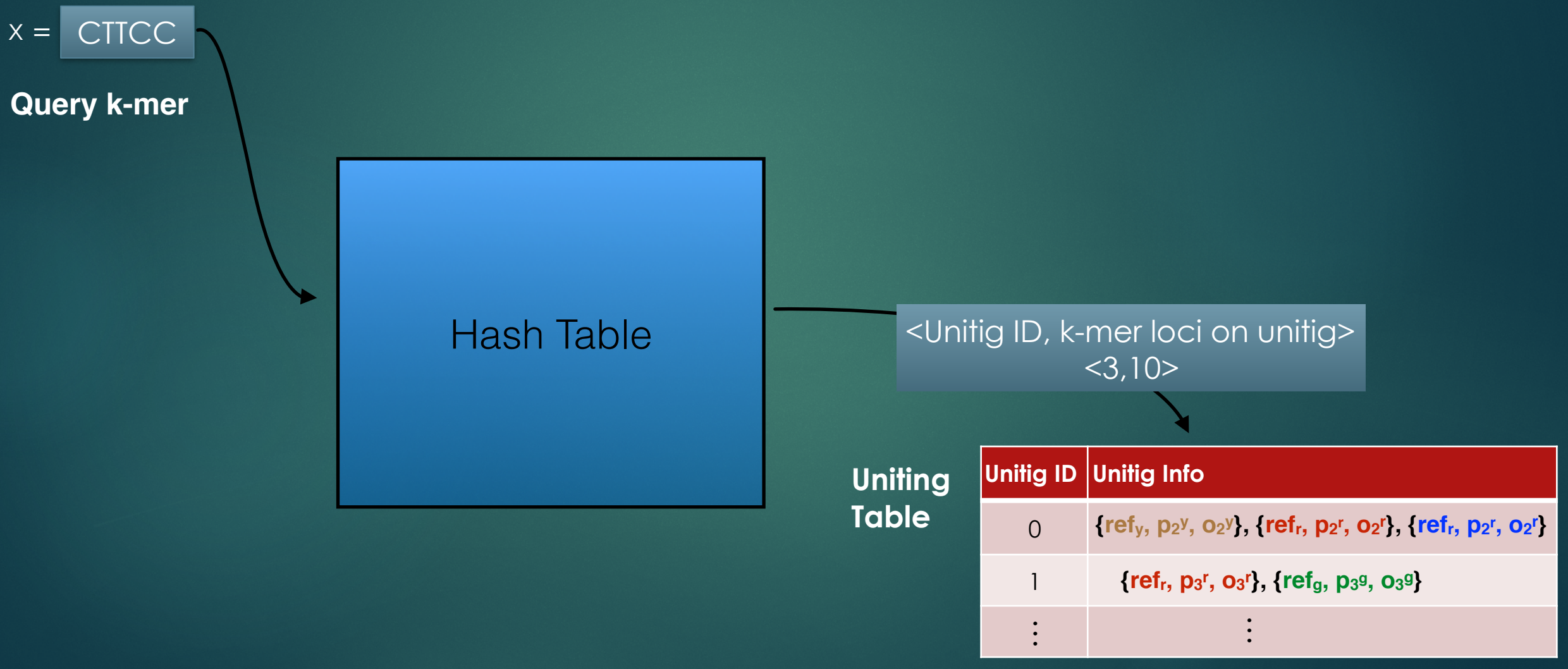


Uniting Info Table

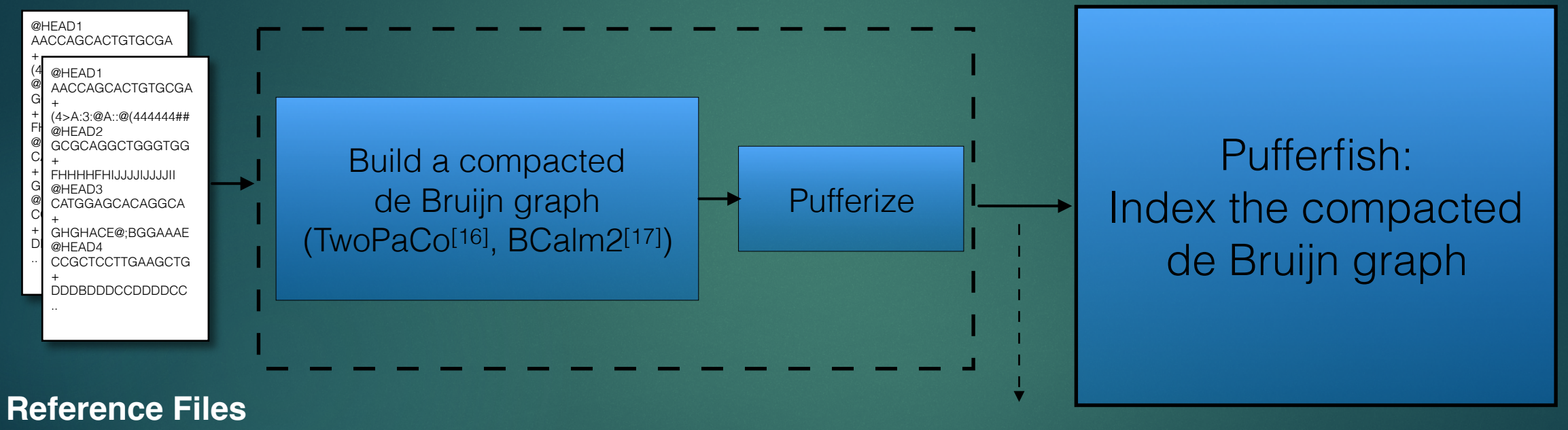
Unitig ID	Unitig Info
0	{ref_y, p_2^y, o_2^y}, {ref_r, p_2^r, o_2^r}, {ref_r, p_2^r, o_2^r}
1	{ref_r, p_3^r, o_3^r}, {ref_g, p_3^g, o_3^g}
⋮	⋮

No notion of k-mers

Kallisto[14] and deBGA[13] use Hash Tables to map k-mers to their corresponding unitigs



Pufferfish Pipeline



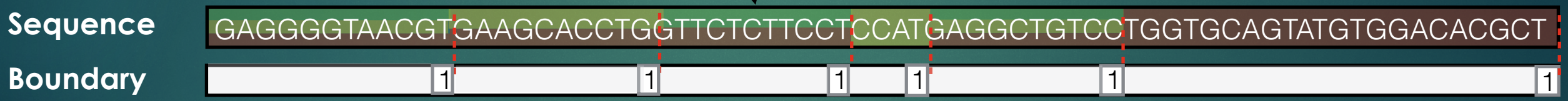
Reference Files

- Unitigs and their ID
- References encoded using unitig IDs

Pufferfish (dense)

x = CTTCC

Query k-mer



rank(p(x))

Uniting Table

Unitig ID	Unitig Info
0	{ref_y, p_2^y, o_2^y}, {ref_r, p_2^r, o_2^r}, {ref_r, p_2^r, o_2^r}
1	{ref_r, p_3^r, o_3^r}, {ref_g, p_3^g, o_3^g}
⋮	⋮

Pufferfish (dense)

Awesome MPHF implementation

Fast and scalable minimal perfect hashing for massive key sets

Antoine Limasset[1], Guillaume Rizk[1], Rayan Chikhi[2], and Pierre Peterlongo[1]

1 IRISA Inria Rennes Bretagne Atlantique, GenScale team, Campus de Beaulieu 35042 Rennes, France
2 CNRS, CRIStAL, Université de Lille, Inria Lille - Nord Europe, France

x = CTTCC

Query k-mer

Minimal Perfect Hash Function

h(x)

- Guarantees a unique index in [0,N) for each k-mer
- no collision
- no order preservation
- False Positive Result: Can generate a number in [0, N) for a non-indexed k-mer

Sequence GAGGGGTAACGTGAAGCACCTGGTTCTCTTCCTCCATGAGGCTGTCCTGGTGCAGTATGTGGACACGCT

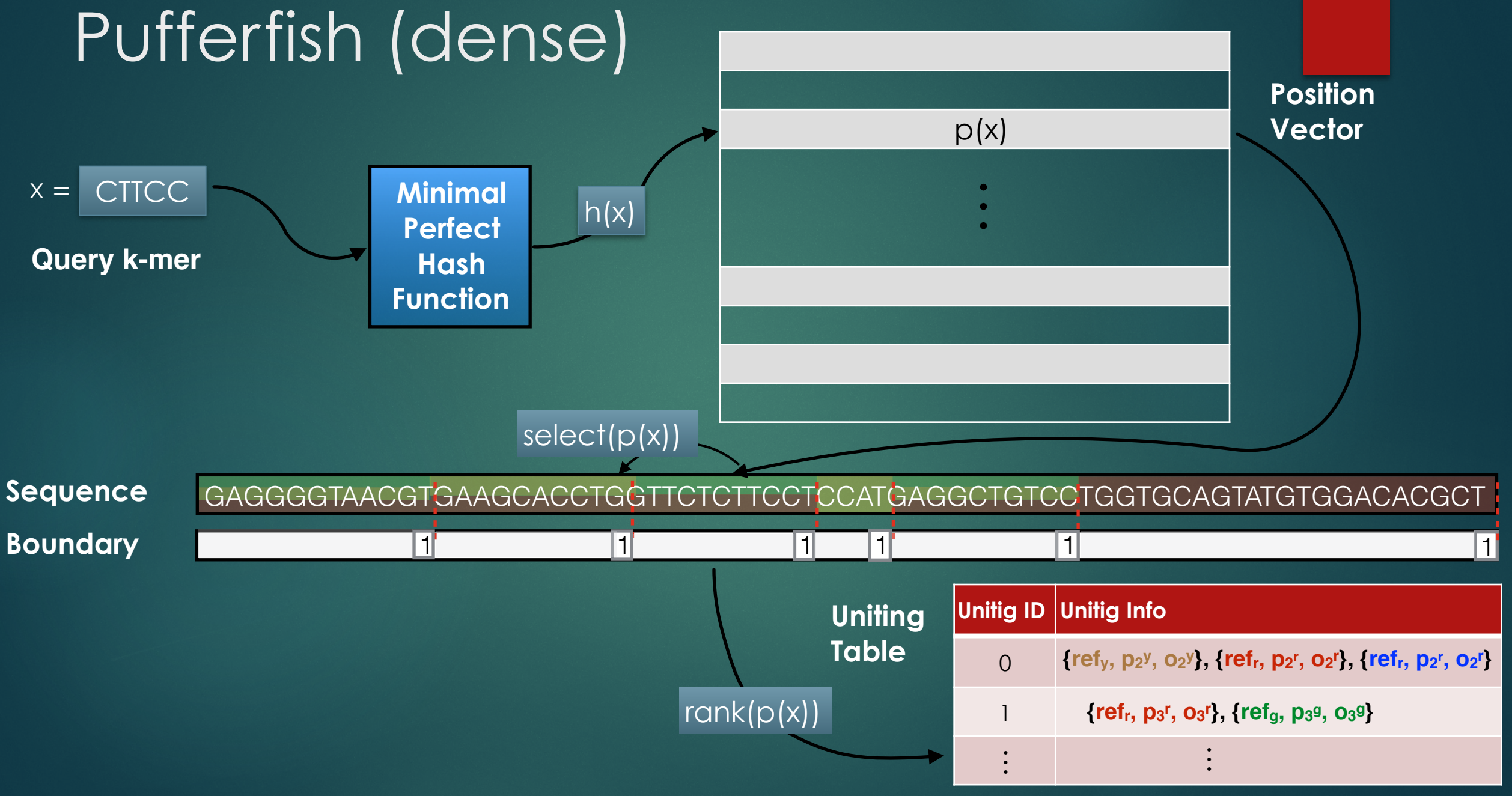
Boundary 1 1 1 1 1 1

rank(p(x))

Uniting Table

Unitig ID	Unitig Info
0	{ref_y, p_2^y, o_2^y}, {ref_r, p_2^r, o_2^r}, {ref_r, p_2^r, o_2^r}
1	{ref_r, p_3^r, o_3^r}, {ref_g, p_3^g, o_3^g}
⋮	⋮

Pufferfish (dense)



Unitig ID	Unitig Info
0	$\{ref_y, p_2^y, o_2^y\}, \{ref_r, p_2^r, o_2^r\}, \{ref_r, p_2^r, o_2^r\}$
1	$\{ref_r, p_3^r, o_3^r\}, \{ref_g, p_3^g, o_3^g\}$
⋮	⋮

Pufferfish

Space of index on disk

Tool	Human Transcriptome	Human Genome	Bacterial Genomes
BWA	347M	5.12G	31G
kallisto	1.7G	58G	120G
pufferfish dense	387M	16G	37G

Pufferfish

Space of index + query in RAM

Tool	Memory (MB)		
	Human Transcriptome	Human Genome	Bacterial Genome
BWA	308	4,439	27,535
kallisto	3,336	110,464	232,353
pufferfish dense	454	17,684	41,532

Pufferfish

Space of index + query in RAM

Tool	Memory (MB)		
	Human Transcriptome	Human Genome	Bacterial Genome
BWA	308	4,439	27,535
kallisto	3,336	110,464	232,353
pufferfish dense	454	17,684	41,532

What does this representation buy us?

What about speed?

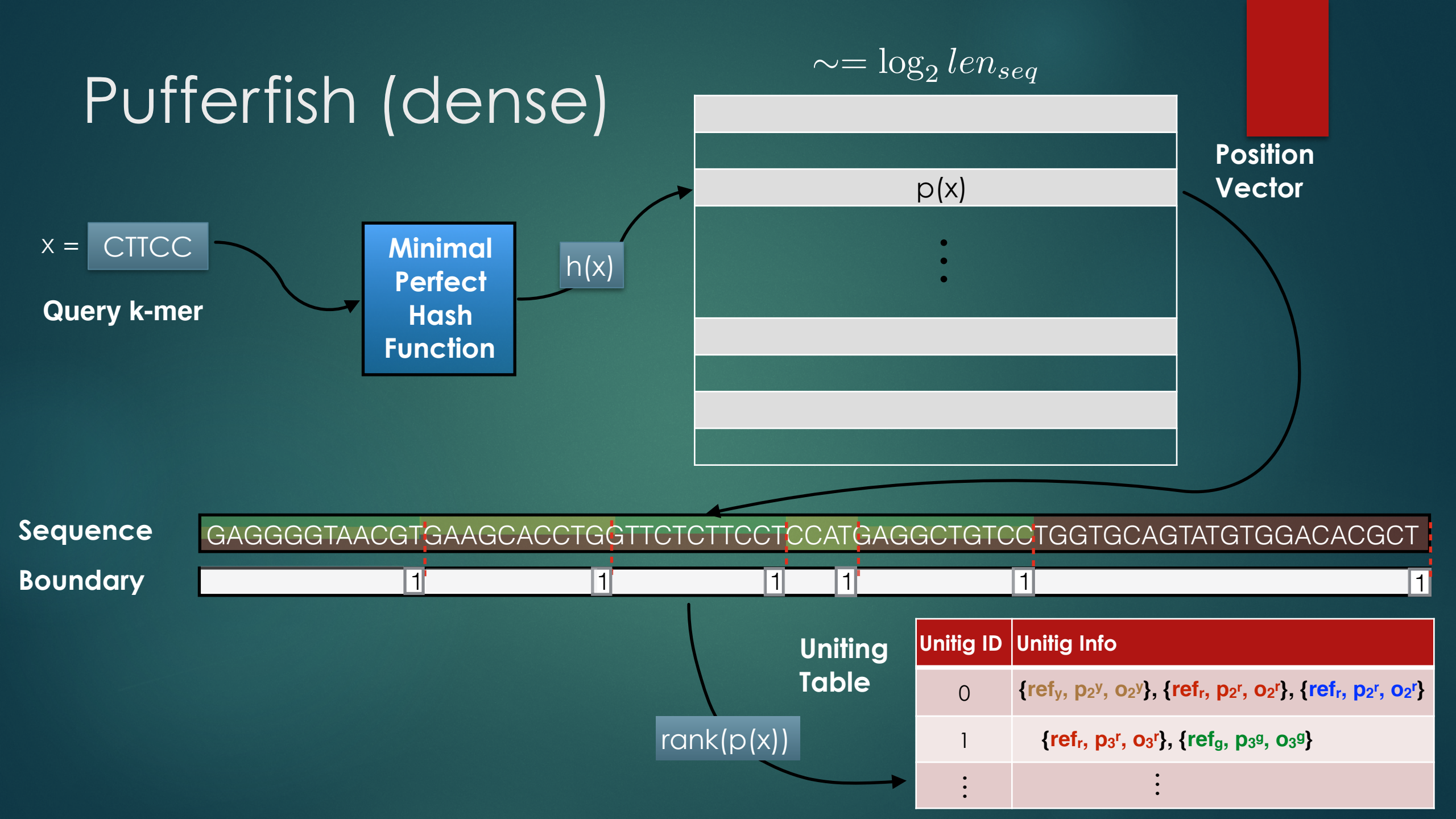
Take experimental sequencing reads, and look-up *all* k-mers (k=31). The time reported is to load the index, map all 31-mers, and record the total number of (multi-mapping) hits.

Tool	Time (h:m:s)		
	Human Transcriptome	Human Genome	Bacterial Genome
BWA	0:17:35	0:50:31	0:14:05
kallisto	0:02:01	0:19:11	0:22:25
pufferfish dense	0:02:46	0:10:37	0:06:03

Dataset	# reads	read length
SRR1215997 (Human txome)	10,683,470	100
SRR5833294 (Human genome)	34,129,891	250
SRR5901135 (From *E. coli* genome)	2,314,288	250

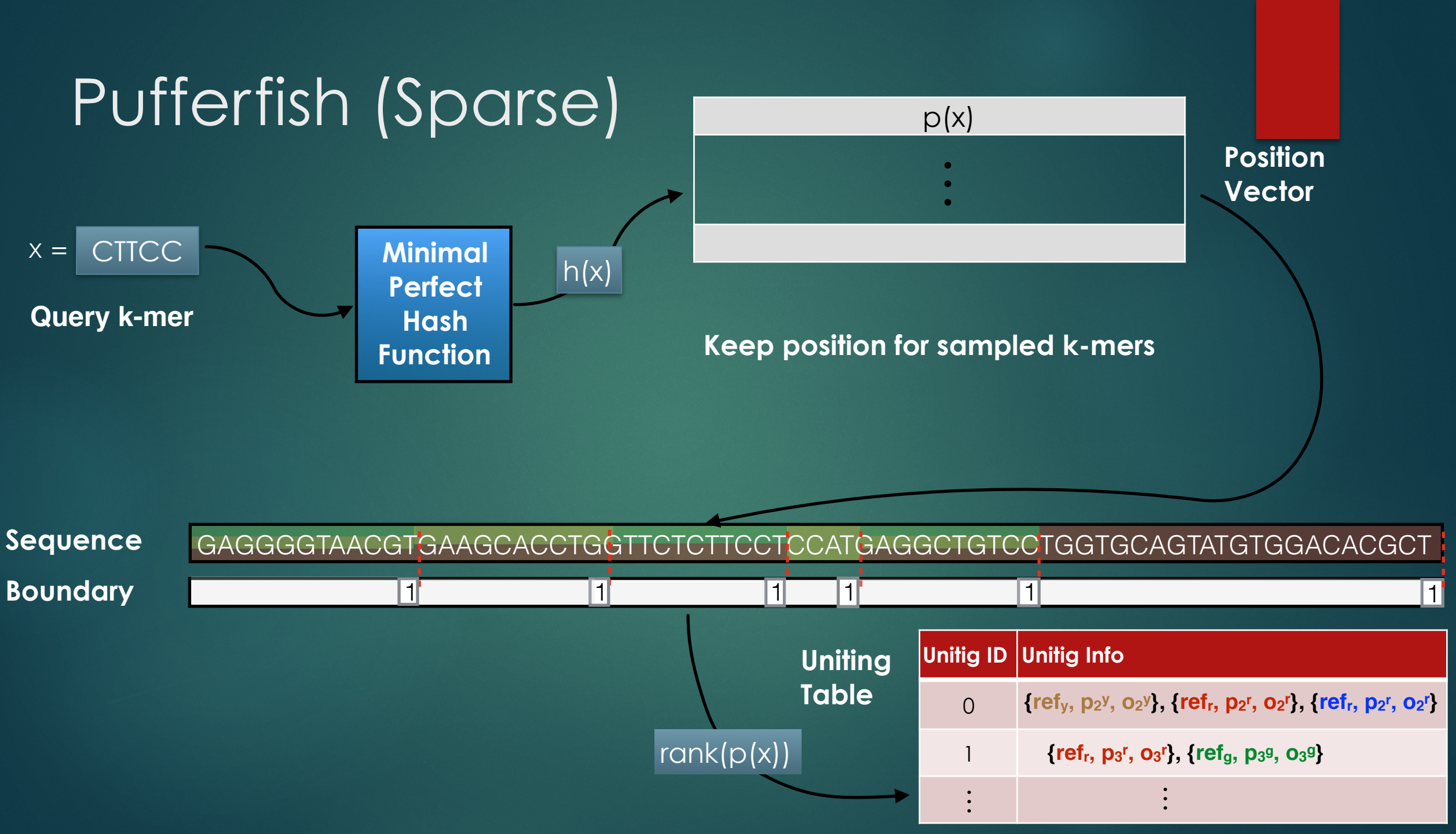
- Pufferfish is either the fastest index, or very close to the fastest in all experiments
- For existing hash-based graph index, i/o & loading burden become substantial as index size grows
- For linear indices, positional enumeration of highly-repetitive content can be slow (e.g. human txome & genome)

Pufferfish (dense)



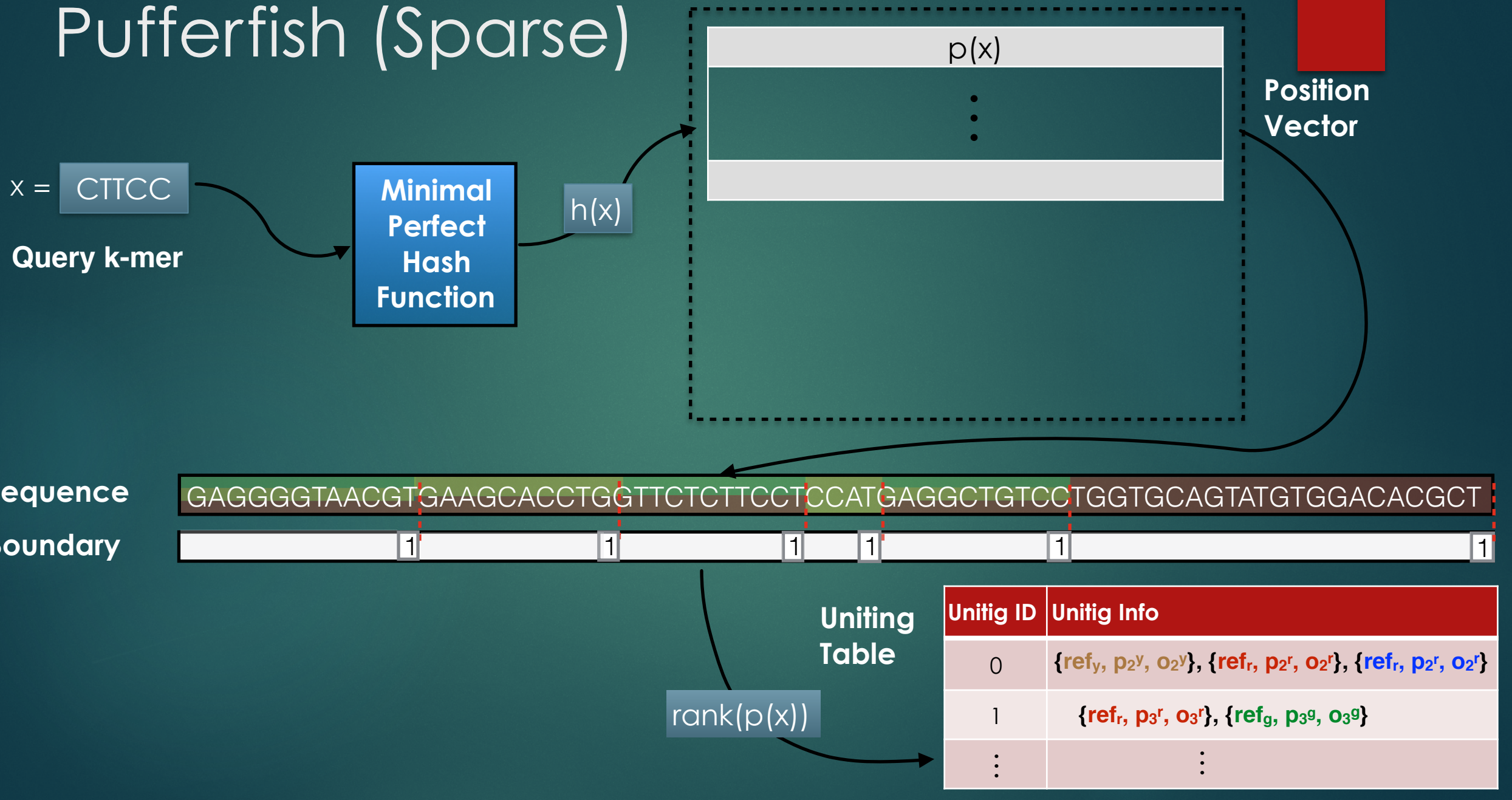
Unitig ID	Unitig Info
0	{ref$_y$, p2^y, o2^y}, {ref$_r$, p2^r, o2^r}, {ref$_r$, p2^r, o2^r}
1	{ref$_r$, p3^r, o3^r}, {ref$_g$, p3^g, o3^g}
⋮	⋮

Pufferfish (Sparse)



Unitig ID	Unitig Info
0	{ref$_y$, p2^y, o2^y}, {ref$_r$, p2^r, o2^r}, {ref$_r$, p2^r, o2^r}
1	{ref$_r$, p3^r, o3^r}, {ref$_g$, p3^g, o3^g}
⋮	⋮

Pufferfish (Sparse)

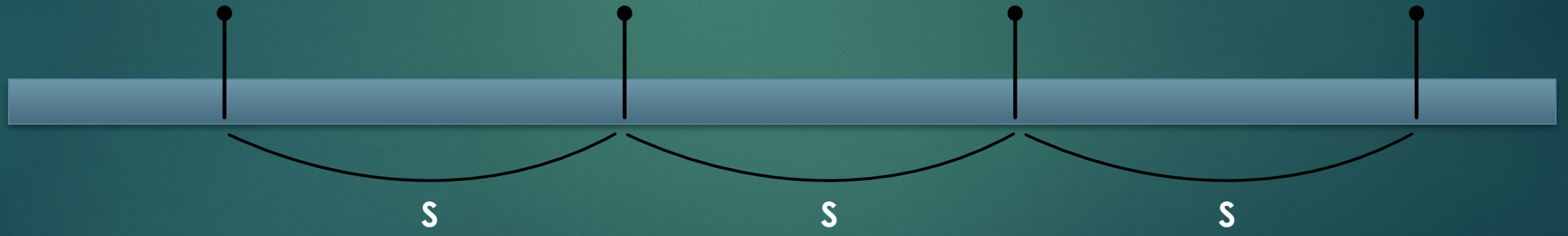


Pufferfish (Sparse)

- Each k-mer appears at most once in unitig array
- Each present k-mer has a unique predecessor + a unique successor

Pufferfish (Sparse)

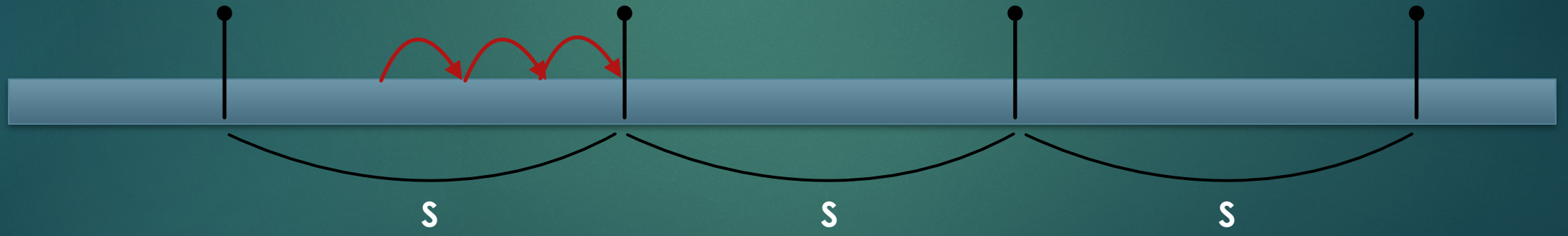
- Each k-mer appears at most once in unitig array
- Each present k-mer has a unique predecessor + a unique successor*



* For simplicity ignore technical issues like boundary k-mers

Pufferfish (Sparse)

- Each k-mer appears at most once in unitig array
- Each present k-mer has a unique predecessor + a unique successor*



* For simplicity ignore technical issues like boundary k-mers

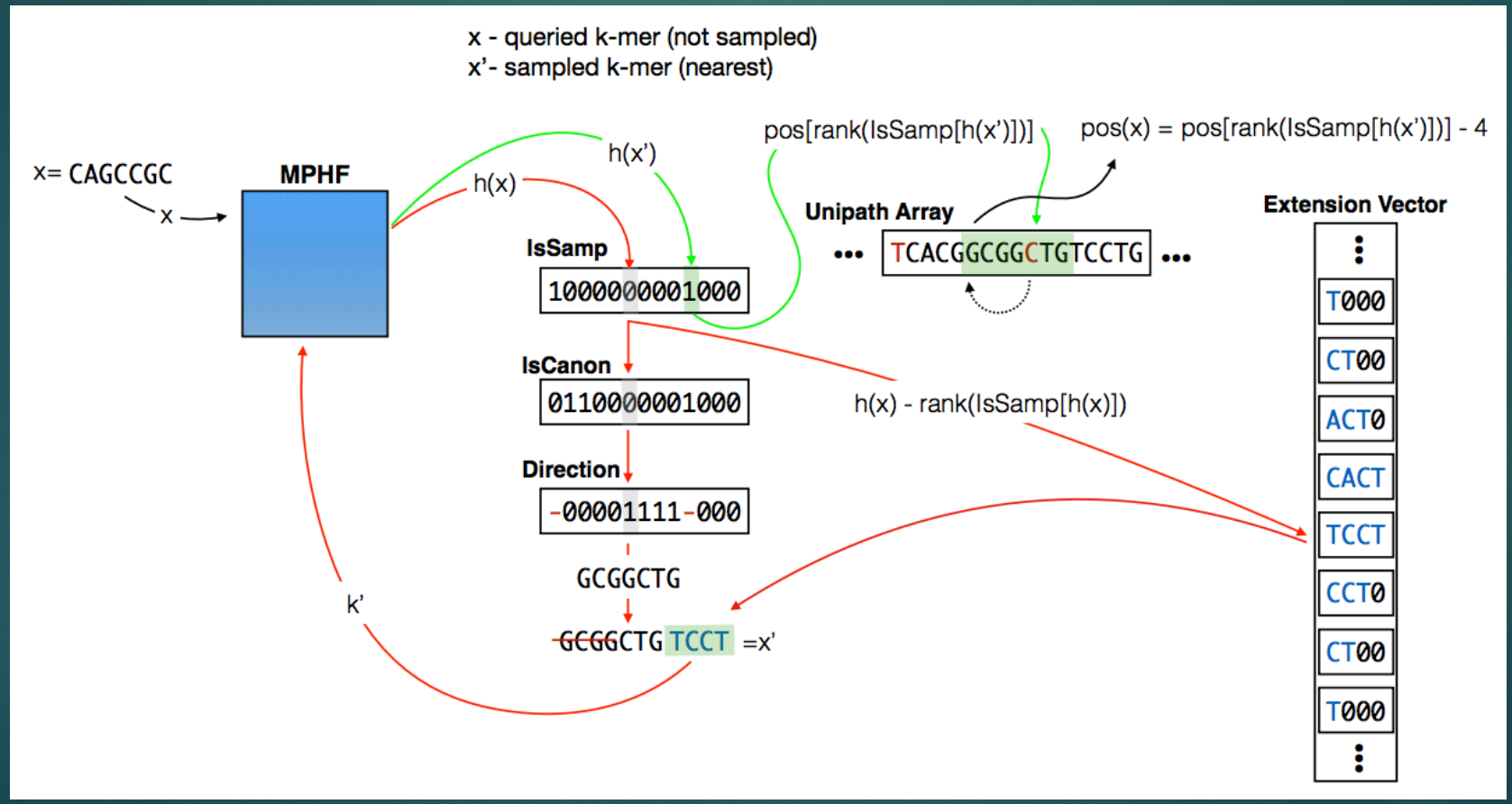
Pufferfish (Sparse)

- Each k-mer appears at most once in unitig array
- Each present k-mer has a unique predecessor + a unique successor*

* For simplicity ignore technical issues like boundary k-mers

Pufferfish (Sparse)

Main Idea: Successors and predecessors in unipaths are *globally unique*, instead of storing position information for all k-mers, store positions only at sampled "landmarks" and say how to navigate to these landmarks (similar to bi-directional sampling in the FM-index).



Space of index on disk

Tool	Human Transcriptome	Human Genome	Bacterial Genomes
BWA	347M	5.12G	31G
kallisto	1.7G	58G	120G
pufferfish dense	387M	16G	37G
pufferfish sparse	271M	11G	26G

Space of index + query in RAM

Tool	Memory (MB)		
	Human Transcriptome	Human Genome	Bacterial Genome
BWA	308	4,439	27,535
kallisto	3,336	110,464	232,353
pufferfish dense	454	17,684	41,532
pufferfish sparse	341	12,533	30,565

What does this representation buy us?

What about speed?

Take experimental sequencing reads, and look-up *all* k-mers (k=31). The time reported is to load the index, map all 31-mers, and record the total number of (multi-mapping) hits.

Tool	Time (h:m:s)		
	Human Transcriptome	Human Genome	Bacterial Genome
BWA	0:17:35	0:50:31	0:14:05
kallisto	0:02:01	0:19:11	0:22:25
pufferfish dense	0:02:46	0:10:37	0:06:03
pufferfish sparse	0:08:34	0:22:11	0:08:26

Dataset	# reads	read length
SRR1215997 (Human txome)	10,683,470	100
SRR5833294 (Human genome)	34,129,891	250
SRR5901135 (From *E. coli* genome)	2,314,288	250

- Pufferfish is either the fastest index, or very close to the fastest in all experiments
- For existing hash-based graph index, i/o & loading burden become substantial as index size grows
- For linear indices, positional enumeration of highly-repetitive content can be slow (e.g. human txome & genome)

References

- [1] Zerbino, D.R. & Birney, E. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome Res. 18, 821–829 (2008).
- [2] Gnerre, S. et al. High-quality draft assemblies of mammalian genomes from massively parallel sequence data. Proc. Natl. Acad. Sci. USA 108, 1513–1518 (2011). 32. Li, R. et al.
- [3] De novo assembly of human genomes with massively parallel short read sequencing. Genome Res. 20, 265–272 (2010).
- [4] Zamin Iqbal, Mario Caccamo, Isaac Turner, Paul Flicek, and Gil McVean. De novo assembly and genotyping of variants using colored de Bruijn graphs. Nature genetics, 44(2):226–232, 2012.
- [5] Alexander Bowe, Taku Onodera, Kunihiko Sadakane, and Tetsuo Shibuya. Succinct de Bruijn graphs. In Proceedings of the International Workshop on Algorithms in Bioinformatics, pages 225–235. Springer, 2012
- [6] Guillaume Holley, Roland Wittler, and Jens Stoye. Bloom filter trie: an alignment-free and reference-free data structure for pan-genome storage. Algorithms Mol. Biol., 11:3, 2016. **BFT**
- [7] Martin D. Muggli, Alexander Bowe, Noelle R. Noyes, Paul Morley, Keith Belk, Robert Raymond, Travis Gagie, Simon J. Puglisi, and Christina Boucher. Succinct Colored de Bruijn Graphs. 2017.
- [8] Simon Gog. Succinct data structure library. https://github.com/simongog/ sdsl-lite, 2017. [online; accessed 01-Feb-2017

References

- [9] Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with bowtie 2. *Nature methods*, 9(4):357–359, 2012.
- [10] Heng Li. Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. *arXiv preprint arXiv:1303.3997*, 2013.
- [11] Korbinian Schneeberger, Jörg Hagmann, Stephan Ossowski, Norman Warthmann, Sandra Gesing, Oliver Kohlbacher, and Detlef Weigel. Simultaneous alignment of short reads against multiple genomes. *Genome biology*, 10(9):R98, 2009.
- [12] Antoine Limasset, Bastien Cazaux, Eric Rivals, and Pierre Peterlongo. Read mapping on de bruijn graphs. *BMC bioinformatics*, 17(1):237, 2016.
- [13] Bo Liu, Hongzhe Guo, Michael Brudno, and Yadong Wang. debga: read alignment with de bruijn graph-based seed and extension. *Bioinformatics*, 32(21):3224–3232, 2016.
- [14] Nicolas L Bray, Harold Pimentel, Páll Melsted, and Lior Pachter. Near-optimal proba- bilistic RNA-seq quantification. *Nature Biotechnology*, 34(5):525–527, 2016.
- [15] Benedict Paten, Adam M Novak, Jordan M Eizenga, and Erik Garrison. Genome graphs and the evolution of genome inference. *Genome research*, 27(5):665–676, 2017.
- [16] Ilia Minkin, Son Pham, and Paul Medvedev. Twopaco: An efficient algorithm to build the compacted de bruijn graph from many complete genomes. *Bioinformatics*, page btw609, 2016.
- [17] Rayan Chikhi, Antoine Limasset, and Paul Medvedev. Compacting de bruijn graphs from sequencing data quickly and in low memory. *Bioinformatics*, 32(12):i201–i208, 2016.