UNIVERSIDAD DE CHILE

Succinct *de* Bruijn graphs

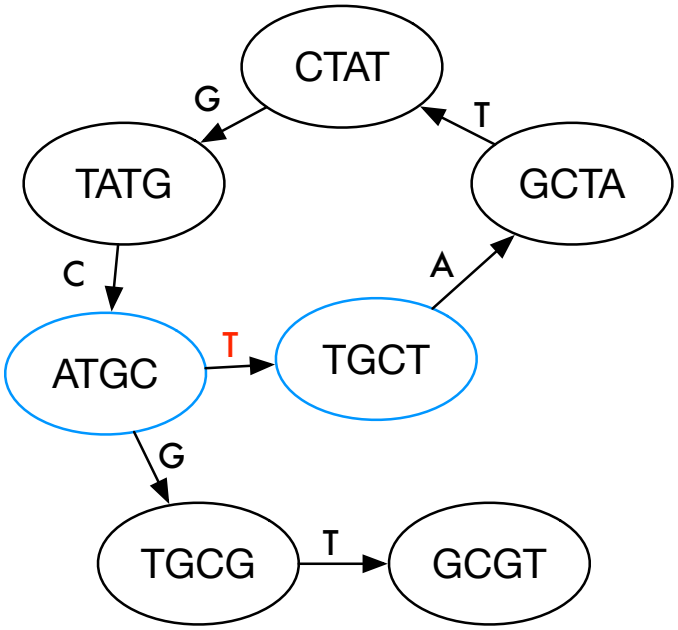
Diego Díaz Domínguez

Department of Computer Science

University of Chile

de Bruijn graphs

Given a parameter k and a set $\mathcal{S}$ of strings, a *de* Bruijn graph (dBG) is a directed graph that encodes the k-1 overlaps between all the k-substrings *(*k-mers) of $\mathcal{S}$.



$\mathcal{S}$ ={ATGCTATGCGT}

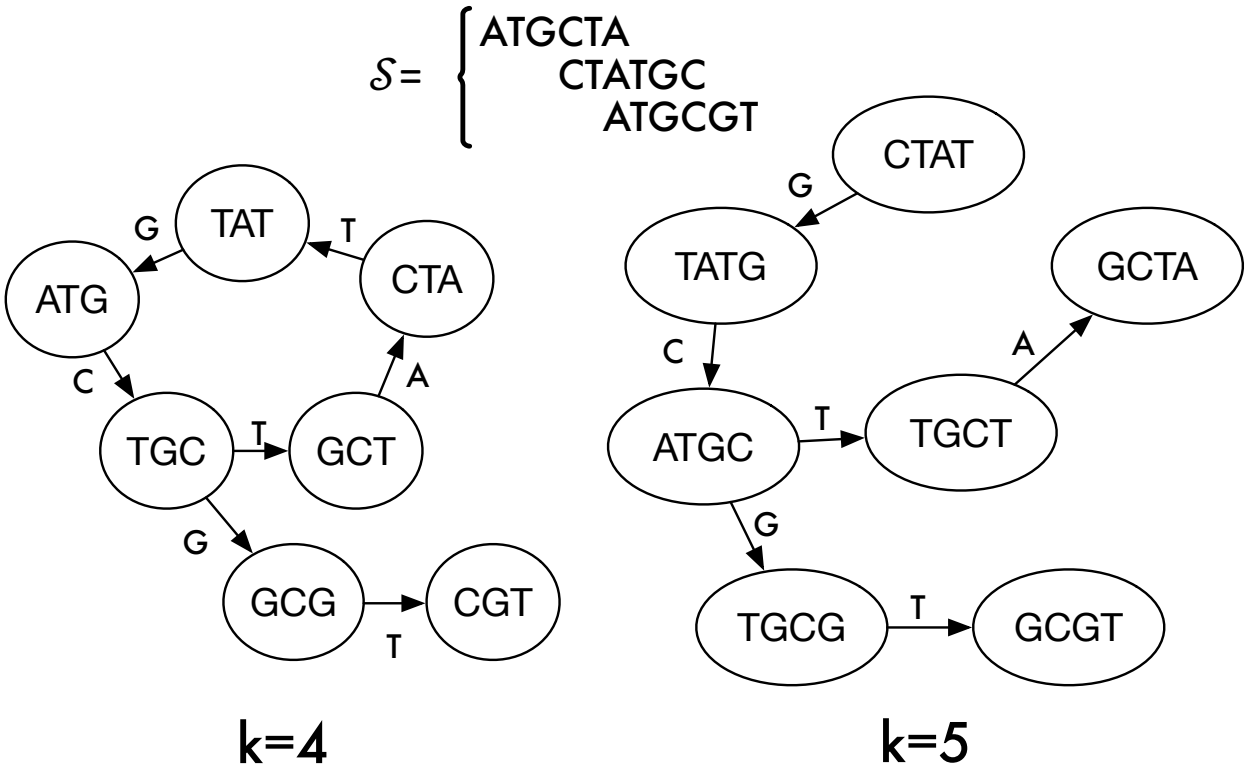
ATGC

TGCT

k-mer = ATGCT

de Bruijn graphs

Different values for k yield different graph topologies



de Bruijn graphs

Advantages:

- They are easy to construct
- They catch the underlying structure of the string set
- They can be used to solve several problems in Bioinformatics

Disadvantages:

- Choosing the right value for k is always a major concern.
- The size of the graph grows with k.
- Having several dBG, with different orders, for the same dataset is expensive.

Encoding a dBG

Things we have to do:

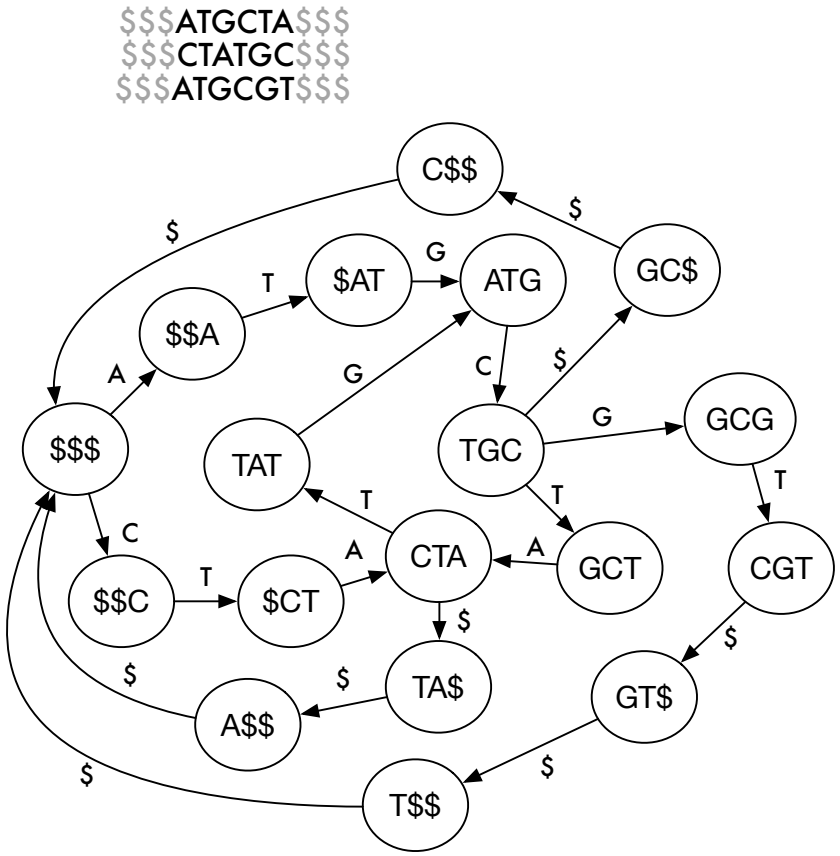
- Represent the nodes and the edges succinctly
- Traverse the graph efficiently
- Combine the information of several graph orders
- Colour the nodes

Operations in a dBG

Navigational operations we are interested in:

- `outdegree(v)`:number of outgoing edges of node `v`
- `indegree(v)`:number of incoming edges of `v`
- `outgoing(v,a)`:follow outgoing edge of `v` labeled with `a`
- `incoming(v,a)`:follow an incoming node of `v` whose label starts with `a`

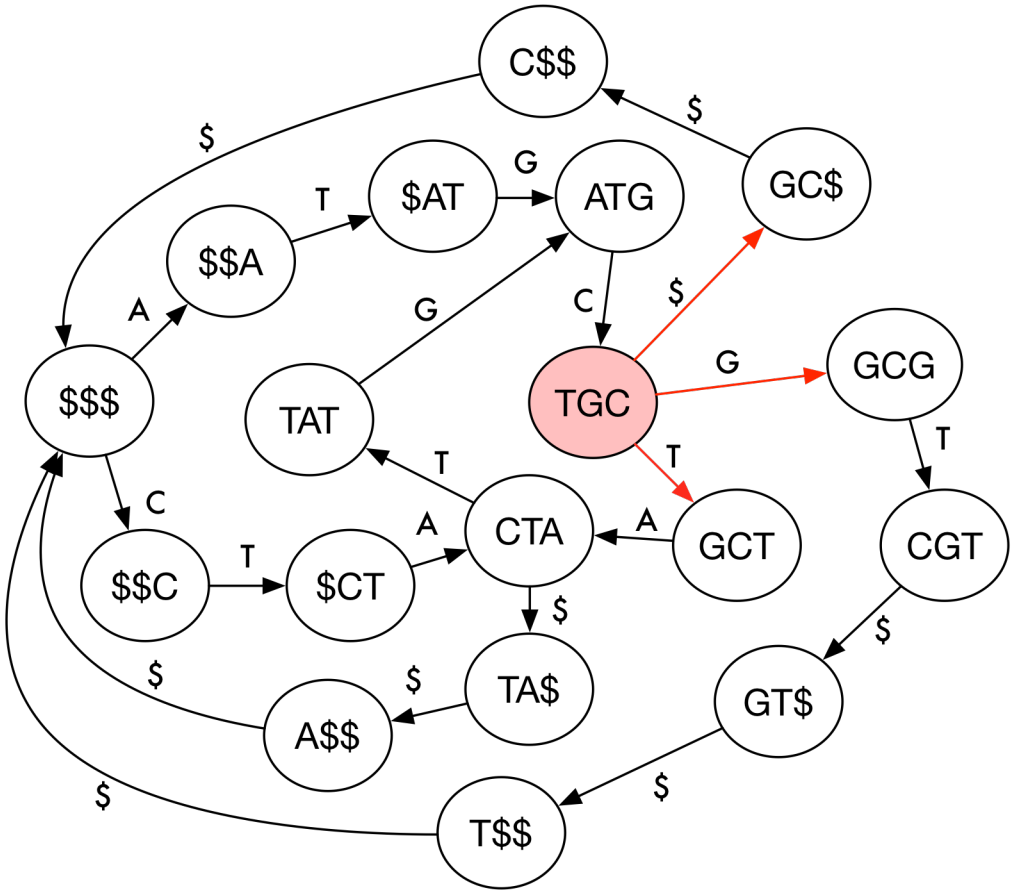
Succinct dBG: BOSS



BOSS Matrix	Edge BWT	Node Marks
$$$	A_1	0
	C_1	1
A$$	$	1
C$$	$	1
T$$	$	1
TA$	$	1
GC$	$	1
GT$	$	1
$$A	T_1	1
CTA	$	0
	T_2	1
$$C	T_3	1
TGC	$	0
	G_1	0
	T_4	1
GCG	T_5	1
ATG	C_2	1
$AT	G_2	1
TAT	$*G_2$	1
$CT	A_2	1
GCT	$*A_2$	1
CGT	$	1

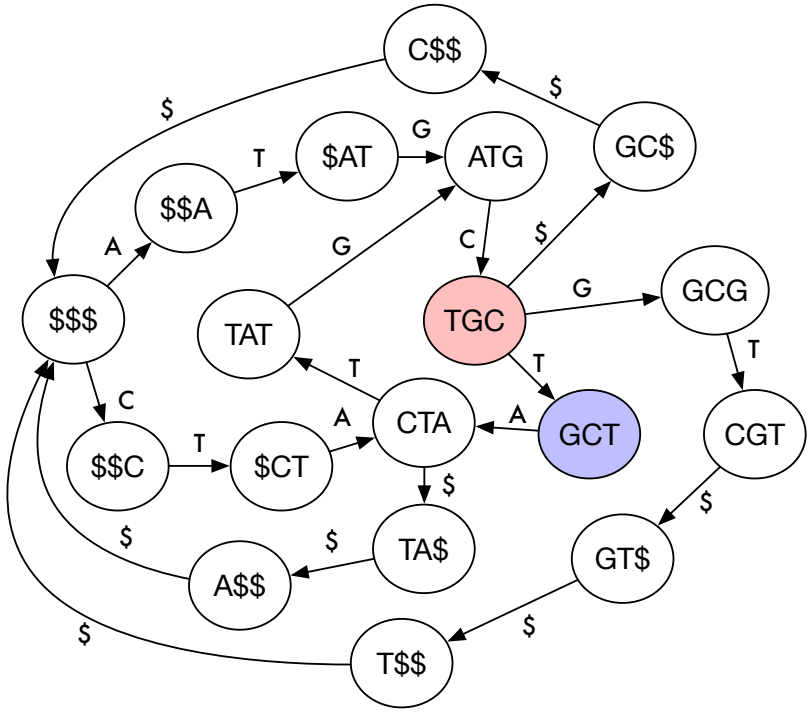
C[$] = 0
C[A] = 7
C[C] = 9
C[G] = 11
C[T] = 13

BOSS:outdegree



BOSS Matrix	Edge BWT	Node Marks	
$$$	A1	0	
	C1	1	
A$$	$	1	
C$$	$	1	
T$$	$	1	
TA$	$	1	
GC$	$	1	
GT$	$	1	
$$A	T1	1	
CTA	$	0	
	T2	1	
$$C	T3	1	
11 TGC	$	0	select(NodeMarks,10)+1
	G1	0	
	T4	1	select(NodeMarks,11)
GCG	T5	1	
ATG	C2	1	
$AT	G2	1	
TAT	*G2	1	
$CT	A1	1	
GCT	*A1	1	
CGT	$	1	

BOSS: forward

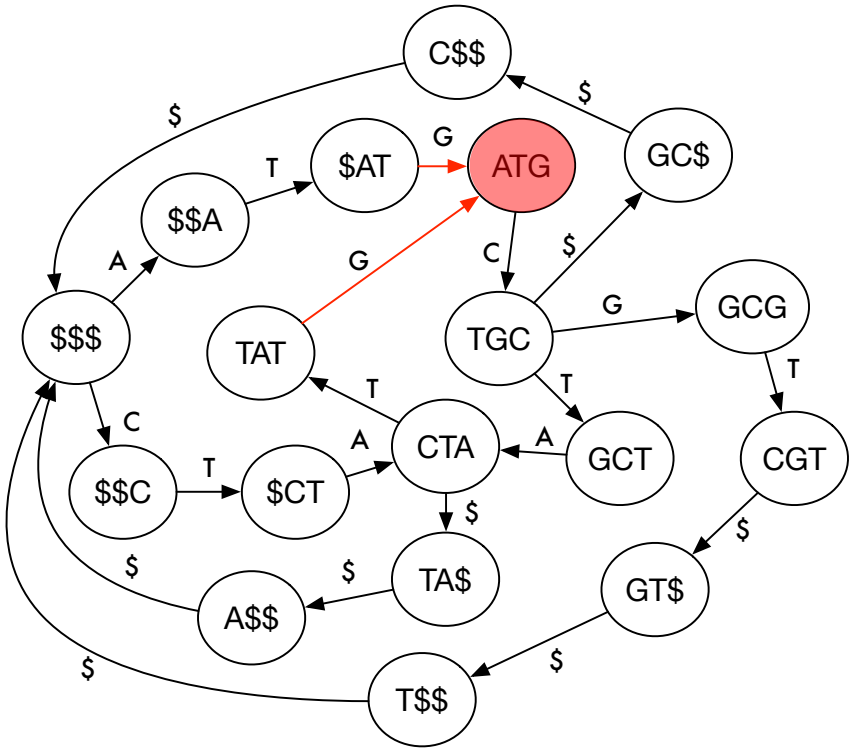


BOSS Matrix	Edge BWT	Node Marks	
$$$	A_1	0	
	C_1	1	
A$$	$	1	
C$$	$	1	
T$$	$	1	
TA$	$	1	
GC$	$	1	
GT$	$	1	
$$A	T_1	1	
CTA	$	0	
	T_2	1	
$$C	T_3	1	
11 TGC	$	0	
	G_1	0	
	T_4	1	15
GCG	T_5	1	
ATG	C_2	1	
$AT	G_2	1	
TAT	$*G_2$	1	
$CT	A_2	1	
17 GCT	$*A_2$	1	
CGT	$	1	

C[$]=0
C[A]=7
C[C]=9
C[G]=11
C[T]=13

`forward(11,T)` = 17
= C[C] + `rank(EdgeBWT,T,15)`
= 13 + 4 = 17

BOSS: indegree



BOSS Matrix	Edge BWT	Node Marks
$$$	A_1	0
	C_1	1
A$$	$	1
C$$	$	1
T$$	$	1
TA$	$	1
GC$	$	1
GT$	$	1
$$A	T_1	1
CTA	$	0
	T_2	1
$$C	T_3	1
TGC	$	0
	G_1	0
	T_4	1
GCG	T_5	1
13 ATG	C_2	1
$AT	a G_2	1
TAT	$*G_2$	1
$CT	A_2	1
GCT	$*A_2$	1
CGT	$	1
	b	

C[$]=0
C[A]=7
C[C]=9
C[G]=11
C[T]=13

`indegree(13) = 2`

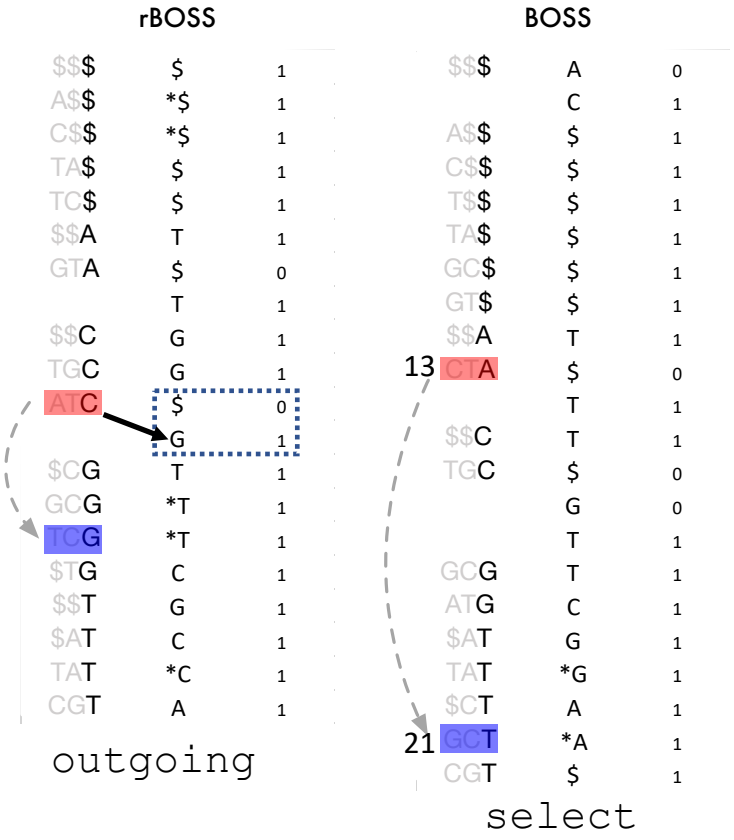
`= 1 + rank(EdgeBWT,*G,b) −`
`rank(EdgeBWT,*G,a) = 2`

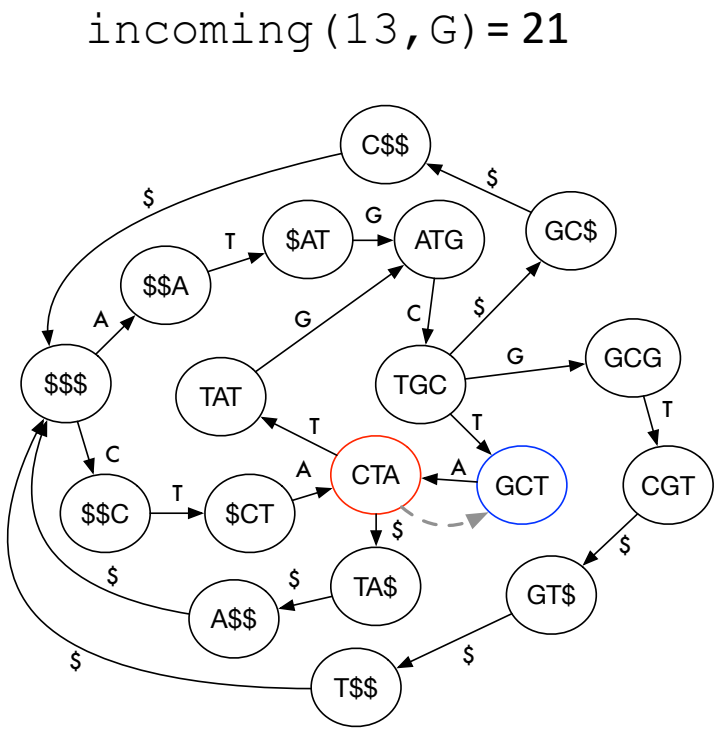
`select(EdgeBWT,G,2)=a`

`select(EdgeBWT,G,3)=b`

Bidirectional BOSS

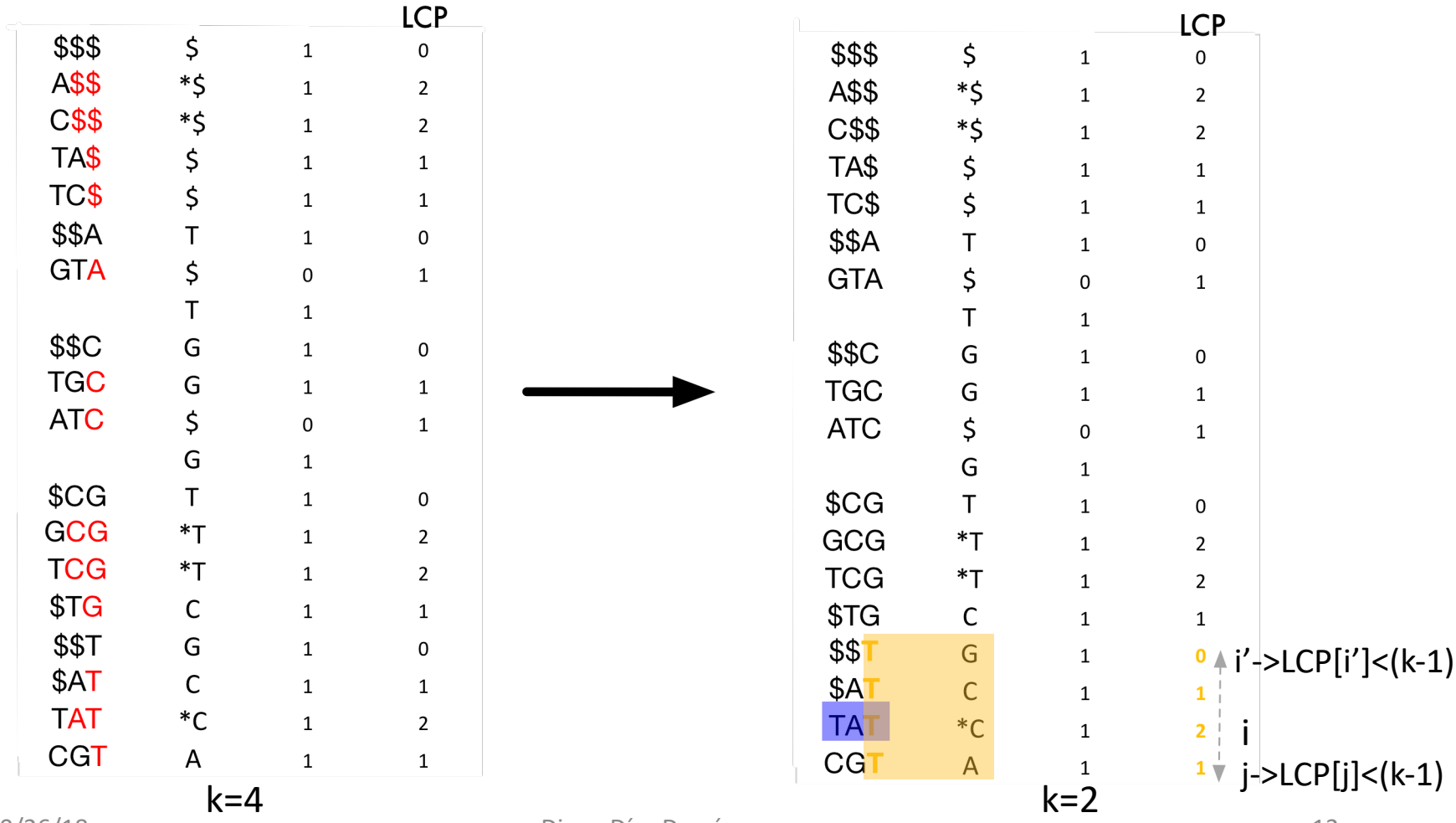
`incoming` is more expensive than `outgoing` because we don't have access to the first symbol of the nodes. By doubling the size of the index, we can compute *incoming* fast.



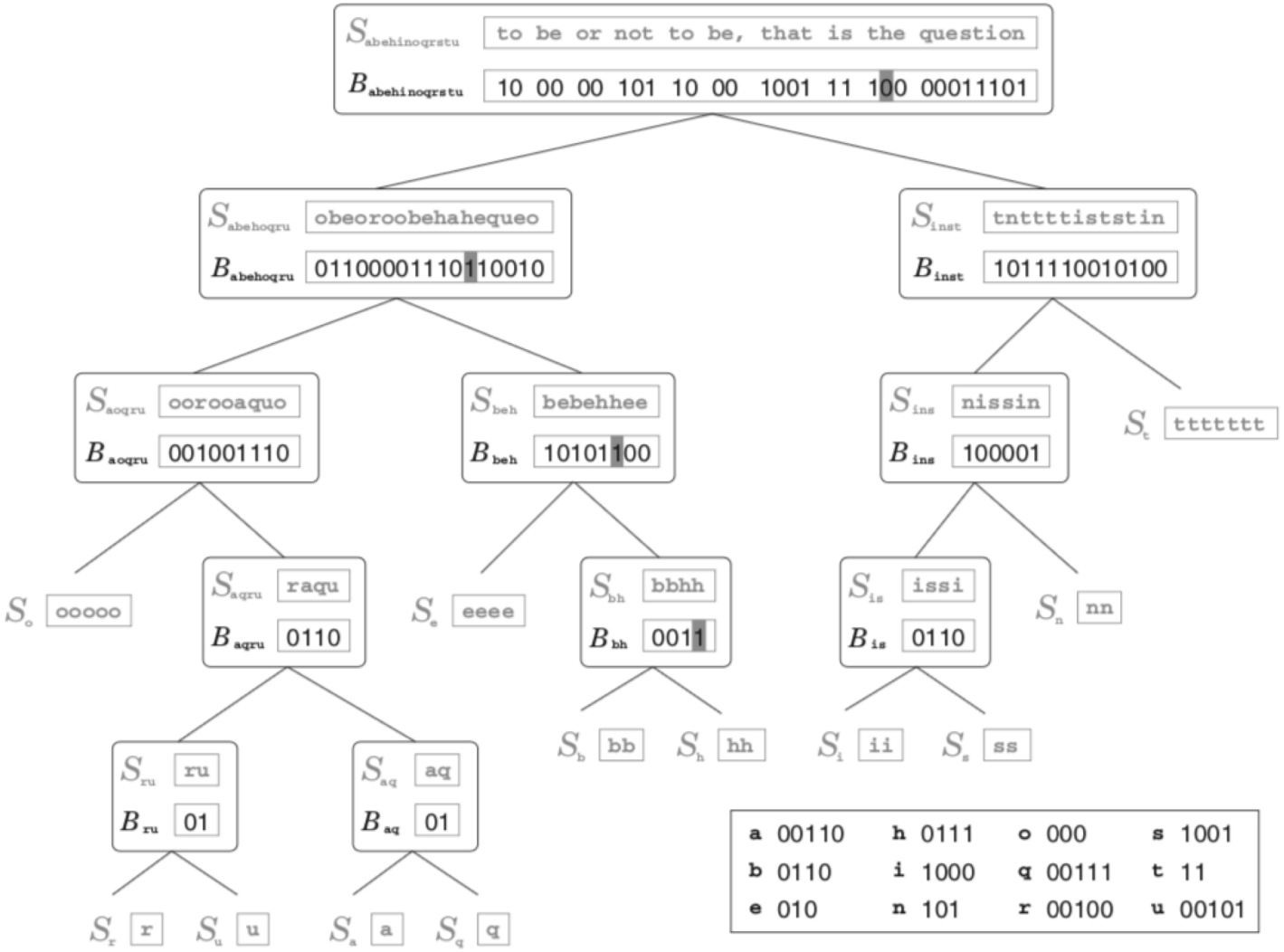


Variable-order BOSS

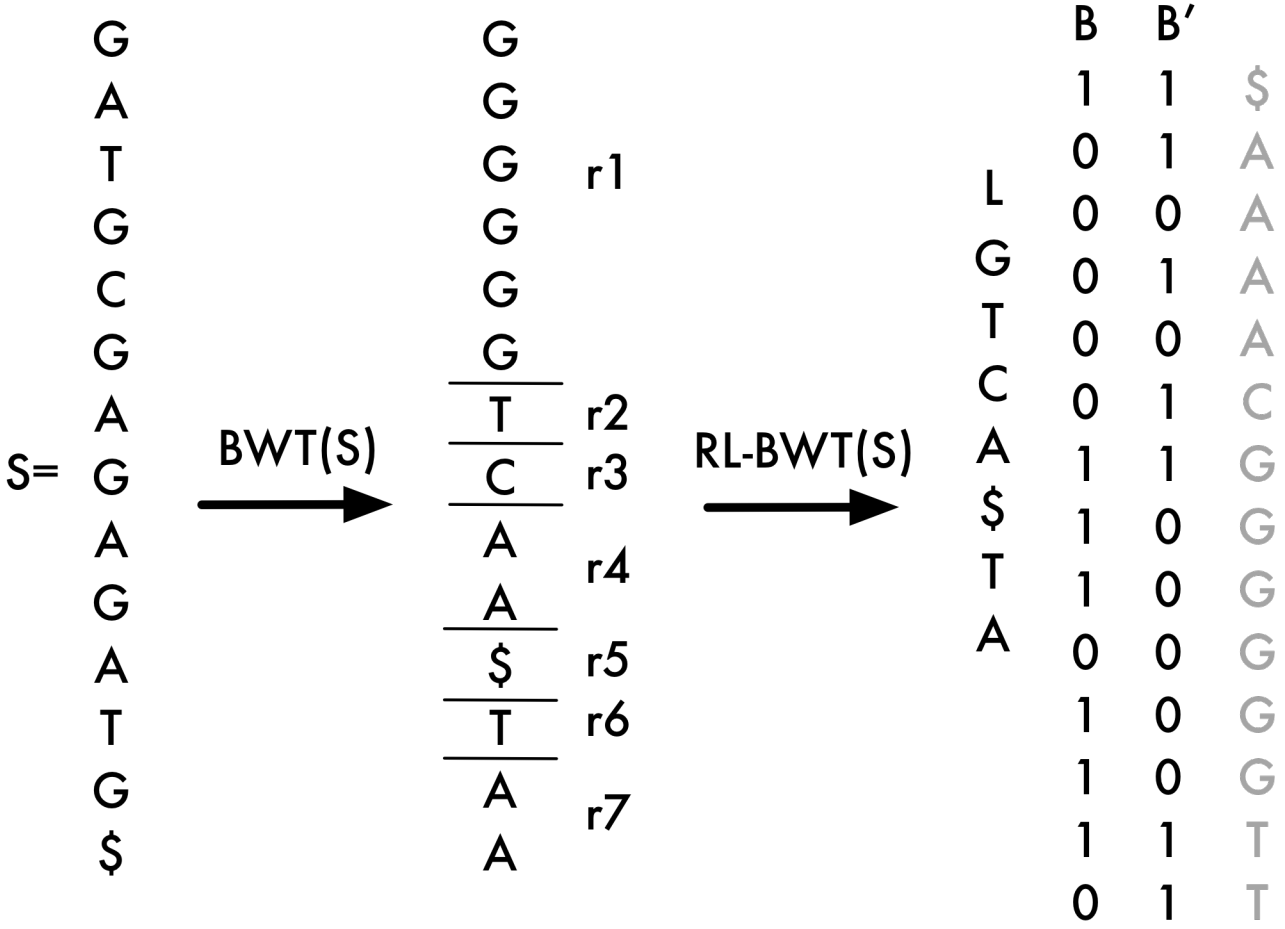
By augmenting BOSS with the LCP, we can represent all the dBG up to order k



Encoding EdgeBWT: Wavelet trees



Encoding EdgeBWT: RL-BWT



Closing remarks

- We can compress several dBG and still supporting fast operations
- It is still necessary to propose new algorithms that work on top of BOSS
- Can we encode the patterns of the graph? (bubbles, tips cross-links)
- Can we combine BOSS with other compression schemes (LZ, grammars) to improve compression even further?

Thanks!

Encoding EdgeBWT: RL-BWT