Indexing Graph Genomes

Jouni Sirén

Human genome

- A ~3 Gbp sequence?
- Two very similar ~3 Gbp sequences?
- A reference sequence biased towards certain individuals and populations?
- A reference sequence and ~88 million common variants?
- A graph?

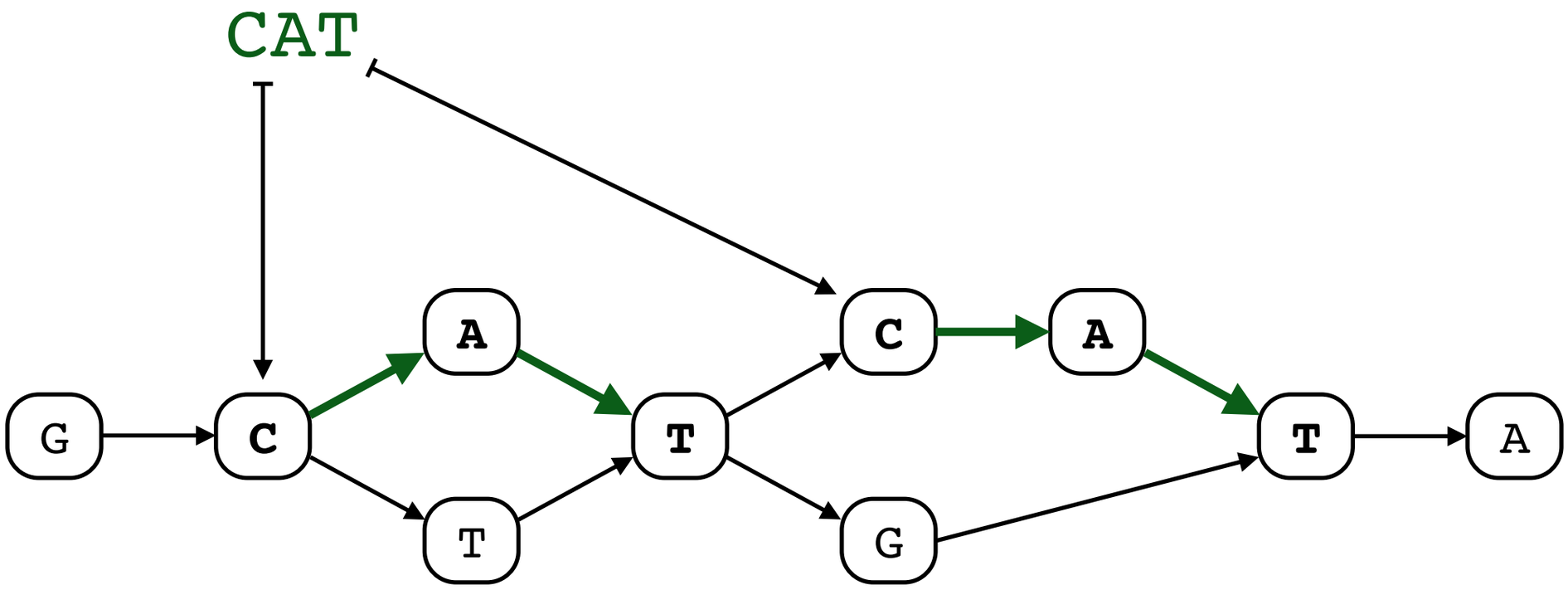
https://github.com/ekg/vg

- Reference sequences are giving way to reference graphs.
- The Global Alliance for Genomics and Health leads an effort for producing the first human graph genome.
- This talk is about the graph indexing approach I have developed for Erik Garrison's vg, which is the main toolkit for working with variation graphs.

Path Indexes

Path queries are one of the basic operations in variation graphs. Given a k-mer, we want to find the paths in the graph labeled with that k-mer.

If the graph is static and large, we may want to index the graph.



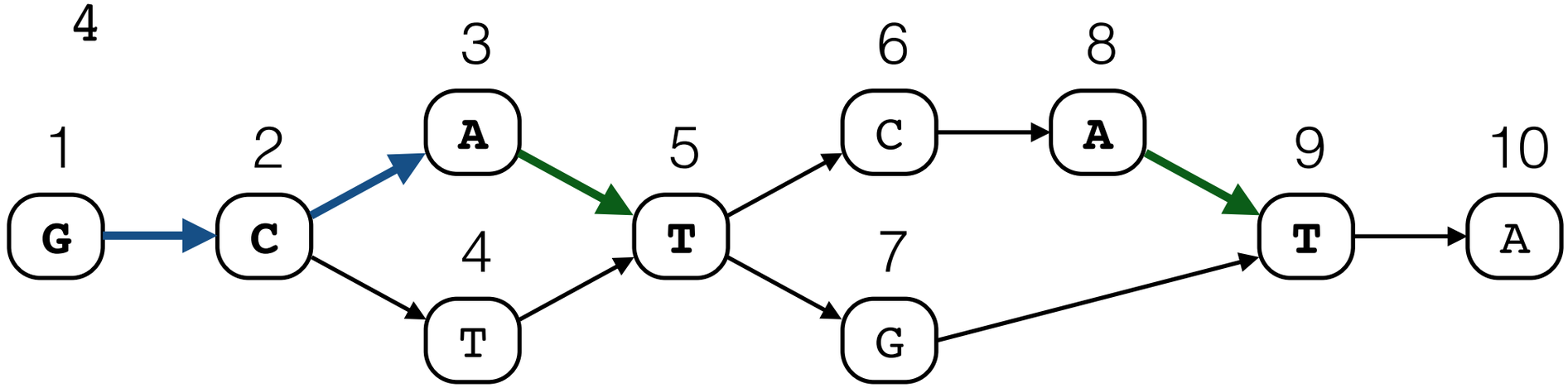
The basic indexing techniques are known but computationally expensive.

J. Sirén, N. Välimäki, V. Mäkinen: Indexing Graphs for Path Queries with Applications in Genome Research. WABI 2011, TCBB 2014.

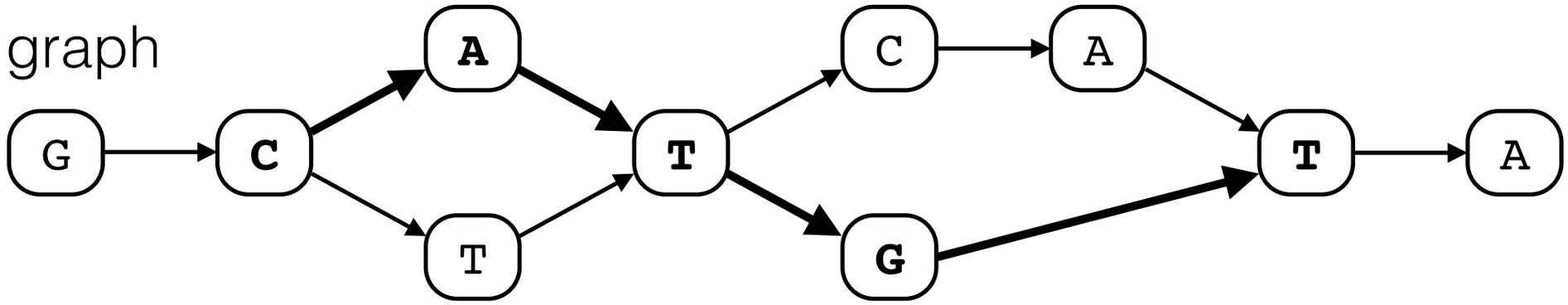
A. Bowe, T. Onodera, K. Sadakane, T. Shibuya: Succinct de Bruijn Graphs. WABI 2012.

```
A$$   10
ATA   8
ATC   3
ATG   3
CAT   2, 6
CTT   2
GCA   1
GCT   1
GTA   7
TA$   9
TCA   5
TGT   5
TTC   4
TTG   4
```

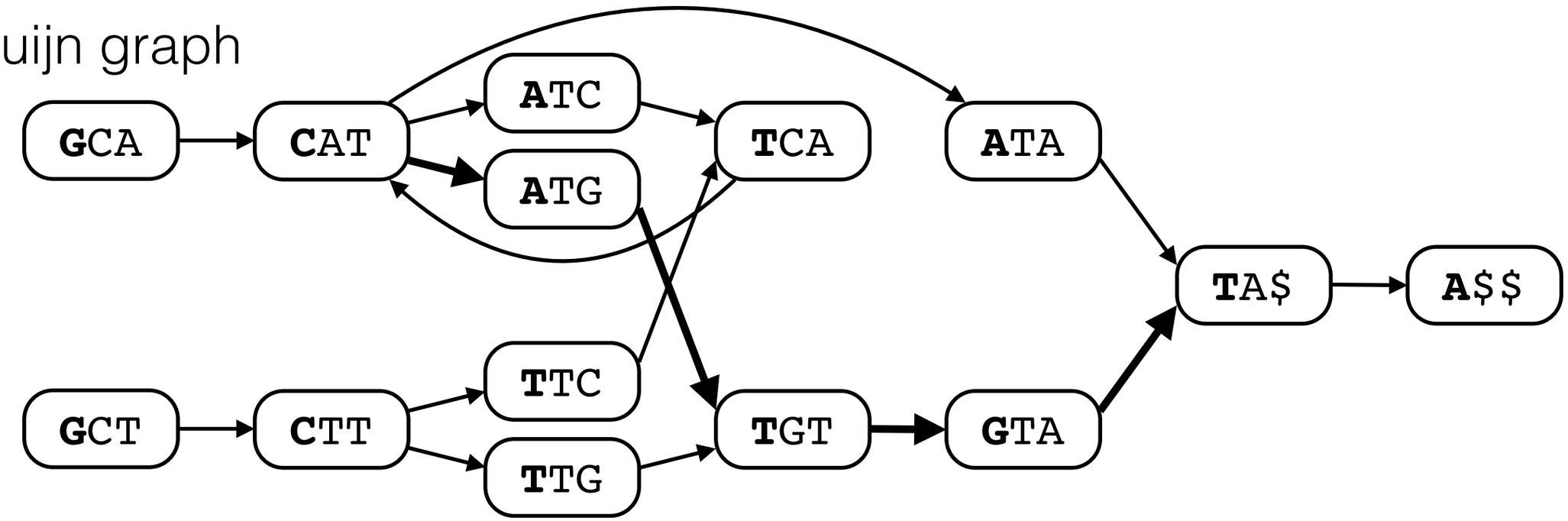
A k-mer index based on a hash table supports queries of length k efficiently. We can also sort the k-mers and use them as a suffix array-like index for shorter queries.



Original graph

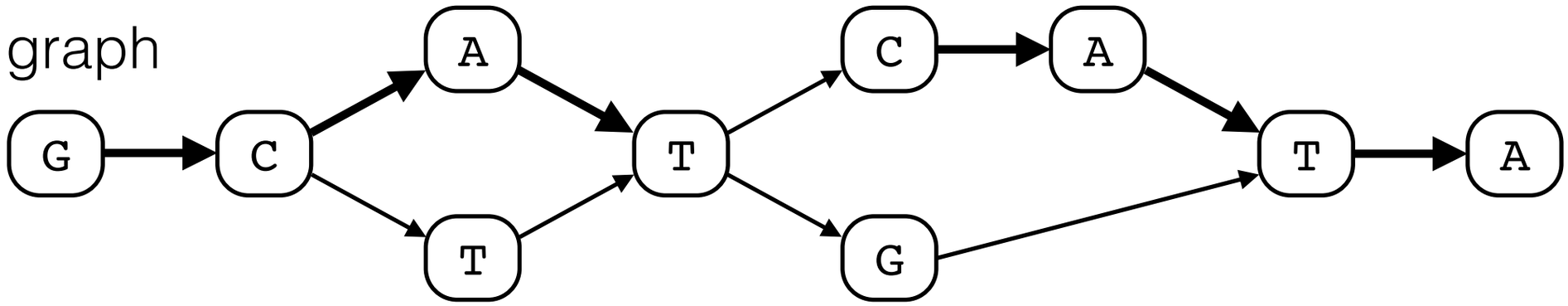


de Bruijn graph

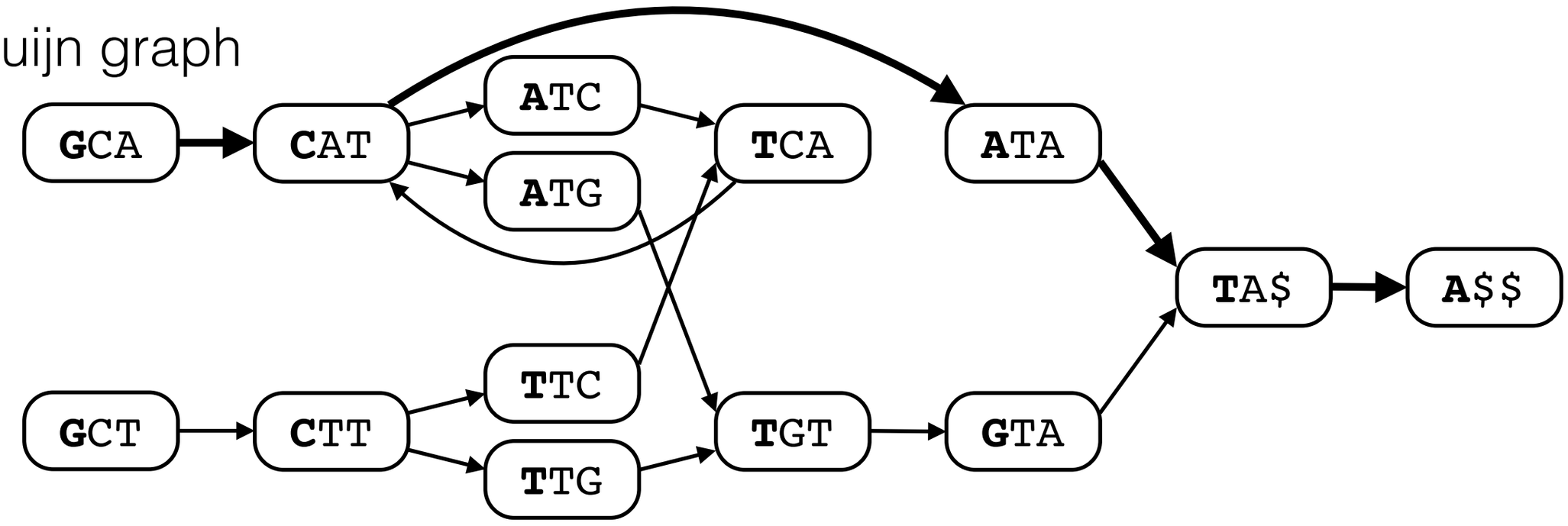


We can search for longer patterns by representing the k-mer index as a de Bruijn graph.

Original graph



de Bruijn graph



The results of long queries must be verified in the original graph to avoid false positives.

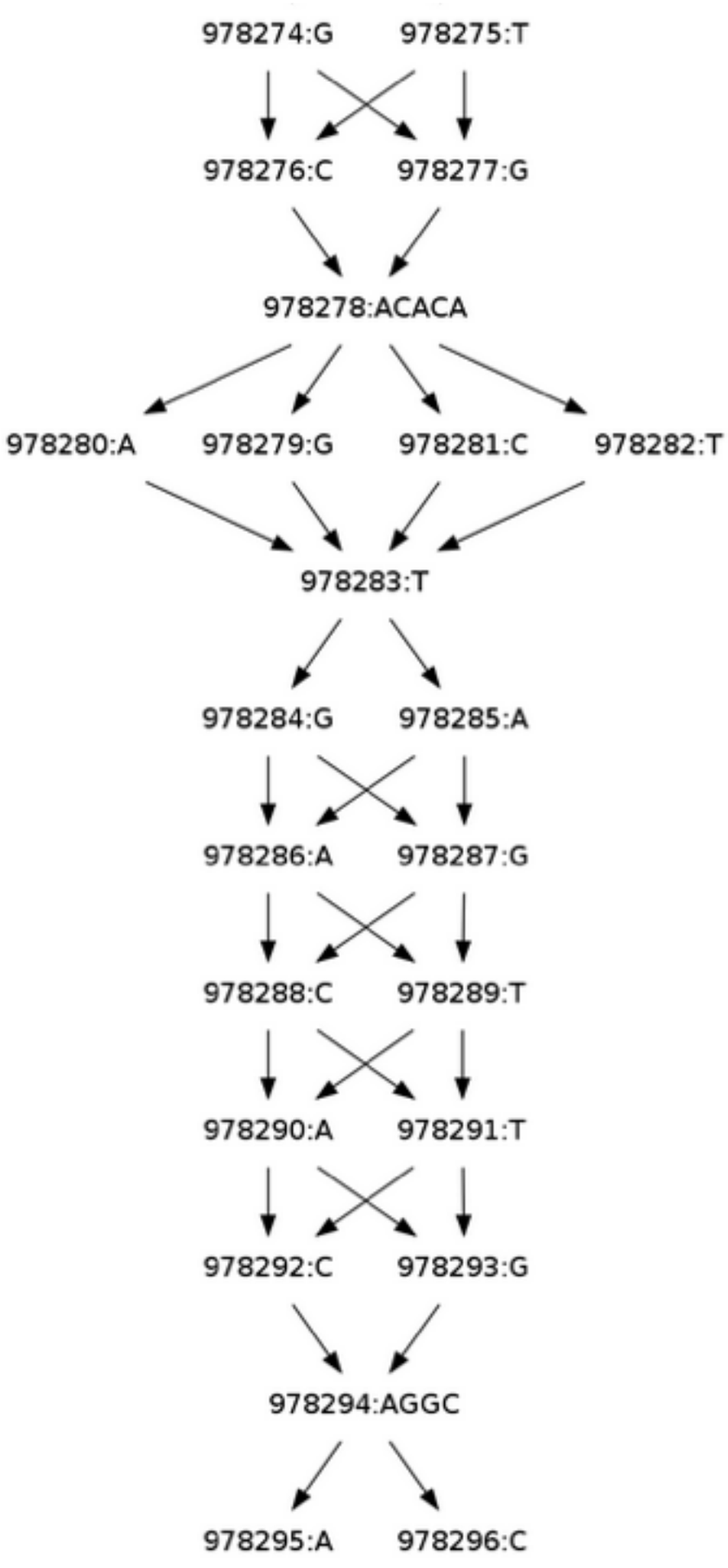
The value k should be large enough that we do not have to verify the results of most queries.

The de Bruijn graph should be small enough to fit in memory.

For a variation graph with ~3 Gbp of sequence data, the de Bruijn graph should ideally have no more than 5 billion nodes.

We can handle graphs with 20 billion nodes.

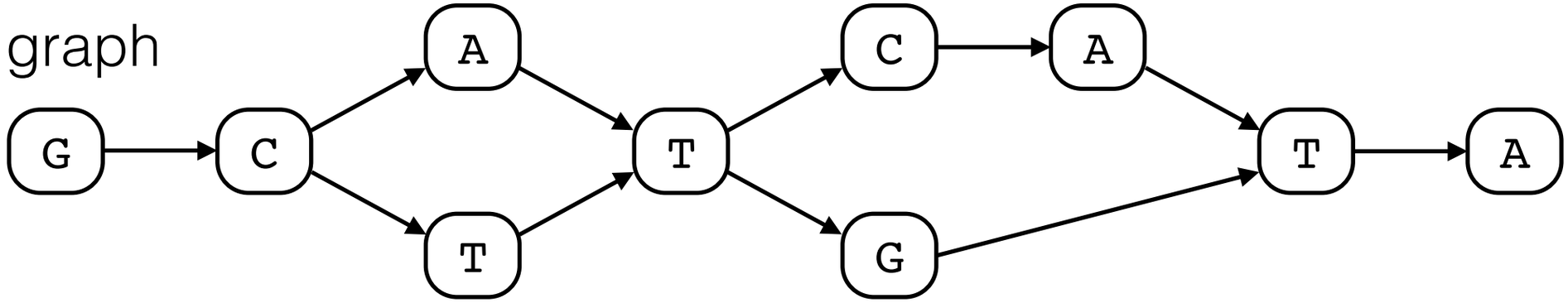
Without any pruning, a de Bruijn graph with $k = 64$ for a human variation graph will have (at least) hundreds of billions of nodes.



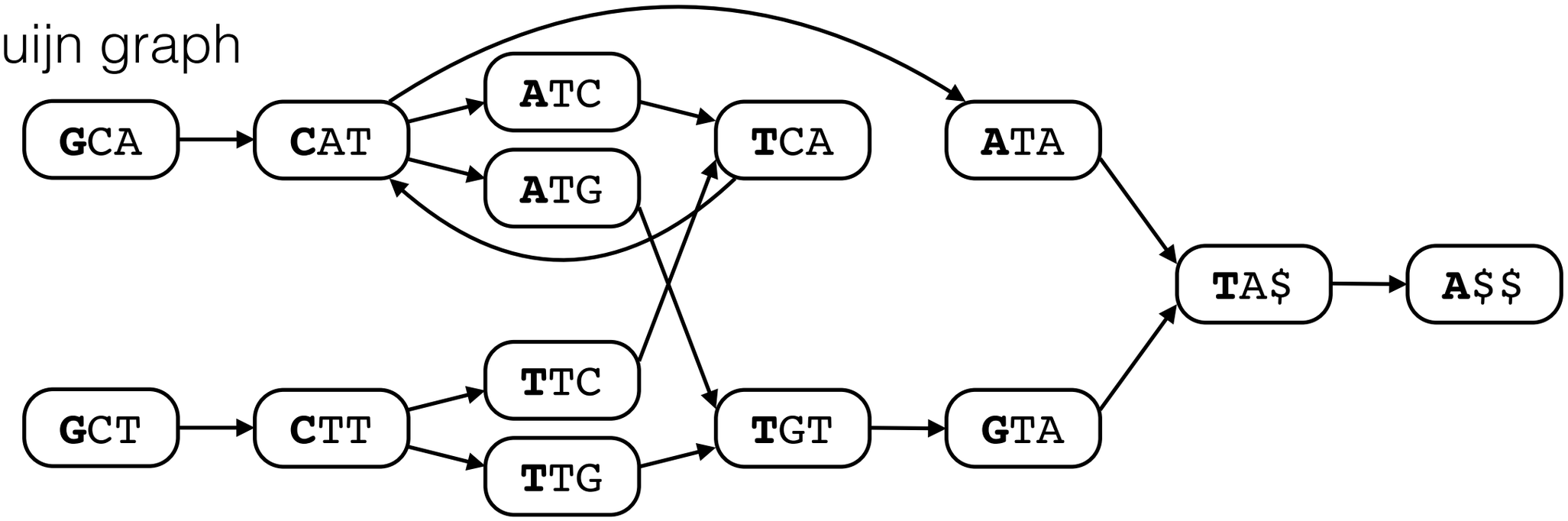
Some parts of the original graph may have **too many paths** through them. Those parts must be **pruned** before indexing.

We can also prune the de Bruijn graph in places, where shorter labels **uniquely determine** the position in the original graph.

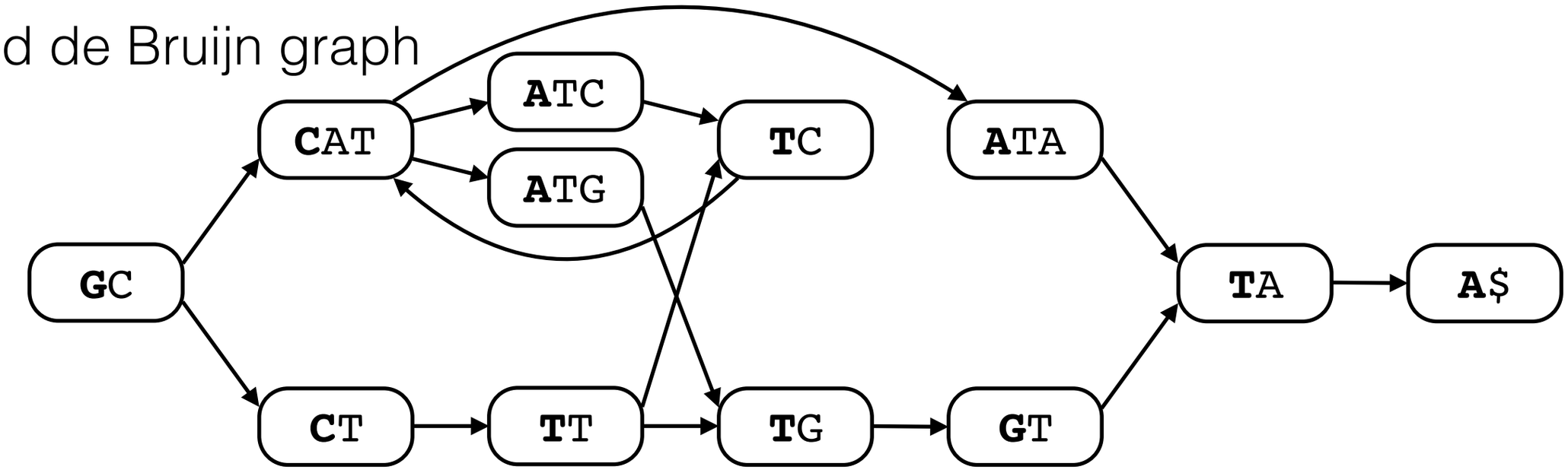
Original graph



de Bruijn graph



Pruned de Bruijn graph



GCSA2

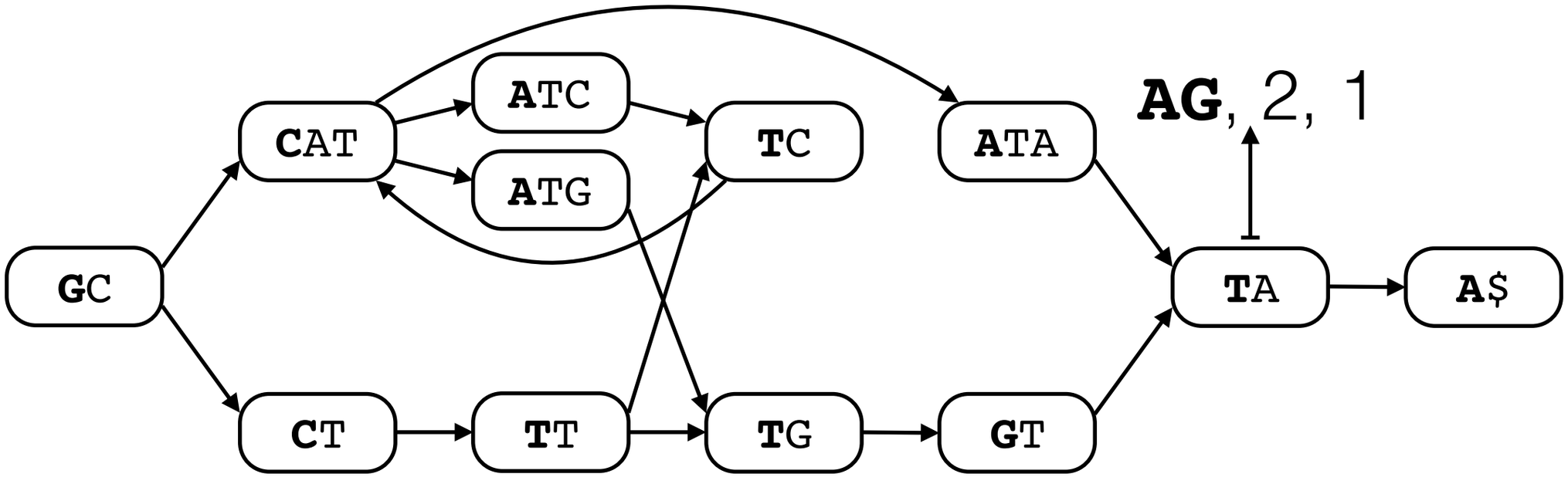
GCSA2

- Graph indexing library based on the Burrows-Wheeler transform and pruned de Bruijn graphs.

- Takes the input as a list of paths of length $k \leq 16$ and builds an index of order $2k$, $4k$, or $8k$.

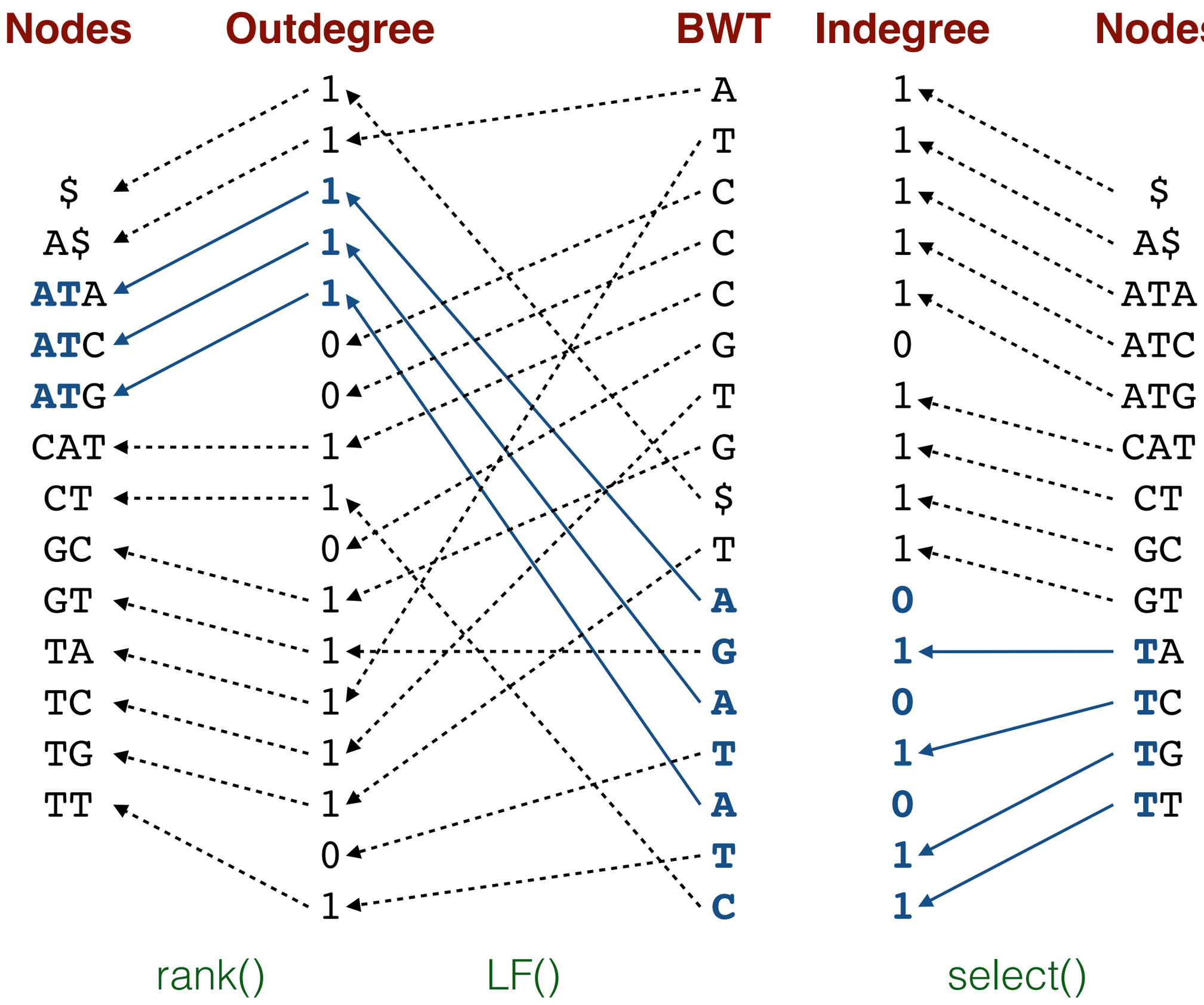
- The implementation is based on the Succinct Data Structures Library (SDSL).

- https://github.com/jltsiren/gcsa2



We store predecessor labels, indegree, and outdegree for each node. For the nodes at the beginning of unary paths, we also store pointers to the original graph. Edges can be determined if the nodes are stored in sorted order.

The encoding is similar to the Burrows-Wheeler transform and the FM-index. Typical space usage is 1–2 bytes/node.


Nodes
Outdegree
BWT
Indegree
Nodes
$
A$
ATA
ATC
ATG
CAT
CT
GC
GT
TA
TC
TG
TT
1
1
1
1
1
0
0
1
1
1
1
1
1
0
1
A
T
C
C
C
G
T
G
$
T
A
G
A
T
A
T
C
1
1
1
1
0
1
1
1
1
1
0
1
1
$
A$
ATA
ATC
ATG
CAT
CT
GC
GT
TA
TC
TG
TT
rank()
LF()
select()

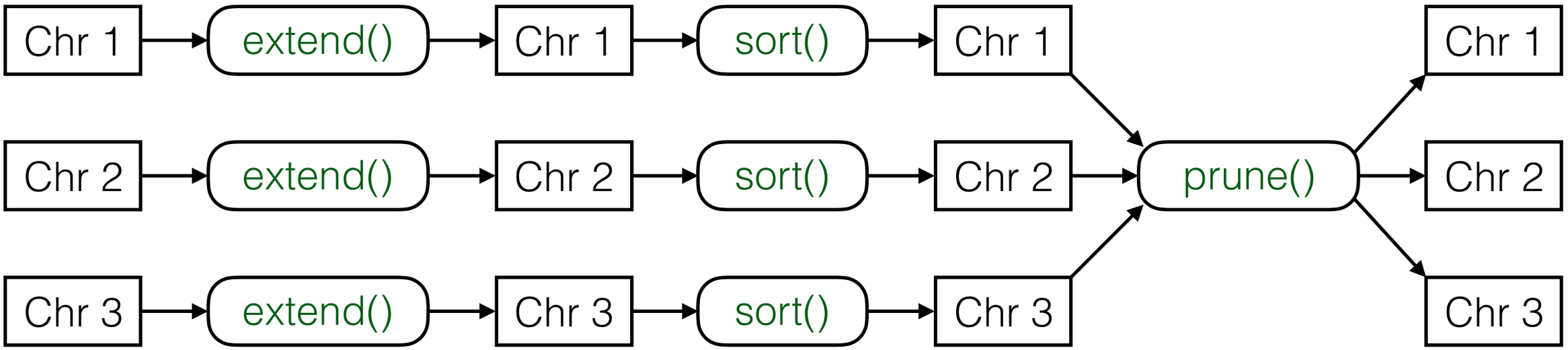
GCSA construction

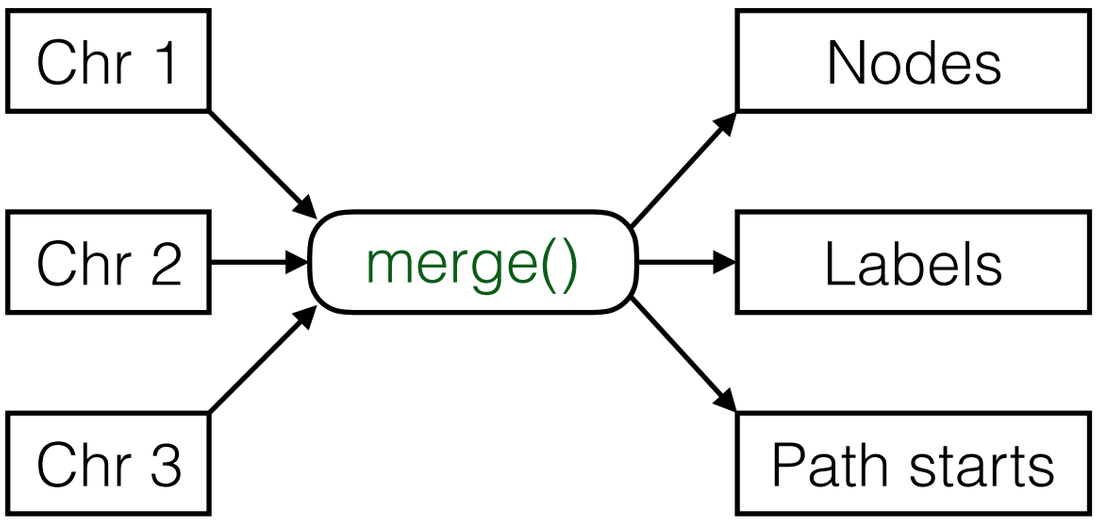
- Start from paths of length k and use a prefix-doubling algorithm to build the pruned de Bruijn graph.

- extend(): Double the path length by joining paths A→B and B→C into paths A→C.

- prune(): If all paths sharing a common prefix start from the same node, merge them into a single path.

- merge(): Merge all paths with the same label.





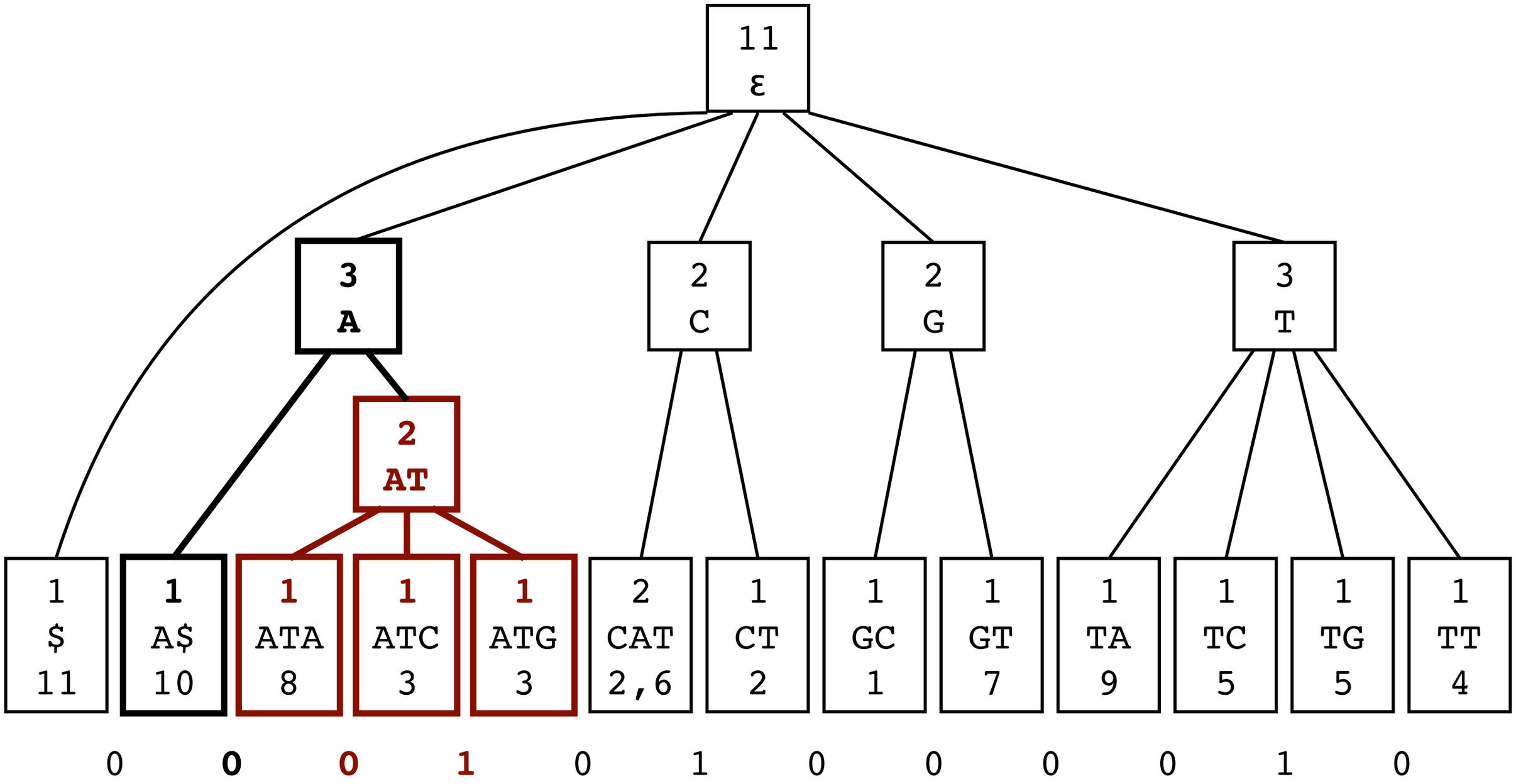
The files are sorted by path labels. GCSA construction determines the lexicographic ranges of potential predecessors of each node in the pruned de Bruijn graph and creates an edge from each node intersecting with the range.

Path length	16→32	16→64	16→128
Nodes:			
de Bruijn graph	6.23G	16.9G	118G
Pruned	4.39G	5.27G	5.76G
Index size:			
Full index	9.99 GB	9.22 GB	9.23 GB
Without pointers	4.10 GB	4.84 GB	5.27 GB
Construction:			
Time	7.20 h	11.4 h	15.5 h
Memory	43.8 GB	43.8 GB	43.8 GB
Disk	401 GB	424 GB	489 GB
I/O:			
Read	1.43 TB	2.11 TB	2.89 TB
Write	1.05 TB	1.71 TB	2.47 TB

1000GP human variation, vg mod -p -l 16 -e 4 | vg mod -S -l 100
32 cores, 256 GB memory, distributed Lustre file system

Extensions

Counting queries

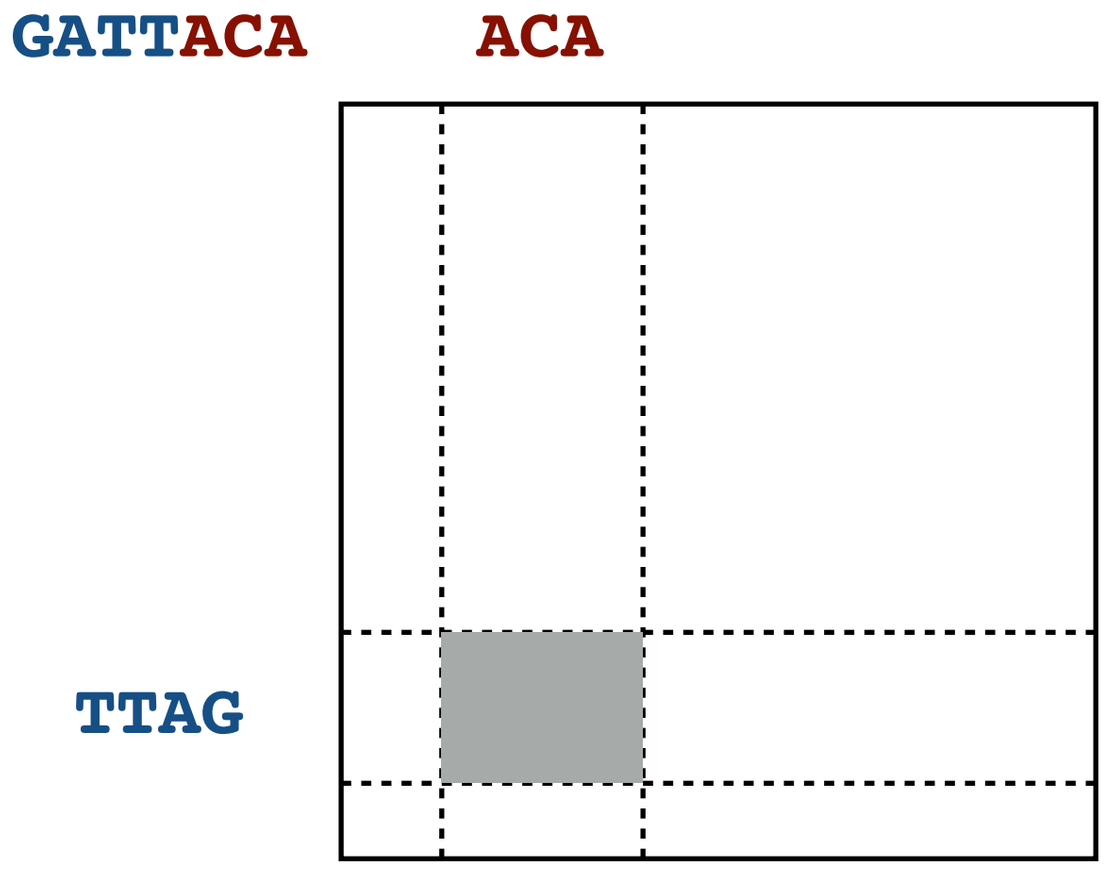


K. Sadakane: Succinct data structures for flexible text retrieval systems. JDA, 2007.

Removed edges

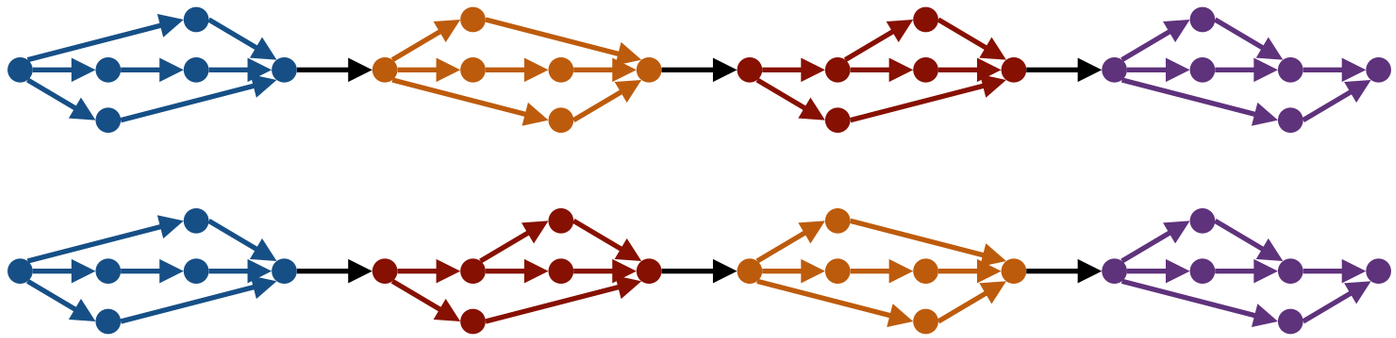
Complex regions of the graph must be pruned before indexing. We can extend the index to find matches crossing one pruned edge.

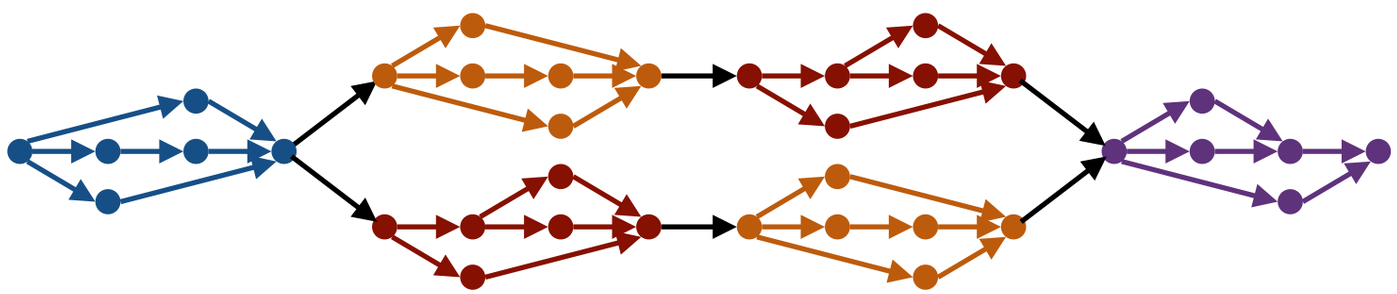
- Split the pattern into prefix-suffix pairs.
- Search for the reverse prefix in the reverse index and for the suffix in the forward index.
- Combine the matches with a 2D range query over the matrix of removed edges.



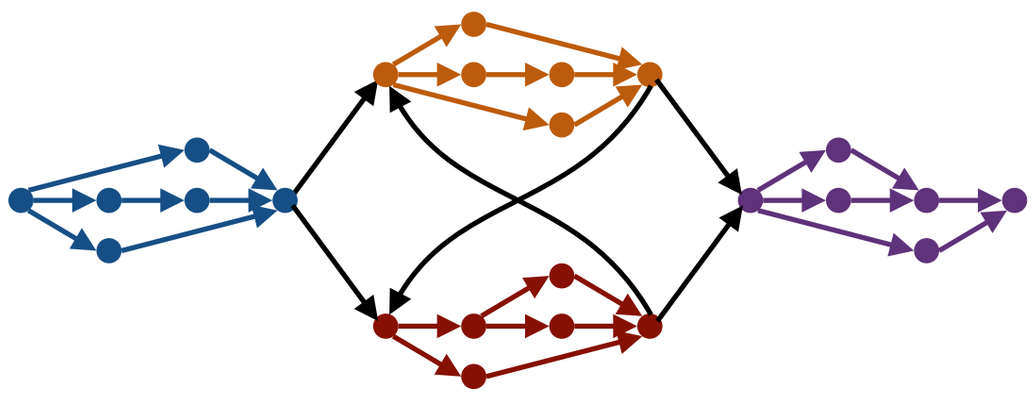
Chris Thachuk: Indexing Hypertext. JDA, 2013.

Rearrangements are difficult to represent with graphs.





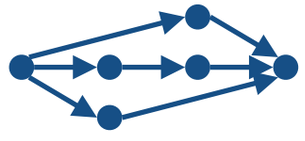
Duplicated subgraphs

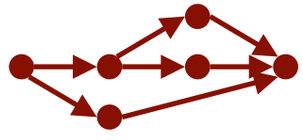


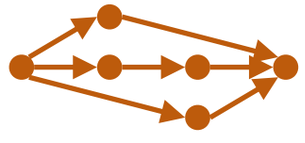
Unsupported paths

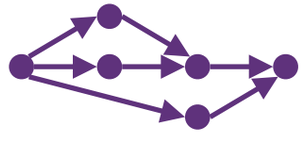
We could represent them with a grammar.

S → ABCD | ACBD

A → 

C → 

B → 

D → 

Conclusions

- We can use pruned de Bruijn graphs encoded using the BWT to index variation graphs.

- GCSA2 is a practical implementation for whole-genome graphs and queries of length up to 128.

- The index is an FM-index: We can extend it with many techniques from text indexing literature.