Where the patterns are: repetition-aware compression for colored de Bruijn Graphs

Alessio Campanelli[1], **Giulio Ermanno Pibiri**[1], **Jason Fan**[2], **Rob Patro**[2]
[1]Ca' Foscari University of Venice
[2]University of Maryland

19th Workshop on Compression, Text, and Algorithms (WCTA 2024)
Puerto Vallarta, Jalisco, México – September 26th, 2024

The colored *k*-mer indexing problem

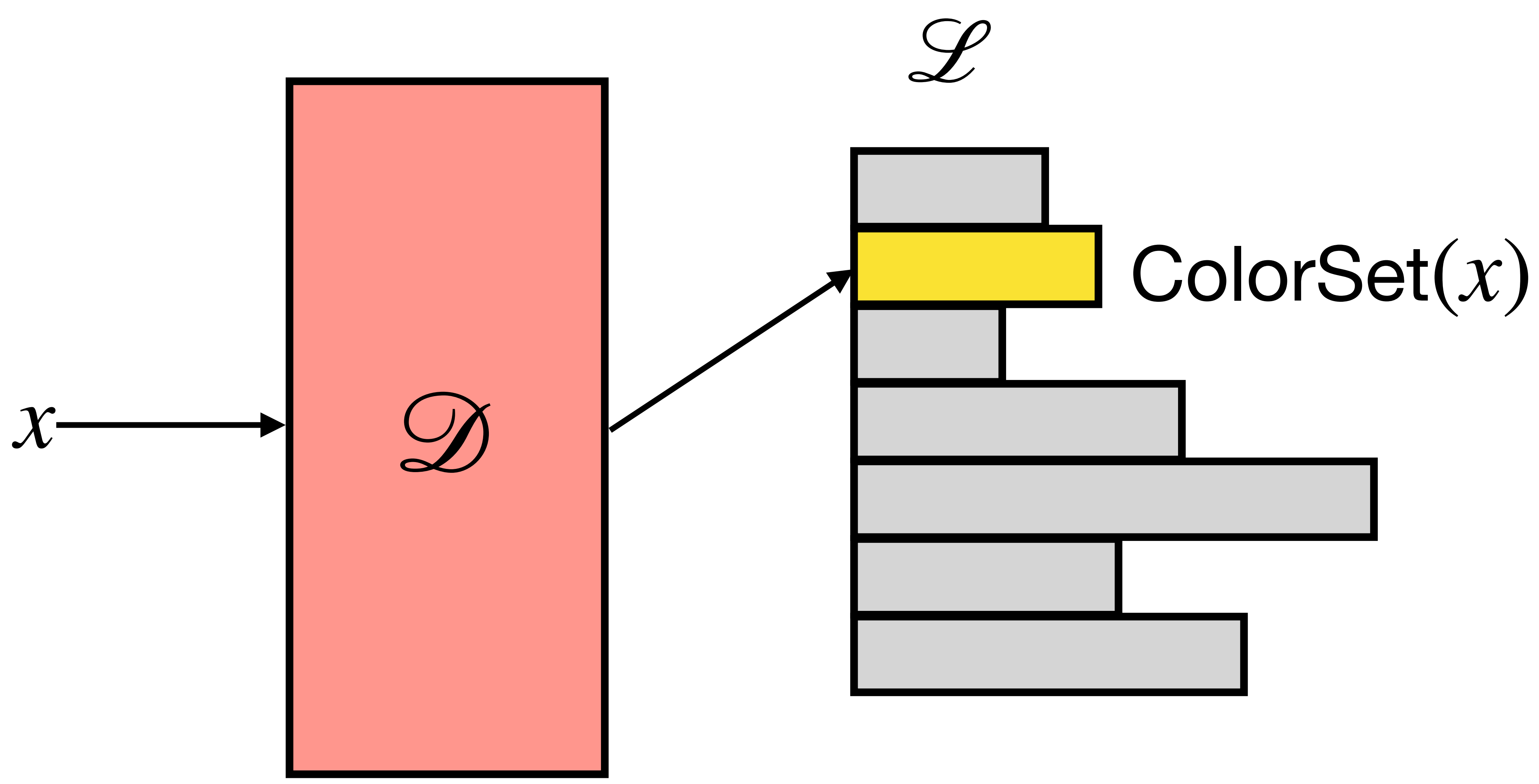
- We are given a collection $\mathscr{R} = \{R_1, \ldots, R_N\}$ of reference sequences. Each R_i is a (long) sequence over the DNA alphabet {A, C, G, T}
- **Problem.** We want to build an index for $\mathscr{R}$ so that we can retrieve ColorSet$(x) = \{i \,|\, x \in R_i\}$ for any *k*-mer x
- A ***k*-mer** is a string of length k

The colored *k*-mer indexing problem

- We are given a collection $\mathscr{R} = \{R_1, \ldots, R_N\}$ of reference sequences. Each R_i is a (long) sequence over the DNA alphabet {A, C, G, T}
- **Problem.** We want to build an index for $\mathscr{R}$ so that we can retrieve ColorSet$(x) = \{i \,|\, x \in R_i\}$ for any *k*-mer x
- A ***k*-mer** is a string of length k
- A lot of hype in the indexing community for the case where $\mathscr{R}$ is a **pangenome**: a collection of related genomes
- Relevant for applications where sequences are first matched against known references.

Modular indexing layout

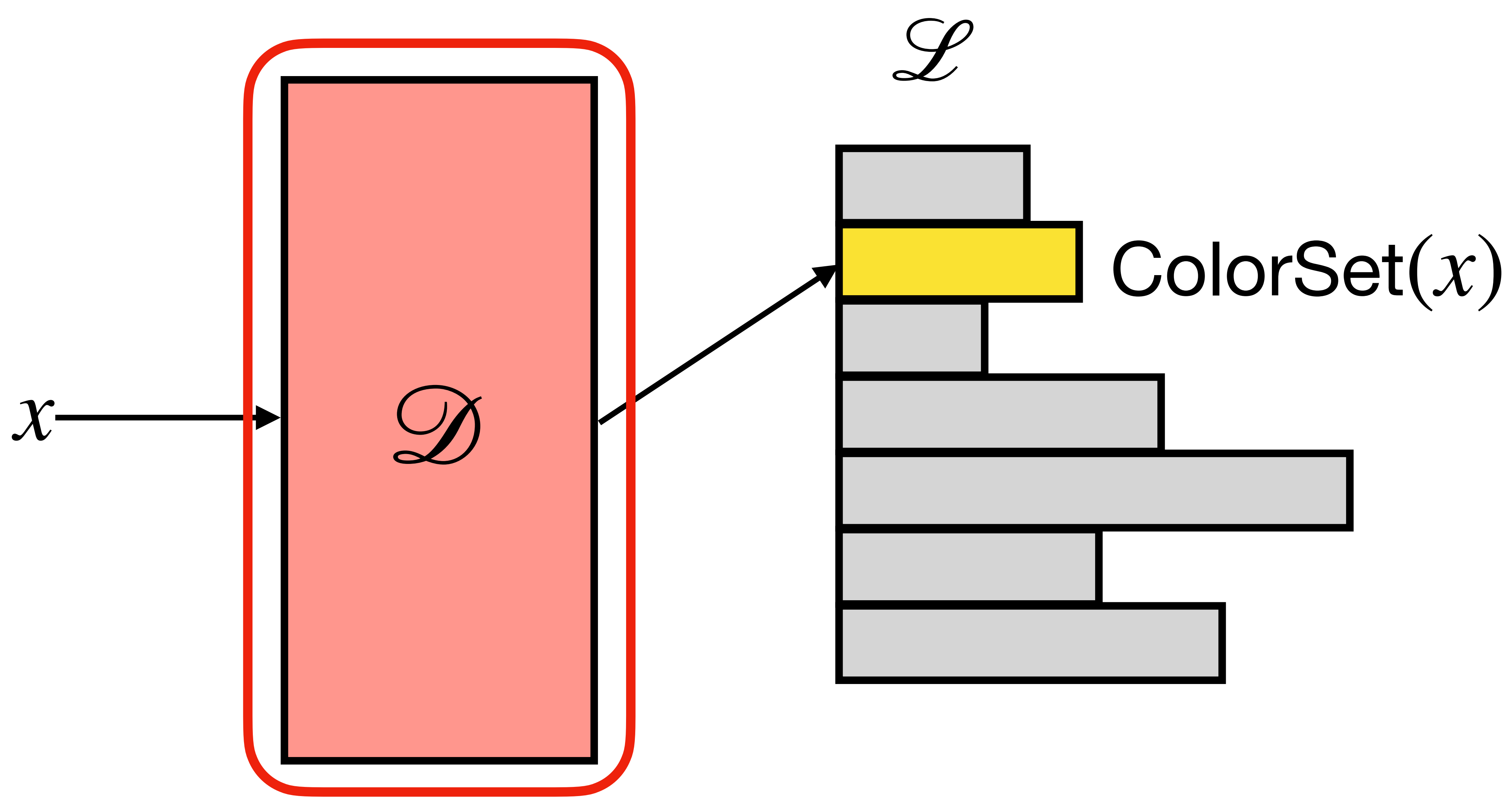
- **Goal.** We want to build the map $x \to \text{ColorSet}(x) = \{i \mid x \in R_i\}$
- Two data structures:



Modular indexing layout

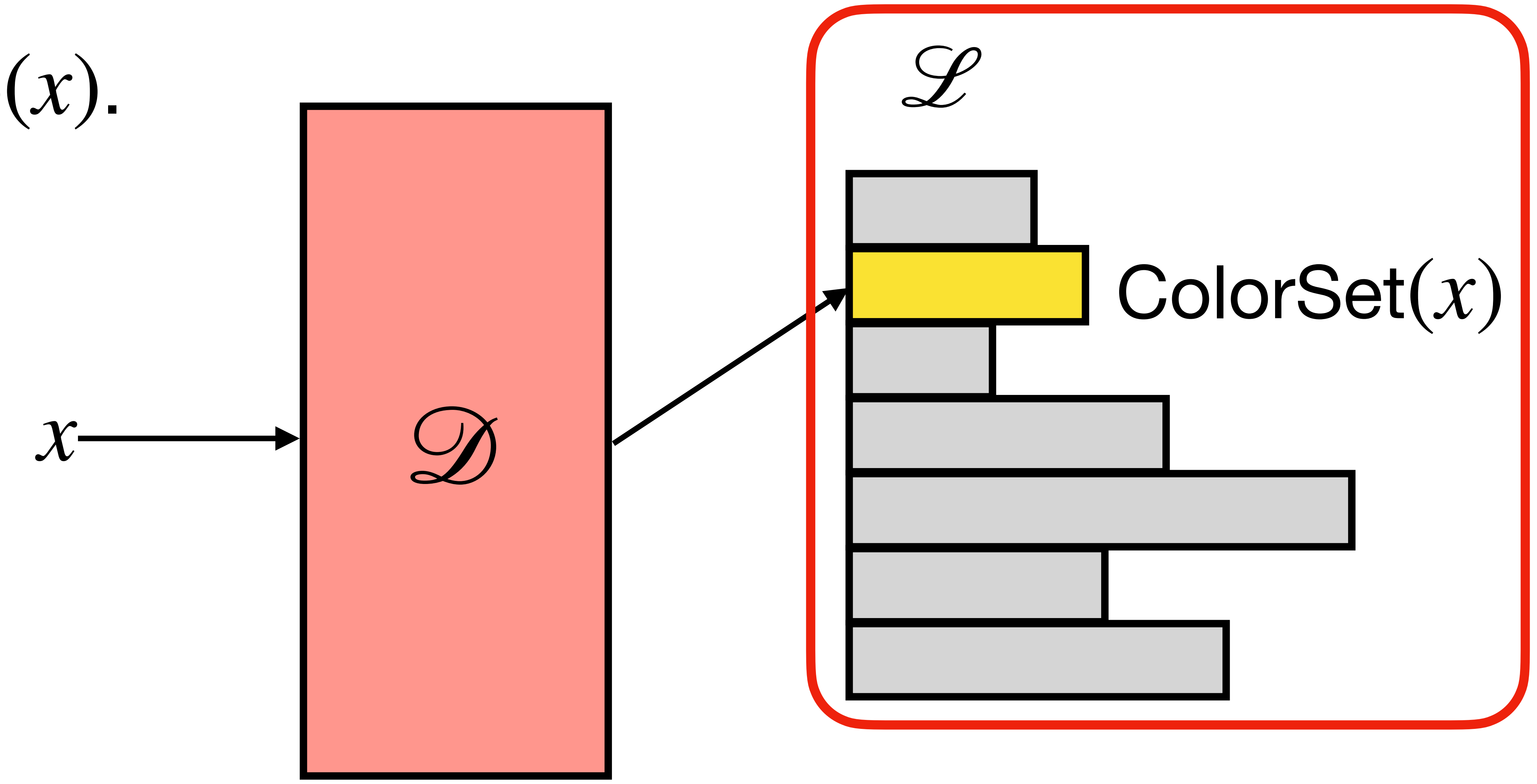
- **Goal.** We want to build the map $x \rightarrow \text{ColorSet}(x) = \{i \,|\, x \in R_i\}$
- Two data structures:

   1. A **dictionary** $\mathscr{D}$ that stores all distinct *k*-mers in $\mathscr{R} = \{R_1, \ldots, R_N\}$. $\mathscr{D}$ stores n distinct *k*-mers and supports $\text{Lookup}(x) = h \in [1,n]$



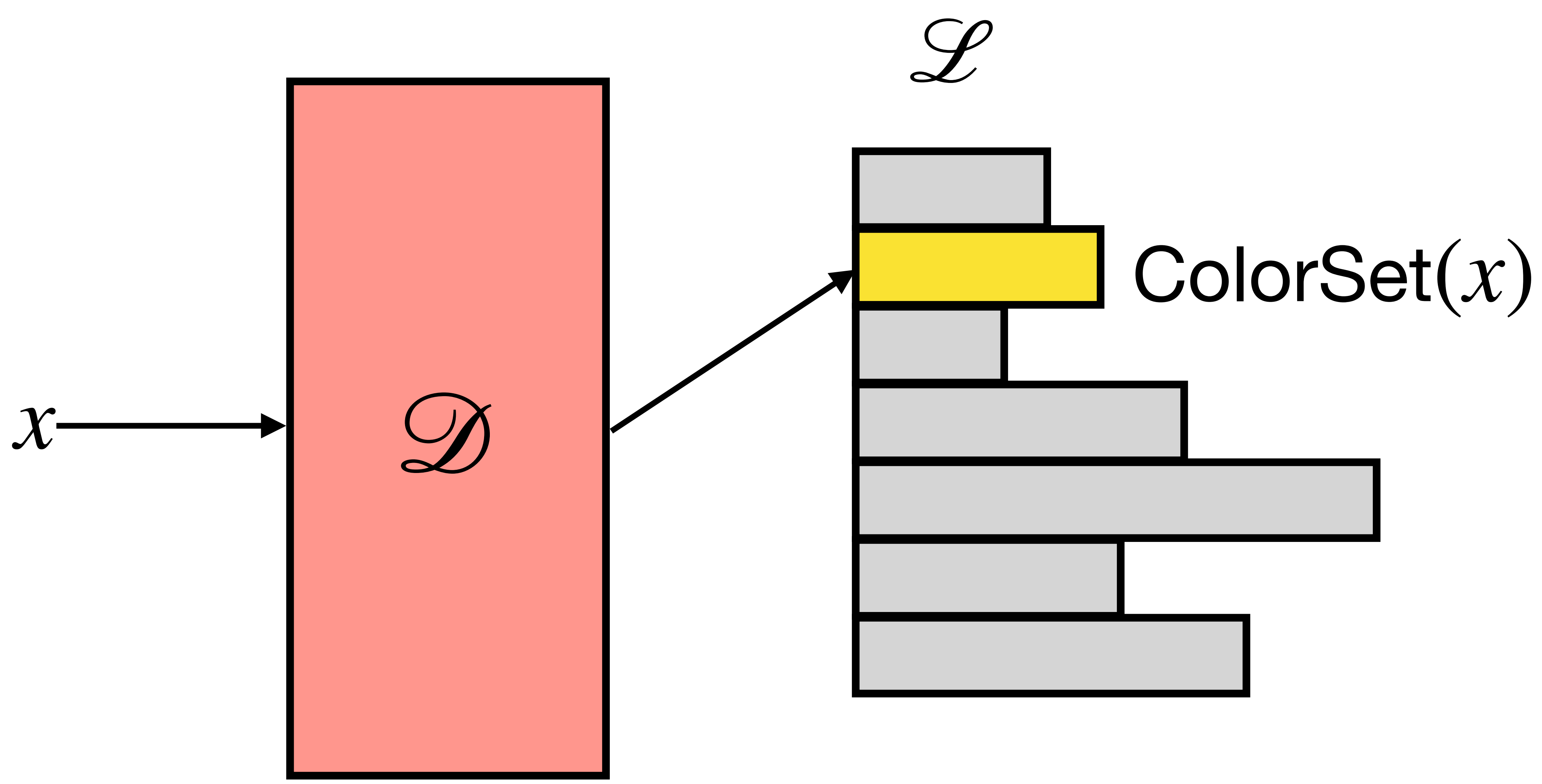
Modular indexing layout

- **Goal.** We want to build the map $x \rightarrow \text{ColorSet}(x) = \{i \,|\, x \in R_i\}$
- Two data structures:
  1. A **dictionary** $\mathscr{D}$ that stores all distinct *k*-mers in $\mathscr{R} = \{R_1, \ldots, R_N\}$.
     $\mathscr{D}$ stores n distinct *k*-mers and supports $\text{Lookup}(x) = h \in [1,n]$
  2. An **inverted index** $\mathscr{L}$ that stores all $\{\text{ColorSet}(x)\}_x$ in the order given by $\text{Lookup}(x)$.



Modular indexing layout

- The initial problem reduces to that of representing the two data structures $\mathscr{D}$ and $\mathscr{L}$.
- To do so at best, we must understand and exploit the properties of our problem



k-mers

- Since we take all consecutive *k*-mers from our references, they **share *(k-1)*-symbol overlaps**

```
ACGGTAGAACCGATTCAAATTCGACGTAGC…
ACGGTAGAACCGA
 CGGTAGAACCGAT
  GGTAGAACCGATT
   GTAGAACCGATTC                     k = 13
    TAGAACCGATTCA
     AGAACCGATTCAA
      GAACCGATTCAAA
       …
```

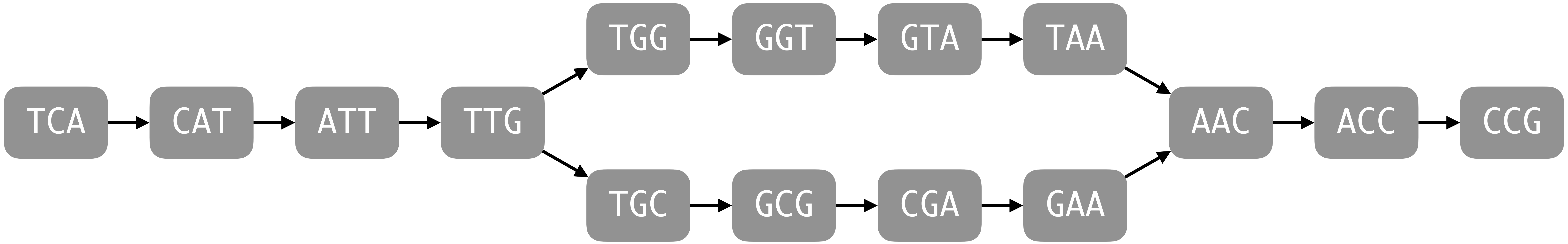
- Two important consequences:

1. It is very likely that, given a *k*-mer x in a query sentence Q, and its answer returned from the index, *next*(x) has a very similar answer (**compression of satellite data**)

2. Given the answer to x, the answer to *next*(x) can be computed faster than the answer for any arbitrary *k*-mer $y \neq$ *next*(x) (**faster query time**)

de Bruijn Graphs

- The dictionary $\mathscr{D}$ is a set of *k*-mers with *(k-1)*-symbol overlaps
- One-to-one correspondence between $\mathscr{D}$ and a *de Bruijn Graph* (dBG)

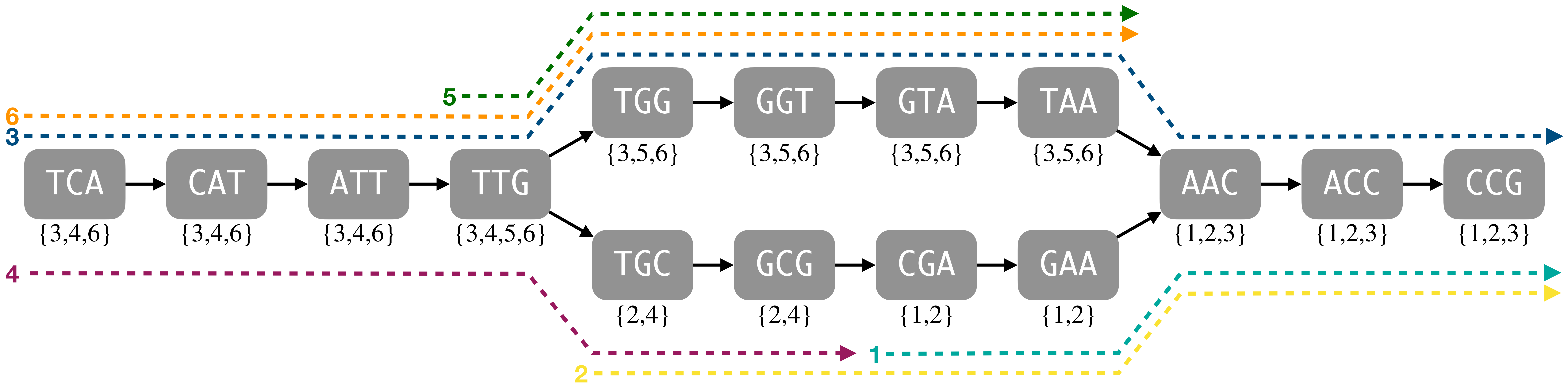


Example for $k = 3$

Colored de Bruijn Graphs

- Colored de Bruijn Graphs store the information of the references (colors) in which each *k*-mer appears

- References in $\mathscr{R}$ are spelled by **paths** in the graph

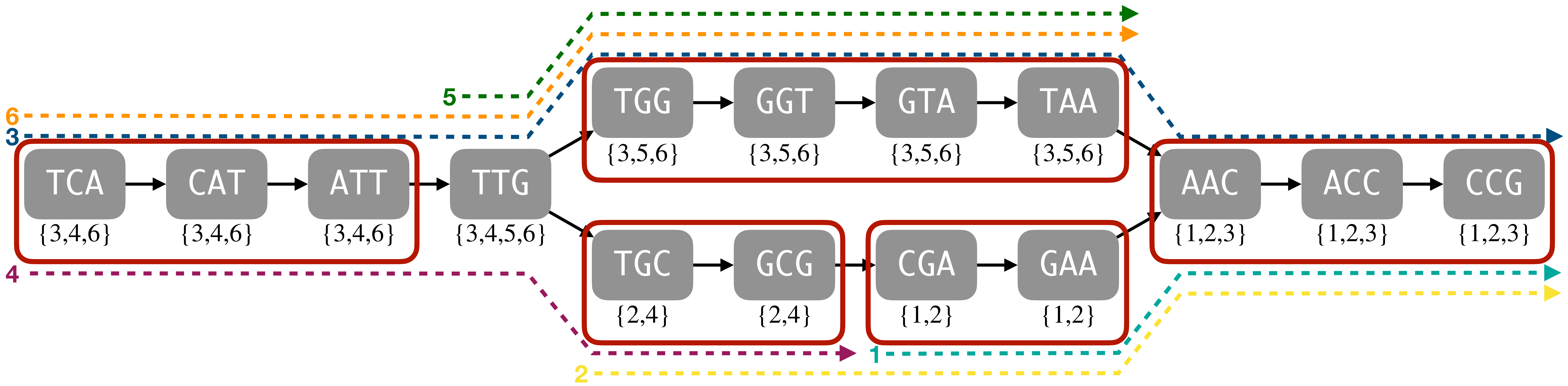


Example for $k = 3$ and $N = 6$ references

Colored de Bruijn Graphs

- Colored de Bruijn Graphs store the information of the references (colors) in which each *k*-mer appears

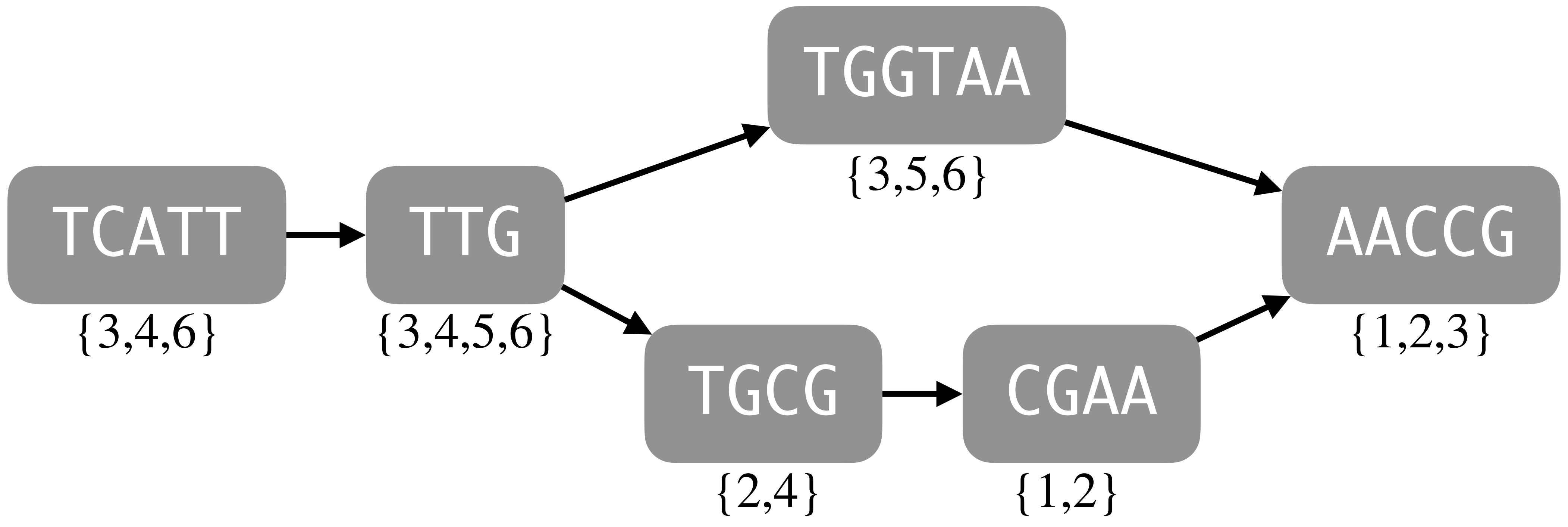
- References in $\mathscr{R}$ are spelled by **paths** in the graph



Example for $k = 3$ and $N = 6$ references

Colored compacted de Bruijn Graphs

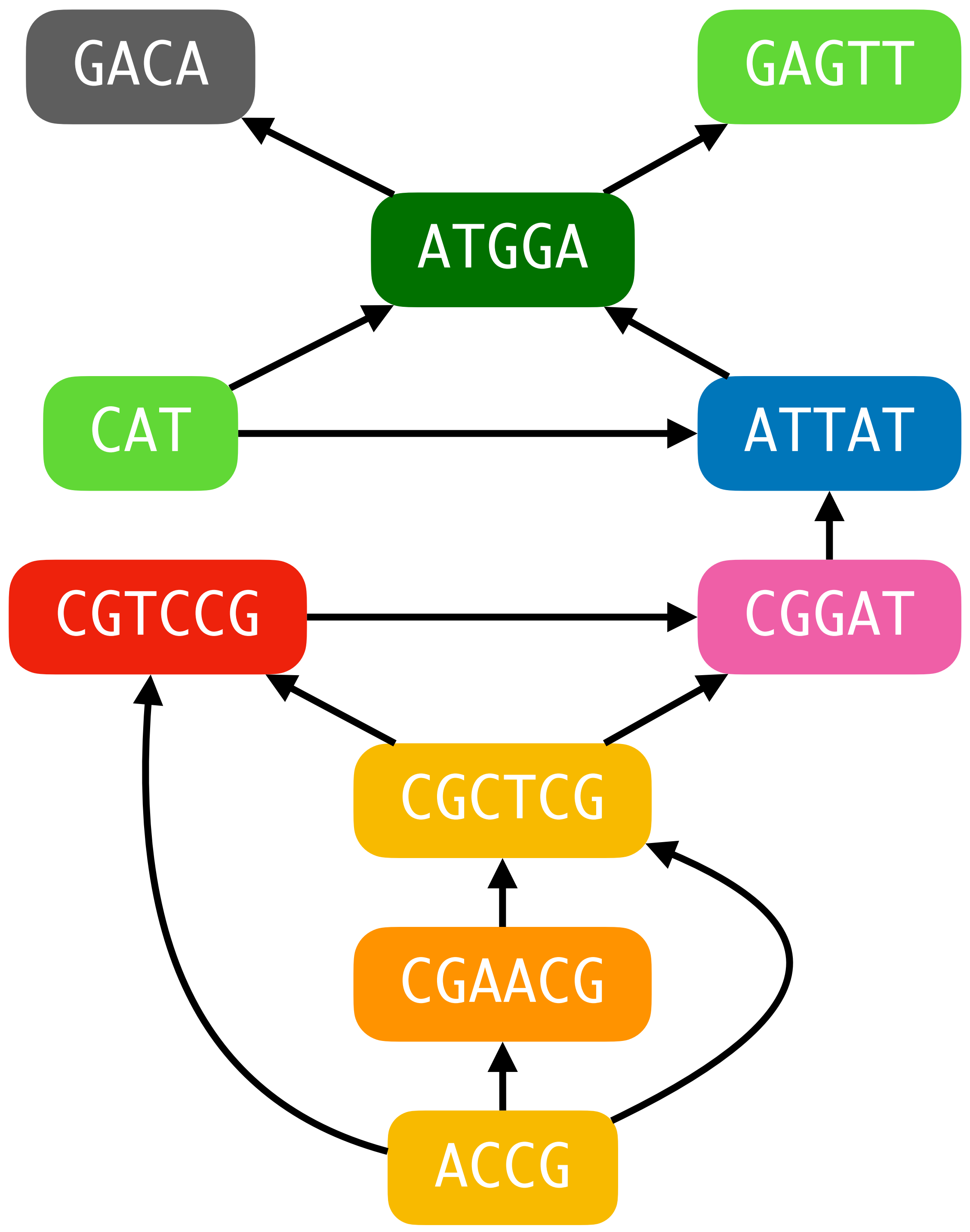
- Paths having the same color along non branching paths can be collapsed into **unitigs**



Example for $k = 3$ and $N = 6$ references

Colored compacted de Bruijn Graphs

- Let's now consider the properties of colored compacted dBGs (c-dBG) and how they can be exploited for efficient indexing.



Larger example for $k = 3$ and $N = 16$ references

- This example will be used in the following discussion
- Nodes with the same color have equal color sets

Properties of c-dBGs (1)

Unitigs spell references in $\mathscr{R}$

- We can represent the set of unitigs instead of the set of *k*-mers.
- Better space effectiveness and cache locality

$\mathscr{D}$

u_1	ACCG
u_2	CGCTCG
u_3	CGAACG
u_4	CGTCCG
u_5	CGGAT
u_6	ATTAT
u_7	CAT
u_8	GAGTT
u_9	ATGGA
u_{10}	GACA

- $\mathscr{D}$ is represented with SSHash [Pibiri, 2022]
- SSHash stores a set of unitigs in any wanted order (*order-preserving*)

Properties of c-dBGs (2)

Unitigs are monochromatic

- We store a color set for each unitig, rather than for each *k*-mer because $\mathit{ColorSet}(x) = \mathit{ColorSet}(y)$ if *k*-mers x and y ar part of the same unitig

- Thus, we need an **efficient map from *k*-mers to unitigs**

$\mathscr{D}$

u_1 ACCG
u_2 CGCTCG
u_3 CGAACG
u_4 CGTCCG
u_5 CGGAT
u_6 ATTAT
u_7 CAT
u_8 GAGTT
u_9 ATGGA
u_{10} GACA

$\mathscr{L}$

C_1 = [3,4,5,9,10,11,13,15]
C_2 = [2,3,15]
C_3 = [1,3,5,7,9,10,11]
C_4 = [1,3,5,7,9,11,13]
C_5 = [1,3,6,7,9,11,12,13,14,16]
C_6 = [6,8]
C_7 = [1,3,8,11,12,13,14,16]
C_8 = [12,16]

- $\mathscr{D}$ is represented with SSHash [Pibiri, 2022]

- SSHash stores a set of unitigs in any wanted order, so it is easy to compute the unitig identifier $\mathit{unitig}(x)$ of any given *k*-mer x

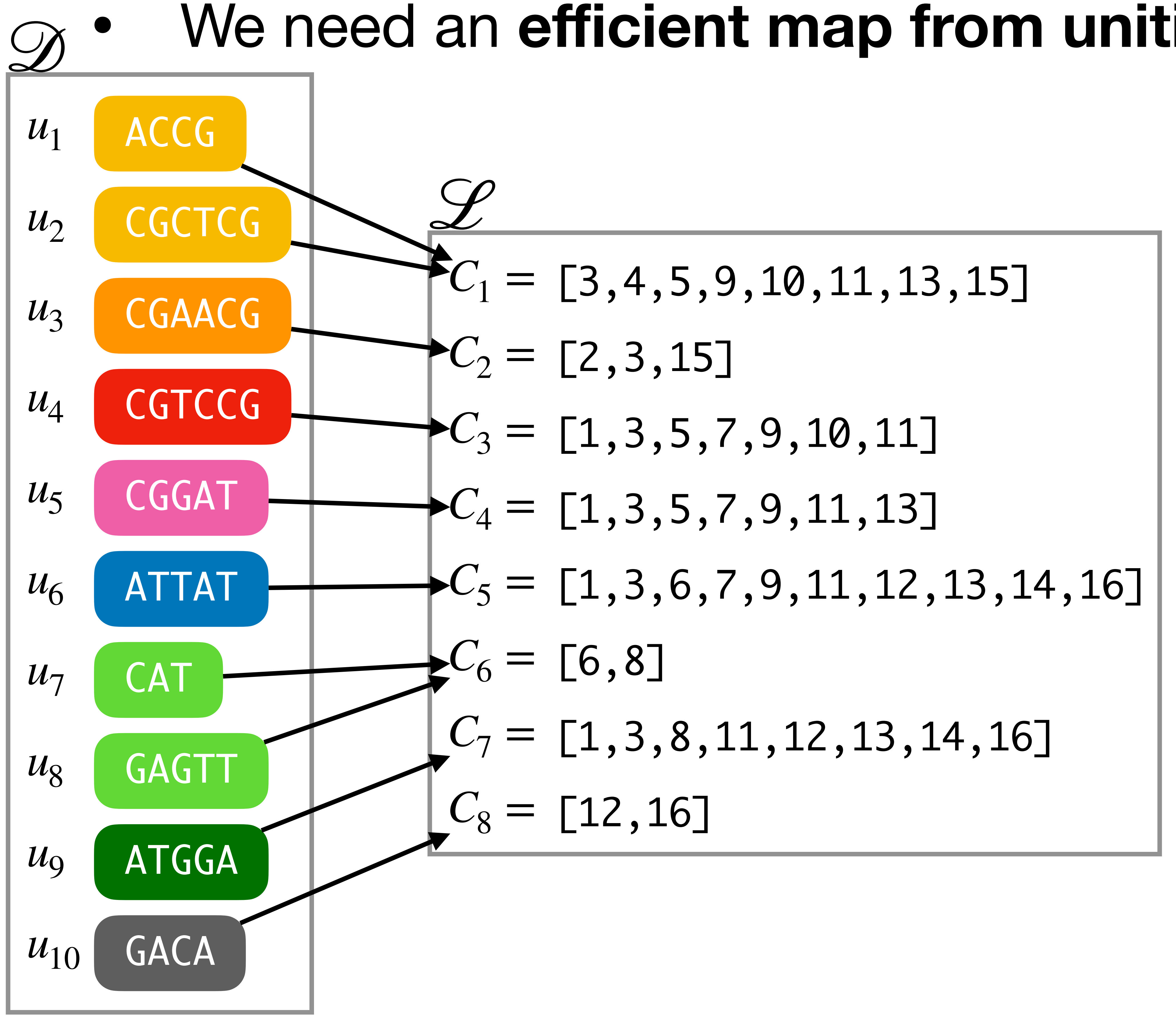
- The inverted index $\mathscr{L}$ stores $\mathit{ColorSet}(x)$ for each unitig in the order given by $\mathit{unitig}(x)$

Properties of c-dBGs (3)

Unitigs co-occur

- Distinct unitigs often have the same color set, i.e. they co-occur in the same subset of references → there are way less distinct color sets than unitigs

- We need an **efficient map from unitigs to color sets**



- $\mathscr{D}$ is represented with SSHash [Pibiri, 2022]

- SSHash stores a set of unitigs in any wanted order, so we can permute unitigs in $\mathscr{D}$ so that consecutive unitigs have the same color

- Then, mapping a unitig to its color set is a simple Rank query over a bitmap

Experimental results

- These observations have been implemented in a tool called **Fulgor** [Fan et al. 2023].
- Fulgor achieves faster query times using less memory than other similar tools.

Dataset	Fulgor		Themisto		MetaGraph-B		MetaGraph-NB		COBS	
	mm:ss	GB	h:mm:ss	GB	mm:ss	GB	h:mm:ss	GB	h:mm:ss	GB
EC	2:10	1.67	3:40	2.46	22:00	30.44	1:05:41	0.40	45:11	34.93
SE-5K	1:10	0.80	3:50	1.82	14:14	36.54	20:32	0.33	38:34	41.93
SE-10K	2:20	2.06	7:35	4.16	28:15	92.18	43:40	0.61	1:01:14	84.20
SE-50K	12:00	18.24	42:02	33.14	NA	NA	4:30:03	2.72	3:54:18	408.82
SE-100K	24:00	42.20	1:22:00	75.93	NA	NA	9:40:06	4.82	8:07:29	522.56
SE-150K	37:00	70.55	2:00:13	124.27	NA	NA	NA	NA	7:47:14	522.63
GB	1:10	36.01	1:20	48.47	28:55	15.86	22:05	9.91	34:45	225.57

The times are the result of the command `/usr/bin/time` when executing a pseudoalignment query

Our contribution

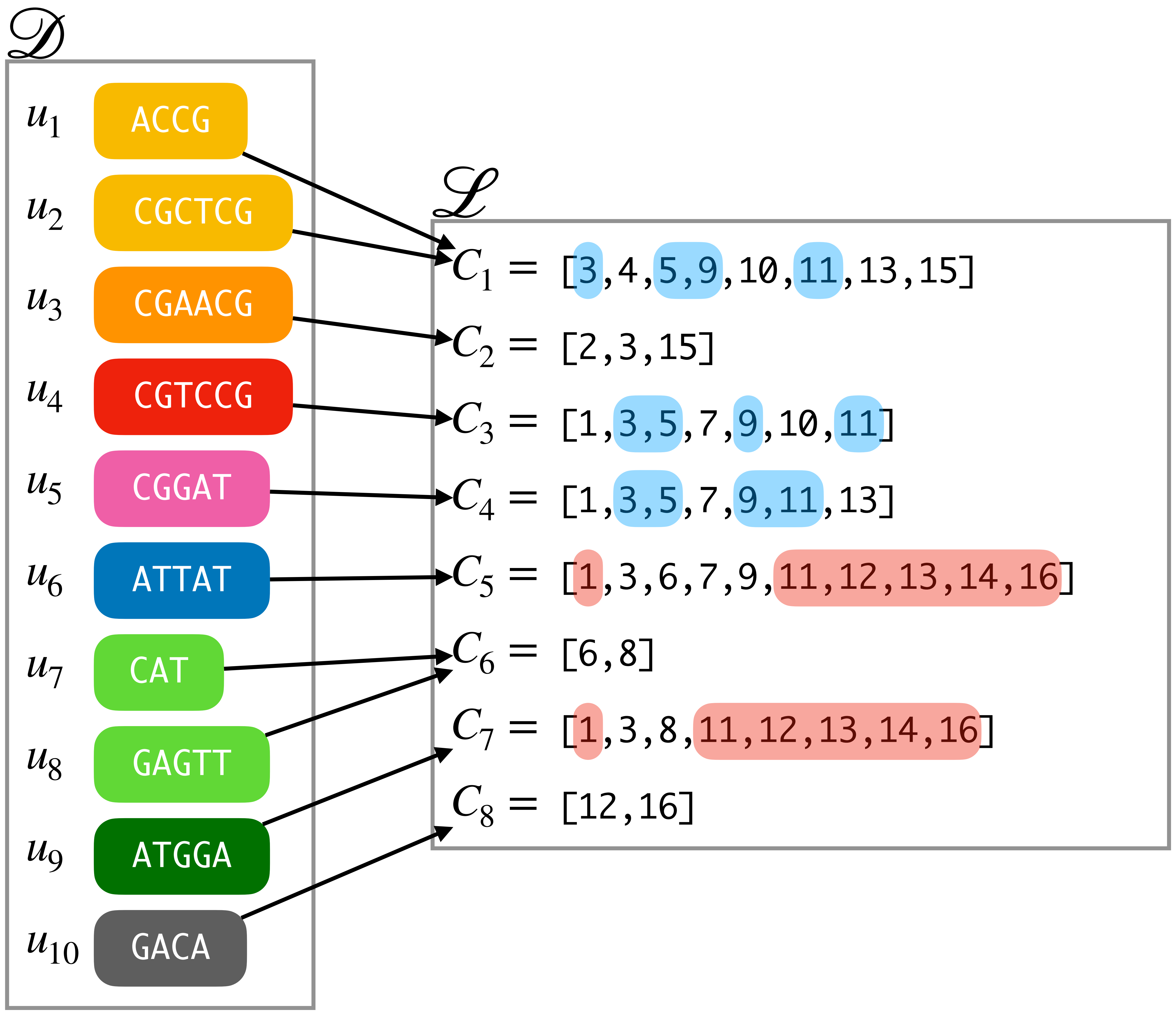
Another important property

1. Unitigs spell references in $\mathscr{R}$.
2. Unitigs are monochromatic.
3. Unitigs co-occur.
4. **Colors are similar when indexing pangenomes.**
    - Implies that it is possible to achieve **better compression** if colors are not compressed individually, but repetitive patterns are factored out and compressed once.

Properties of c-dBGs (4)

Colors are similar when indexing pangenomes

- Consider the example from before

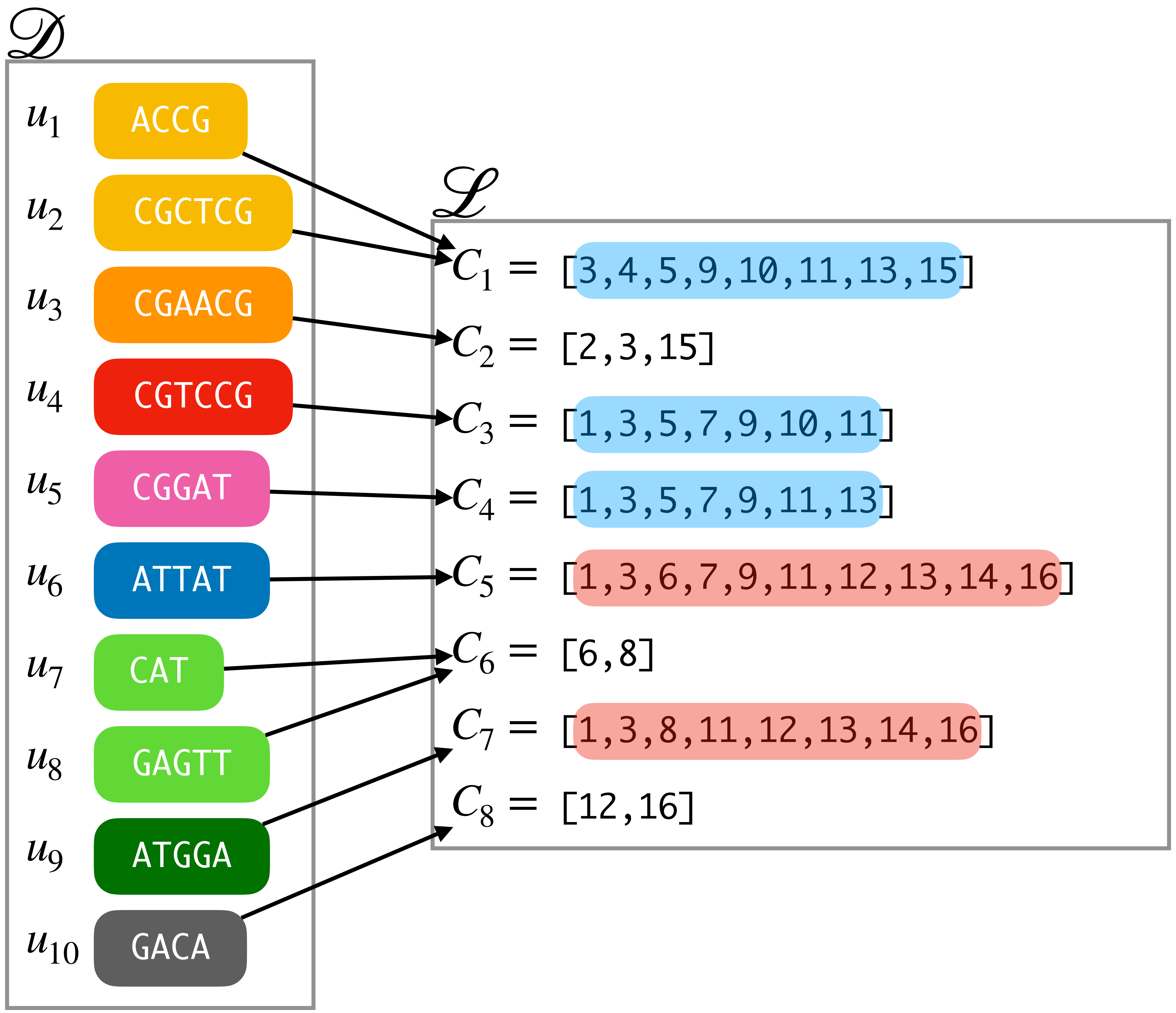


- The pattern $\{3,5,9,11\}$ is represented three times
- The pattern $\{1,11,12,13,14,16\}$ is represented twice

Properties of c-dBGs (4)

Colors are similar when indexing pangenomes

- Consider the example from before



- Similarly, color sets C_1, C_3, and C_4 have very few differences
- The same is true for color sets C_5 and C_7

Two ways of partitioning color sets

$C_1 = [3,4,5,9,10,11,13,15]$

$C_2 = [2,3,15]$

$C_3 = [1,3,5,7,9,10,11]$

$C_4 = [1,3,5,7,9,11,13]$

$C_5 = [1,3,6,7,9,11,12,13,14,16]$

$C_6 = [6,8]$

$C_7 = [1,3,8,11,12,13,14,16]$

$C_8 = [12,16]$

Horizontal partitioning

$C_1 = [3,4,5,9,10,11,13,15]$

$C_2 = [2,3,15]$

$C_3 = [1,3,5,7,9,10,11]$

$C_4 = [1,3,5,7,9,11,13]$

$C_5 = [1,3,6,7,9,11,12,13,14,16]$

$C_6 = [6,8]$

$C_7 = [1,3,8,11,12,13,14,16]$

$C_8 = [12,16]$

Vertical partitioning

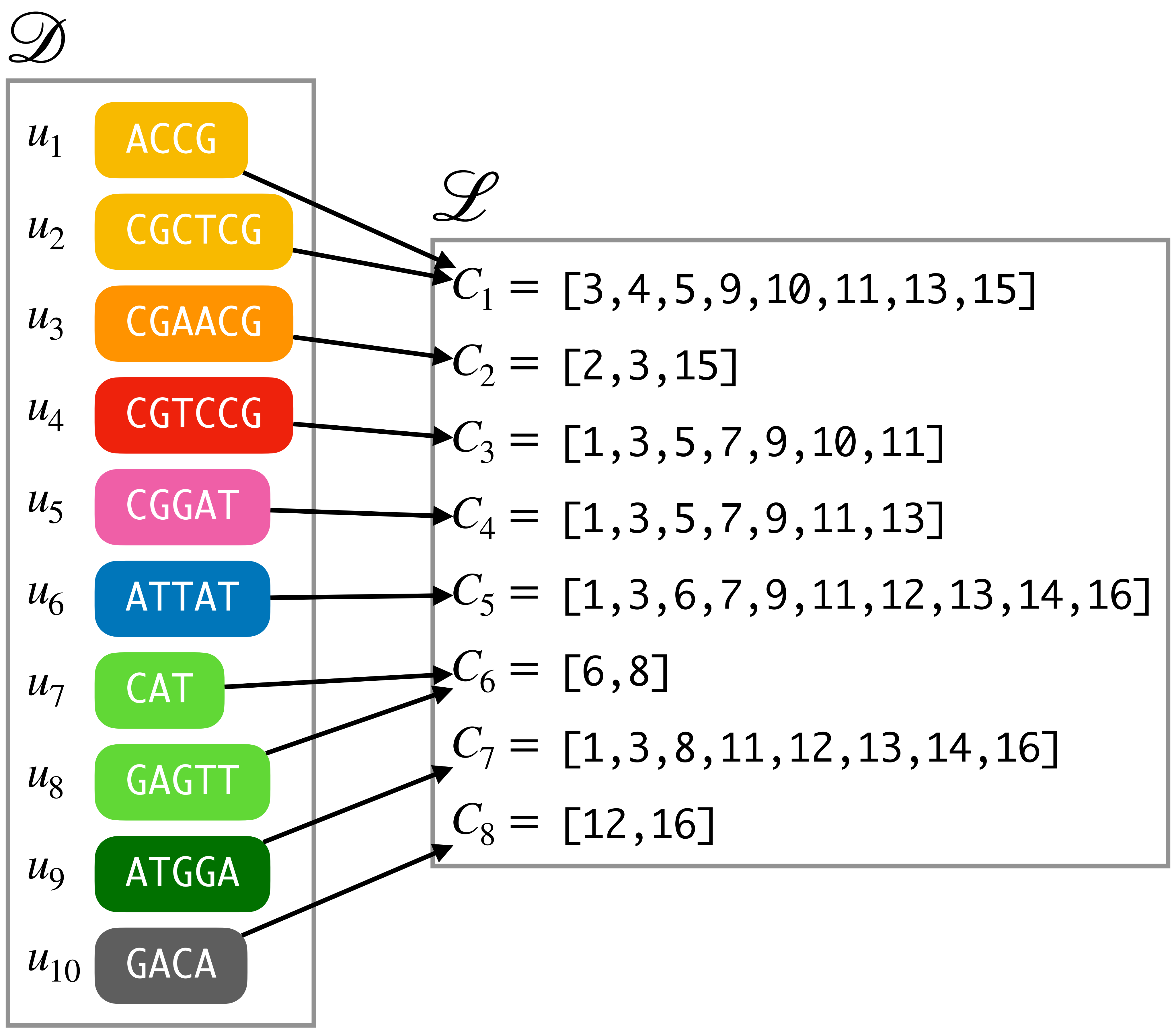
Horizontal partitioning (d-Fulgor)

- Let z be the number of distinct color sets
- Determine a partition of $[z] = \{1,...,z\}$ so that color sets in the same partition are similar
- **Intuition:** similar color sets share most of their colors. Create a **representative** color set for each partition that expresses its repetitiveness.
- All color sets within a partition are encoded as the **symmetric difference** between the original color set and the representative set.

Horizontal partitioning - Example

- Example for $N = 16$ references, $z = 8$ color sets and 3 horizontal partitions

$\{1,3,4\}\{2,6\}\{5,7,8\}$

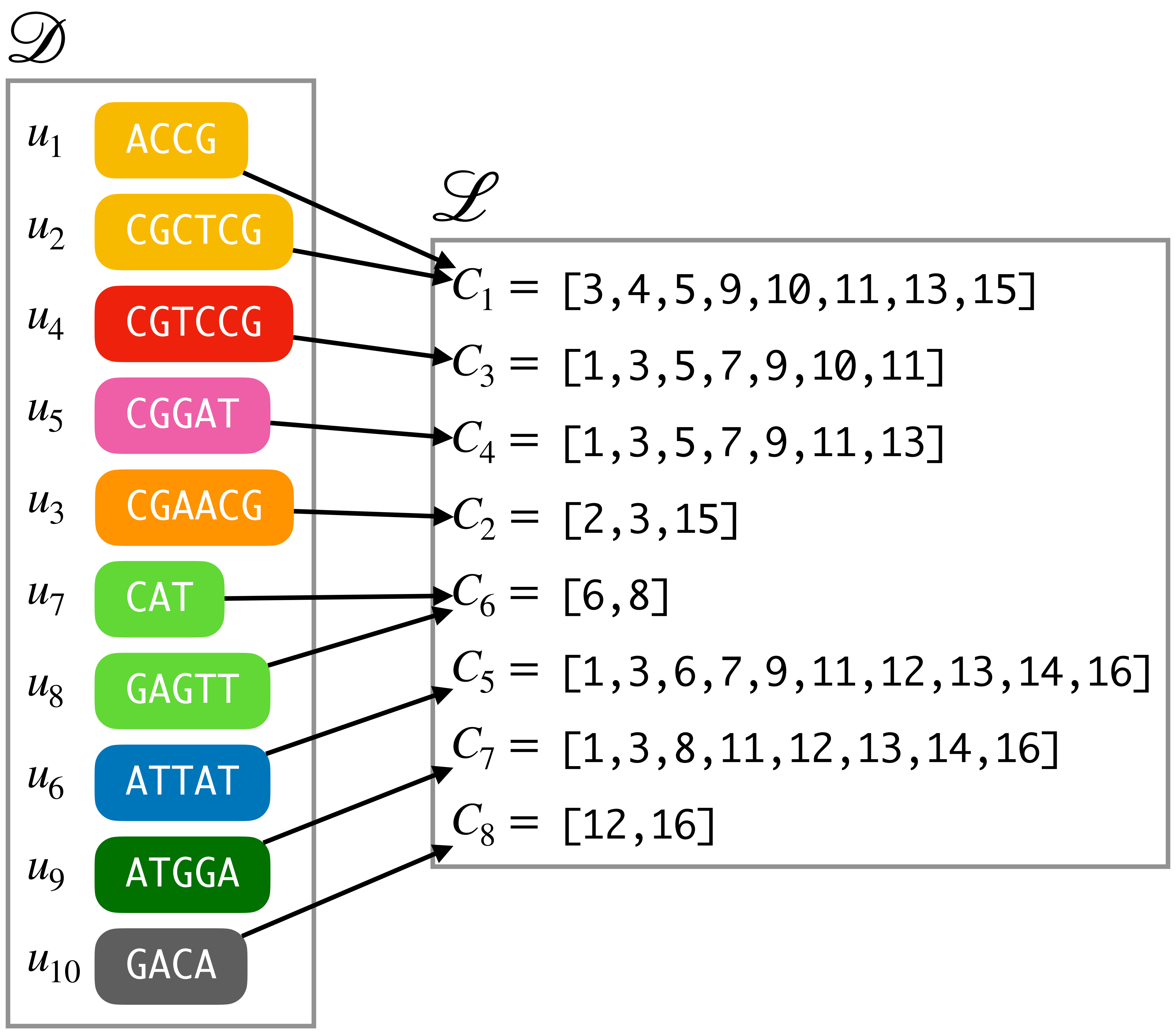


1. Permute unitigs and color sets they map to according to the partition

Horizontal partitioning - Example

- Example for $N = 16$ references, $z = 8$ color sets and 3 horizontal partitions

$\{1,3,4\}\{2,6\}\{5,7,8\}$

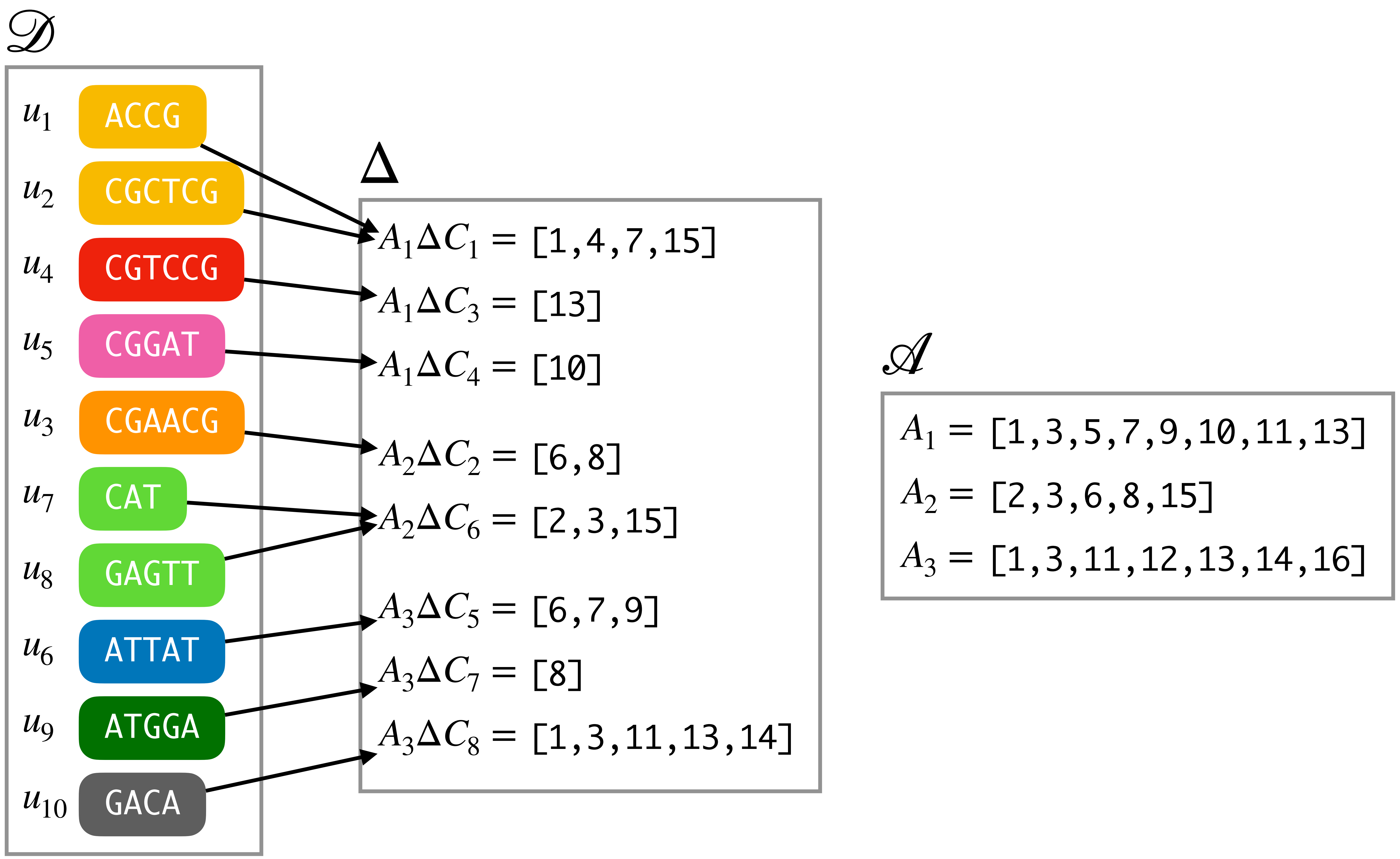


2. Generate representative sets ($\mathscr{A}$) and differential color sets ($A_i \Delta C_j$)

Horizontal partitioning - Example

- Example for $N = 16$ references, $z = 8$ color sets and 3 horizontal partitions

$$\{1,3,4\}\{2,6\}\{5,7,8\}$$



Vertical partitioning (m-Fulgor)

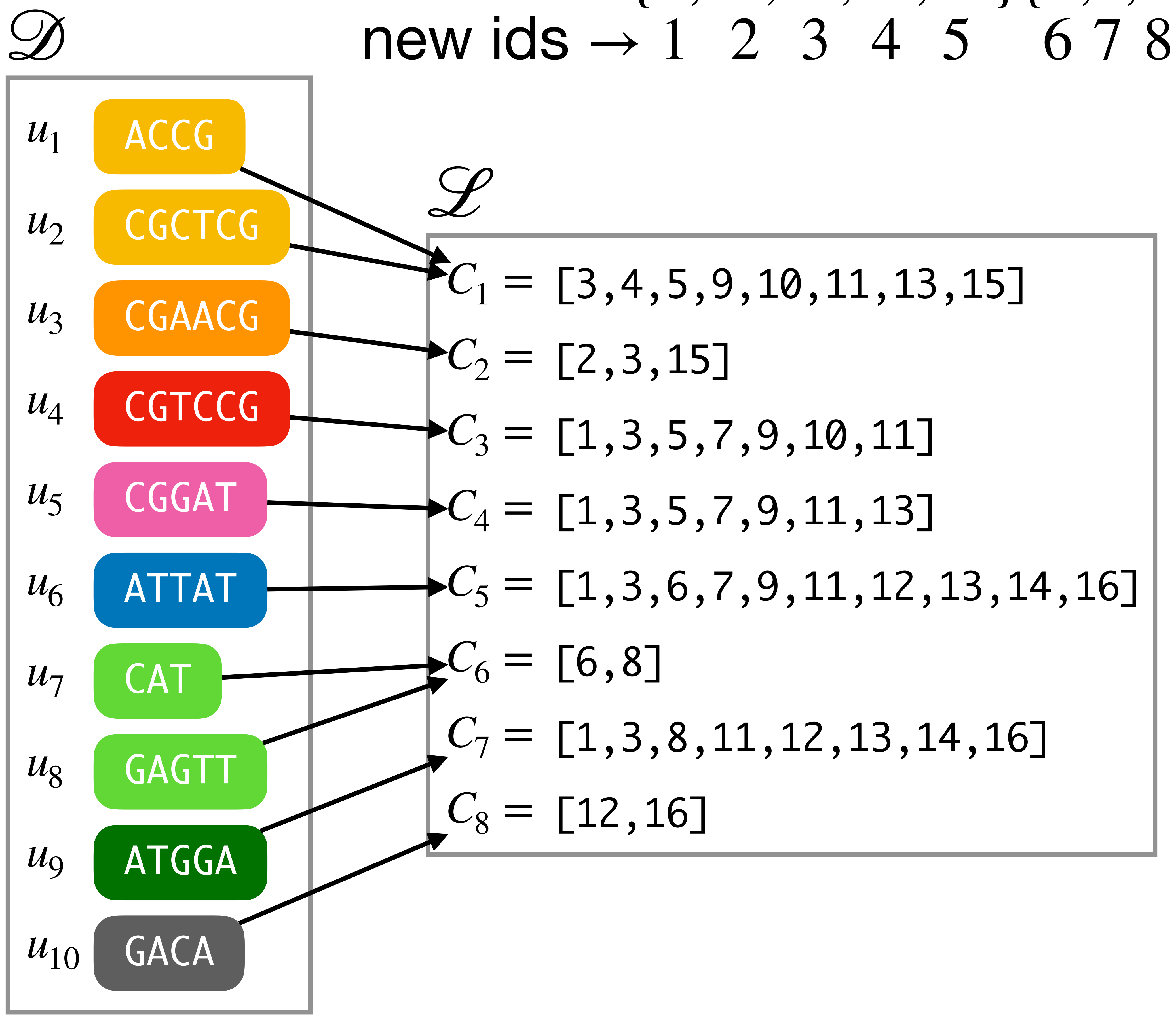
- Recall that N is the number of references in $\mathscr{R}$
- Similarly, determine a partition of $[N] = \{1,\ldots,N\}$ so that references in the same partition are similar
- **Intuition:** similar references induce similar color sets. Sequences of repeating colors in a partition are stored as **partial** color sets
- Color sets can be represented as sequences of references, or **meta** color sets, to those partial sets.

Vertical partitioning - Example

- Example for $N = 16$ references, $z = 8$ color sets and 4 vertical partitions

$\{1,12,13,14,16\}\{3,5,9\}\{7,11\}\{2,\ 4,\ 6,\ 8,\ 10,\ 15\}$

new ids → $1\ \ 2\ \ 3\ \ 4\ \ 5\quad 6\ 7\ 8\quad 9\ 10\quad 11\ \ 12\ \ 13\ \ 14\ \ 15\ \ 16$



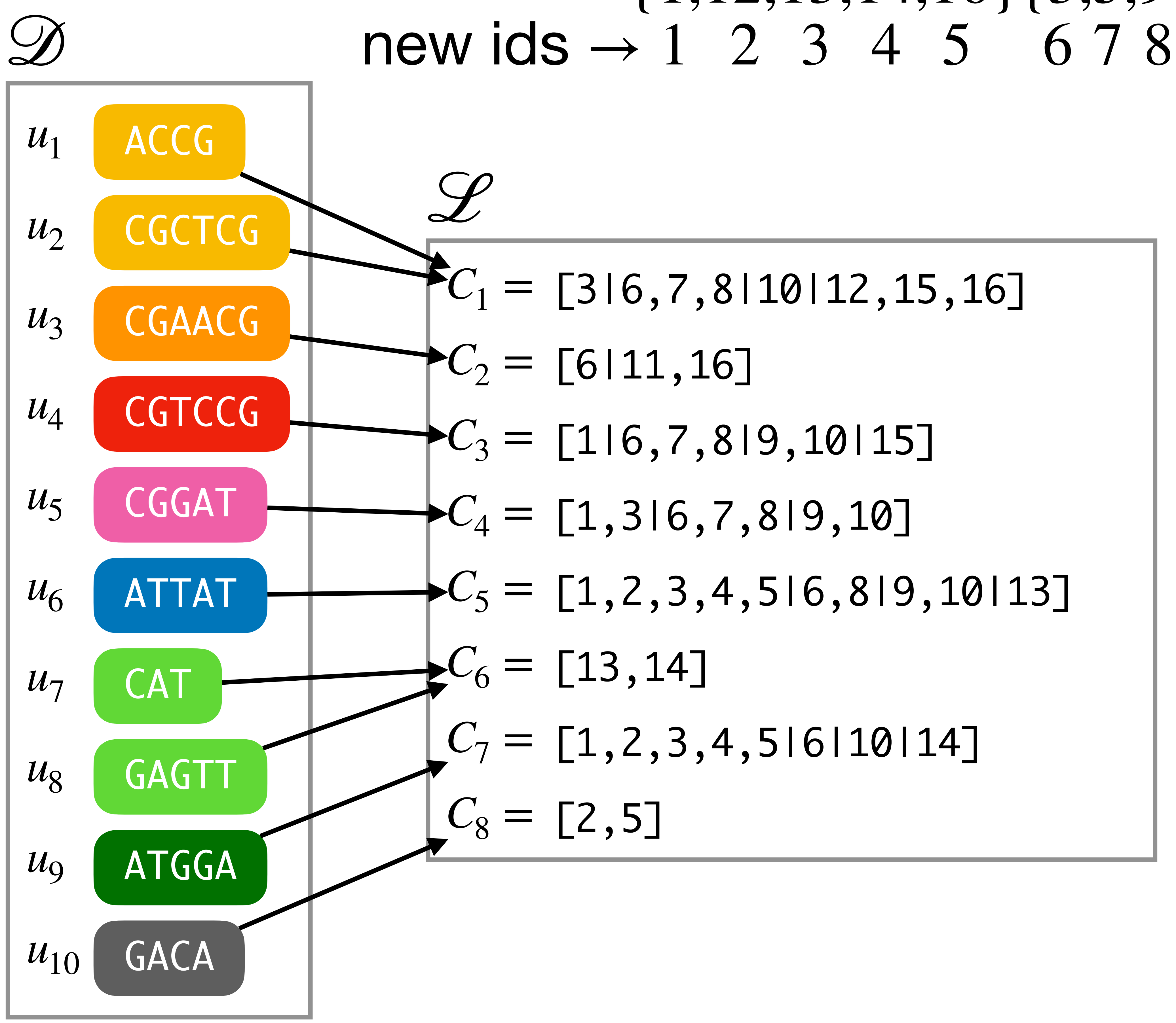
1. Assign the new identifiers to the color sets and sort them

Vertical partitioning - Example

- Example for $N = 16$ references, $z = 8$ color sets and 4 vertical partitions

$$\{1,12,13,14,16\}\{3,5,9\}\{7,11\}\{2,\ 4,\ 6,\ 8,\ 10,\ 15\}$$

new ids $\rightarrow$ $1\ \ 2\ \ 3\ \ 4\ \ 5\ \ \ 6\ 7\ 8\ \ \ 9\ 10\ \ \ 11\ \ 12\ \ 13\ \ 14\ \ 15\ \ 16$



2. Compute the partial color sets

Vertical partitioning - Example

- Example for $N = 16$ references, $z = 8$ color sets and 4 vertical partitions

$\{1,12,13,14,16\}\{3,5,9\}\{7,11\}\{2,\ 4,\ 6,\ 8,\ 10,\ 15\}$

new ids → 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

2. Compute the partial color sets

$\mathscr{D}$

u_1	ACCG
u_2	CGCTCG
u_3	CGAACG
u_4	CGTCCG
u_5	CGGAT
u_6	ATTAT
u_7	CAT
u_8	GAGTT
u_9	ATGGA
u_{10}	GACA

$\mathscr{L}$

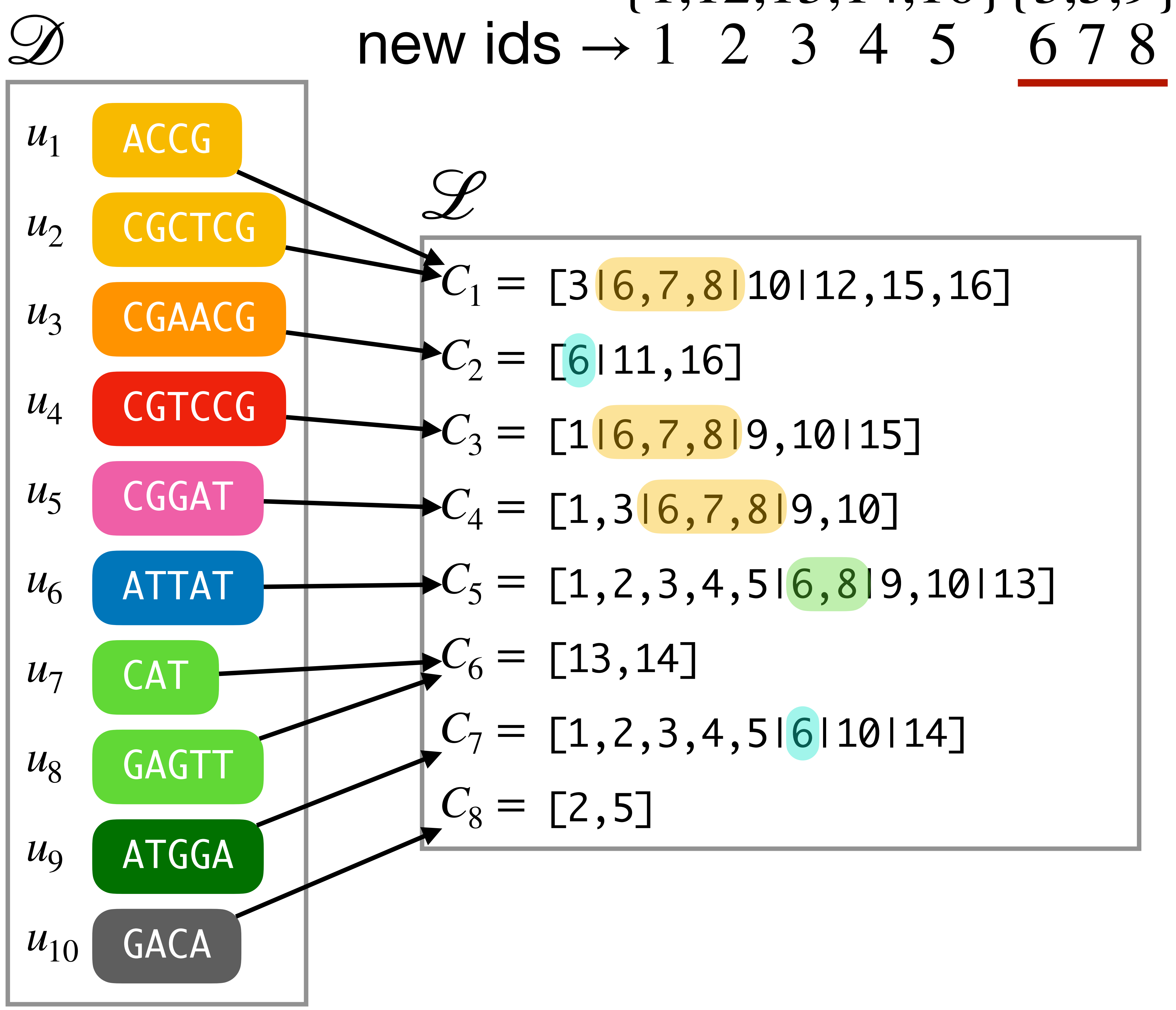
C_1 = [3|6,7,8|10|12,15,16]
C_2 = [6|11,16]
C_3 = [1|6,7,8|9,10|15]
C_4 = [1,3|6,7,8|9,10]
C_5 = [1,2,3,4,5|6,8|9,10|13]
C_6 = [13,14]
C_7 = [1,2,3,4,5|6|10|14]
C_8 = [2,5]

Vertical partitioning - Example

- Example for $N = 16$ references, $z = 8$ color sets and 4 vertical partitions

$\{1,12,13,14,16\}\{3,5,9\}\{7,11\}\{2,\ 4,\ 6,\ 8,\ 10,\ 15\}$

new ids → $1\ \ 2\ \ 3\ \ 4\ \ 5\ \ \ \underline{6\ 7\ 8}\ \ 9\ 10\ \ 11\ 12\ 13\ 14\ 15\ 16$



2. Compute the partial color sets

[6,7,8]
[6]
[6,8]

Vertical partitioning - Example

- Example for $N = 16$ references, $z = 8$ color sets and 4 vertical partitions

$$\{1,12,13,14,16\}\{3,5,9\}\{7,11\}\{2,\ 4,\ 6,\ 8,\ 10,\ 15\}$$

new ids $\rightarrow$ $1\ \ 2\ \ 3\ \ 4\ \ 5\ \ \ \underline{6\ 7\ 8}\ \ 9\ 10\ \ 11\ \ 12\ \ 13\ \ 14\ \ 15\ \ 16$

$\mathscr{D}$

u_1	ACCG
u_2	CGCTCG
u_3	CGAACG
u_4	CGTCCG
u_5	CGGAT
u_6	ATTAT
u_7	CAT
u_8	GAGTT
u_9	ATGGA
u_{10}	GACA

$\mathscr{L}$

C_1 = [3|6,7,8|10|12,15,16]

C_2 = [6|11,16]

C_3 = [1|6,7,8|9,10|15]

C_4 = [1,3|6,7,8|9,10]

C_5 = [1,2,3,4,5|6,8|9,10|13]

C_6 = [13,14]

C_7 = [1,2,3,4,5|6|10|14]

C_8 = [2,5]

2. Compute the partial color sets

[6,7,8]

[6] $\quad -5 =$

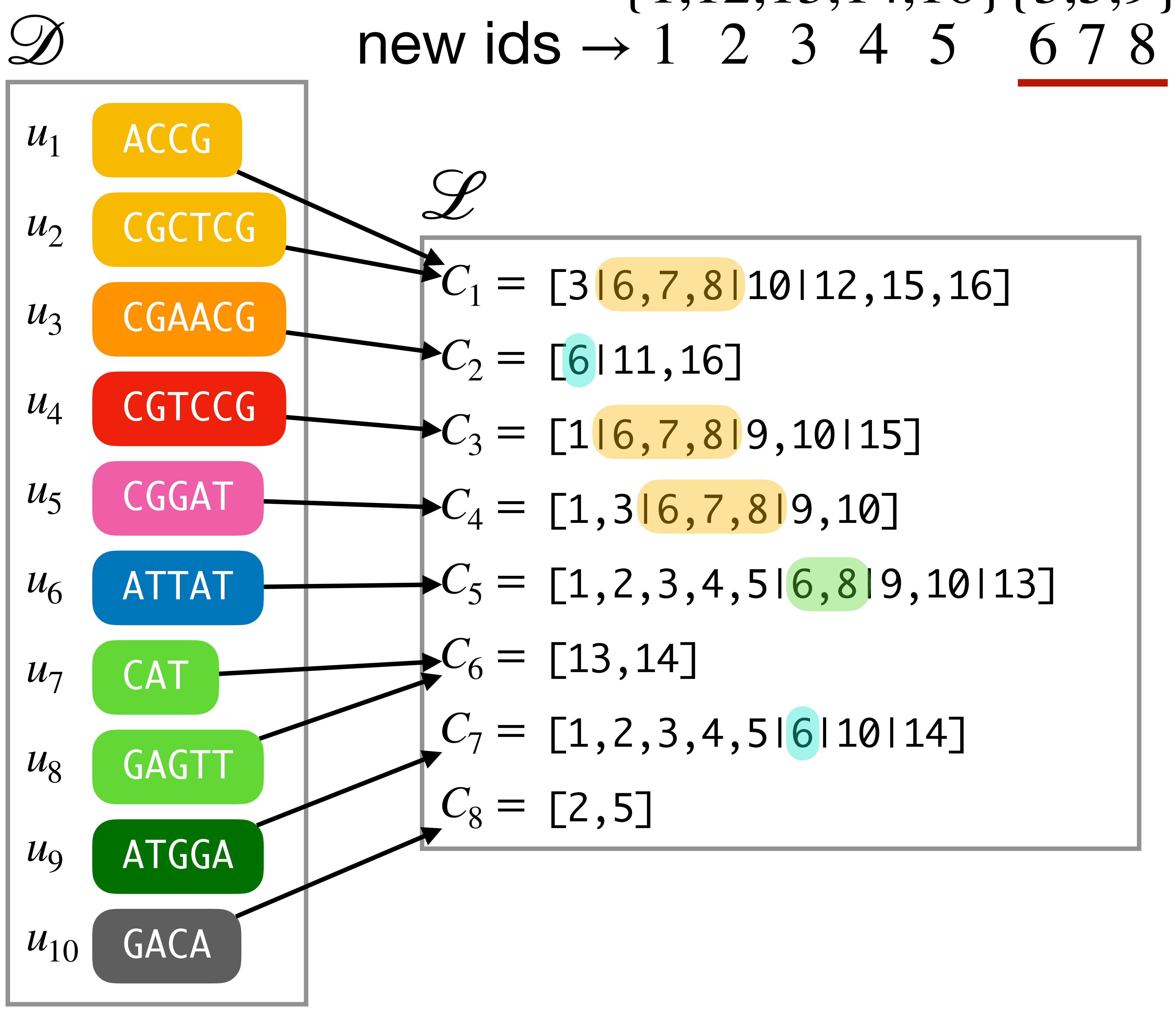
[6,8]

Vertical partitioning - Example

- Example for $N = 16$ references, $z = 8$ color sets and 4 vertical partitions

$\{1,12,13,14,16\}\{3,5,9\}\{7,11\}\{2,\ 4,\ 6,\ 8,\ 10,\ 15\}$

new ids → 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16



2. Compute the partial color sets

[6,7,8] [1,2,3]
[6] $-5 =$ [1]
[6,8] [1,3]

Vertical partitioning - Example

- Example for $N = 16$ references, $z = 8$ color sets and 4 vertical partitions

$$\{1,12,13,14,16\}\{3,5,9\}\{7,11\}\{2,\ 4,\ 6,\ 8,\ 10,\ 15\}$$

new ids $\rightarrow$ $1\ \ 2\ \ 3\ \ 4\ \ 5\ \ \ 6\ 7\ 8\ \ \ 9\ 10\ \ \ 11\ \ 12\ \ 13\ \ 14\ \ 15\ \ 16$

$\mathscr{D}$

u_1 ACCG

u_2 CGCTCG

u_3 CGAACG

u_4 CGTCCG

u_5 CGGAT

u_6 ATTAT

u_7 CAT

u_8 GAGTT

u_9 ATGGA

u_{10} GACA

$\mathscr{M}$

$M_1 = [(1,1),(2,1),(3,1),(4,1)]$

$M_2 = [(2,2),(4,2)]$

$M_3 = [(1,2),(2,1),(3,2),(4,3)]$

$M_4 = [(1,3),(2,1),(3,2)]$

$M_5 = [(1,4),(2,3),(3,2),(4,4)]$

$M_6 = [(4,5)]$

$M_7 = [(1,4),(2,2),(3,1),(4,6)]$

$M_8 = [(1,5)]$

$\mathscr{P}$

$P_{1,1} = [3]$

$P_{1,2} = [1]$

$P_{1,3} = [1,3]$

$P_{1,4} = [1,2,3,4,5]$

$P_{1,5} = [2,5]$

$P_{2,1} = [1,2,3]$

$P_{2,2} = [1]$

$P_{2,3} = [1,3]$

$P_{3,1} = [3]$

$P_{3,2} = [1]$

$P_{4,1} = [1,3]$

$P_{4,2} = [1,2,3,4,5]$

$P_{4,3} = [2,5]$

$P_{4,4} = [1,2,3]$

$P_{4,5} = [1]$

$P_{4,6} = [1,3]$

Vertical partitioning - Example

- Example for $N = 16$ references, $z = 8$ color sets and 4 vertical partitions

$$\{1,12,13,14,16\}\{3,5,9\}\{7,11\}\{2,\ 4,\ 6,\ 8,\ 10,\ 15\}$$

new ids → $1\ \ 2\ \ 3\ \ 4\ \ 5\quad 6\ 7\ 8\quad 9\ 10\quad 11\ \ 12\ \ 13\ \ 14\ \ 15\ \ 16$

$\mathscr{D}$

u_1	ACCG
u_2	CGCTCG
u_3	CGAACG
u_4	CGTCCG
u_5	CGGAT
u_6	ATTAT
u_7	CAT
u_8	GAGTT
u_9	ATGGA
u_{10}	GACA

$\mathscr{M}$

$M_1 =$ [(1,1),(2,1),(3,1),(4,1)]

$M_2 =$ [(2,2),(4,2)]

$M_3 =$ [(1,2),(2,1),(3,2),(4,3)]

$M_4 =$ [(1,3),(2,1),(3,2)]

$M_5 =$ [(1,4),(2,3),(3,2),(4,4)]

$M_6 =$ [(4,5)]

$M_7 =$ [(1,4),(2,2),(3,1),(4,6)]

$M_8 =$ [(1,5)]

$\mathscr{P}$

$P_{1,1} =$ [3]

$P_{1,2} =$ [1]

$P_{1,3} =$ [1,3]

$P_{1,4} =$ [1,2,3,4,5]

$P_{1,5} =$ [2,5]

$P_{2,1} =$ [1,2,3]

$P_{2,2} =$ [1]

$P_{2,3} =$ [1,3]

$P_{3,1} =$ [3]

$P_{3,2} =$ [1]

$P_{4,1} =$ [1,3]

$P_{4,2} =$ [1,2,3,4,5]

$P_{4,3} =$ [2,5]

$P_{4,4} =$ [1,2,3]

$P_{4,5} =$ [1]

$P_{4,6} =$ [1,3]

Vertical partitioning - Example

- Example for $N = 16$ references, $z = 8$ color sets and 4 vertical partitions

$$\{1,12,13,14,16\}\{3,5,9\}\{7,11\}\{2,\ 4,\ 6,\ 8,\ 10,\ 15\}$$

new ids $\rightarrow$ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

$\mathscr{D}$

u_1 ACCG
u_2 CGCTCG
u_3 CGAACG
u_4 CGTCCG
u_5 CGGAT
u_6 ATTAT
u_7 CAT
u_8 GAGTT
u_9 ATGGA
u_{10} GACA

$\mathscr{M}$

$M_1 =$ [(1,1),(2,1),(3,1),(4,1)]
$M_2 =$ [(2,2),(4,2)]
$M_3 =$ [(1,2),(2,1),(3,2),(4,3)]
$M_4 =$ [(1,3),(2,1),(3,2)]
$M_5 =$ [(1,4),(2,3),(3,2),(4,4)]
$M_6 =$ [(4,5)]
$M_7 =$ [(1,4),(2,2),(3,1),(4,6)]
$M_8 =$ [(1,5)]

$\mathscr{P}$

$P_{1,1} =$ [3]
$P_{1,2} =$ [1]
$P_{1,3} =$ [1,3]
$P_{1,4} =$ [1,2,3,4,5]
$P_{1,5} =$ [2,5]

$P_{2,1} =$ [1,2,3]
$P_{2,2} =$ [1]
$P_{2,3} =$ [1,3]

$P_{3,1} =$ [3]
$P_{3,2} =$ [1]

$P_{4,1} =$ [1,3]
$P_{4,2} =$ [1,2,3,4,5]
$P_{4,3} =$ [2,5]
$P_{4,4} =$ [1,2,3]
$P_{4,5} =$ [1]
$P_{4,6} =$ [1,3]

Combining the representations (md-Fulgor)

- Horizontal and vertical partitioning are based on orthogonal paradigms
- A combined representation could exploit the advantages of both approaches
- **Intuition:** Encode the color sets using vertical partitioning, then encode the partial sets using horizontal partitioning.

Space in GB

Dataset	Fulgor			d-Fulgor		m-Fulgor		md-Fulgor	
	dBG	Color sets	Total	Color sets	Total	Color sets	Total	Color sets	Total
EC	0.29	1.36 (83%)	1.65	0.45 (61%)	0.74	0.40 (58%)	0.69	0.24 (45%)	0.52
SE-5K	0.16	0.59 (79%)	0.75	0.20 (56%)	0.36	0.16 (50%)	0.32	0.11 (40%)	0.27
SE-10K	0.35	1.66 (83%)	2.01	0.48 (58%)	0.83	0.34 (49%)	0.70	0.22 (39%)	0.57
SE-50K	1.25	17.03 (93%)	18.29	4.31 (77%)	5.57	2.08 (62%)	3.34	1.38 (52%)	2.64
SE-100K	1.71	40.71 (96%)	42.43	9.37 (84%)	11.10	3.75 (68%)	5.47	2.26 (57%)	3.98
SE-150K	2.02	68.61 (97%)	70.65	15.73 (89%)	17.77	5.27 (72%)	7.31	3.22 (61%)	5.26
GB	21.29	15.54 (42%)	36.83	7.51 (26%)	28.81	9.16 (30%)	30.46	6.19 (23%)	27.48

All experiments used $k = 31$

Space in GB

Dataset	Fulgor			d-Fulgor		m-Fulgor		md-Fulgor	
	dBG	Color sets	Total	Color sets	Total	Color sets	Total	Color sets	Total
EC	0.29	1.36 (83%)	1.65	0.45 (61%)	0.74	0.40 (58%)	0.69	0.24 (45%)	0.52
SE-5K	0.16	0.59 (79%)	0.75	0.20 (56%)	0.36	0.16 (50%)	0.32	0.11 (40%)	0.27
SE-10K	0.35	1.66 (83%)	2.01	0.48 (58%)	0.83	0.34 (49%)	0.70	0.22 (39%)	0.57
SE-50K	1.25	17.03 (93%)	18.29	4.31 (77%)	5.57	2.08 (62%)	3.34	1.38 (52%)	2.64
SE-100K	1.71	40.71 (96%)	42.43	9.37 (84%)	11.10	3.75 (68%)	5.47	2.26 (57%)	3.98
SE-150K	2.02	68.61 (97%)	70.65	15.73 (89%)	17.77	5.27 (72%)	7.31	3.22 (61%)	5.26
GB	21.29	15.54 (42%)	36.83	7.51 (26%)	28.81	9.16 (30%)	30.46	6.19 (23%)	27.48

All experiments used $k = 31$

Pseudoalignment efficiency

Dataset	Hit rate	Fulgor		d-Fulgor		m-Fulgor		md-Fulgor	
		mm:ss	GB	h:mm:ss	GB	mm:ss	GB	h:mm:ss	GB
EC	98.99	2:10	1.67	5:20	0.78	2:30	0.73	5:00	0.57
SE-5K	89.49	1:10	0.80	2:00	0.41	1:16	0.37	1:48	0.32
SE-10K	89.71	2:20	2.06	4:30	0.90	2:28	0.77	3:34	0.65
SE-50K	91.25	12:00	18.24	29:00	5.82	13:10	3.64	22:25	2.95
SE-100K	91.41	24:00	42.20	1:02:00	11.58	27:00	6.08	50:00	4.62
SE-150K	91.52	37:00	70.55	1:38:00	18.51	41:30	8.29	1:15:00	6.28
GB	92.91	1:10	36.01	1.00	28.17	1:09	29.79	1.03	26.88

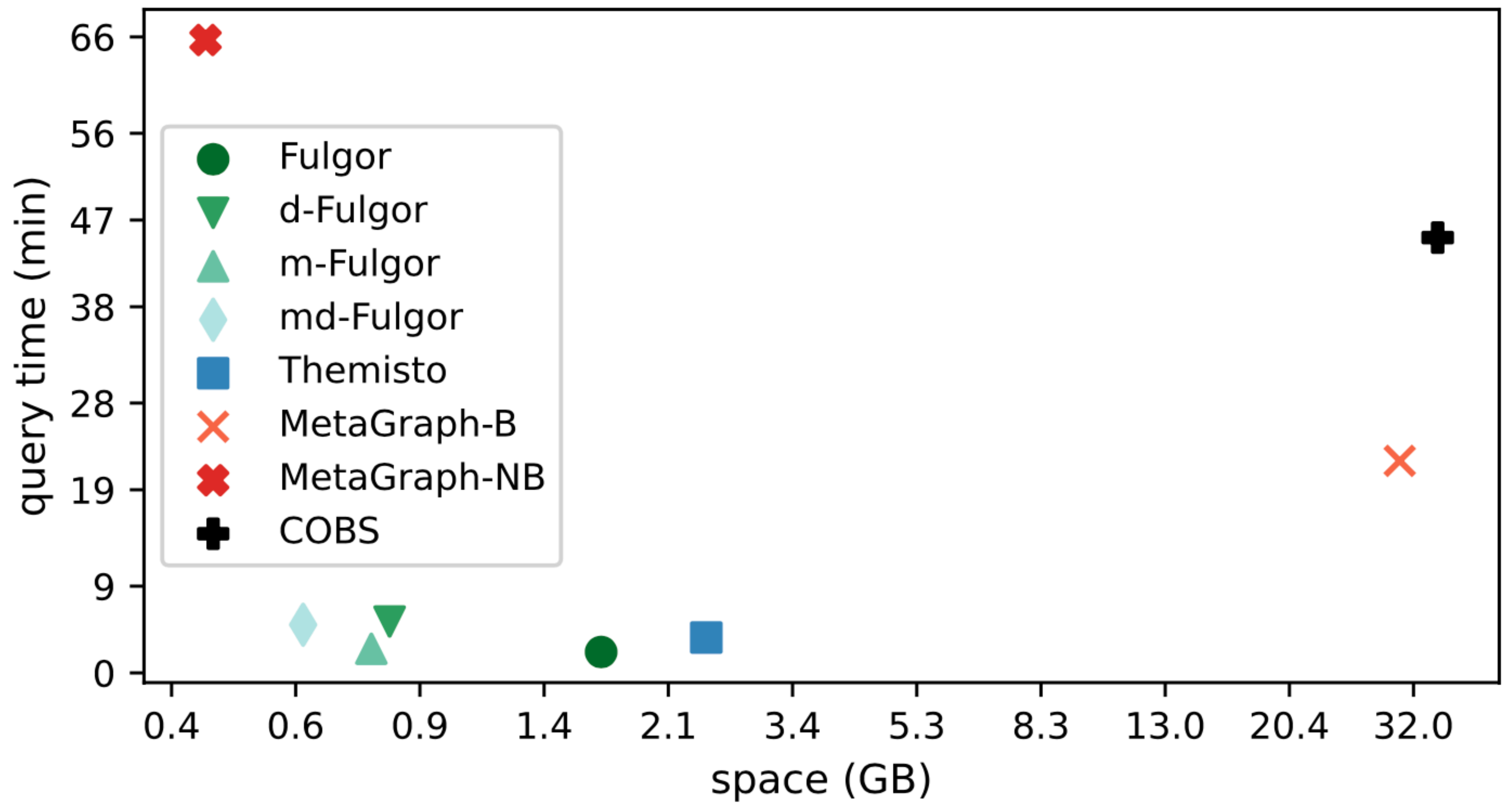
All experiments used $k = 31$

Pseudoalignment efficiency

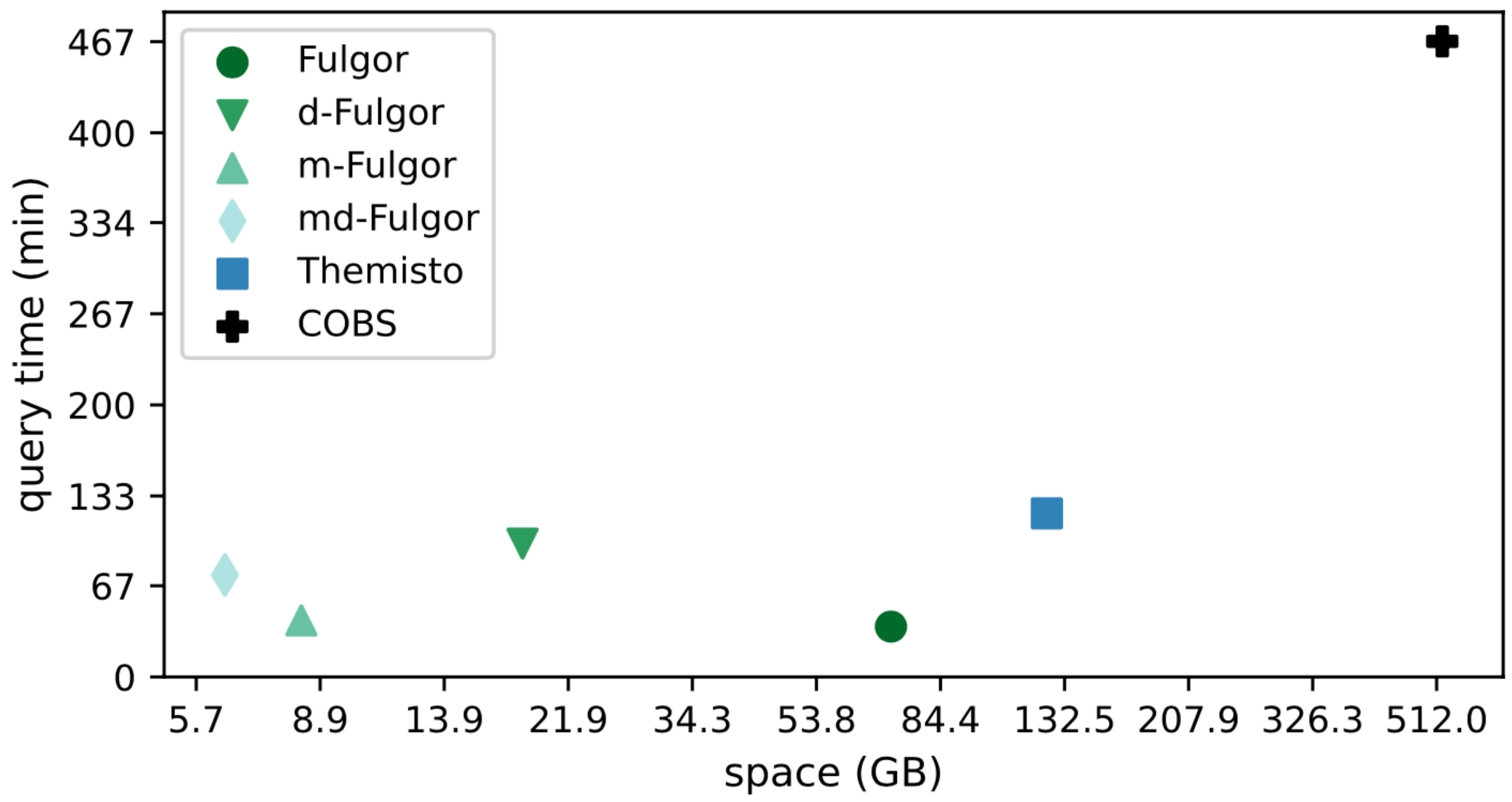
Dataset	Hit rate	Fulgor		d-Fulgor		m-Fulgor		md-Fulgor	
		mm:ss	GB	h:mm:ss	GB	mm:ss	GB	h:mm:ss	GB
EC	98.99	2:10	1.67	5:20	0.78	2:30	0.73	5:00	0.57
SE-5K	89.49	1:10	0.80	2:00	0.41	1:16	0.37	1:48	0.32
SE-10K	89.71	2:20	2.06	4:30	0.90	2:28	0.77	3:34	0.65
SE-50K	91.25	12:00	18.24	29:00	5.82	13:10	3.64	22:25	2.95
SE-100K	91.41	24:00	42.20	1:02:00	11.58	27:00	6.08	50:00	4.62
SE-150K	91.52	37:00	70.55	1:38:00	18.51	41:30	8.29	1:15:00	6.28
GB	92.91	1:10	36.01	1.00	28.17	1:09	29.79	1.03	26.88

All experiments used $k = 31$

Overall space/time trade-off

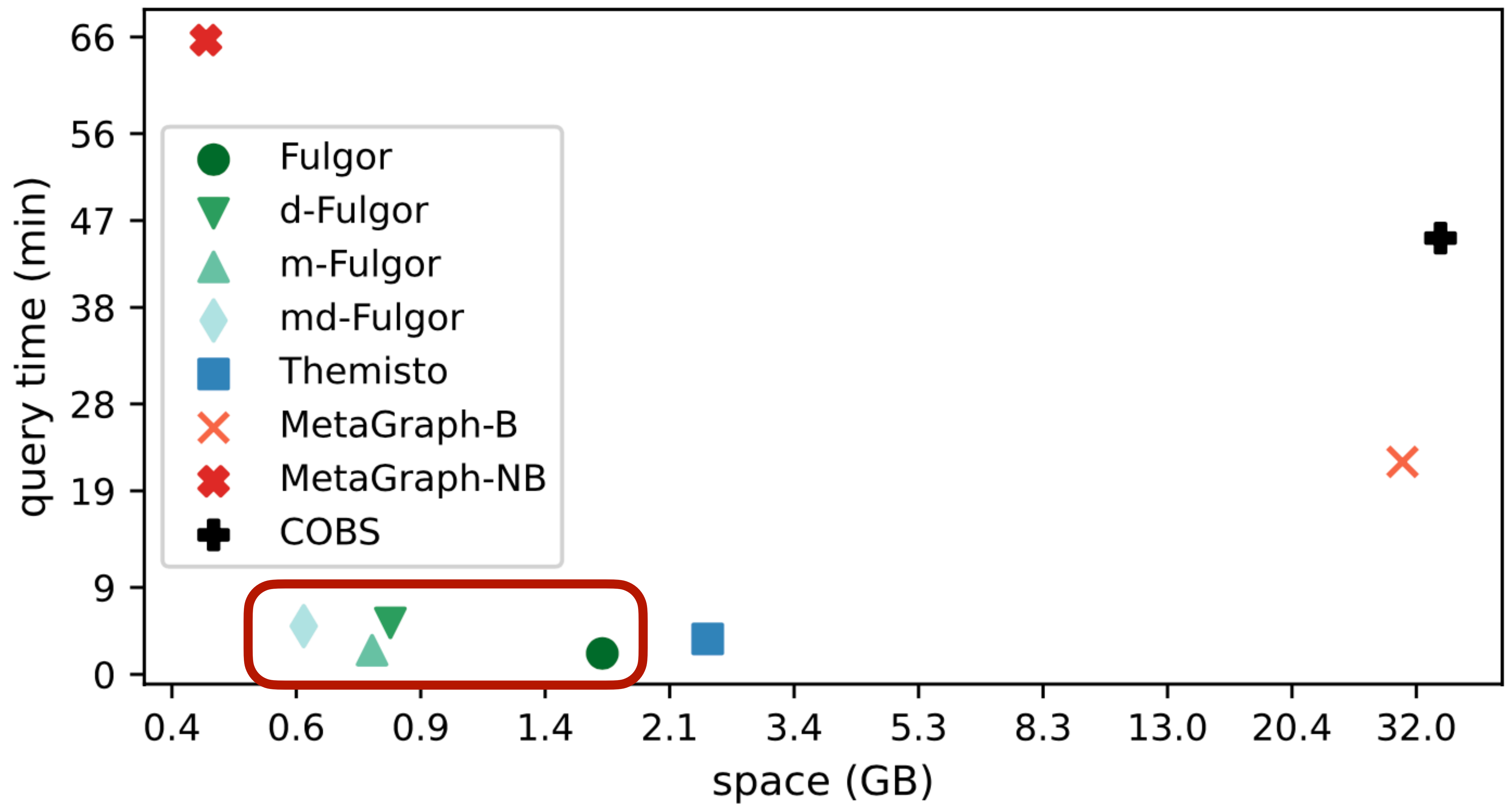


EC

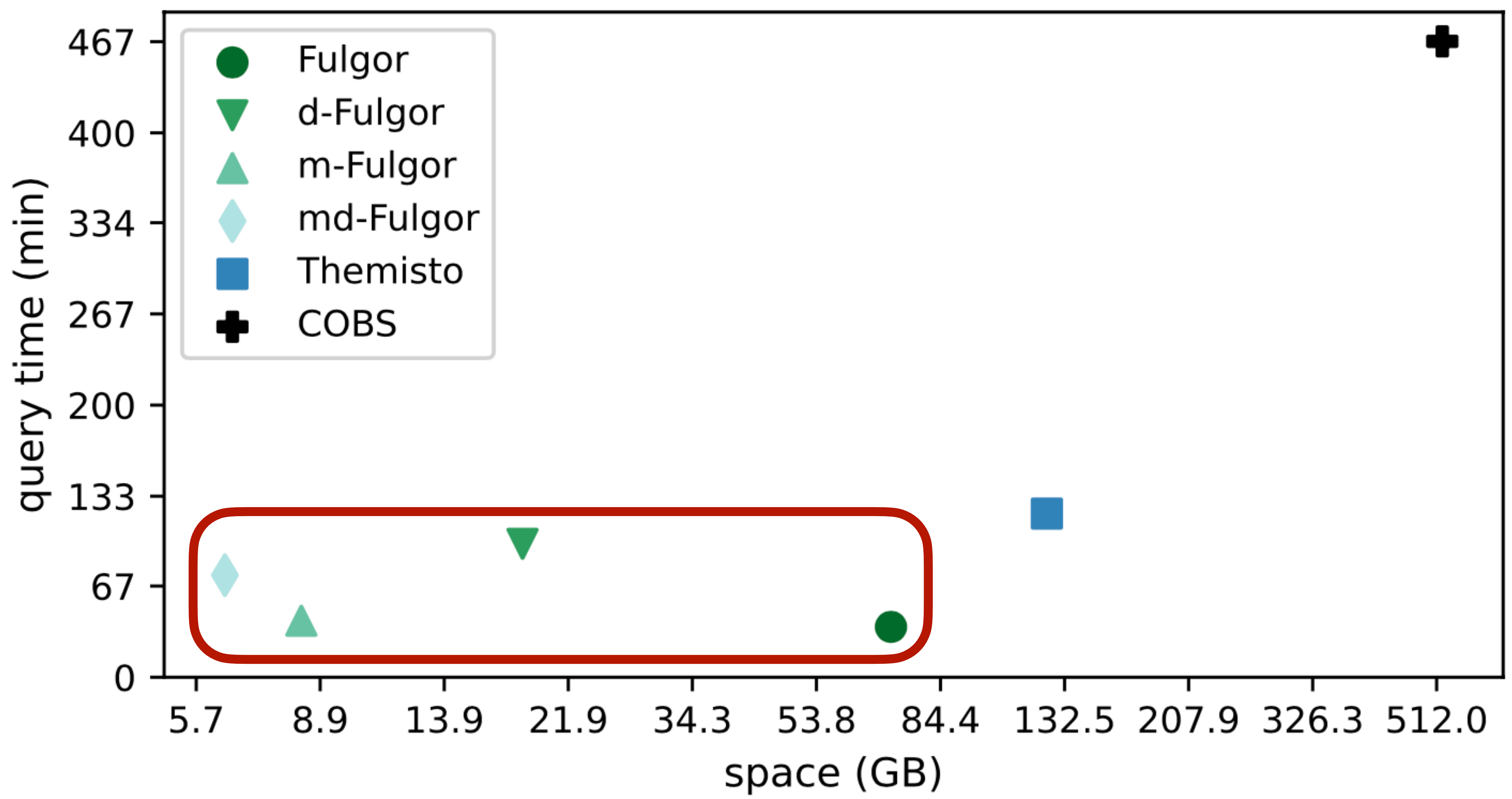


SE-150k

Overall space/time trade-off



EC



SE-150k

Thank you!

Thank you! Any questions?