

The FMD and Graph FM Indices

The FMD Index

Motivation : The FM-index naturally searches in one direction (from the end of the string to the front)

To find MEMs and SMEMs, it will be useful to extend matches in both directions.

Why not two indices? This could be accomplished with 2 FM-indices, but having the search work within a single index will be more efficient.

First formal description of FMD index

BIOINFORMATICS

ORIGINAL PAPER

Vol. 28 no. 14 2012, pages 1838–1844
doi:10.1093/bioinformatics/bts280

Sequence analysis

Advance Access publication May 7, 2012

Exploring single-sample SNP and INDEL calling with whole-genome *de novo* assembly

Heng Li

Medical Population Genetics Program, Broad Institute, 7 Cambridge Center, MA 02142, USA

Associate Editor: Michael Brudno

The FMD Index

Some notation. For pattern P and text T

$$I^l(P) = \min\{k : P \text{ is the prefix of } T_{S(k)}\}$$

$$I^u(P) = \max\{k : P \text{ is the prefix of } T_{S(k)}\}$$

Then $[I^l(P), I^u(P)]$ is the suffix array interval for P

The length of this is given by $I^s(P) = I^u(P) - I^l(P) + 1$

Some notation. For pattern P and text T

The FMD Index

Let R_0, R_1, \dots, R_{n-1} denote a series of DNA/RNA texts

Define a new text $T = R_0 \bar{R}_0 R_1 \bar{R}_1 \dots R_{n-1} \bar{R}_{n-1}$

Where \bar{R} is the reverse complement of R

Consider *bi-intervals* of the index of the form $[I^l(P), I^l(\bar{P}), I^s(P)]$

Also, recall that we can extend a “normal” interval as

$$I^l(aP) = C(a) + O(a, I^l(P) - 1)$$

$$I^u(aP) = C(a) + O(a, I^u(P)) - 1$$

The FMD Index

Assume we have the bi-interval of P , $[I^l(P), I^l(\overline{P}), I^s(P)]$

How do we compute the bi-interval of aP ?

We know that $[I^l(\overline{aP}), I^u(\overline{aP})]$ is a subinterval of

$[I^l(\overline{P}), I^u(\overline{P})]$, why?

Because P is a prefix of $\overline{aP} = \overline{P} \circ \overline{a}$

Further, because of the symmetry of T , $I^s(cP) = I^s(\overline{cP}), \forall c$

The FMD Index

Example:

$W = \text{AACG}$

$a = \text{G}$

$aW = \text{GAACG}$

$\overline{Wa} = \text{CGTTC}$

Consider symmetry of T:

$\# \text{AACG} = \# \text{CGTT}$

$\# \text{AAACG} = \# \text{CGTTT}$

$\# \text{CAACG} = \# \text{CGTTG}$

$\# \text{GAACG} = \# \text{CGTTC}$

$\# \text{TAACG} = \# \text{CGTTA}$

So, given \overline{W} , to extend to \overline{Wa} ,
we can simply *count*!

Algorithm 2: Backward extension

Input: Bi-interval $[k, l, s]$ of string W and a symbol a

Output: Bi-interval of string aW

Function BACKWARDEXT($[k, l, s], a$) **begin**

for $b \leftarrow 0$ **to** 5 **do**

$k_b \leftarrow C(b) + O(b, k - 1)$

$s_b \leftarrow O(b, k + s - 1) - O(b, k - 1)$

$l_0 \leftarrow l;$

$l_4 \leftarrow l_0 + s_0;$

for $b \leftarrow 3$ **to** 1 **do**

$l_b \leftarrow l_{b+1} + s_{b+1}$

$l_5 \leftarrow l_1 + s_1;$

return $[k_a, l_a, s_a]$

This is the part that
requires some thought

$\$, A, C, G, T, N$

$0, 1, 2, 3, 4, 5$

The FMD Index

Forward extension is simply backward extension in the reverse complement!

Algorithm 3: Forward extension

Input: Bi-interval $[k, l, s]$ of string W and a symbol a

Output: Bi-interval of string Wa

Function FORWARDEXT($[k, l, s], a$) **begin**

$[l', k', s'] \leftarrow \text{BACKWARDEXT}([l, k, s], \bar{a});$
 return $[k', l', s']$

Finding SMEMs with the FMD Index

Algorithm 5: Finding SMEMs

Input: String P and start position i_0 ; $P[-1]=0$

Output: Set of bi-intervals of SMEMs overlapping i_0

Function SUPERMEM1(P, i_0) **begin**

Initialize Curr, Prev and Match as empty arrays;

$[k, l, s] \leftarrow [C(P[i_0]), C(\overline{P[i_0]}), C(P[i_0]+1) - C(P[i_0])]$;

for $i \leftarrow i_0 + 1$ **to** $|P|$ **do**

if $i = |P|$ **then**

 Append $[k, l, s]$ to Curr

else

$[k', l', s'] \leftarrow \text{FORWARD_EXT}([k, l, s], P[i])$;

if $s' \neq s$ **then**

 Append $[k, l, s]$ to Curr

if $s' = 0$ **then**

break;

$[k, l, s] \leftarrow [k', l', s']$

Swap array Curr and Prev;

$i' \leftarrow |P|$;

for $i \leftarrow i_0 - 1$ **to** -1 **do**

 Reset Curr to empty;

$s'' \leftarrow -1$;

for $[k, l, s]$ in Prev **do**

$[k', l', s'] \leftarrow \text{BACKWARD_EXT}([k, l, s], P[i])$;

if $s' = 0$ **or** $i = -1$ **then**

if Curr is empty **and** $i + 1 < i' + 1$ **then**

$i' \leftarrow i$;

 Append $[k, l, s]$ to Match

if $s' \neq 0$ **and** $s' \neq s''$ **then**

$s'' \leftarrow s'$;

 Append $[k, l, s]$ to Curr

if Curr is empty **then**

break

 Swap Curr and Prev;

return Match

Extend “forward”

i.e. find left-maximal matches
(intervals)


Extend “backward”

i.e. find right-maximal for left-
maximal intervals

The Graph FM-Index & HISAT2

Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype

Daehwan Kim , Joseph M. Paggi, Chanhee Park, Christopher Bennett & Steven L. Salzberg

Nature Biotechnology **37**, 907–915 (2019) | [Download Citation](#) 

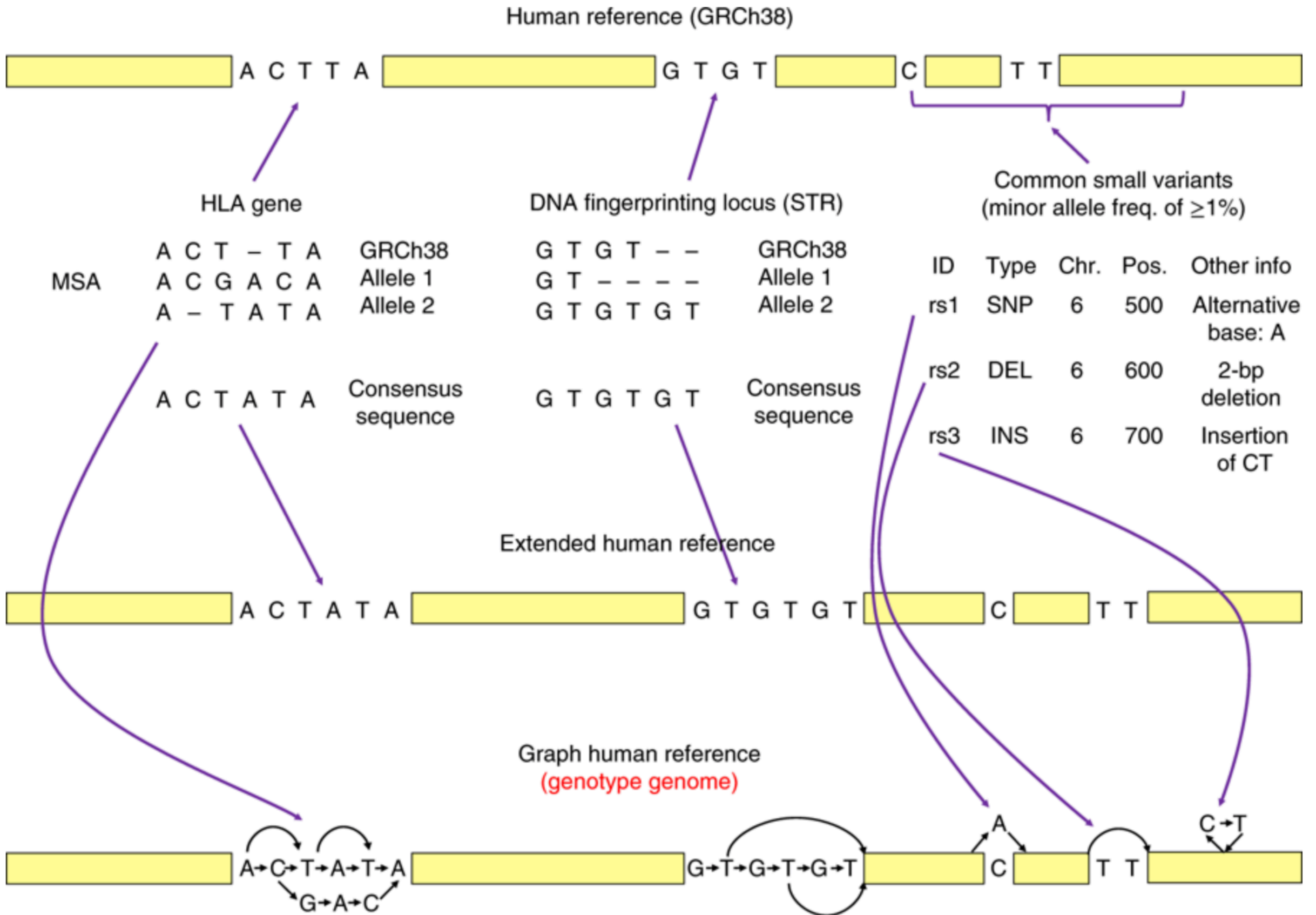
The Graph FM-Index

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.

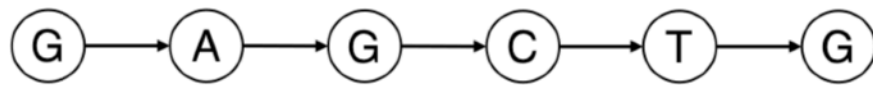
HISAT2 is one of a new breed of “graph” aligners, that views the genome as a graph rather than a simple string. This framework allows encoding variants as alternative “paths” through the genome.

The Graph FM-Index



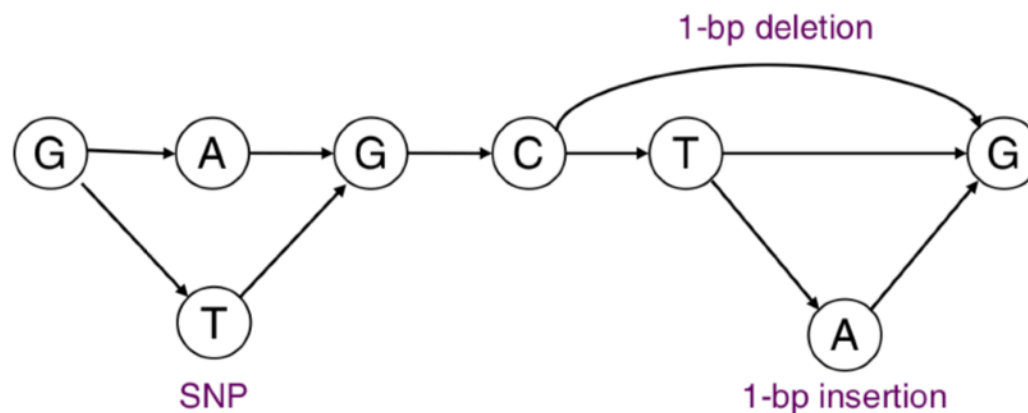
The Graph FM-Index

1. Reference sequence (6 bp long)



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

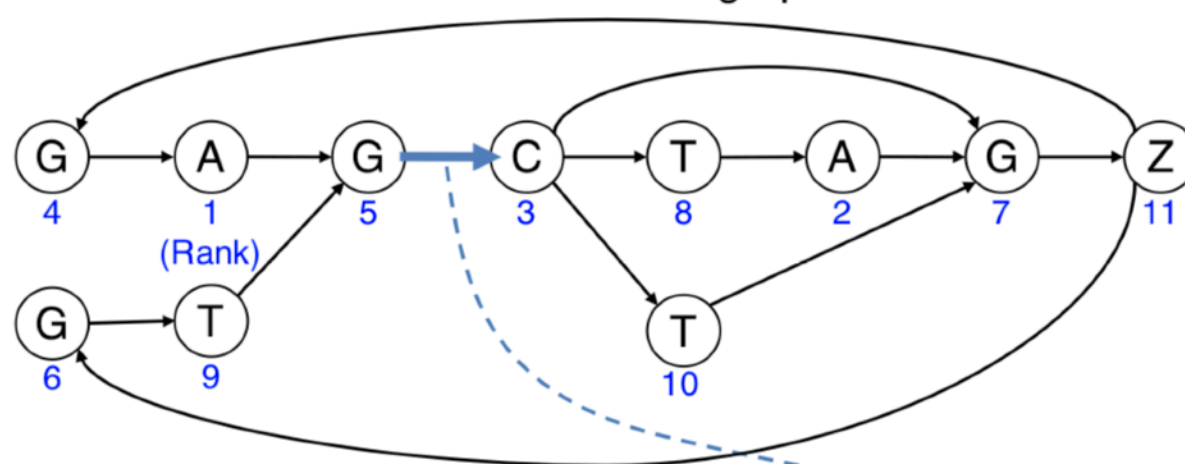
2. Graphical representation (original graph)



Prefix doubling and pruning



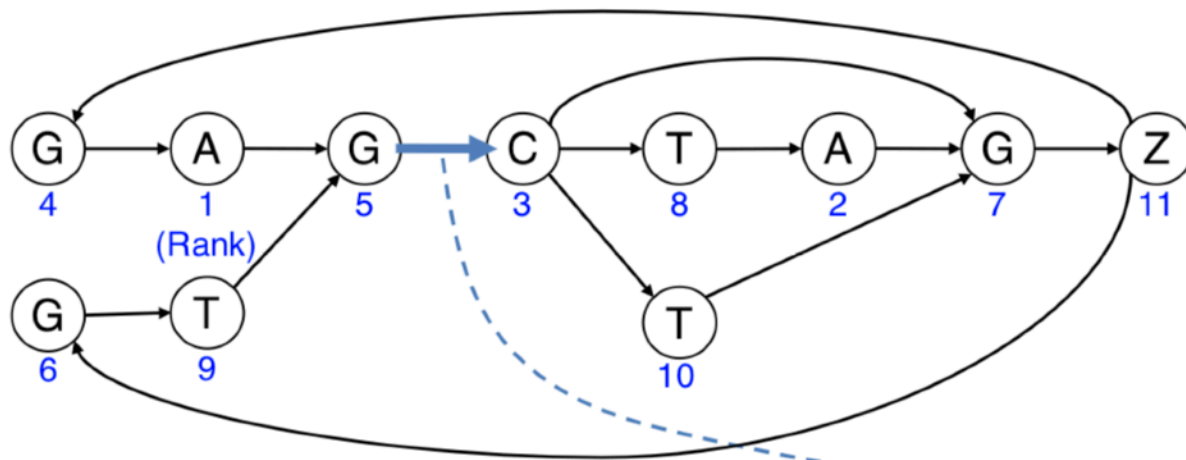
3. Prefix-sorted graph



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		G	3
	C		Z	4
	C		A	5
4	G		T	7
5	G		Z	
6	G		A	8
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



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
	C		A	5
4	G		T	6
5	G		Z	
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

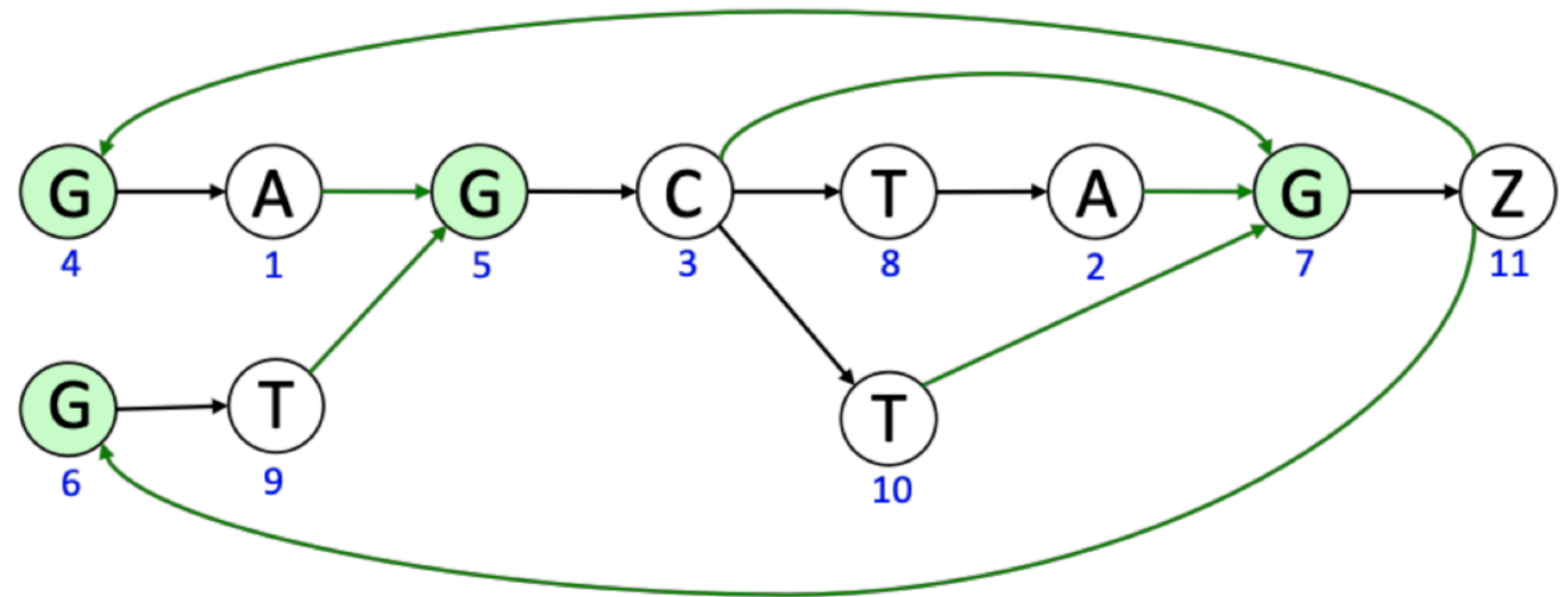
The Graph FM-Index

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

Searching the Graph FM-Index

Query : TAG

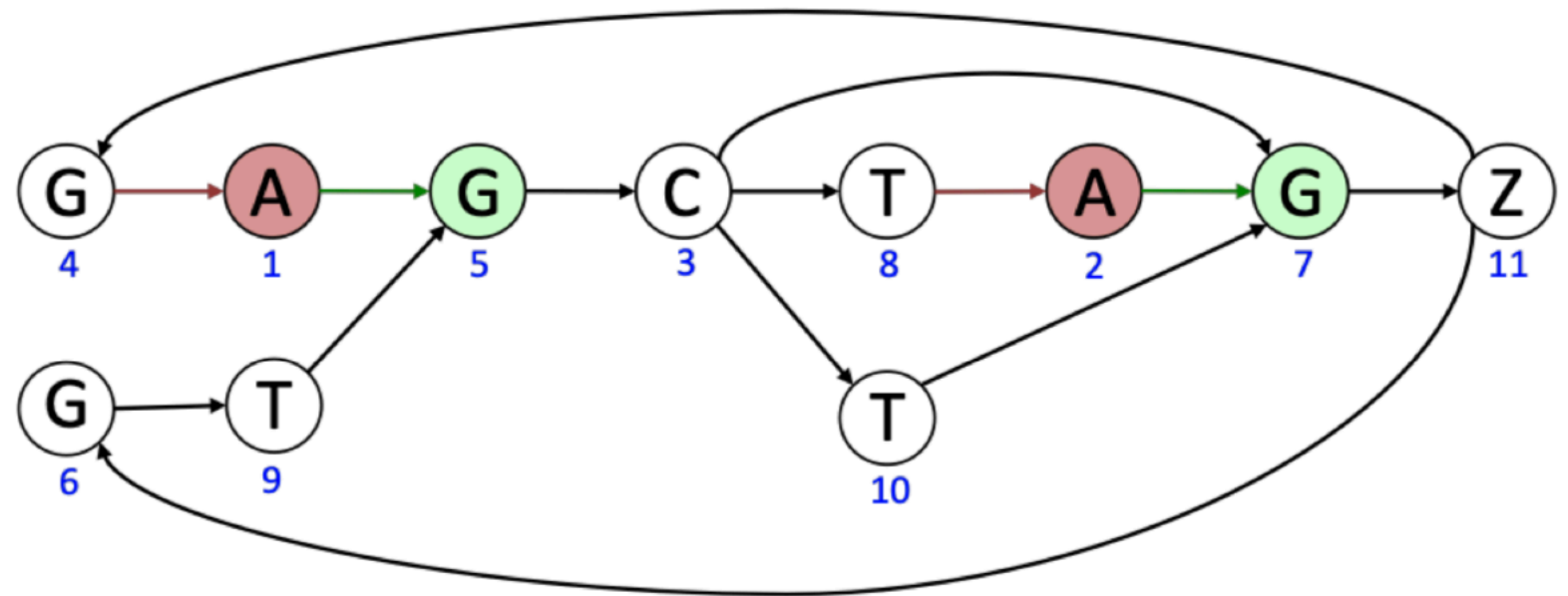
Outgoing edge(s)			Incoming edge(s)	
Node ID	First		Last	Node ID
1	A	1 ↑	G	1
2	A		T	2
3	C		G	3
	C		Z	4
	C		A	5
4	G		T	5
5	G		Z	6
6	G		A C T	7
7	G			
8	T			
9	T		C	8
10	T		G	9
11	Z		C	10
	Z		G	11



Searching the Graph FM-Index

Query : TAG

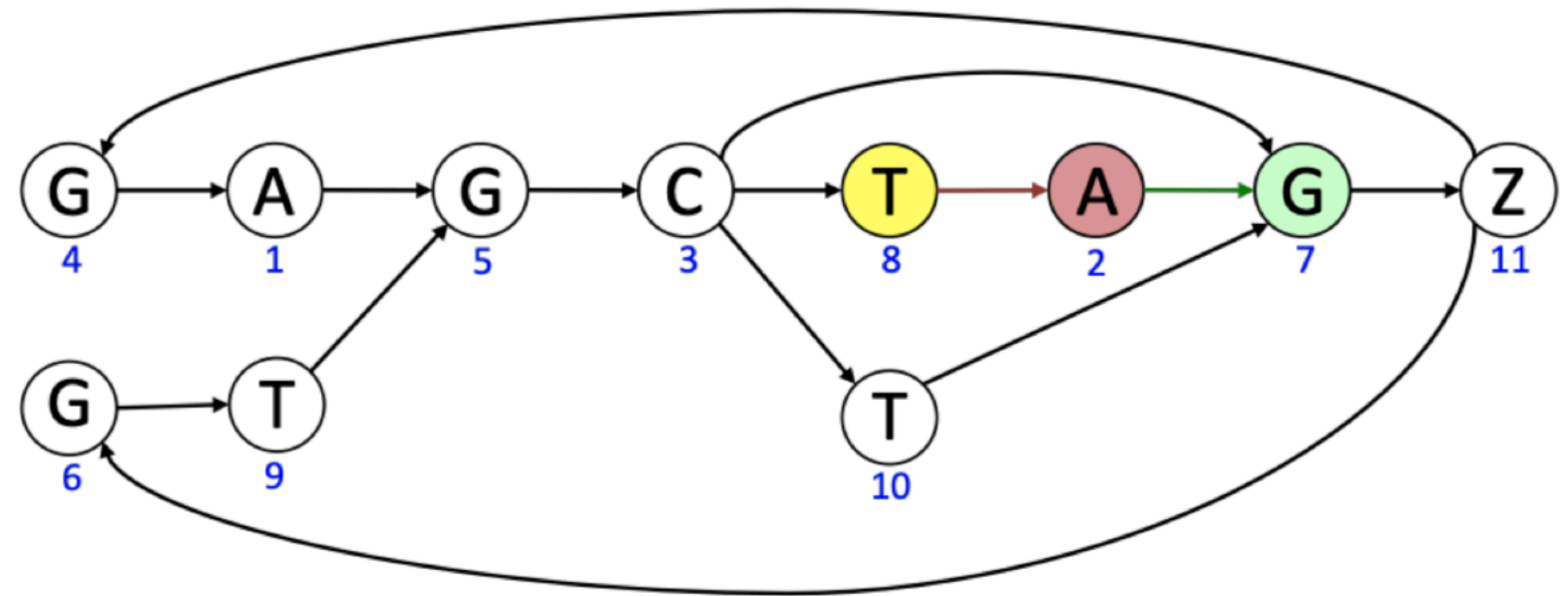
Outgoing edge(s)			Incoming edge(s)	
Node ID	First		Last	Node ID
1	A	2 ↙	G	1
2	A		T	2
3	C		G	3
4	G	1 ↗	Z	4
5	G		A	5
6	G		T	5
7	G		Z	6
8	T		A	7
9	T		C	
10	T		T	
11	Z		C	8
	Z		G	9
			C	10
			G	11



Searching the Graph FM-Index

Query : TAG

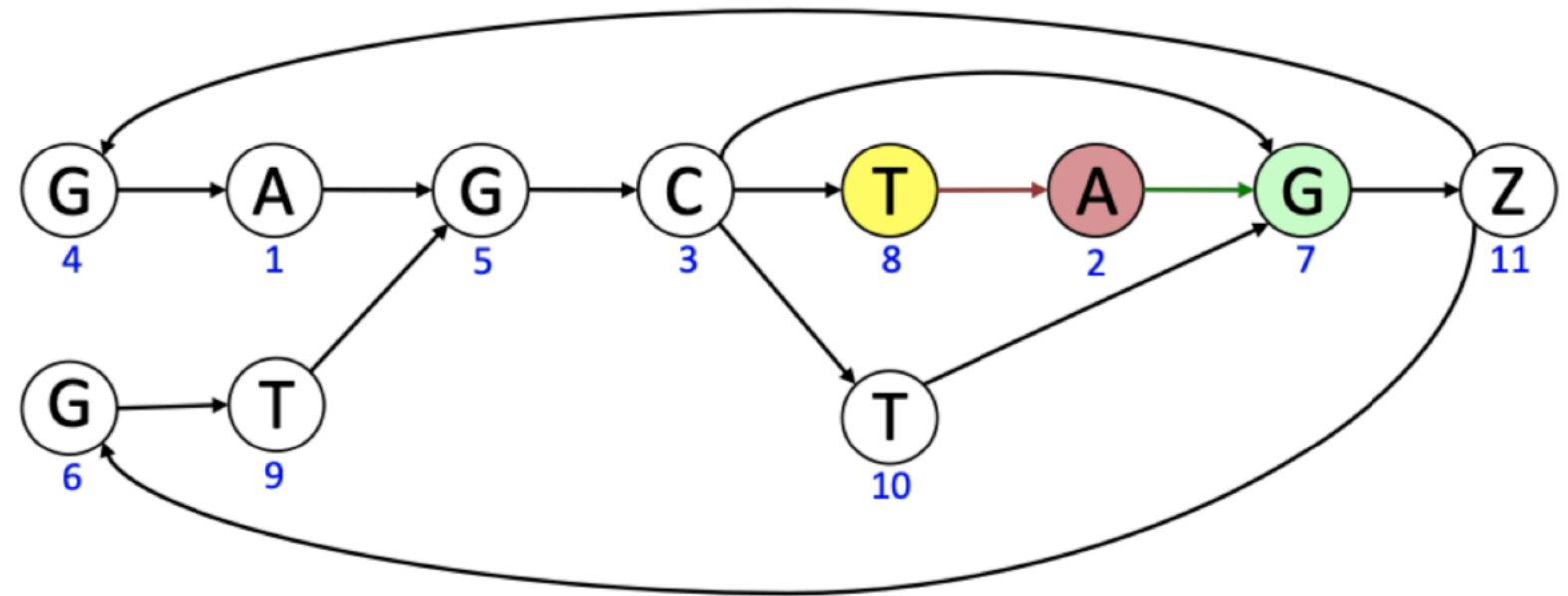
Outgoing edge(s)			Incoming edge(s)	
Node ID	First		Last	Node ID
1	A	3 →	G	1
2	A		T	2
3	C	2 ←	G	3
	C		Z	4
	C		A	5
4	G	1 ↗	T	5
5	G		Z	6
6	G		A C T	7
7	G			
8	T			
9	T		C	8
10	T		G	9
11	Z		C	10
	Z		G	11



Searching the Graph FM-Index

Query : TAG

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



How to store the GFM efficiently

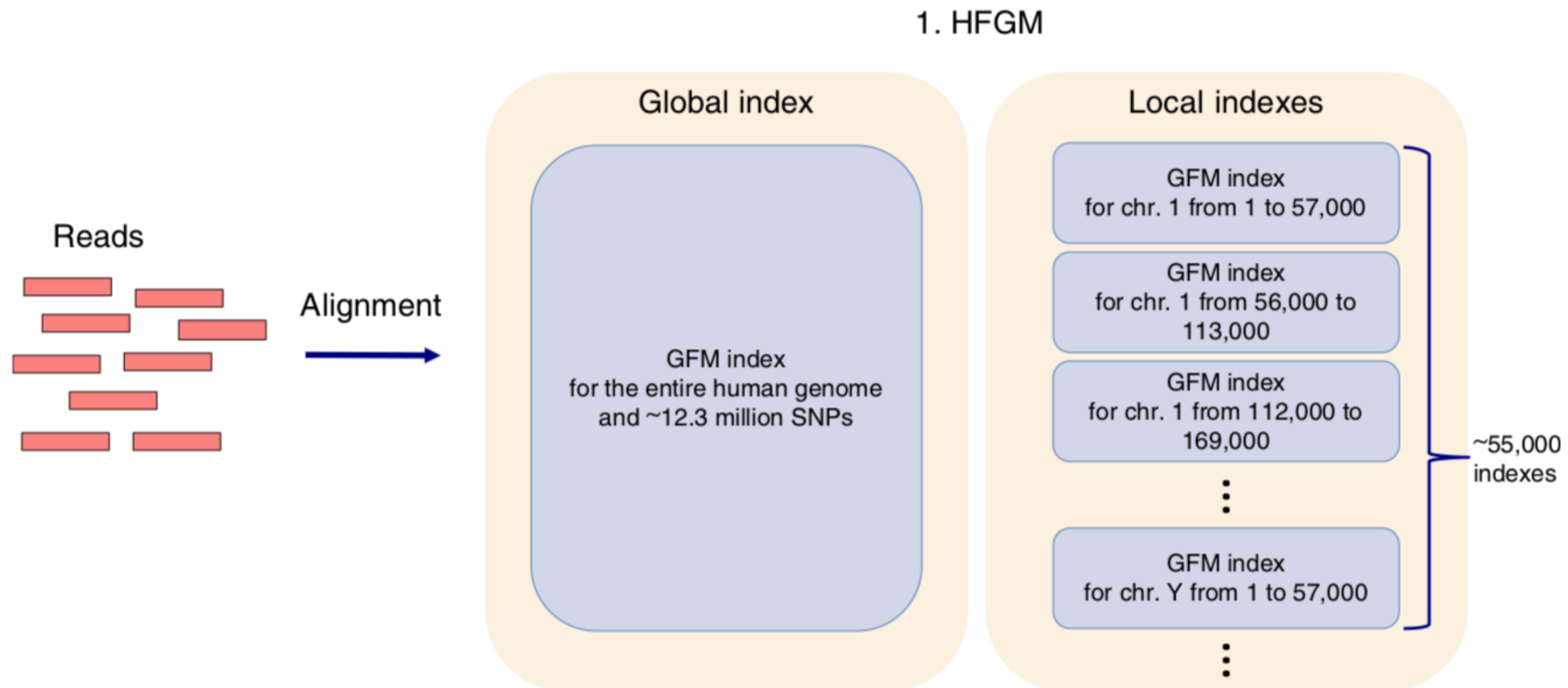
Outgoing edge(s)			Incoming edge(s)	
Node rank	First		Last	Node rank
1	A		G	1
2	A		T	2
3	C		G	3
	C		Z	4
	C		A	5
4	G		T	6
5	G		Z	
6	G		A	
7	G		C	7
8	T		T	
9	T		C	
10	T		G	8
11	Z		C	9
	Z		G	10



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

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

