Efficient construction of a compressed de Bruijn graph for pan-genome analysis

Timo Beller and Enno Ohlebusch

Institute of Theoretical Computer Science
Ulm University

Ischia, June 30, 2015

What the heck is pan-genome analysis?

Next generation sequencers produce vast amounts of DNA sequence information:

- multiple genomes of the same or closely related species are available
- 1000 Genomes Project: goal was to sequence the genomes of at least 1000 humans from all over the world and to produce a catalog of all variations (SNPs, indels, etc.) in the human population
- the genomic sequences together with this catalog is called the "pan-genome" of the population

Related work

There are several approaches that try to capture variations between many individuals/strains in a population graph:

- Schneeberger et al. (Genome Biology 2009), Huang et al. (Bioinformatics 2013), Rahn et al. (Bioinformatics 2014) require a multi-alignment as input
- Marcus et al. (Bioinformatics 2014) use a compressed de Bruijn graph of maximal exact matches (MEMs) as a graphical representation of the relationship between genomes
- the contracted de Bruijn graph introduced by Cazaux et al. (CPM 2014) cannot be used

Definition

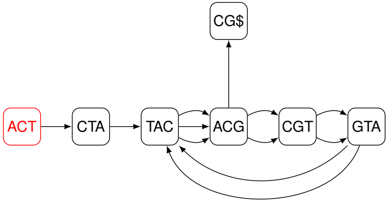
Uncompressed de Bruijn graph

Given $k > 0$ and string S, the de Bruijn graph of S

- contains a node for each distinct length k substring of S, called a k-mer
- two nodes u and v are connected by a directed edge (u, v) if $u = S[i..i + k - 1]$ and $v = S[i + 1..i + k]$

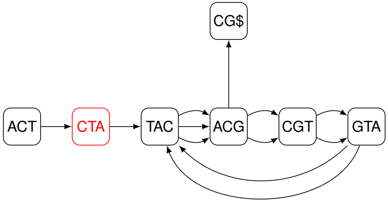
Example

The uncompressed and compressed de Bruijn graphs for $k = 3$ and the string $S =$ ACTACGTACGTACG\$



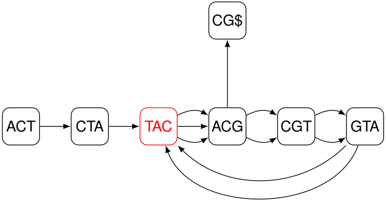
Example

The uncompressed and compressed de Bruijn graphs for $k = 3$ and the string $S =$ ACTACGTACGTACG$



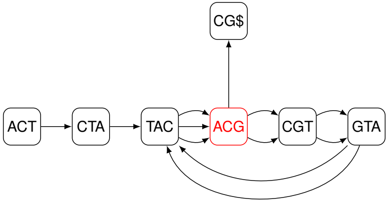
Example

The uncompressed and compressed de Bruijn graphs for $k = 3$ and the string $S =$ ACTACGTACGTACG$



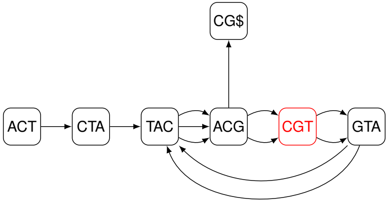
Example

The uncompressed and compressed de Bruijn graphs for $k = 3$ and the string $S =$ ACTACGTACGTACG$



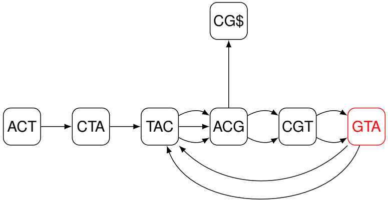
Example

The uncompressed and compressed de Bruijn graphs for $k = 3$ and the string $S =$ ACTACGTACGTACG$



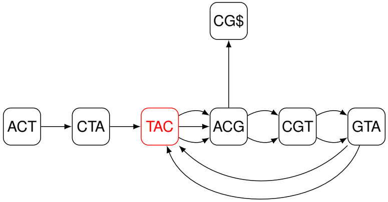
Example

The uncompressed and compressed de Bruijn graphs for $k = 3$ and the string $S =$ ACTACGTACGTACG$



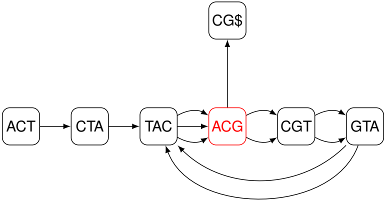
Example

The uncompressed and compressed de Bruijn graphs for $k = 3$ and the string $S =$ ACTACGTACGTACG$



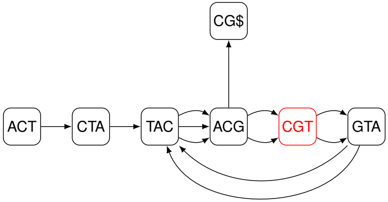
Example

The uncompressed and compressed de Bruijn graphs for $k = 3$ and the string $S =$ ACTACGTACGTACG$



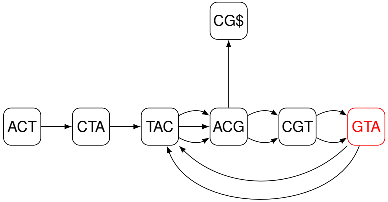
Example

The uncompressed and compressed de Bruijn graphs for $k = 3$ and the string $S =$ ACTACGTACGTACG$



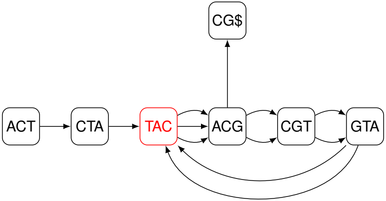
Example

The uncompressed and compressed de Bruijn graphs for $k = 3$ and the string $S =$ ACTACGTACGTACG$



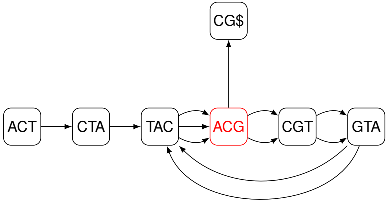
Example

The uncompressed and compressed de Bruijn graphs for $k = 3$ and the string $S =$ ACTACGTACGTACG$



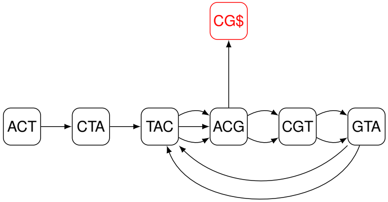
Example

The uncompressed and compressed de Bruijn graphs for $k = 3$ and the string $S =$ ACTACGTACGTACG$



Example

The uncompressed and compressed de Bruijn graphs for $k = 3$ and the string $S =$ ACTACGTACGTACG$



Definition

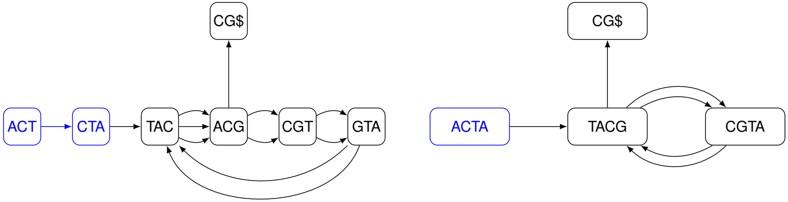
Compressed de Bruijn graph

A de Bruijn graph can be compressed as follows:

- if node u is the only predecessor of node v and v is the only successor of u (but there may be multiple edges (u, v)), then u and v can be merged into a single node that has the predecessors of u and the successors of v.

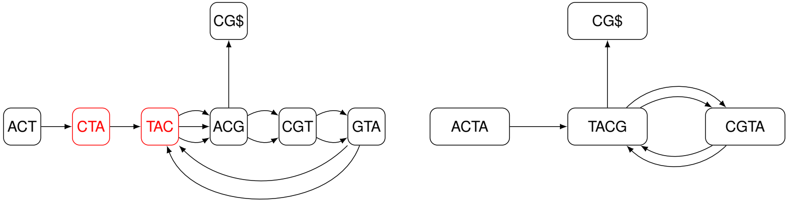
Example

The uncompressed and compressed de Bruijn graphs for $k = 3$ and the string $S =$ACTACGTACGTACG$



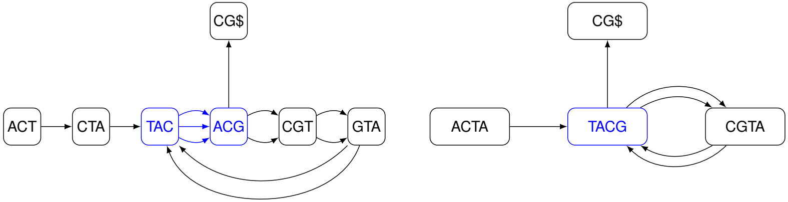
Example

The uncompressed and compressed de Bruijn graphs for $k = 3$ and the string $S =$ACTACGTACGTACG$



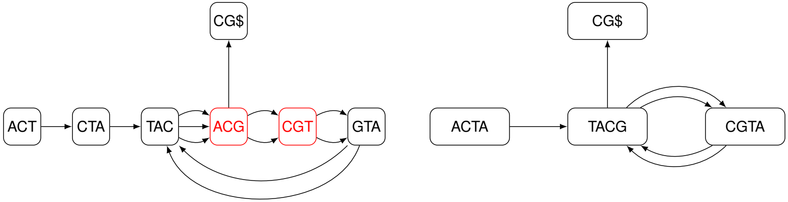
Example

The uncompressed and compressed de Bruijn graphs for $k = 3$ and the string $S =$ACTACGTACGTACG$



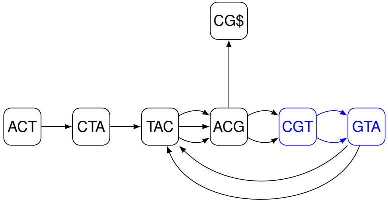
Example

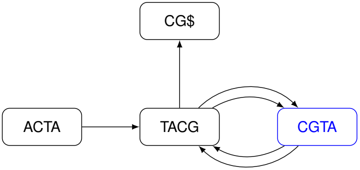
The uncompressed and compressed de Bruijn graphs for $k = 3$ and the string $S =$ACTACGTACGTACG$



Example

The uncompressed and compressed de Bruijn graphs for $k = 3$ and the string $S =$ACTACGTACGTACG$





Xpdf: Schatz.pdf

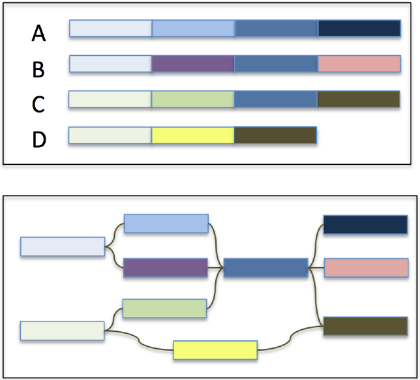


Fig. 1. Overview of a graphical representation of a pan-genome. The four input genomes (A-D) are decomposed into segments shared or specific to the individuals in the population with edges maintaining the adjacencies of the segments.

Page 3 of 8 125% Quit

Lemma

Of course, the compressed de Bruijn graph can be built from its uncompressed counterpart (a much larger graph), but this is disadvantageous because of the huge space consumption.

Lemma

Of course, the compressed de Bruijn graph can be built from its uncompressed counterpart (a much larger graph), but this is disadvantageous because of the huge space consumption.

Characterization of nodes in the de Bruijn graph

Let ω be a node in the de Bruijn graph.

- If ω is not the start node, then it has at least two different predecessors if and only if the length k prefix of ω is a left-maximal repeat.
- It has at least two different successors if and only if the length k suffix of ω is a right-maximal repeat.

Example: left- and right-maximal repeat

$S =$ ACTACGTACGTACG$ contains the repeat ACG

Example: left- and right-maximal repeat

$S =$ ACTACGTACGTACG$ contains the repeat ACG

Left-context: ACTACGTACGTACG$

Example: left- and right-maximal repeat

$S =$ ACTACGTACGTACG$ contains the repeat ACG

Left-context: ACTACGTACGTACG$

Left-context(ACG) = {T} is a singleton set

Example: left- and right-maximal repeat

$S =$ ACTACGTACGTACG$ contains the repeat ACG

Left-context: ACTACGTACGTACG$

Left-context(ACG) = $\{$T$\}$ is a singleton set

$\Rightarrow$ ACG is not left-maximal

Example: left- and right-maximal repeat

$S =$ `ACTACGTACGTACG$` contains the repeat `ACG`

Left-context: `ACTACGTACGTACG$`

Left-context(`ACG`) = {`T`} is a singleton set

⇒ `ACG` is not left-maximal

⇒ node `ACG` has just one predecessor

Example: left- and right-maximal repeat

$S =$ ACTACGTACGTACG\$ contains the repeat ACG

Left-context: ACTACGTACGTACG\$

Left-context(ACG) = {T} is a singleton set

⇒ ACG is not left-maximal

⇒ node ACG has just one predecessor

Right-context: ACTACGTACGTACG\$

Example: left- and right-maximal repeat

$S =$ ACTACGTACGTACG$ contains the repeat ACG

Left-context: ACTACGTACGTACG$

Left-context(ACG) = {T} is a singleton set

⇒ ACG is not left-maximal

⇒ node ACG has just one predecessor

Right-context: ACTACGTACGTACG$

Right-context(ACG) = {T, $} is not a singleton set

Example: left- and right-maximal repeat

$S =$ ACTACGTACGTACG$ contains the repeat ACG

Left-context: ACTACGTACGTACG$

Left-context(ACG) = {T} is a singleton set

⇒ ACG is not left-maximal

⇒ node ACG has just one predecessor

Right-context: ACTACGTACGTACG$

Right-context(ACG) = {T, $} is not a singleton set

⇒ ACG is right-maximal

Example: left- and right-maximal repeat

$S =$ ACTACGTACGTACG$ contains the repeat ACG

Left-context: ACTACGTACGTACG$

Left-context(ACG) = {T} is a singleton set

⇒ ACG is not left-maximal

⇒ node ACG has just one predecessor

Right-context: ACTACGTACGTACG$

Right-context(ACG) = {T, $} is not a singleton set

⇒ ACG is right-maximal

⇒ node ACG has (at least) two different successors

Idea of the algorithm

Compute all right-maximal k-mers, using an index of the text.

Idea of the algorithm

Compute all right-maximal k-mers, using an index of the text.

Extend each right-maximal k-mer as long as possible, character by character, to the left.

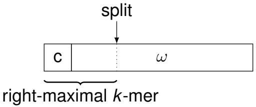
Idea of the algorithm

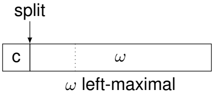
Compute all right-maximal k-mers, using an index of the text.

Extend each right-maximal k-mer as long as possible, character by character, to the left.

The left-extension of a string ω must stop if

- the length k prefix of $c\omega$ is a right-maximal repeat for some $c \in \Sigma$.
- (the length k prefix of) ω is a left-maximal repeat





Algorithm

Add the intervals of all right-maximal k-mers to a queue Q.
Add the interval [1..1] to Q (the stop node ends with \$).

while the queue Q is not empty **do**

- remove an interval from Q
- extend the corresponding string ω to the left: this yields $c\omega$
- **if** the length k prefix of $c\omega$ is a right-maximal repeat **then** add an edge to the graph
- **else if** ω is a left-maximal repeat **then** add an edge to the graph and add the interval of the length k prefix of $c\omega$ to Q
- **else** proceed with the $c\omega$-interval

Preproccessing

i	LCP	B	BWT	$S[SA[i] \ldots |S|]$
1	−1	0	C	$
2	0	0	$	ACTAGTTTTCTAGTCC$
3	1	0	T	AGTCC$
4	3	0	T	AGTTTTCTAGTCC$
5	0	0	C	C$
6	1	0	T	CC$
7	1	0	T	CTAGTCC$
8	5	0	A	CTAGTTTTCTAGTCC$
9	0	0	A	GTCC$
10	2	0	A	GTTTTCTAGTCC$
11	0	0	C	TAGTCC$
12	4	0	C	TAGTTTTCTAGTCC$
13	1	0	G	TCC$
14	2	0	T	TCTAGTCC$
15	1	0	T	TTCTAGTCC$
16	2	0	T	TTTCTAGTCC$
17	3	0	T	TTTTCTAGTCC$
18	4	0	G	TTTTTCTAGTCC$

Preproccessing

i	LCP	B	BWT	$S[SA[i]\ldots\lvert S\rvert]$
1	−1	0	C	$
2	0	0	$	ACTAGTTTTTCTAGTCC$
3	1	0	T	AGTCC$
4	3	0	T	AGTTTTTCTAGTCC$
5	0	0	C	C$
6	1	0	T	CC$
7	1	0	T	CTAGTCC$
8	5	0	A	CTAGTTTTTCTAGTCC$
9	0	0	A	GTCC$
10	2	0	A	GTTTTTCTAGTCC$
11	0	0	C	TAGTCC$
12	4	0	C	TAGTTTTTCTAGTCC$
13	1	0	G	TCC$
14	2	0	T	TCTAGTCC$
15	1	0	T	TTCTAGTCC$
16	2	0	T	TTTCTAGTCC$
17	3	0	T	TTTTCTAGTCC$
18	4	0	G	TTTTTCTAGTCC$

Preproccessing

i	LCP	B	BWT	$S[SA[i]\ldots\|S\|]$
1	−1	0	C	$
2	0	0	$	ACTAGTTTTTCTAGTCC$
3	1	0	T	AGTCC$
4	3	0	T	AGTTTTTCTAGTCC$
5	0	0	C	C$
6	1	0	T	CC$
7	1	0	T	CTAGTCC$
8	5	0	A	CTAGTTTTTCTAGTCC$
9	0	0	A	GTCC$
10	2	0	A	GTTTTTCTAGTCC$
11	0	0	C	TAGTCC$
12	4	0	C	TAGTTTTTCTAGTCC$
13	1	0	G	TCC$
14	2	0	T	TCTAGTCC$
15	1	0	T	TTCTAGTCC$
16	2	0	T	TTTCTAGTCC$
17	3	0	T	TTTTCTAGTCC$
18	4	0	G	TTTTTCTAGTCC$

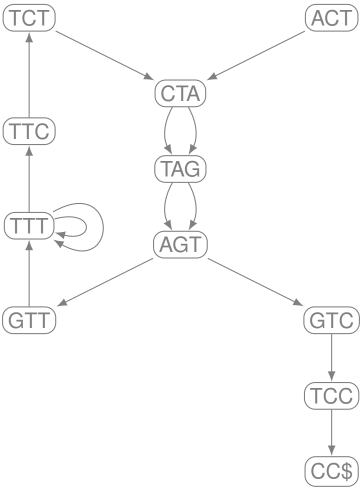
Preproccessing

i	LCP	B	BWT	$S[SA[i] \ldots \|S\|]$
1	−1	0	C	$
2	0	0	$	ACTAGTTTTTCTAGTCC$
3	1	1	T	AGTCC$
4	3	1	T	AGTTTTTCTAGTCC$
5	0	0	C	C$
6	1	0	T	CC$
7	1	0	T	CTAGTCC$
8	5	0	A	CTAGTTTTTCTAGTCC$
9	0	0	A	GTCC$
10	2	0	A	GTTTTTCTAGTCC$
11	0	0	C	TAGTCC$
12	4	0	C	TAGTTTTTCTAGTCC$
13	1	0	G	TCC$
14	2	0	T	TCTAGTCC$
15	1	0	T	TTCTAGTCC$
16	2	1	T	TTTCTAGTCC$
17	3	0	T	TTTTCTAGTCC$
18	4	1	G	TTTTTCTAGTCC$

Compressing

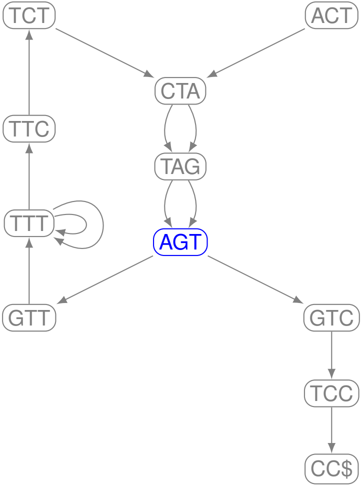
i	B	BWT	$S[SA[i] \dots |S|]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



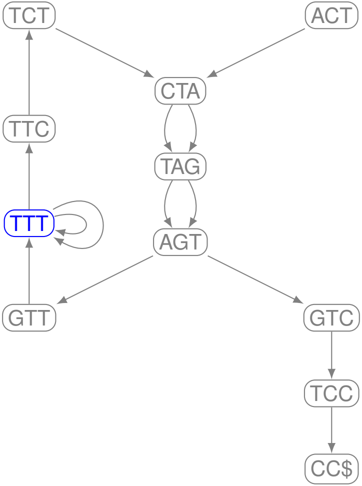
i	B	BWT	$S[SA[i] \ldots \lvert S \rvert]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



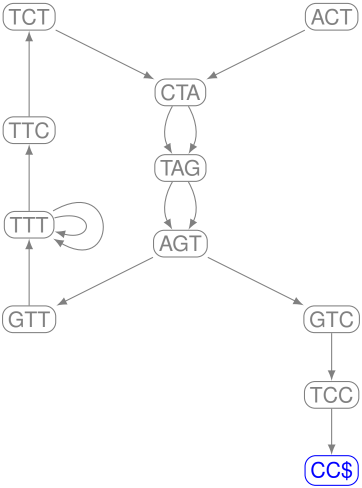
i	B	BWT	$S[SA[i] \ldots \vert S \vert]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



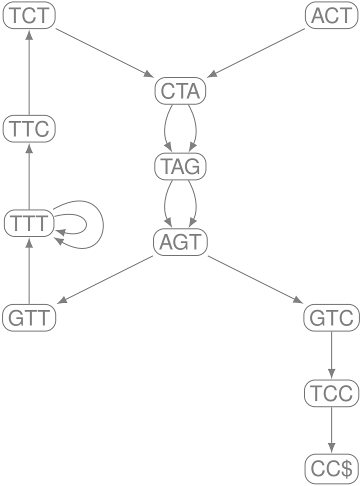
i	*B*	BWT	*S*[SA[*i*] . . . |*S*|]
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



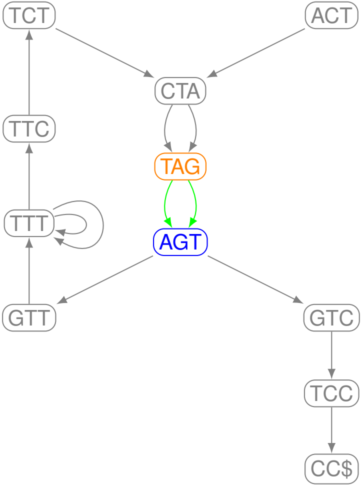
i	*B*	BWT	*S*[SA[*i*] . . . |*S*|]
1	0	C	$
2	0	$	ACTAGTTTTT. . .
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA. . .
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC. . .
9	0	A	GTCC$
10	0	A	GTTTTTCTAG. . .
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT. . .
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



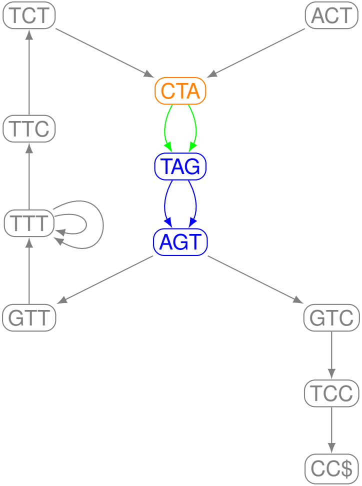
i	B	BWT	$S[SA[i] \dots \|S\|]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



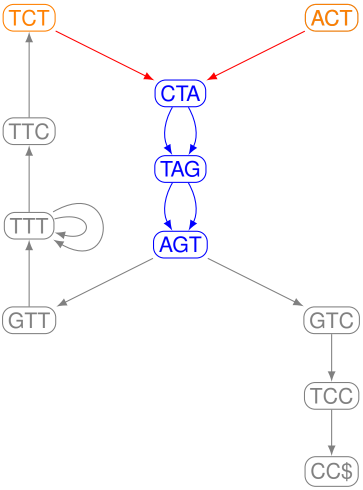
i	B	BWT	$S[SA[i] \ldots \lvert S \rvert]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



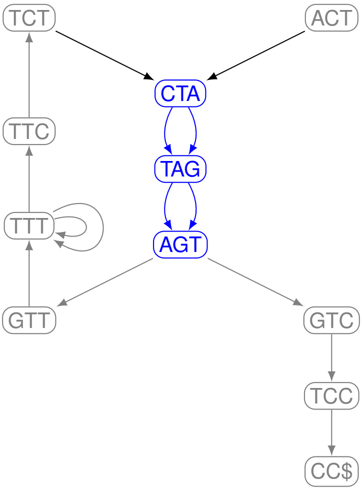
i	B	BWT	$S[SA[i] \ldots \|S\|]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



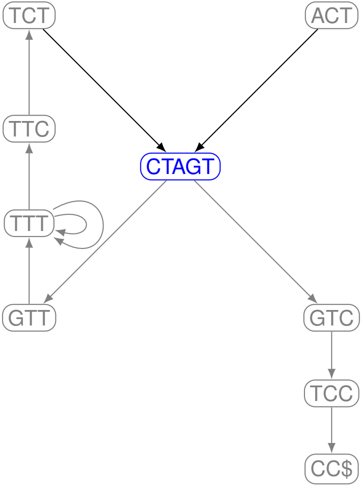
i	B	BWT	$S[SA[i]\ldots\|S\|]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



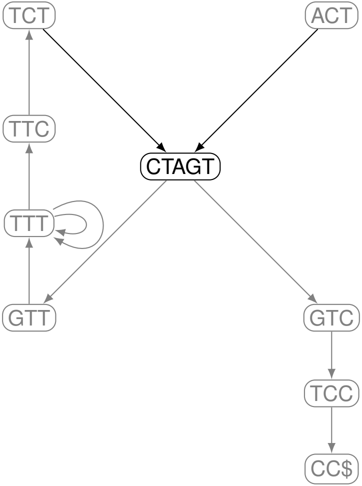
i	B	BWT	$S[SA[i] \ldots \|S\|]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



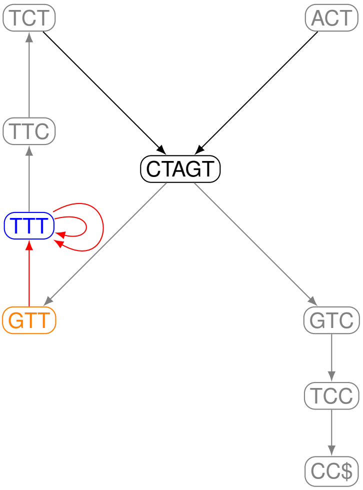
i	B	BWT	$S[\mathrm{SA}[i]\ldots\|S\|]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



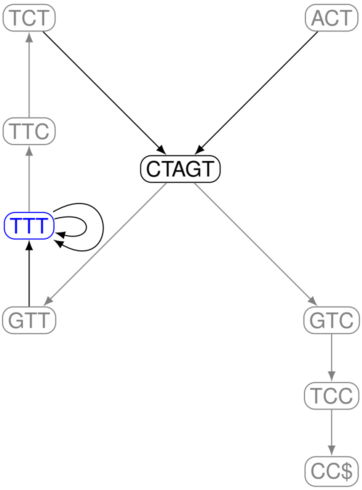
i	*B*	BWT	*S*[SA[*i*] . . . |*S*|]
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



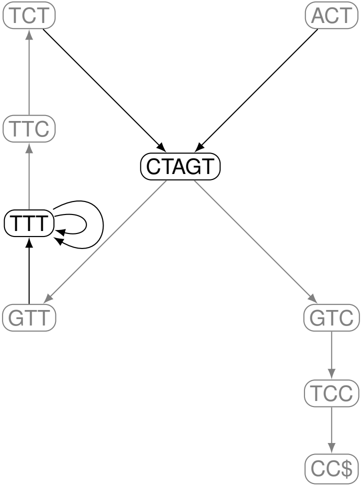
i	B	BWT	$S[SA[i] \ldots \|S\|]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



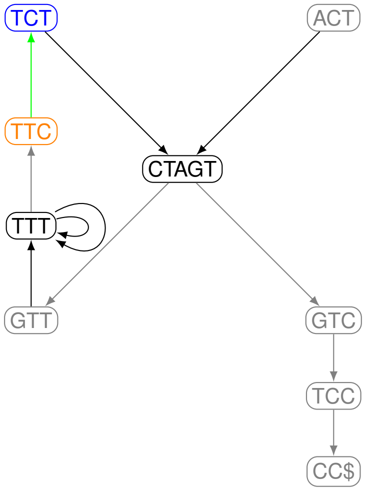
i	B	BWT	$S[SA[i] \ldots \lvert S \rvert]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



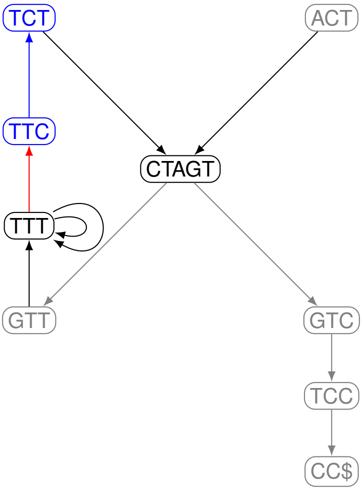
i	B	BWT	$S[SA[i] \ldots \|S\|]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



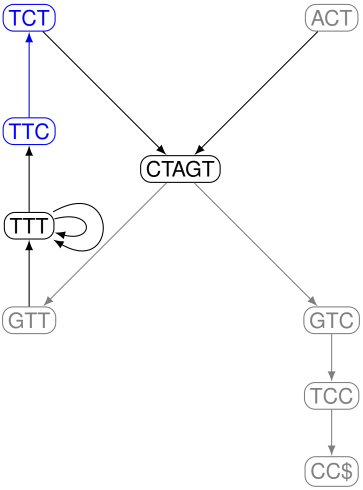
i	B	BWT	$S[SA[i]\ldots|S|]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



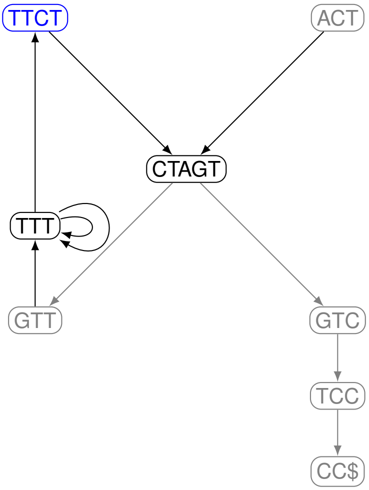
i	B	BWT	$S[SA[i] \ldots \vert S\vert]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



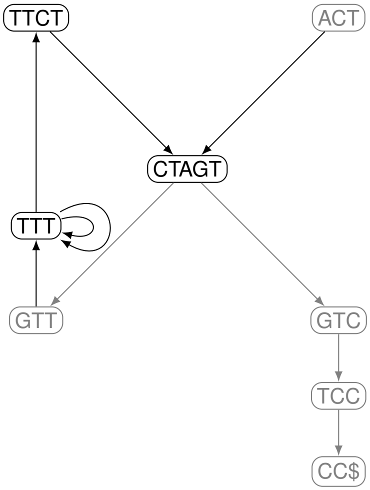
i	B	BWT	$S[SA[i] \dots \lvert S \rvert]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



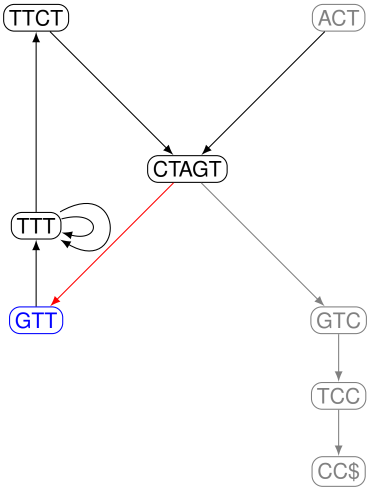
i	B	BWT	$S[SA[i] \ldots \|S\|]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



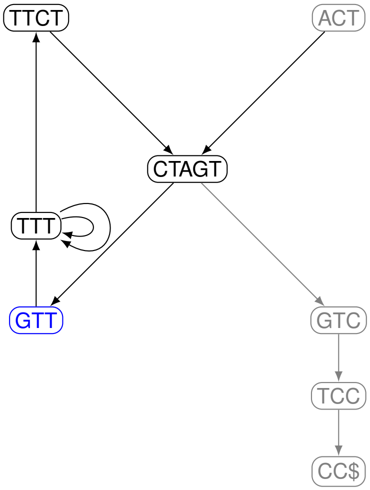
i	B	BWT	$S[SA[i] \ldots \|S\|]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



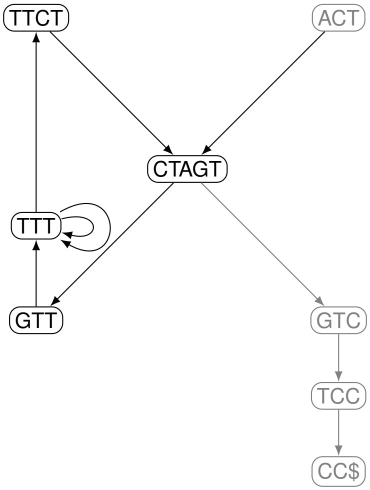
i	B	BWT	$S[SA[i] \ldots \lvert S \rvert]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



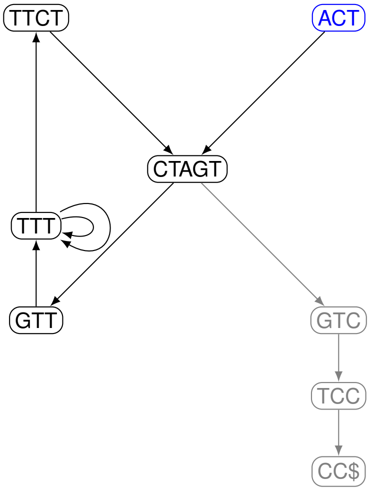
i	B	BWT	$S[SA[i] \ldots \|S\|]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



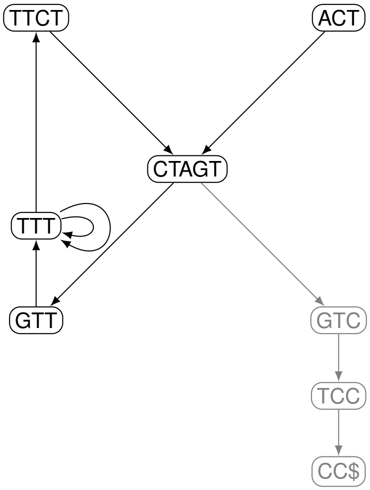
i	B	BWT	$S[SA[i] \ldots \|S\|]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



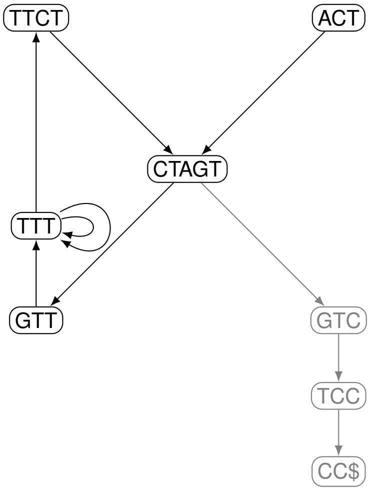
i	B	BWT	$S[SA[i] \ldots \|S\|]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



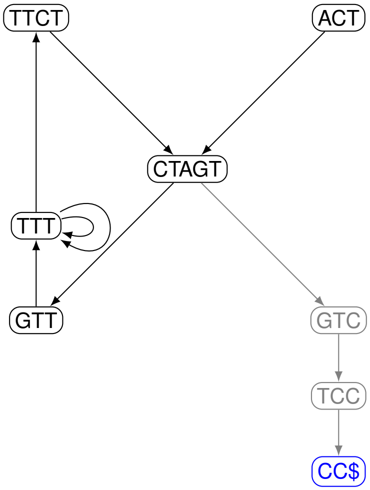
i	B	BWT	$S[SA[i] \dots \lvert S \rvert]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



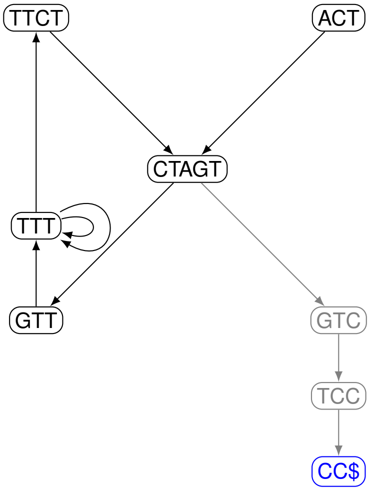
i	B	BWT	$S[SA[i] \ldots \|S\|]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



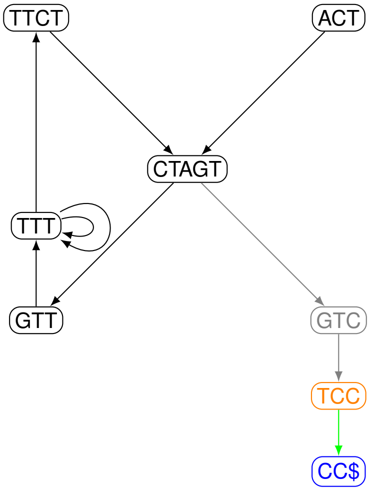
i	B	BWT	$S[SA[i] \ldots \lvert S \rvert]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



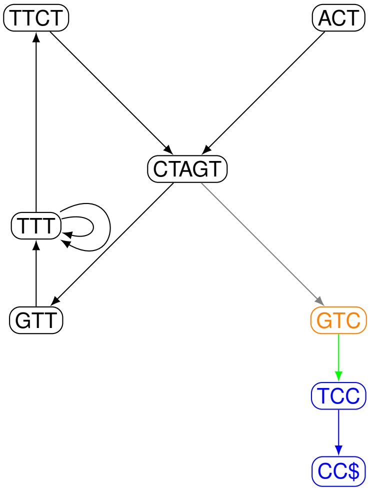
i	B	BWT	$S[SA[i] \ldots \lvert S \rvert]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



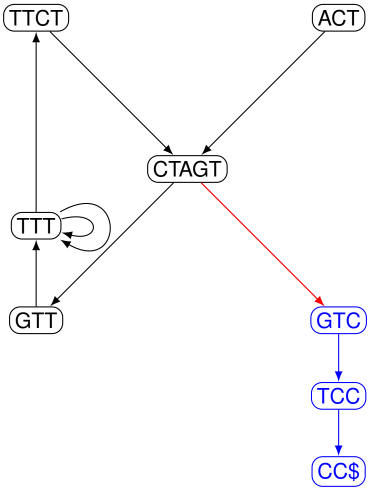
i	B	BWT	$S[SA[i] \ldots \|S\|]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



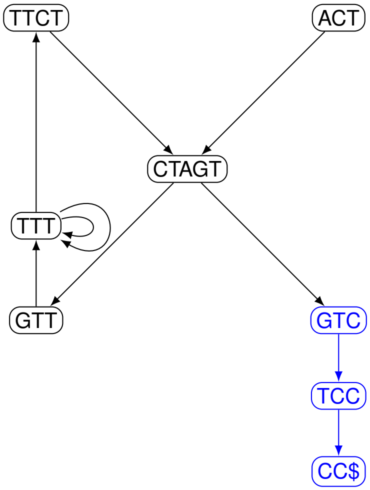
i	B	BWT	$S[SA[i]\ldots\vert S\vert]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



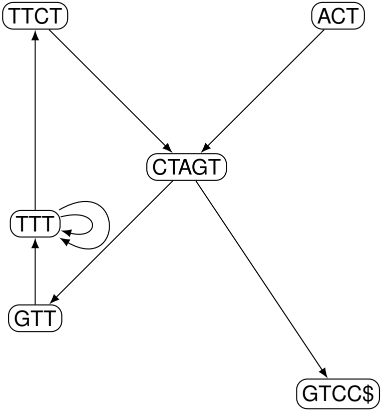
i	B	BWT	$S[SA[i]\ldots\lvert S\rvert]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



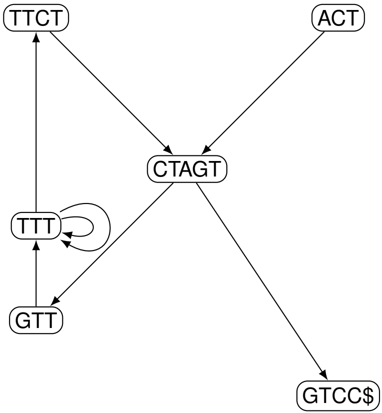
i	B	BWT	$S[SA[i] \ldots \|S\|]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



i	B	BWT	$S[SA[i] \dots \lvert S \rvert]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

Compressing



i	B	BWT	$S[SA[i] \ldots \|S\|]$
1	0	C	$
2	0	$	ACTAGTTTTT...
3	1	T	AGTCC$
4	1	T	AGTTTTTCTA...
5	0	C	C$
6	0	T	CC$
7	0	T	CTAGTCC$
8	0	A	CTAGTTTTTC...
9	0	A	GTCC$
10	0	A	GTTTTTCTAG...
11	0	C	TAGTCC$
12	0	C	TAGTTTTTCT...
13	0	G	TCC$
14	0	T	TCTAGTCC$
15	0	T	TTCTAGTCC$
16	1	T	TTTCTAGTCC$
17	0	T	TTTTCTAGTCC$
18	1	G	TTTTTCTAGTCC$

A different representation of the graph

The algorithm computes a compressed de Bruijn graph, in which a node is represented by the quadruple (id, lb, rb, len)

- id is the identifier of the node
- [$lb..rb$] is the suffix array interval
- len is the length of the corresponding string ω

A different representation of the graph

The algorithm computes a compressed de Bruijn graph, in which a node is represented by the quadruple (*id*, *lb*, *rb*, *len*)

- *id* is the identifier of the node
- [*lb*..*rb*] is the suffix array interval
- *len* is the length of the corresponding string ω

Marcus et al. store the positions SA[*lb*], $\ldots$, SA[*rb*] at which ω occurs in *S*—together with the corresponding outgoing edges—in ascending order.

A different representation of the graph

The algorithm computes a compressed de Bruijn graph, in which a node is represented by the quadruple (*id*, *lb*, *rb*, *len*)

- *id* is the identifier of the node
- [*lb*..*rb*] is the suffix array interval
- *len* is the length of the corresponding string ω

Marcus et al. store the positions SA[*lb*], $\ldots$, SA[*rb*] at which ω occurs in *S*—together with the corresponding outgoing edges—in ascending order.

To model this, a node now contains the sorted list of positions *posList* and the corresponding adjacency list *adjList*.

Implementation

Now the walk through the graph G that gives S is induced by the adjacency lists: if node v is visited for the i-th time, then its successor is the node that can be found at position i in the adjacency list of v.

Implementation

Now the walk through the graph G that gives S is induced by the adjacency lists: if node v is visited for the i-th time, then its successor is the node that can be found at position i in the adjacency list of v.

The following three algorithms are implemented:

- A1 Uses a comparison based sorting algorithm.
- A2 Uses a non-comparison based sorting algorithm.
- A3 Uses a backward search to track the suffixes of S in G.

Experimental results

Data

- 40-62 different strains of the bacterium *E.coli*
- 7 different human genomes
- *chr1* denotes their first chromosome

The experiments were conducted on a 64 bit Ubuntu 14.04.1 LTS (Kernel 3.13) system equipped with two ten-core Intel Xeon processors E5-2680v2 with 2.8 GHz and 128GB of RAM (but no parallelism was used).

Experimental results

file (Mbp)	k (size)	A1	A2	A3	splitMEM
40 *E.coli*	-	**38** (5.00)	**38** (5.00)	127 (**1.32**)	94 (315)
(199)	25 (1.50)	**55** (6.06)	57 (9.18)	190 (**2.87**)	2,170 (572)
	100 (0.65)	**58** (5.00)	65 (7.89)	207 (**1.63**)	1,684 (572)
	1000 (0.06)	**76** (5.00)	81 (7.08)	190 (**1.49**)	1,671 (572)
62 *E.coli*	-	**64** (5.00)	**64** (5.00)	201 (**1.24**)	134 (316)
(310)	25 (1.57)	**86** (6.06)	93 (9.38)	295 (**2.83**)	-
	100 (0.68)	**92** (5.00)	105 (8.19)	331 (**1.68**)	-
	1000 (0.06)	134 (5.00)	**123** (7.33)	305 (**1.53**)	-
7 *chr1*	-	**399** (5.00)	**399** (5.00)	1,163 (**1.24**)	-
(1,736)	25 (3.10)	**601** (7.70)	646 (11.44)	1,910 (**4.45**)	-
	100 (1.59)	**549** (5.88)	598 (9.70)	1,628 (**2.76**)	-
	1000 (1.50)	**606** (5.86)	621 (9.57)	1,655 (**2.66**)	-
7 *genomes*	-	-	-	**22,038** (**1.24**)	-
(21,201)	25 (3.34)	-	-	**33,247** (**4.84**)	-
	100 (1.16)	-	-	**29,641** (**2.22**)	-
	1000 (1.01)	-	-	**29,962** (**2.04**)	-

Thank you!
Any Questions?