The Graph FM Index

UNIVERSITY OF MARYLAND

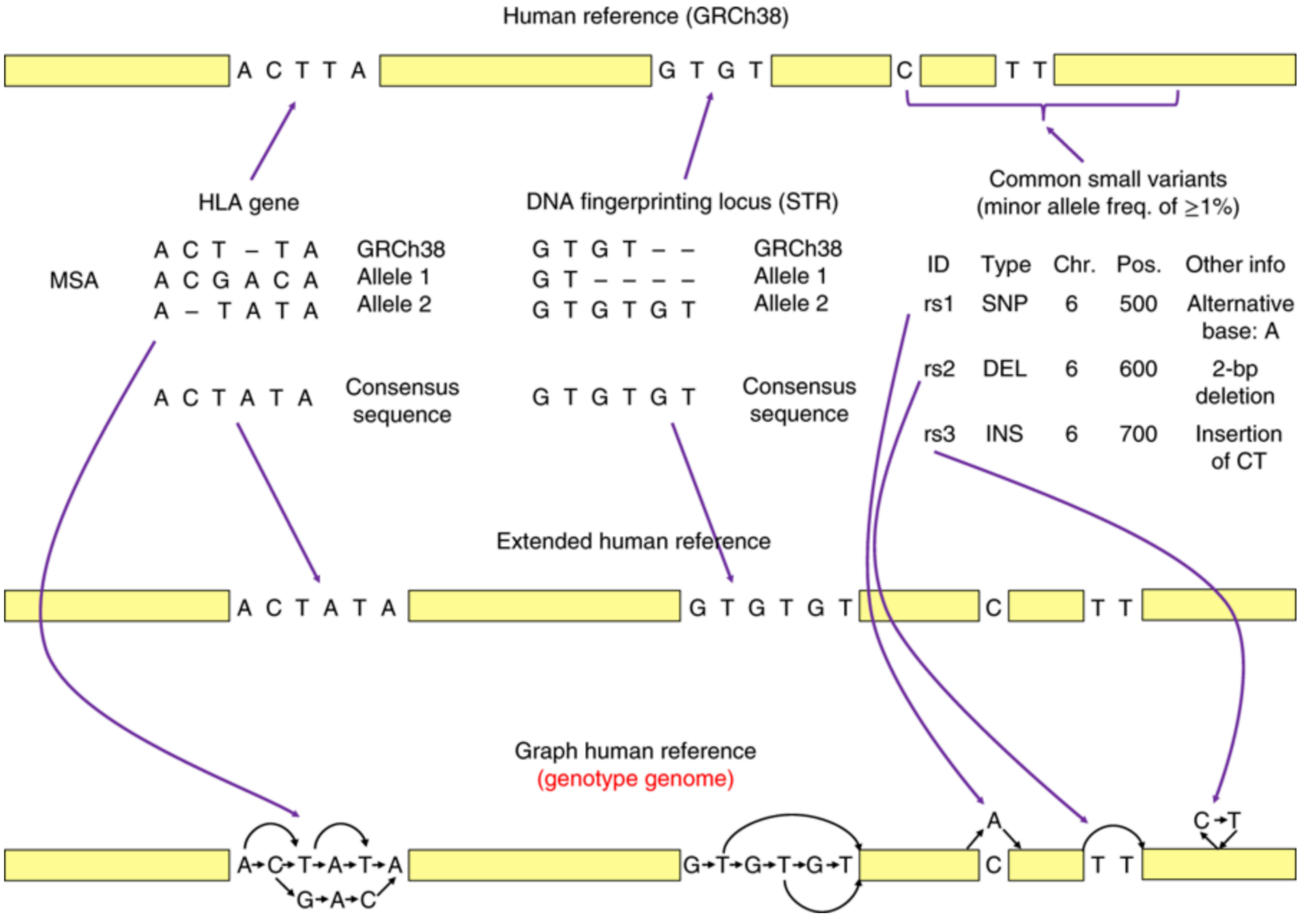
The Graph FM-Index

Idea / motivation : **No sample is the reference**

We have spent a lot of effort characterizing major human variants, yet most aligners simply map against a single human reference genome that doesn't even have the most likely variant at each locus.

HISAT2 is one of a new breed of "graph" aligners, that views the genome as a graph rather than a simple string. This framework allows encoding variants as alternative "paths" through the genome.

The Graph FM-Index



Graph FM-Index

Siren, J., Valimaki, N. & Makinen, V. Indexing graphs for path queries with applications in genome research. *IEEE-ACM Trans. Comput. Biol. Bioinform.* **11**, 375–388 (2014).

Construction of graph FM index relies on creation of prefix-range-sorted automata

- Key property needed for backward search:

  - For list (u,v) of outgoing edges, sorted by pairs (p(u), p(v)), l(u)p(v) must be sorted by sequences

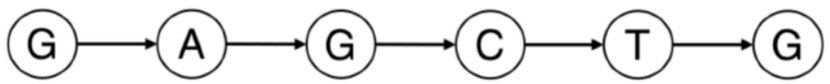
  - For any c, all outgoing edges from nodes labeled with c are lexicographically adjacent and are sorted by the prefix p(v) of the destination node

  - All occurrences of c in BWT encode an incoming edge from a node with label c, and thus are sorted by prefix p(v) of the destination

  - Hence, incoming edge labeled by nth occurrence of c is the same as the outgoing edge for rank C[c]+j

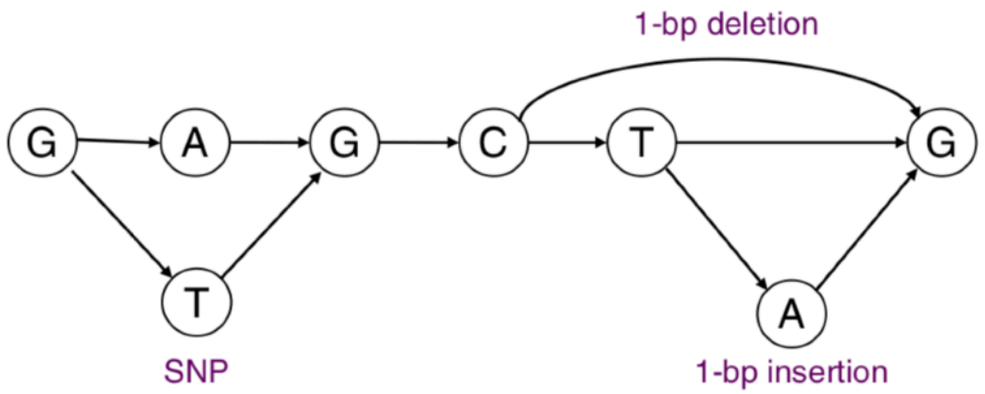
The Graph FM-Index

1. Reference sequence (6 bp long)



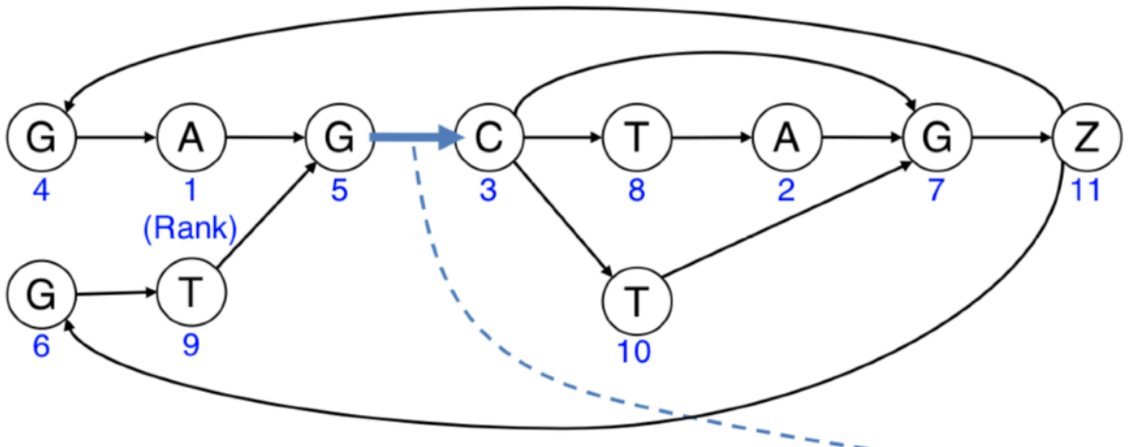
single-nucleotide variant (A/T), a 1-bp deletion (T) and a 1-bp insertion (A)

2. Graphical representation (original graph)



Prefix doubling and pruning

3. Prefix-sorted graph



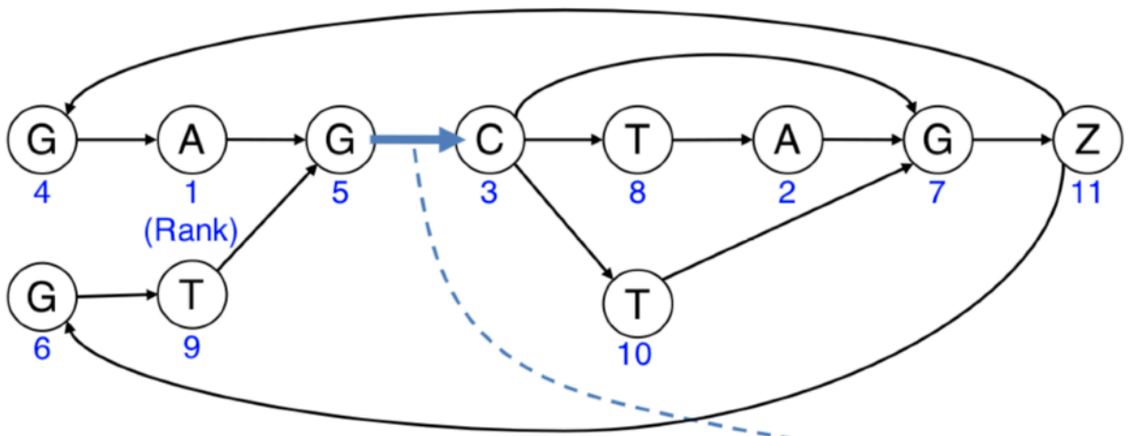
4. Tabular representation of the prefix-sorted graph

Outgoing edge(s)			Incoming edge(s)	
Node rank	First		Last	Node rank
1	A		G	1
2	A		T	2
3	C C C		G Z A T	3 4 5
4	G			
5	G		Z	6
6	G		A C T	7
7	G			
8	T			
9	T		C	8
10	T		G	9
11	Z Z		C G	10 11

Siren, J., Valimaki, N. & Makinen, V. Indexing graphs for path queries with applications in genome research. *IEEE-ACM Trans. Comput. Biol. Bioinform.* **11**, 375–388 (2014).

(the transformation from 2 -> 3 is crucial to allow indexing)

The Graph FM-Index



Outgoing edge(s)			Incoming edge(s)	
Node rank	First		Last	Node rank
1	A		G	1
2	A		T	2
3	C C C		G	3
			Z	4
			A	5
4	G		T	
5	G		Z	6
6	G		A	7
7	G		C	
8	T		T	
9	T		C	8
10	T		G	9
11	Z		C	10
	Z		G	11

The Graph FM-Index

Note: We have an LF mapping here, just like a normal BWT

This 1-to-1 correspondence isn't possible without the graph transformation.

Hint: try and search for the pattern "GTG"

Outgoing edge(s)		Incoming edge(s)	
Node rank	First	Last	Node rank
1	A_0	G_0	1
2	A_1	T_0	2
3	C_0 C_1 C_2	G_1	3
		Z_0	4
4	G_0	A_0 T_1	5
5	G_1	Z_1	6
6	G_2	A_1 C_0 T_2	7
7	G_3		
8	T_0		
9	T_1	C_1	8
10	T_2	G_2	9
11	Z_0 Z_1	C_2	10
		G_3	11

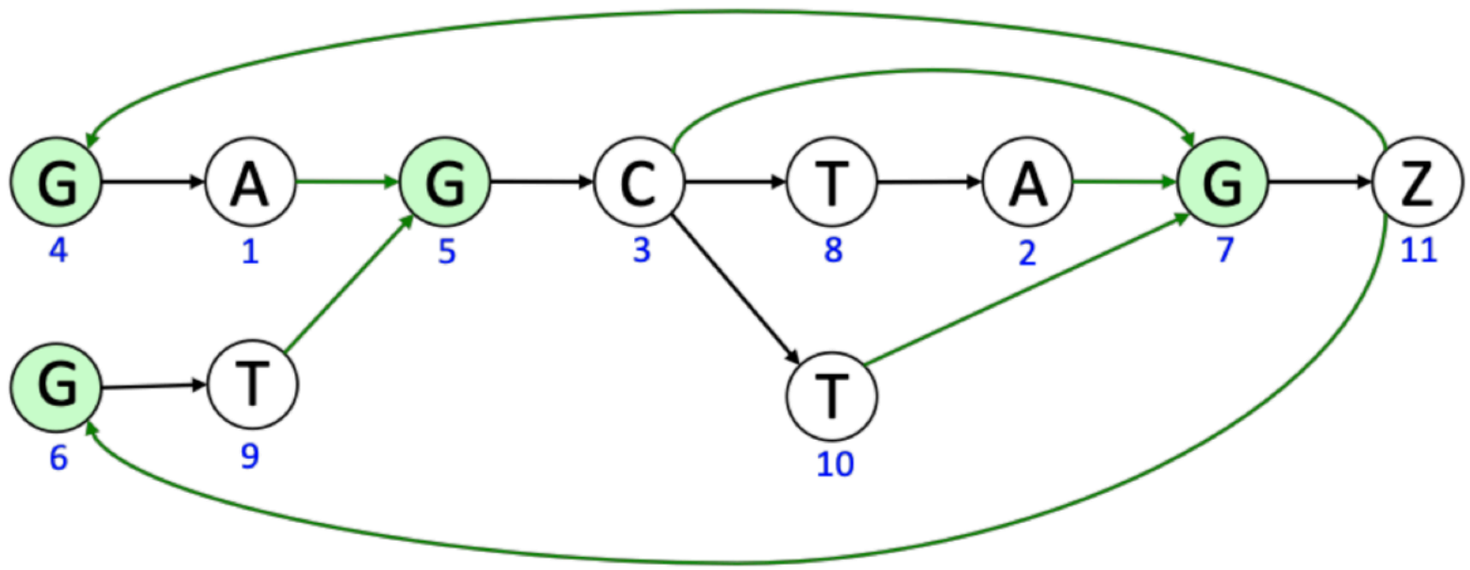
The Graph FM-Index

Outgoing edge(s)		Incoming edge(s)	
Node rank	First	Last	Node rank
1	A_0	G_0	1
2	A_1	T_0	2
3	C_0 C_1 C_2	G_1	3
		Z_0	4
4	G_0	A_0 T_1	5
5	G_1	Z_1	6
6	G_2	A_1 C_0 T_2	7
7	G_3		
8	T_0		
9	T_1	C_1	8
10	T_2	G_2	9
11	Z_0 Z_1	C_2	10
		G_3	11

Searching the Graph FM-Index

Outgoing edge(s)		Incoming edge(s)	
Node rank	First	Last	Node rank
1	A_0	G_0	1
2	A_1	T_0	2
3	C_0 C_1 C_2	G_1	3
		Z_0	4
		A_0 T_1	5
4	G_0	Z_1	6
5	G_1	A_1 C_0 T_2	7
6	G_2		
7	G_3		
8	T_0	C_1	8
9	T_1	G_2	9
10	T_2	C_2	10
11	Z_0 Z_1	G_3	11

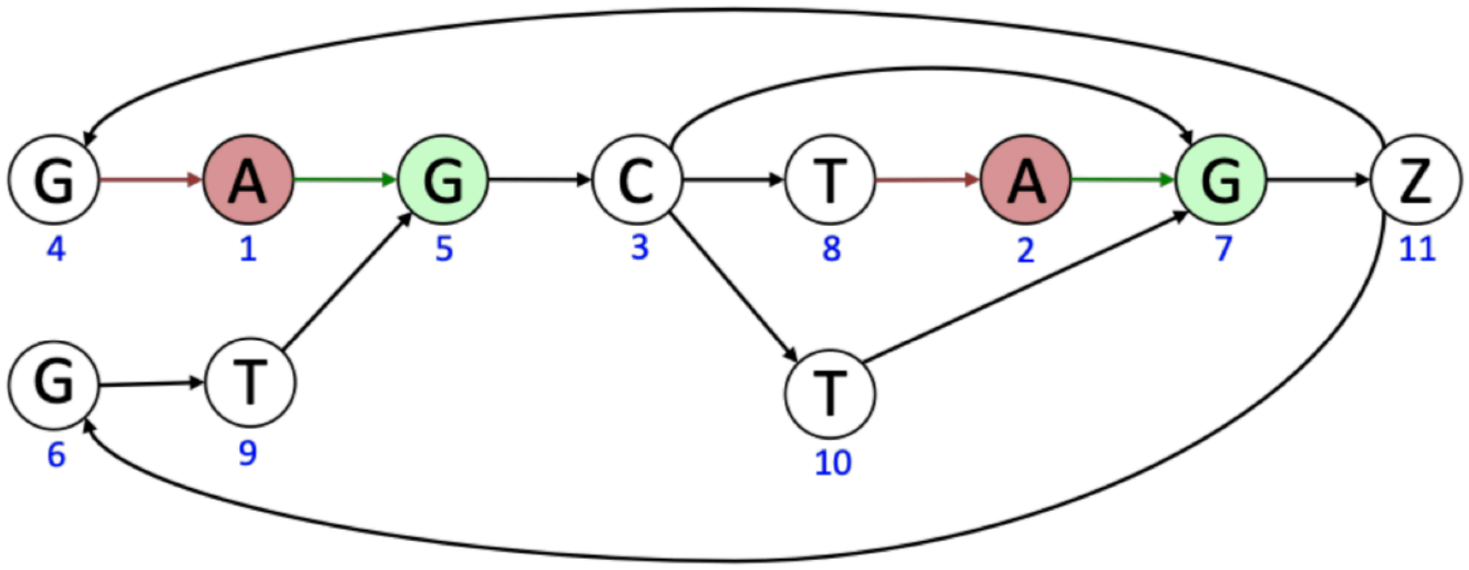
Query : TAG



Searching the Graph FM-Index

Outgoing edge(s)			Incoming edge(s)	
Node rank	First		Last	Node rank
1	A_0		G_0	1
2	A_1		T_0	2
3	C_0 C_1 C_2		G_1	3
4	G_0		Z_0 A_0 T_1	4 5
5	G_1	1	Z_1	6
6	G_2		A_1 C_0 T_2	7
7	G_3			
8	T_0		C_1	8
9	T_1		G_2	9
10	T_2		C_2	10
11	Z_0 Z_1		G_3	11

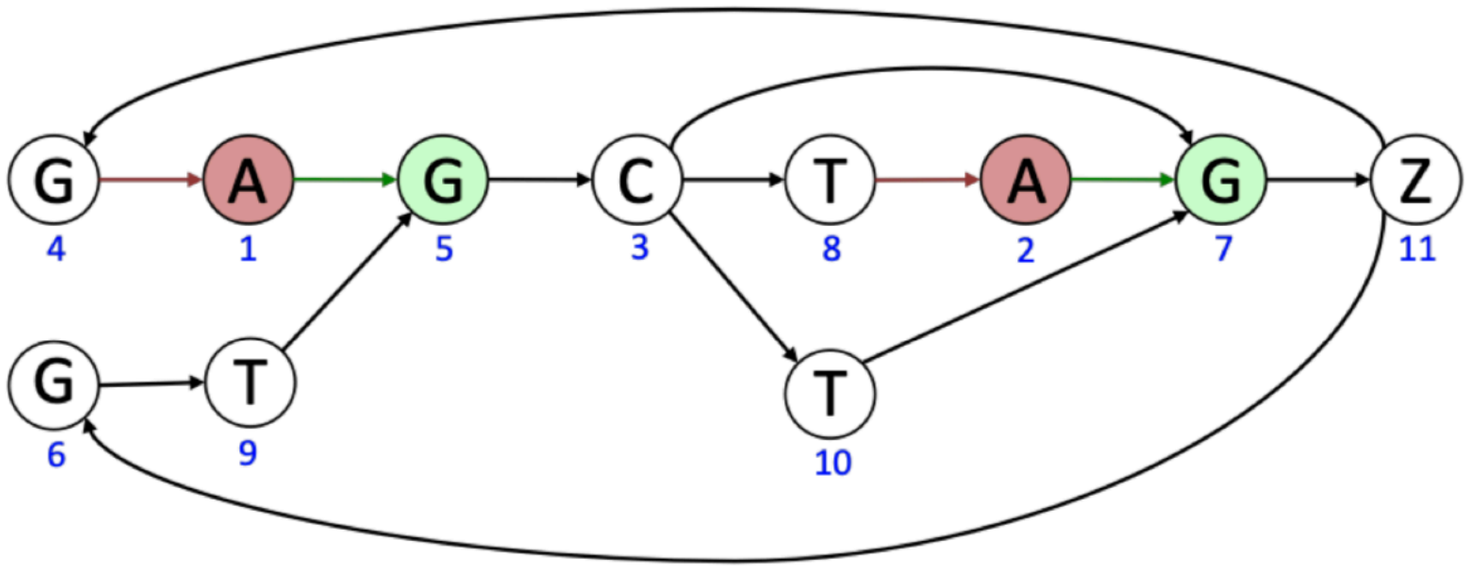
Query : TAG



Searching the Graph FM-Index

Outgoing edge(s)		Incoming edge(s)	
Node rank	First	Last	Node rank
1	A_0	G_0	1
2	A_1	T_0	2
3	C_0 C_1 C_2	G_1	3
4	G_0	Z_0	4
5	G_1	A_0 T_1	5
6	G_2	Z_1	6
7	G_3	A_1 C_0 T_2	7
8	T_0	C_1	8
9	T_1	G_2	9
10	T_2	C_2	10
11	Z_0 Z_1	G_3	11

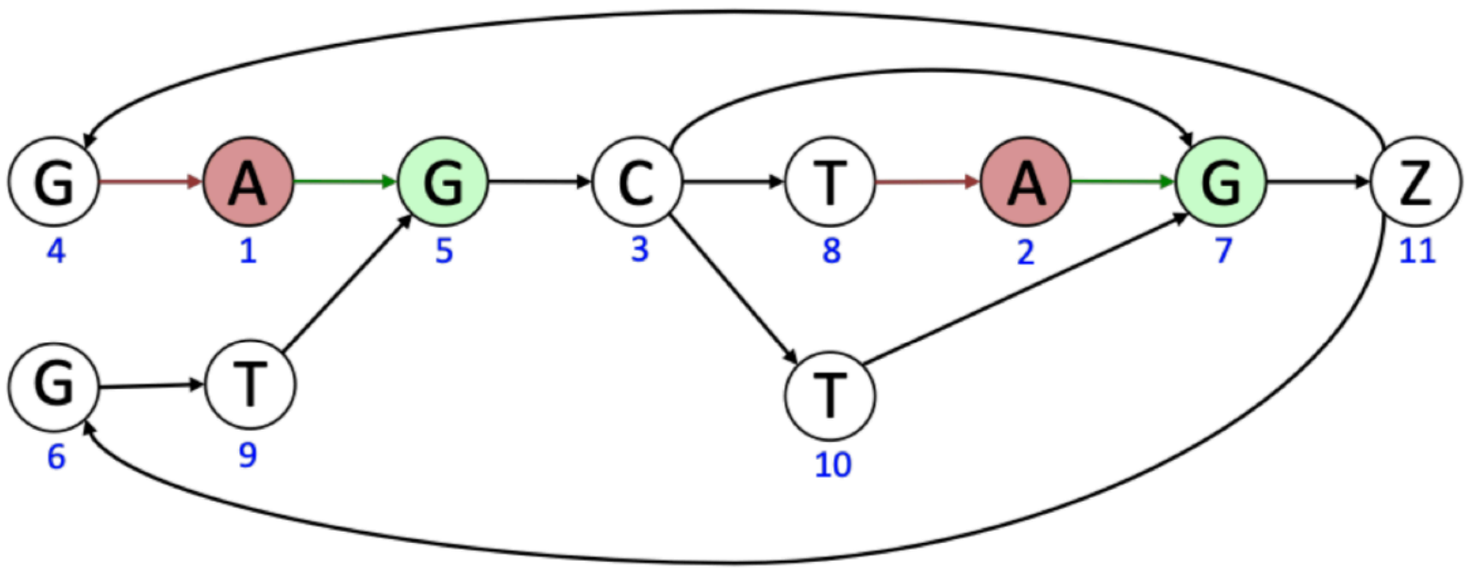
Query : TAG



Searching the Graph FM-Index

Outgoing edge(s)			Incoming edge(s)	
Node rank	First		Last	Node rank
1	A_0		G_0	1
2	A_1		T_0	2
3	C_0 C_1 C_2	2	G_1	3
4	G_0		Z_0	4
5	G_1		A_0 T_1	5
6	G_2		Z_1	6
7	G_3		A_1 C_0 T_2	7
8	T_0		C_1	8
9	T_1		G_2	9
10	T_2		C_2	10
11	Z_0 Z_1		G_3	11

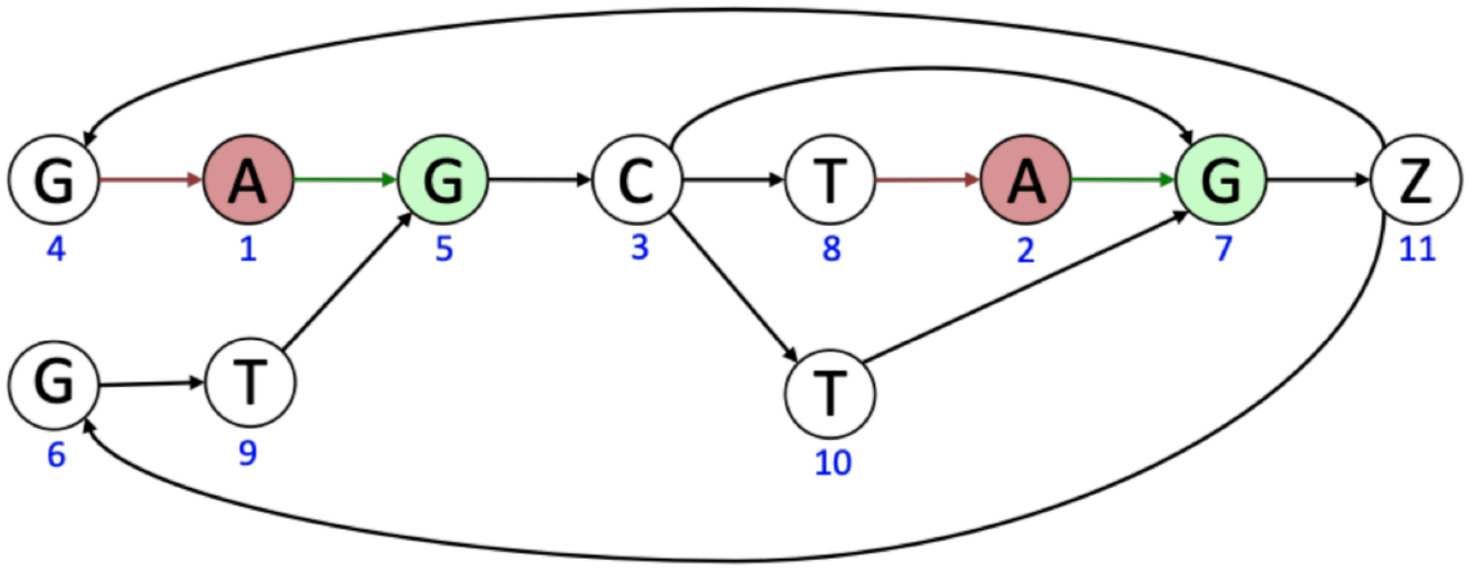
Query : TAG



Searching the Graph FM-Index

Outgoing edge(s)			Incoming edge(s)	
Node rank	First		Last	Node rank
1	A_0	3	G_0	1
2	A_1		T_0	2
3	C_0		G_1	3
	C_1		Z_0	4
	C_2		A_0	5
4	G_0		T_1	
5	G_1		Z_1	6
6	G_2		A_1	7
7	G_3		C_0	
8	T_0		T_2	
9	T_1		C_1	8
10	T_2		G_2	9
11	Z_0		C_2	10
	Z_1		G_3	11

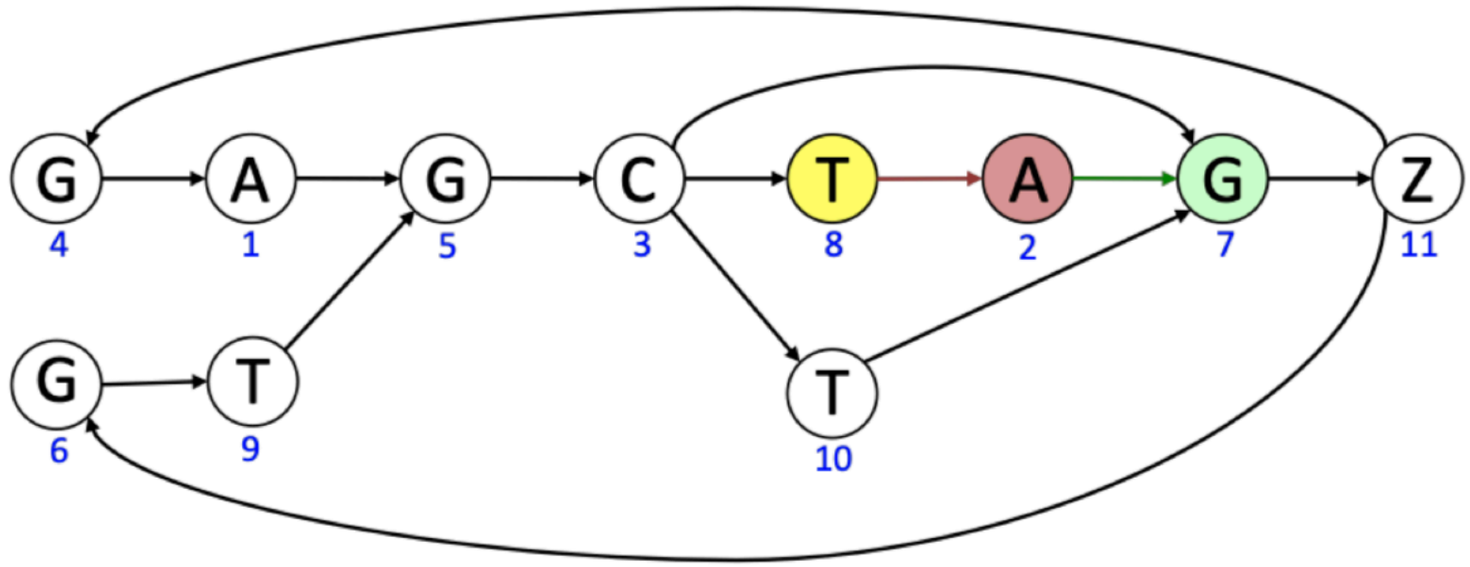
Query : TAG



Searching the Graph FM-Index

Outgoing edge(s)		Incoming edge(s)	
Node rank	First	Last	Node rank
1	A_0	G_0	1
2	A_1	T_0	2
3	C_0 C_1 C_2	G_1	3
4	G_0	Z_0	4
5	G_1	A_0 T_1	5
6	G_2	Z_1	6
7	G_3	A_1 C_0 T_2	7
8	T_0	C_1	8
9	T_1	G_2	9
10	T_2	C_2	10
11	Z_0 Z_1	G_3	11

Query : TAG

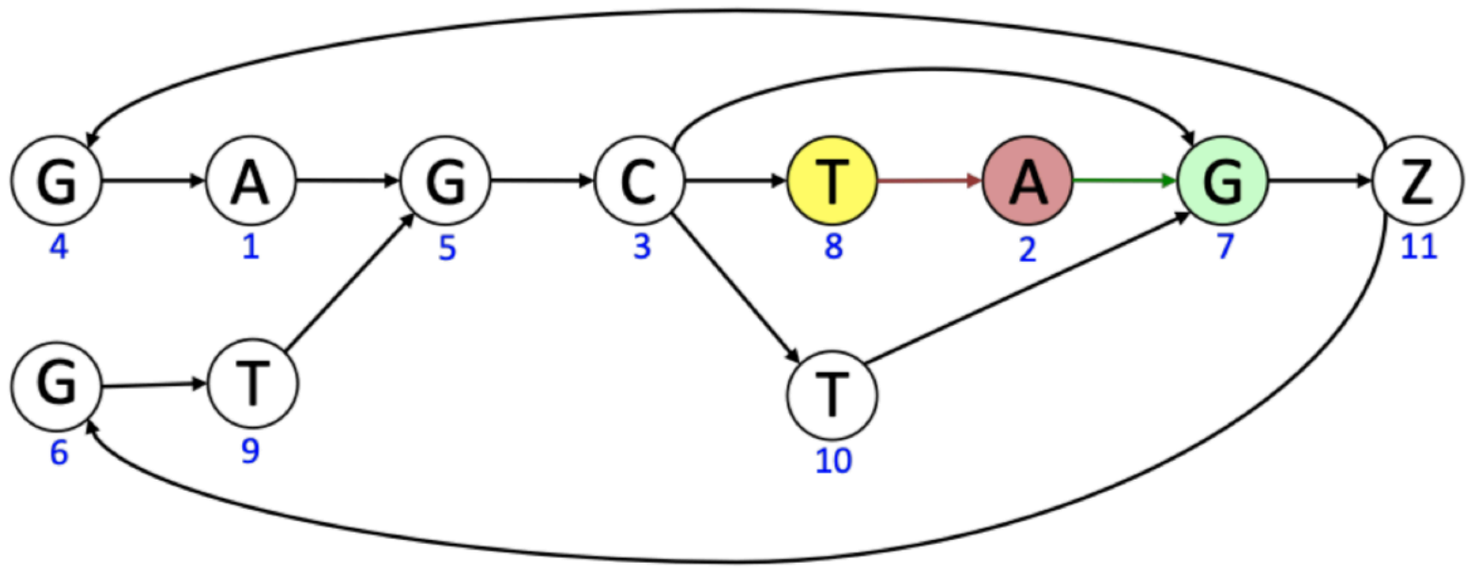


Searching the Graph FM-Index

Outgoing edge(s)		Incoming edge(s)	
Node rank	First	Last	Node rank
1	A_0	G_0	1
2	A_1	T_0	2
3	C_0 C_1 C_2	G_1	3
4	G_0	Z_0	4
5	G_1	A_0 T_1	5
6	G_2	Z_1	6
7	G_3	A_1 C_0 T_2	7
8	T_0	C_1	8
9	T_1	G_2	9
10	T_2	C_2	10
11	Z_0 Z_1	G_3	11

4

Query : TAG

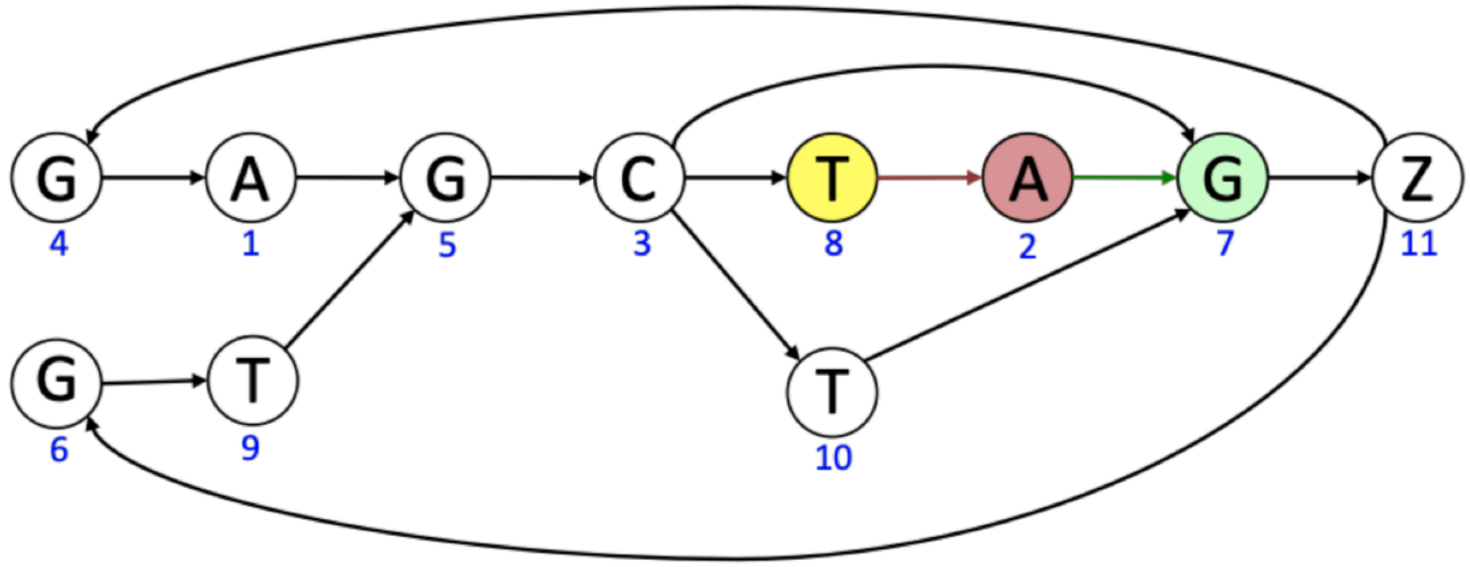


Searching the Graph FM-Index

Outgoing edge(s)		Incoming edge(s)	
Node rank	First	Last	Node rank
1	A_0	G_0	1
2	A_1	T_0	2
3	C_0 C_1 C_2	G_1	3
4	G_0	Z_0	4
5	G_1	A_0 T_1	5
6	G_2	Z_1	6
7	G_3	A_1 C_0 T_2	7
8	T_0	C_1	8
9	T_1	G_2	9
10	T_2	C_2	10
11	Z_0 Z_1	G_3	11

4

Query : TAG



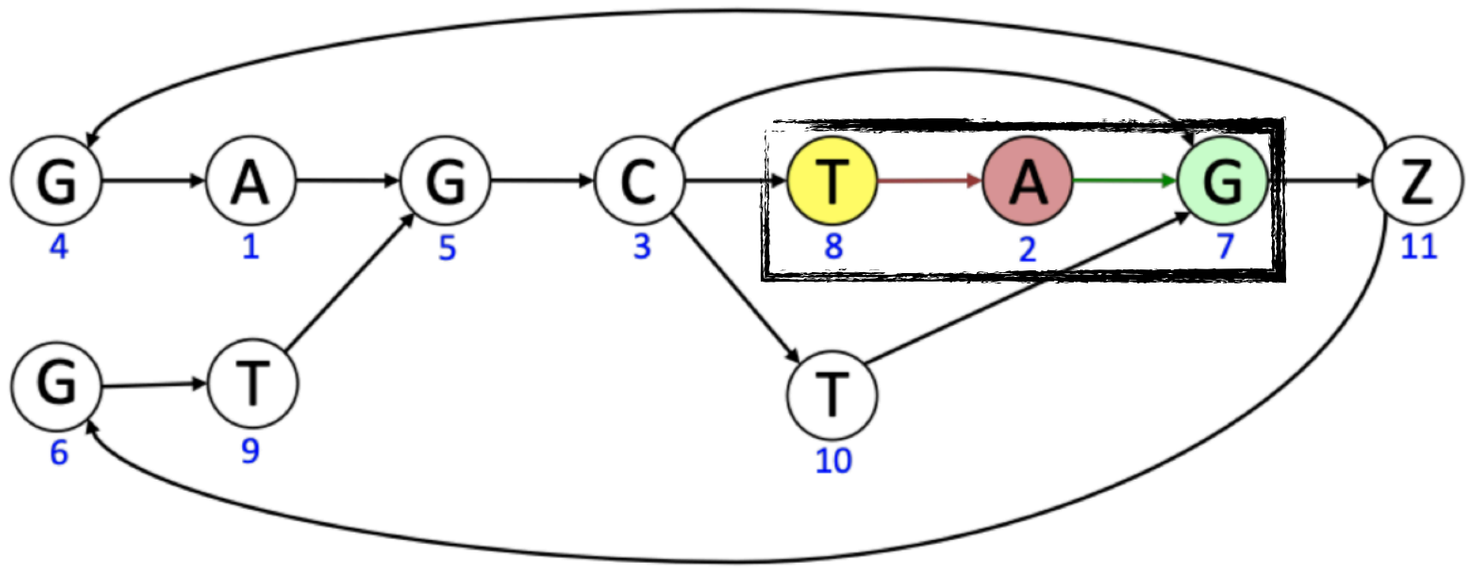
The last step, to row 8, gives us the **ID** of the node corresponding to the prefix.

Searching the Graph FM-Index

Outgoing edge(s)		Incoming edge(s)	
Node rank	First	Last	Node rank
1	A_0	G_0	1
2	A_1	T_0	2
3	C_0 C_1 C_2	G_1	3
4	G_0	Z_0	4
5	G_1	A_0 T_1	5
6	G_2	Z_1	6
7	G_3	A_1 C_0 T_2	7
8	T_0	C_1	8
9	T_1	G_2	9
10	T_2	C_2	10
11	Z_0 Z_1	G_3	11

4

Query : TAG



We've found our pattern in the graph!

The last step, to row 8, gives us the **ID** of the node corresponding to the prefix.

How to store the GFM efficiently

Outgoing edge(s)		Incoming edge(s)	
Node rank	First	Last	Node rank
1	A	G	1
2	A	T	2
3	C	G	3
	C	Z	4
		A	5
4	G	T	
5	G	Z	6
6	G	A	7
7	G	C	
8	T	T	
9	T	C	8
10	T	G	9
11	Z	C	10
	Z	G	11

Outgoing edge(s)	Incoming edge(s)	
Node rank	Last	Node rank
1	10	1
1	11	1
1	10	1
0	00	1
0	00	1
1	11	0
1	00	1
1	00	1
1	01	0
1	11	0
1	01	1
1	10	1
1	01	1
0	10	1

First	
A	2
C	3
G	4
T	3
Z	2

How to store the GFM efficiently

Outgoing edge(s)	
Node rank	First
1	A
2	A
3	
4	
5	
6	
7	
8	
9	T
10	T
11	Z Z

Incoming edge(s)	
Last	Node rank
G	1
T	2
C	8
G	9
C	10
G	11

Outgoing edge(s)
Node rank
1
1
1
1
1
1
1
1
1
1
1 0

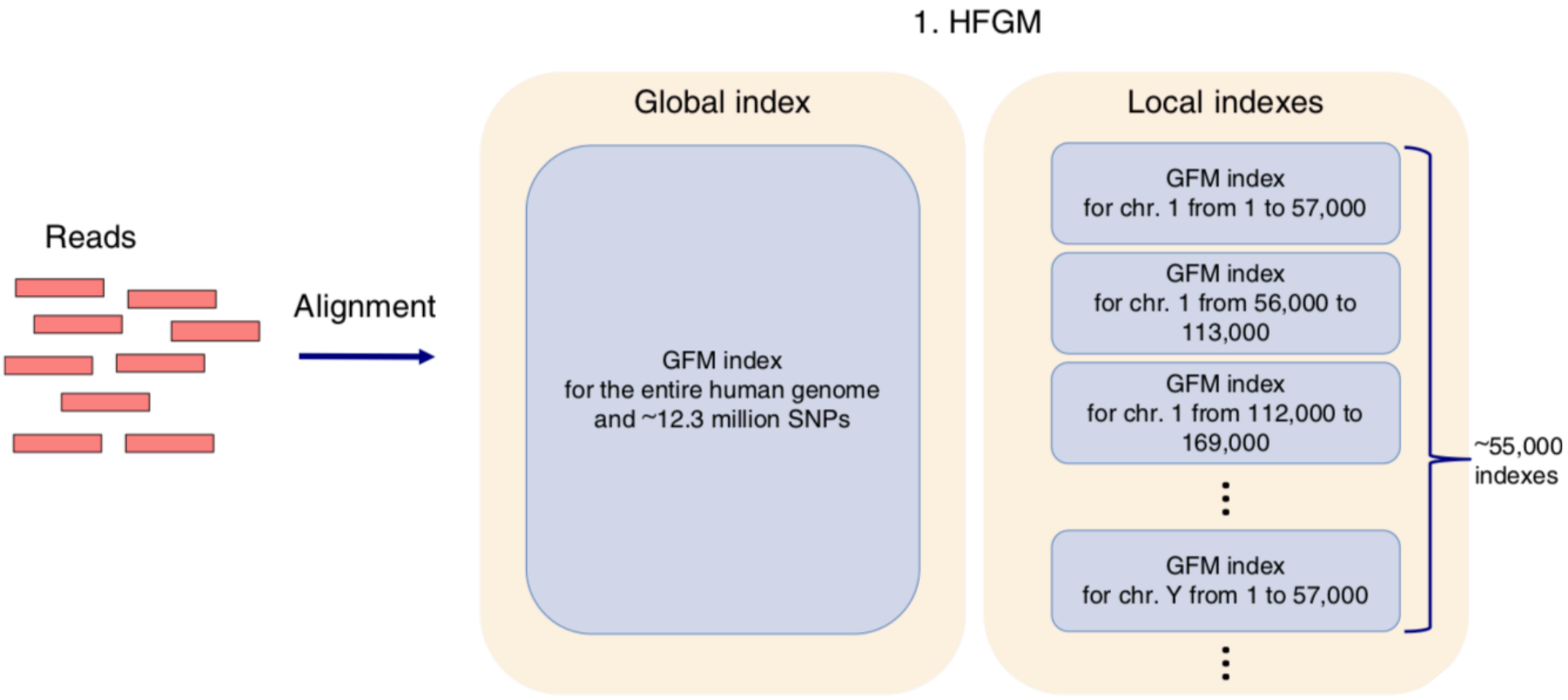
Incoming edge(s)	
Last	Node rank
10	1
11	1
1	1
1	1
1	1
1 0	1
1	1
1 0 0	1
01	1
10	1
01	1
10	1

First	
A	2
C	3
G	4
T	3
Z	2

Think of the GFM index as an index over the edges of the prefix-sorted graph. The tables store the information associated with the starting and ending vertices of the edges, and the edges are grouped by their destination (respectively source) vertices in the two tables.

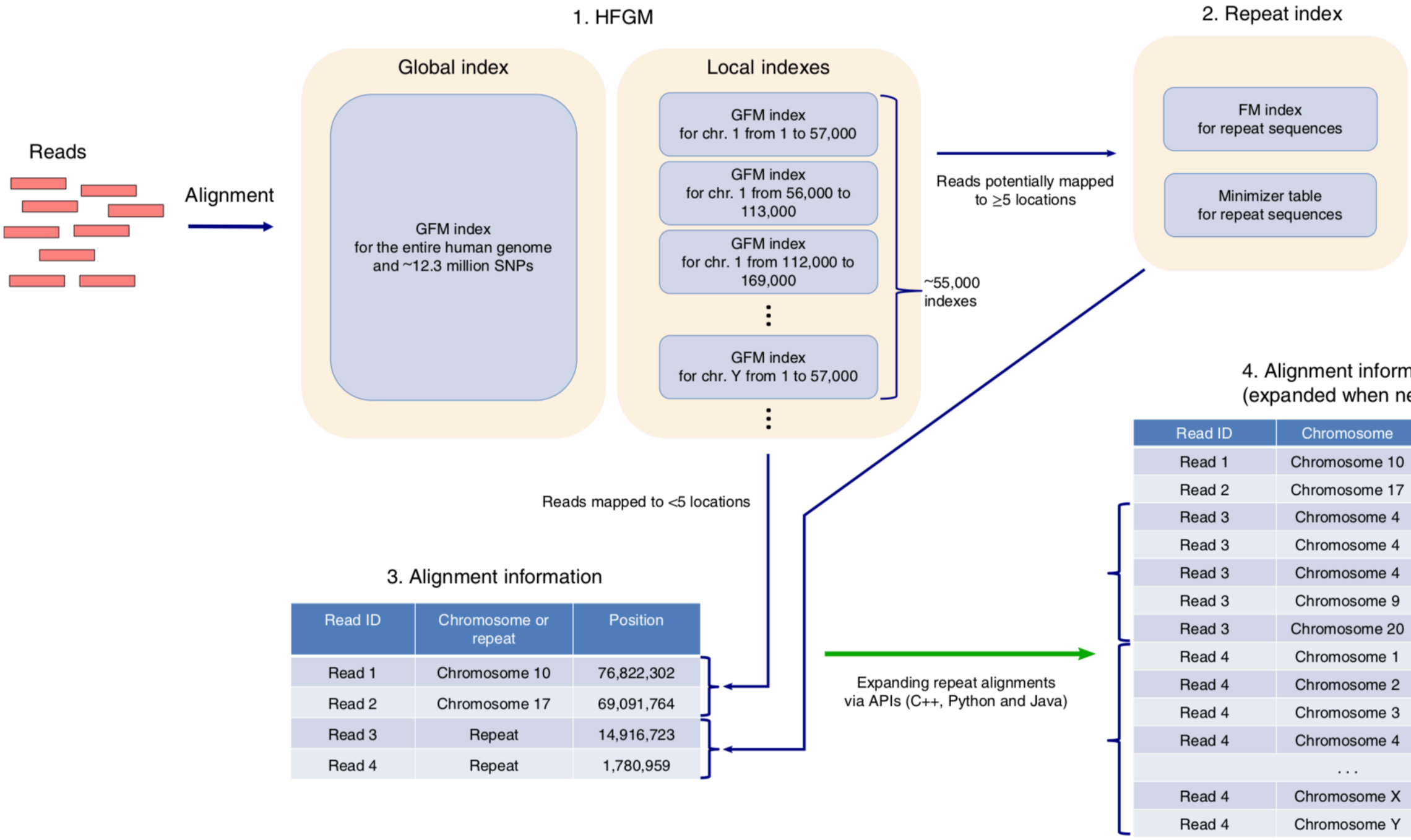
The tables are tied together by the same LF mapping that we used in the linear text FM-index.

Uses same idea as HISAT to make GFM Cache-efficient



Uses same idea as HISAT to make GFM
Cache-efficient

Special handling of repetitive sequences



Read ID	Chromosome or repeat	Position
Read 1	Chromosome 10	76,822,302
Read 2	Chromosome 17	69,091,764
Read 3	Repeat	14,916,723
Read 4	Repeat	1,780,959

Read ID	Chromosome	Position
Read 1	Chromosome 10	76,822,302
Read 2	Chromosome 17	69,091,764
Read 3	Chromosome 4	122,942,920
Read 3	Chromosome 4	123,001,589
Read 3	Chromosome 4	127,862,299
Read 3	Chromosome 9	89,130,544
Read 3	Chromosome 20	16,910,713
Read 4	Chromosome 1	102,931,088
Read 4	Chromosome 2	105,208,801
Read 4	Chromosome 3	174,262,680
Read 4	Chromosome 4	15,844,480
	. . .	
Read 4	Chromosome X	69,679,816
Read 4	Chromosome Y	22,342,611