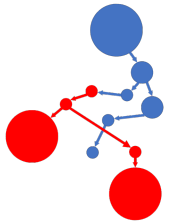
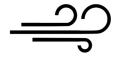


WGT
Wheeler Graph Toolkit



WGT: Tools and algorithms for recognizing, visualizing and generating Wheeler graphs

2023.04.14

@KuanHaoChao

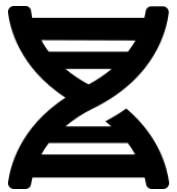
Presenter : Kuan-Hao Chao (khchao.com)

Kuan-Hao Chao†, Pei-Wei Chen†, Sanjit A. Seshia, Ben Langmead

Berkeley
UNIVERSITY OF CALIFORNIA

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Department of Computer Science



RECOMB-SEQ

Why is Wheeler graph interesting?: XBW *2005*

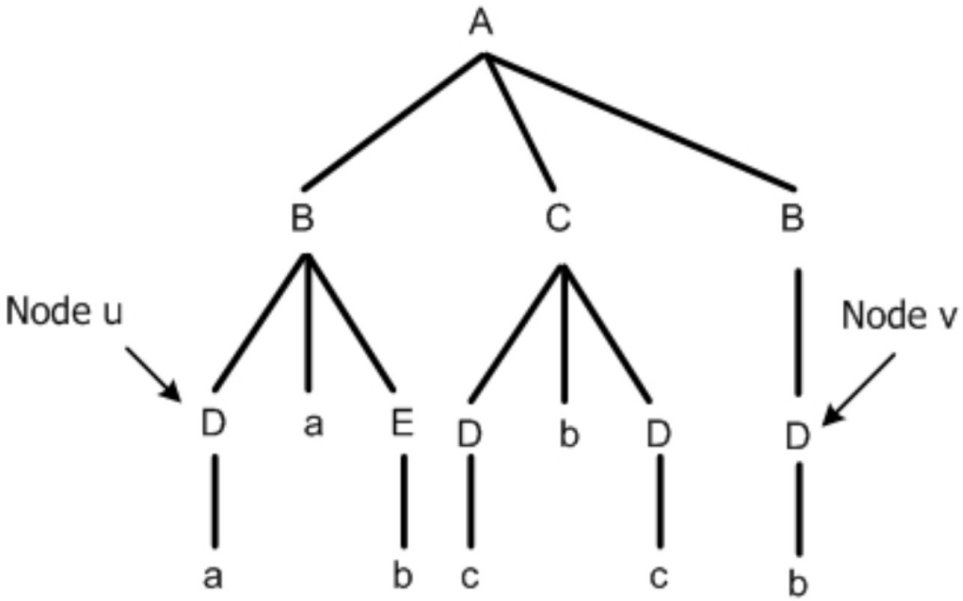
Browse Conferences > Annual IEEE Symposium on Found... > 46th Annual IEEE Symposium on ...

Annual IEEE Symposium on Foundations of Computer Science

Structuring labeled trees for optimal succinctness, and beyond

Publisher: IEEE | Cite This | PDF

P. Ferragina ; F. Luccio ; G. Manzini ; S. Muthukrishnan



(a)

S_{last}	S_α	S_π
0	A	*empty string*
0	B	A
0	D	BA
1	a	DBA
0	a	BA
1	E	BA
1	b	EBA
0	C	A
0	D	CA
1	c	DCA
0	b	CA
1	D	CA
1	c	DCA
1	B	A
1	D	BA
1	b	DBA

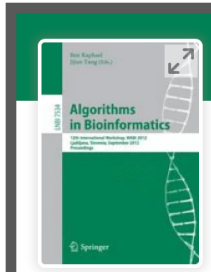
(b)

stable sort

	S_{last}	S_α	S_π	
1	0	A	*empty string*	
2	0	B	A	p(u)
3	0	C	A	
4	1	B	A	p(v)
5	0	D	BA	u
6	0	a	BA	u's siblings
7	1	E	BA	
8	1	D	BA	v
9	0	D	CA	
10	0	b	CA	
11	1	D	CA	
12	1	a	DBA	
13	1	b	DBA	
14	1	c	DCA	
15	1	c	DCA	
16	1	b	EBA	

(c)

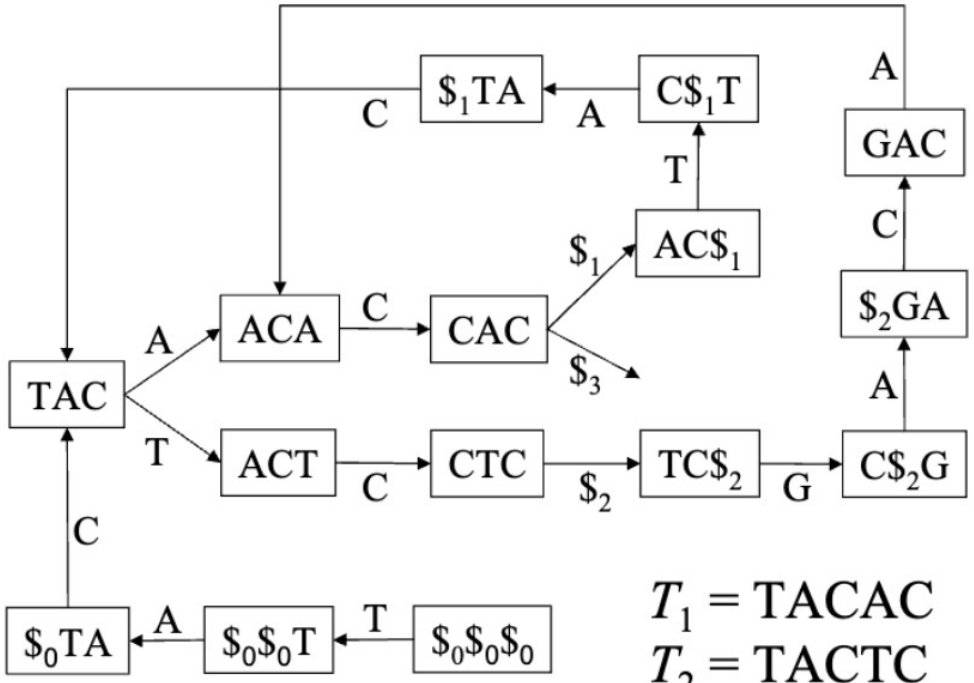
Why is Wheeler graph interesting?: BOSS *2012*



Succinct de Bruijn Graphs

Alexander Bowe, Taku Onodera, Kunihiko Sadakane & Tetsuo Shibuya

Conference paper 2012



$T_1 = \text{TACAC}$
$T_2 = \text{TACTC}$
$T_3 = \text{GACTC}$

F	
$\$_i$	0
A	3
C	7
G	13
T	14

Incoming edge label **Outgoing edge label**

	last	Node	*W*
$\$_i$	1	$\$_0\$_0\$_0$	T
	1	AC$\$_1$	T
	1	TC$\$_2$	G
A	1	ACA	C
	1	$\$_2$GA	C
	1	$\$_0$TA	C
	1	$\$_1$TA	C-
C	0	CAC	$\$_3$
	1	CAC	$\$_1$
	1	GAC	A
	0	TAC	T
	1	TAC	A-
	1	CTC	$\$_2$
G	1	C$\$_2$G	A
T	1	$\$_0\$_0$T	A
	1	C$\$_1$T	A
	1	ACT	C

Why is Wheeler graph interesting?: GCSA *2014*

IEEE/ACM TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS, VOL. 11, NO. 2, MARCH/APRIL 2014 375

Indexing Graphs for Path Queries with Applications in Genome Research

Jouni Sirén, Niko Välimäki, and Veli Mäkinen

2014

```
G A C G T A - C T G
G A C G T A - - - G
G A T G T A - C T G
G A C - T A C C T G
```

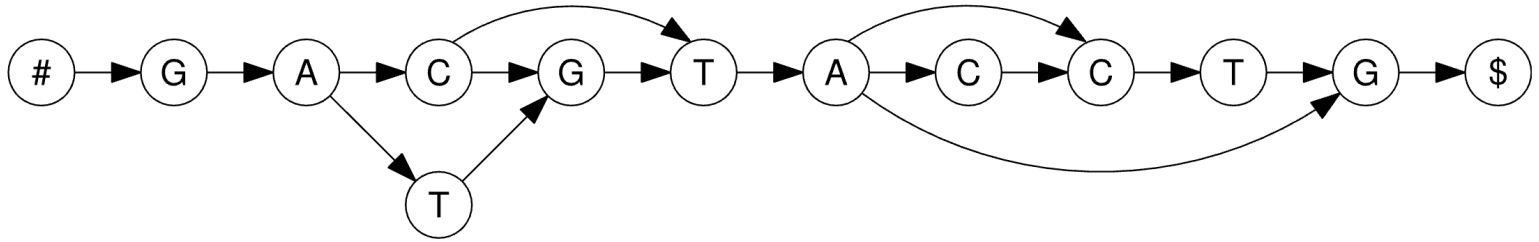


Fig. 4. A reverse deterministic automaton corresponding to the first 10 positions of the multiple alignment in Fig. 1.

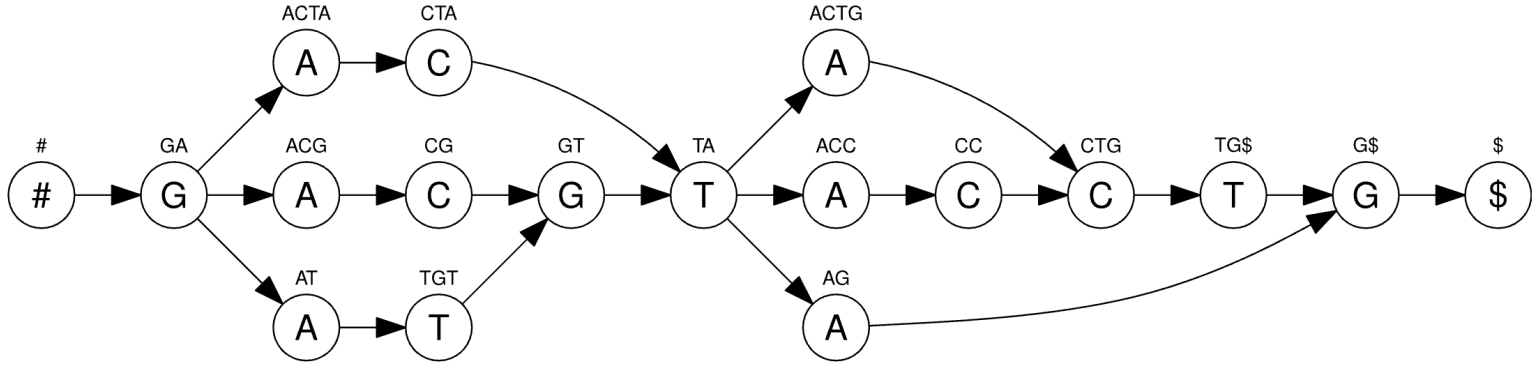


Fig. 5. A prefix-sorted automaton built for the automaton in Fig. 4. The strings above nodes are prefixes $p(v)$.

Why is Wheeler graph interesting?: VG *2017-18*

OPEN ACCESS

Indexing Variation Graphs

Author: Jouni Sirén | AUTHORS INFO & AFFILIATIONS

https://doi.org/10.1137/1.9781611974768.2

2017

Published: 01 October 2018

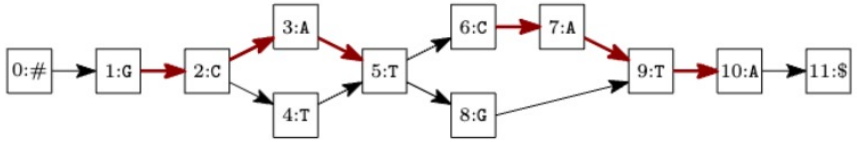
Variation graph toolkit improves read mapping by representing genetic variation in the reference

Erik Garrison, Jouni Sirén, Adam M Novak, Glenn Hickey, Jordan M Eizenga, Eric T Dawson, William Jones, Shilpa Garg, Charles Markello, Michael F Lin, Benedict Paten & Richard Durbin

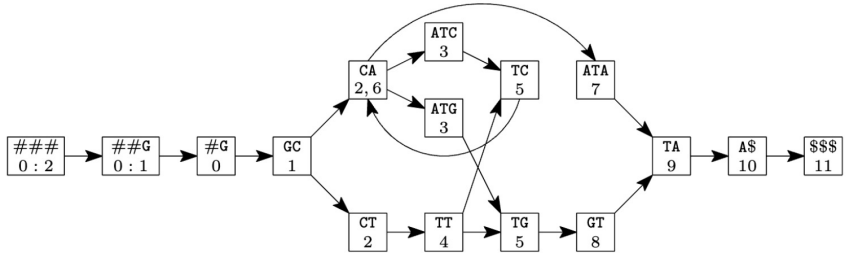
Nature Biotechnology **36**, 875–879 (2018) | Cite this article

2018

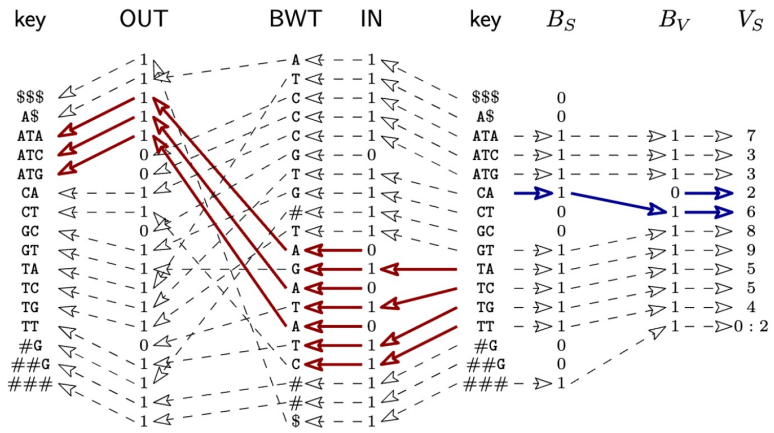
***G*: Input graph**

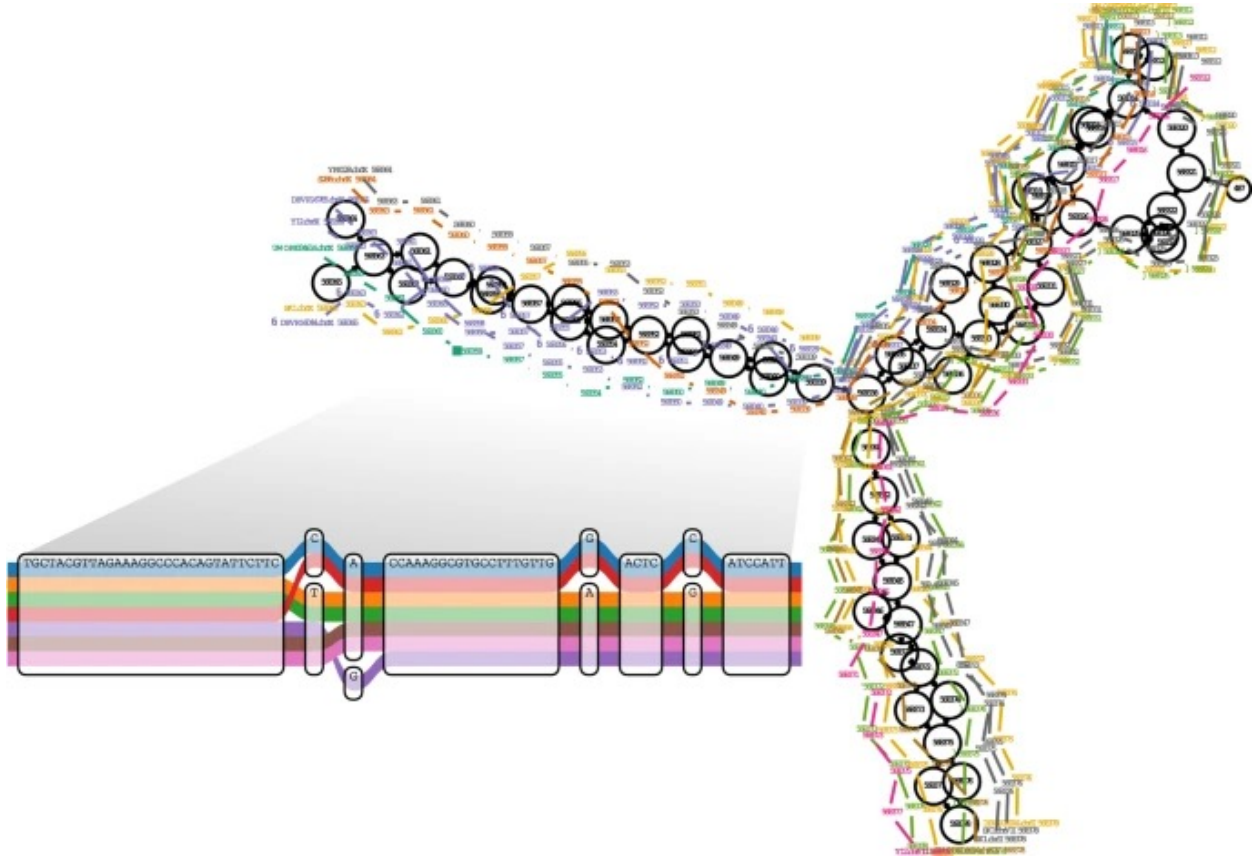


***G′*: Order-3 de Bruijn graph**



GCSA* for graph *G″





Why is Wheeler graph interesting?: HISAT2 *2019*

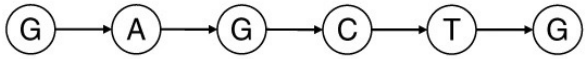
Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype

nature biotechnology

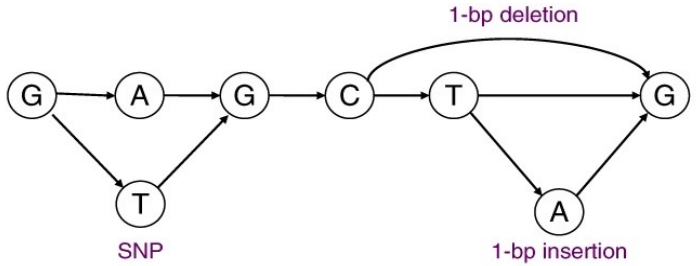
Daehwan Kim[1*], Joseph M. Paggi[2], Chanhee Park[1], Christopher Bennett[1] and Steven L. Salzberg[3,4]

2019

1. Reference sequence (6 bp long)

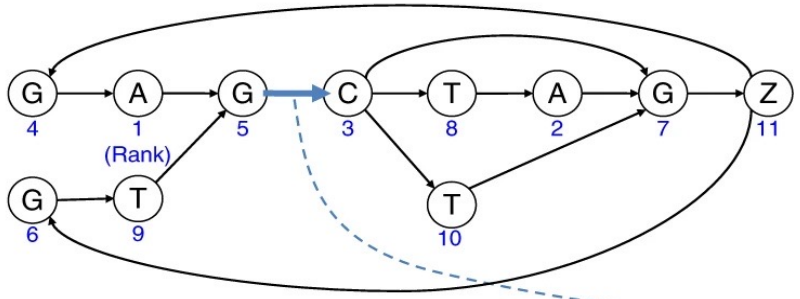


2. Graphical representation (original graph)



Prefix doubling and pruning

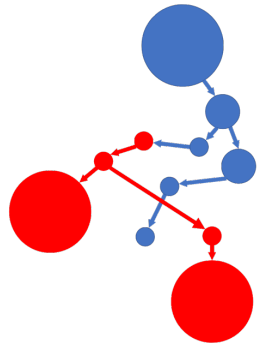
3. Prefix-sorted graph



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	
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

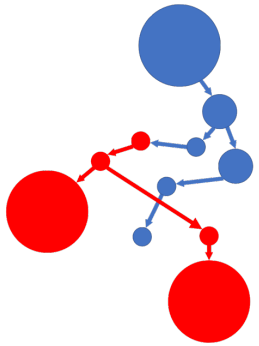
Our Contribution



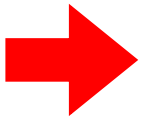
WGT
Wheeler Graph Toolkit

- A Wheeler graph visualizer: bipartite representation
- Implemented the first Wheeler graph *recognizer*
- Wheeler graph *generators*: Random WG / Trie / DBG / RDG

Our Contribution



WGT
Wheeler Graph Toolkit



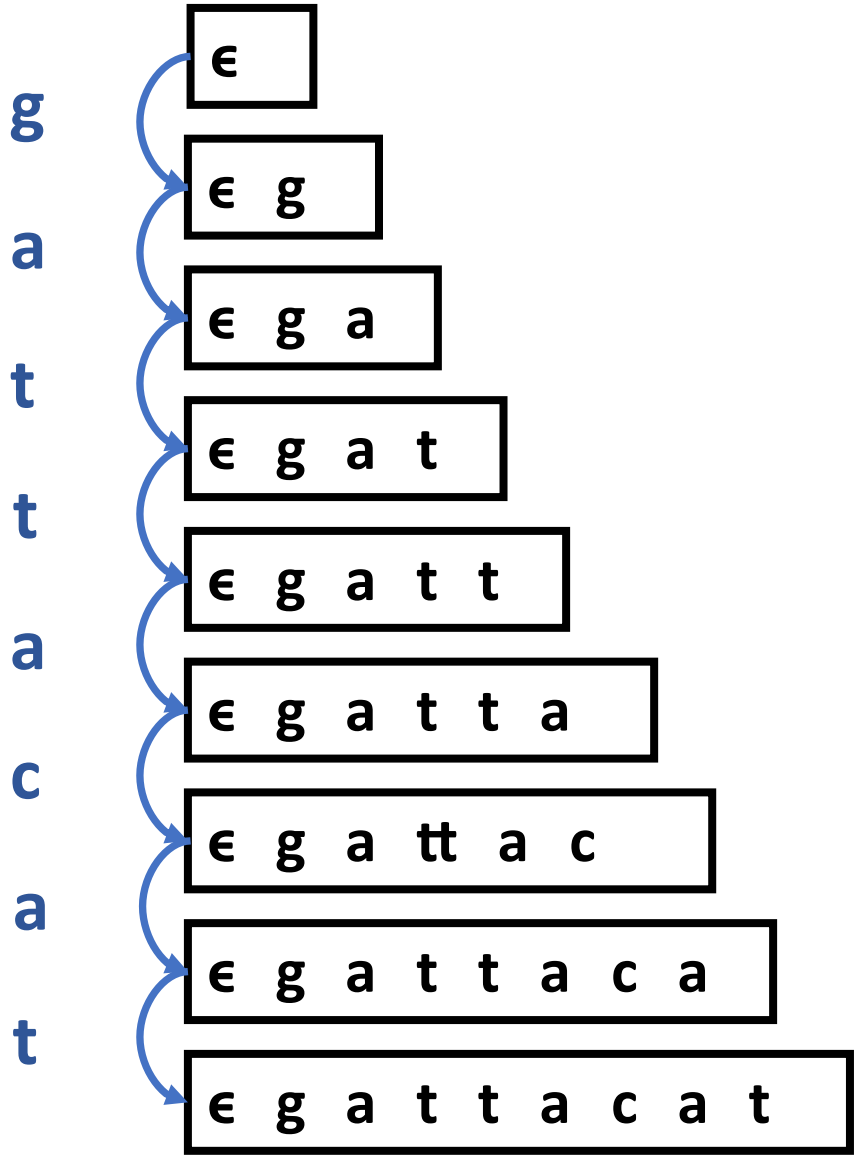
- ***A Wheeler graph visualizer: bipartite representation***

- Implemented the first Wheeler graph *recognizer*

- Wheeler graph *generators*: Random WG / Trie / DBG / RDG

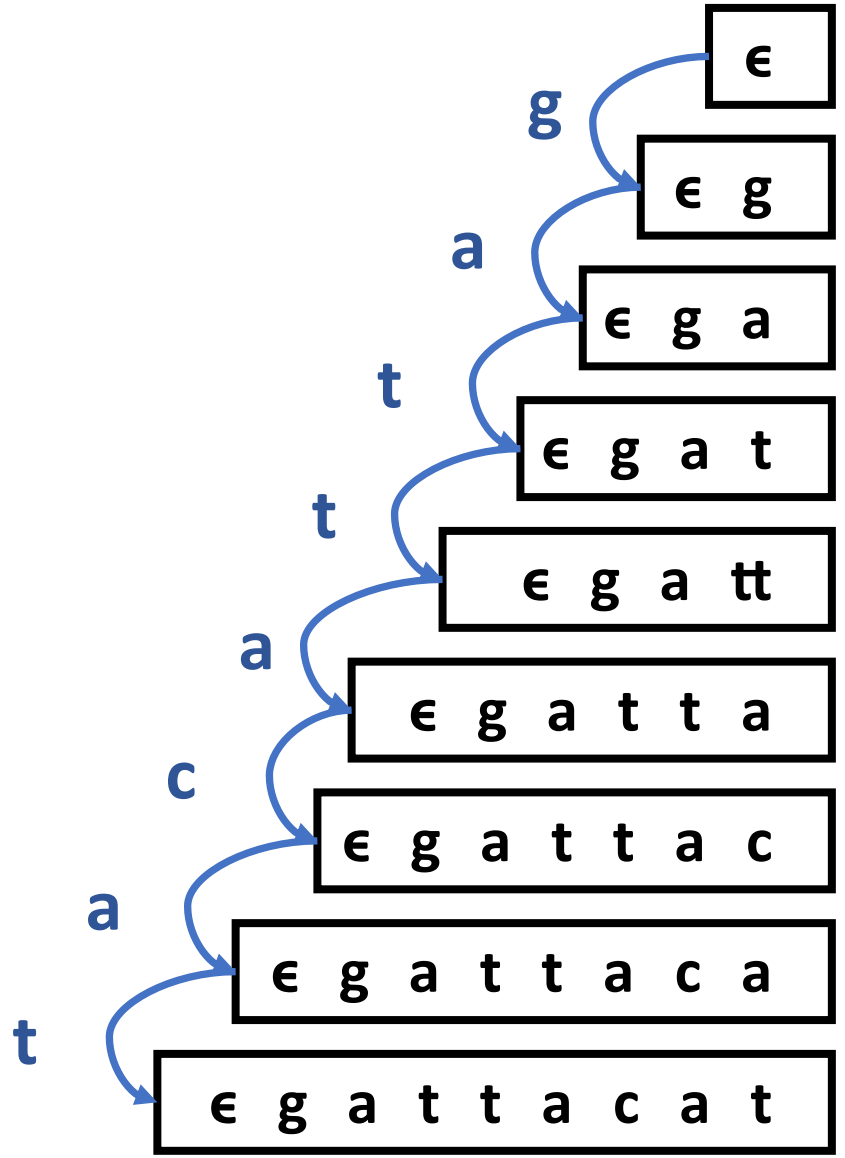
Representing a Sequence in Wheeler Graph

T: ϵgattacat



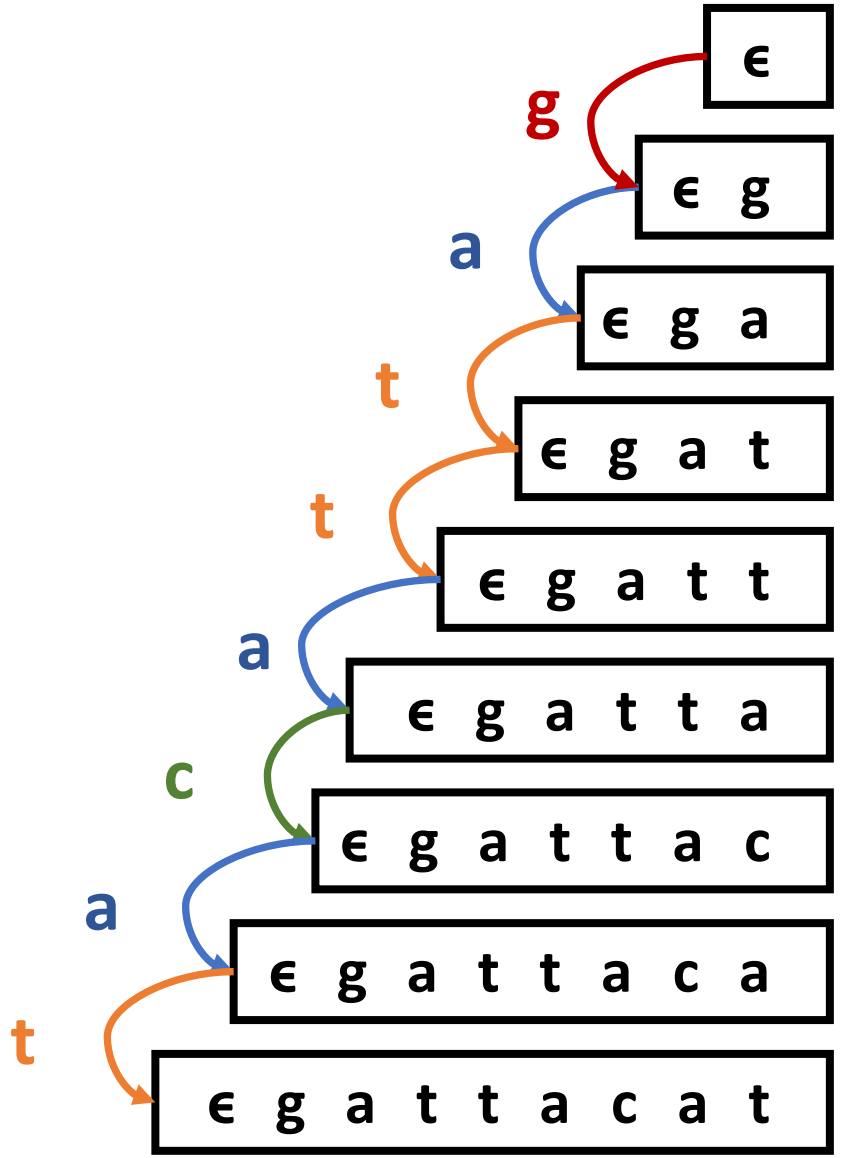
Representing a Sequence in Wheeler Graph

T: ϵgattacat



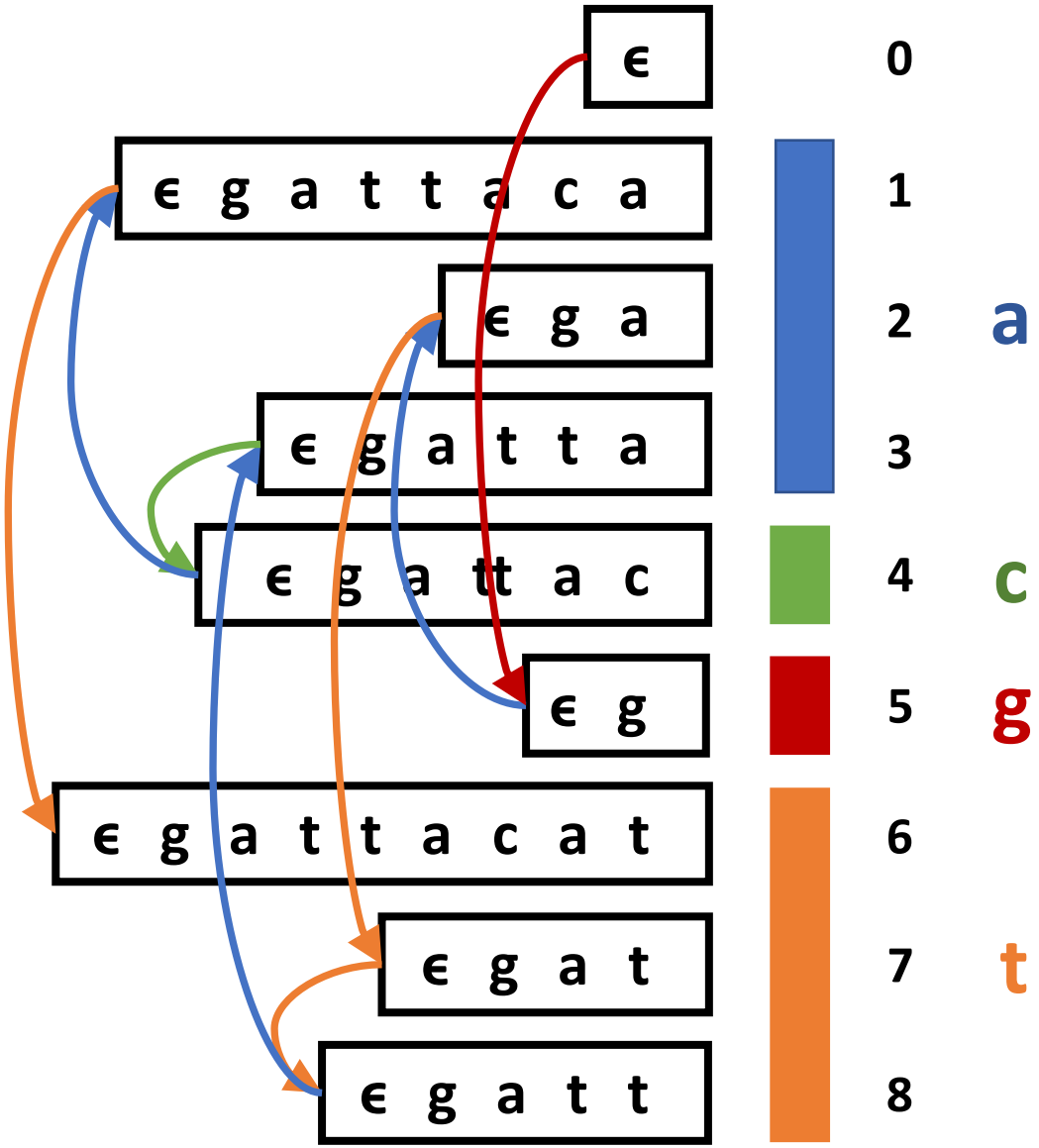
Representing a Sequence in Wheeler Graph

T: ϵgattacat



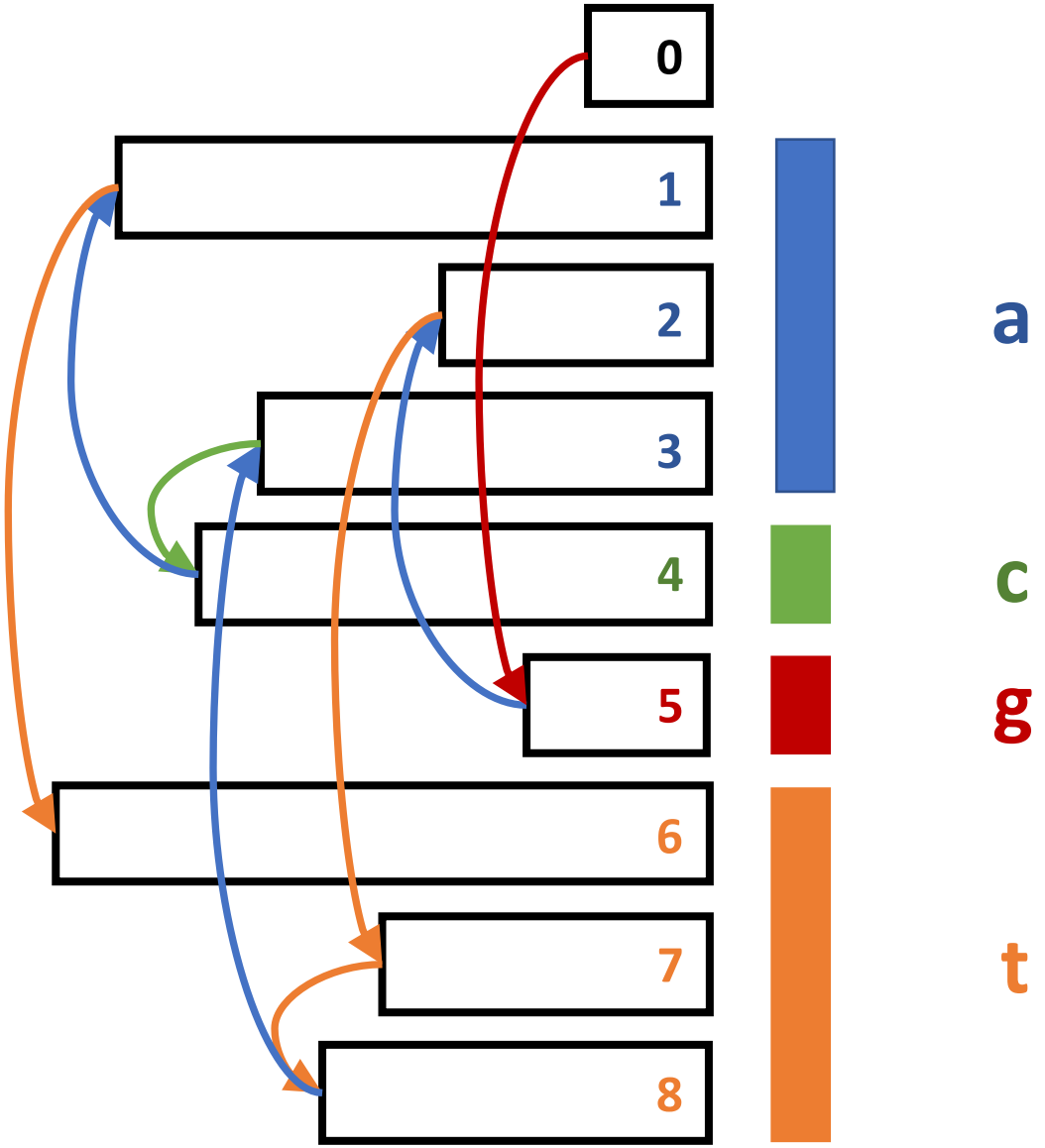
Representing a Sequence in Wheeler Graph

T: ϵgattacat



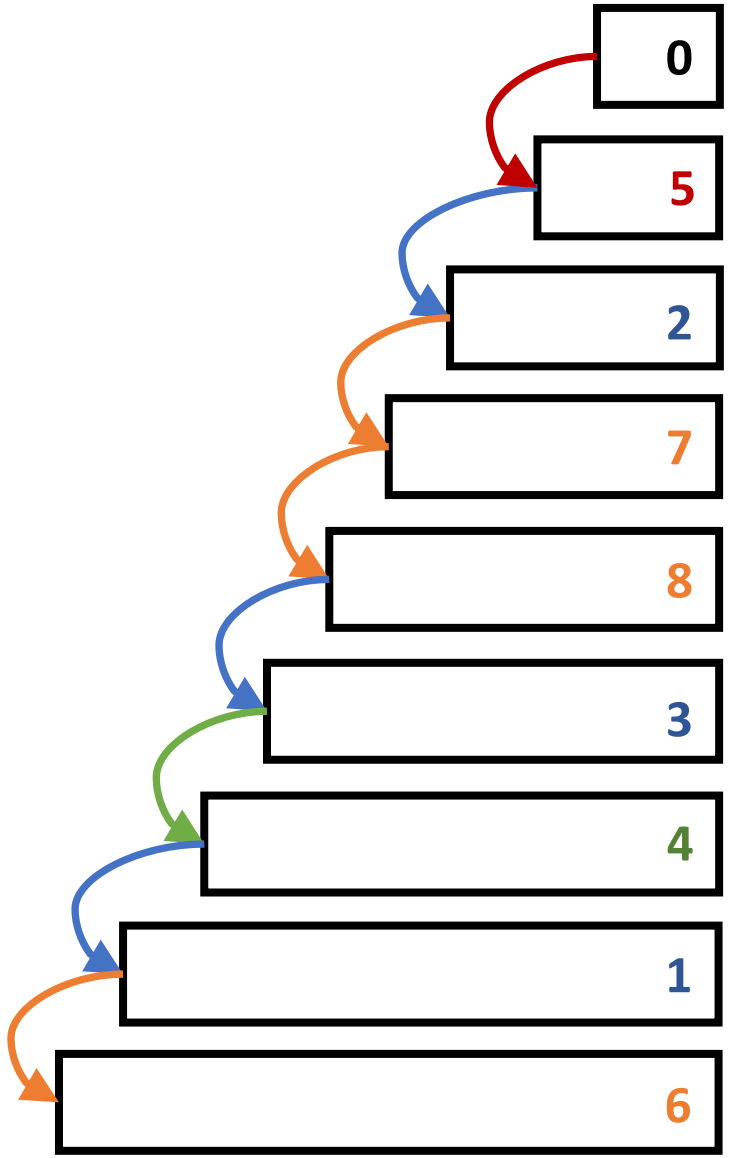
Representing a Sequence in Wheeler Graph

T: ϵgattacat



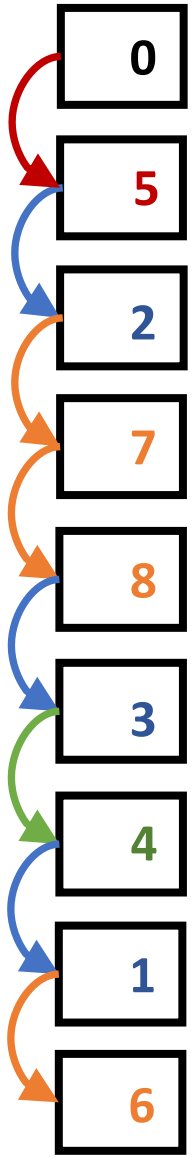
Representing a Sequence in Wheeler Graph

T: ϵgattacat



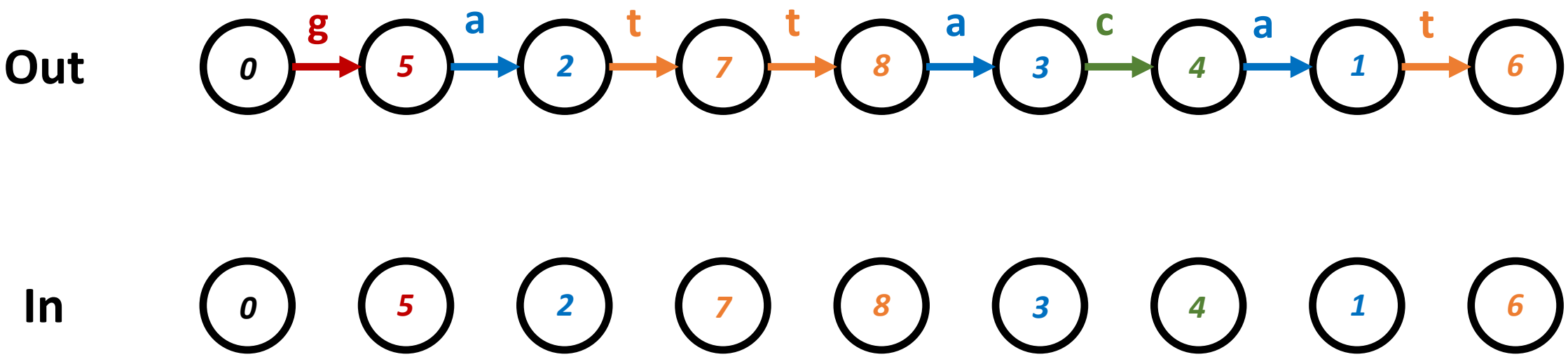
Representing a Sequence in Wheeler Graph

T: ϵgattacat



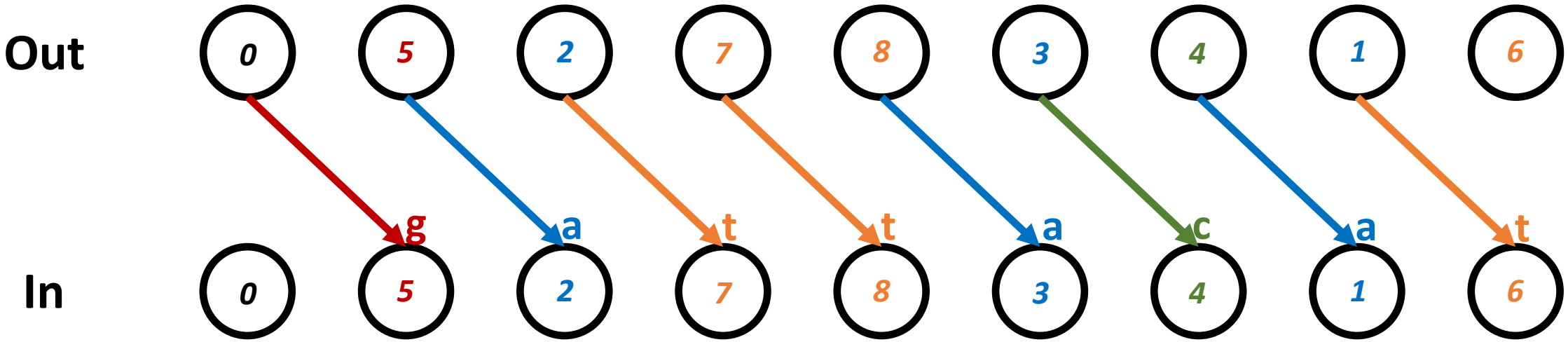
Visualizer: Bipartite WG Representation

T: ∈gattacat



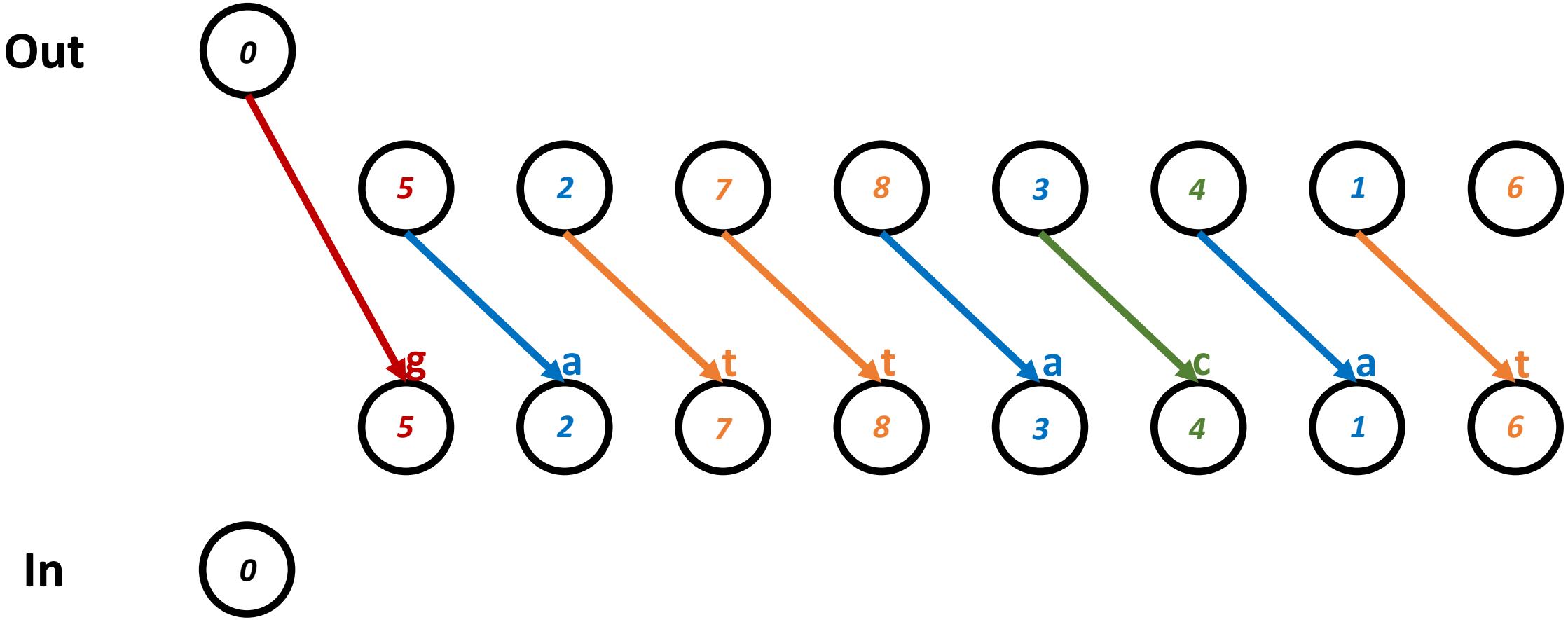
Visualizer: Bipartite WG Representation

T: ϵgattacat



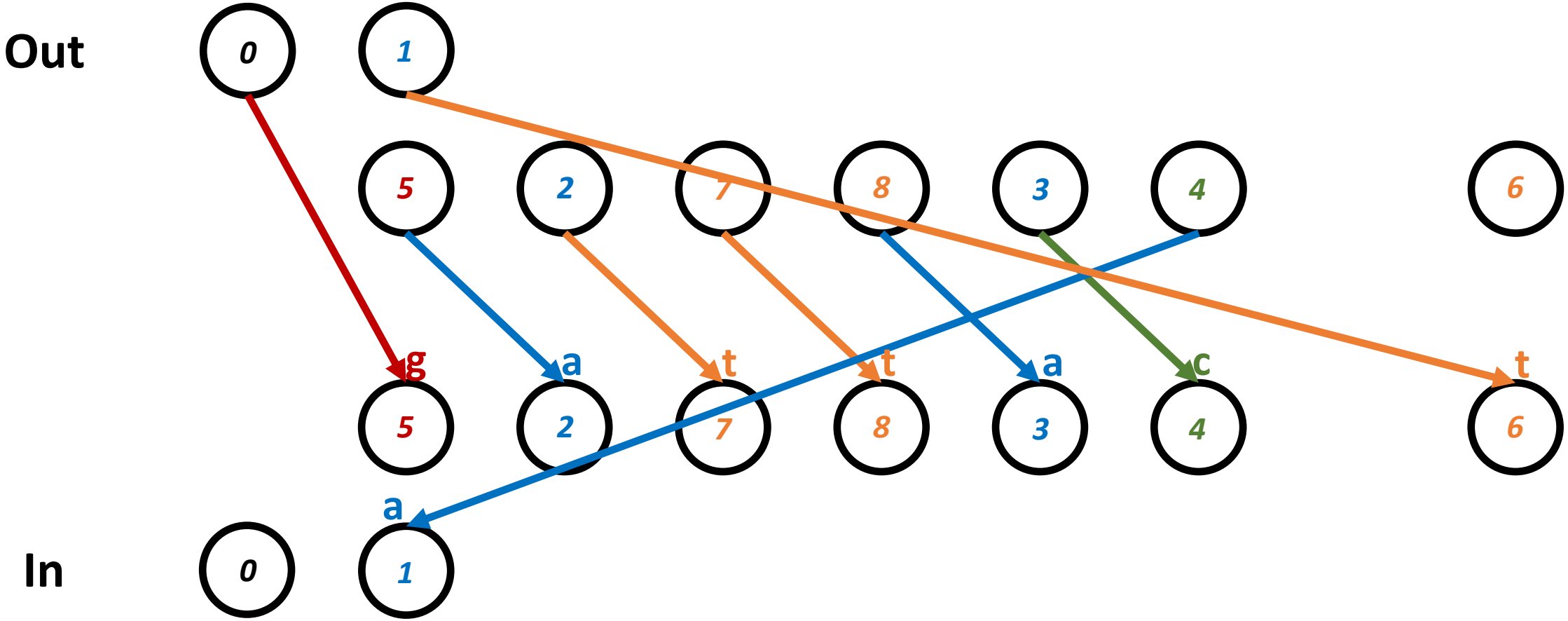
Visualizer: Bipartite WG Representation

T: ϵgattacat



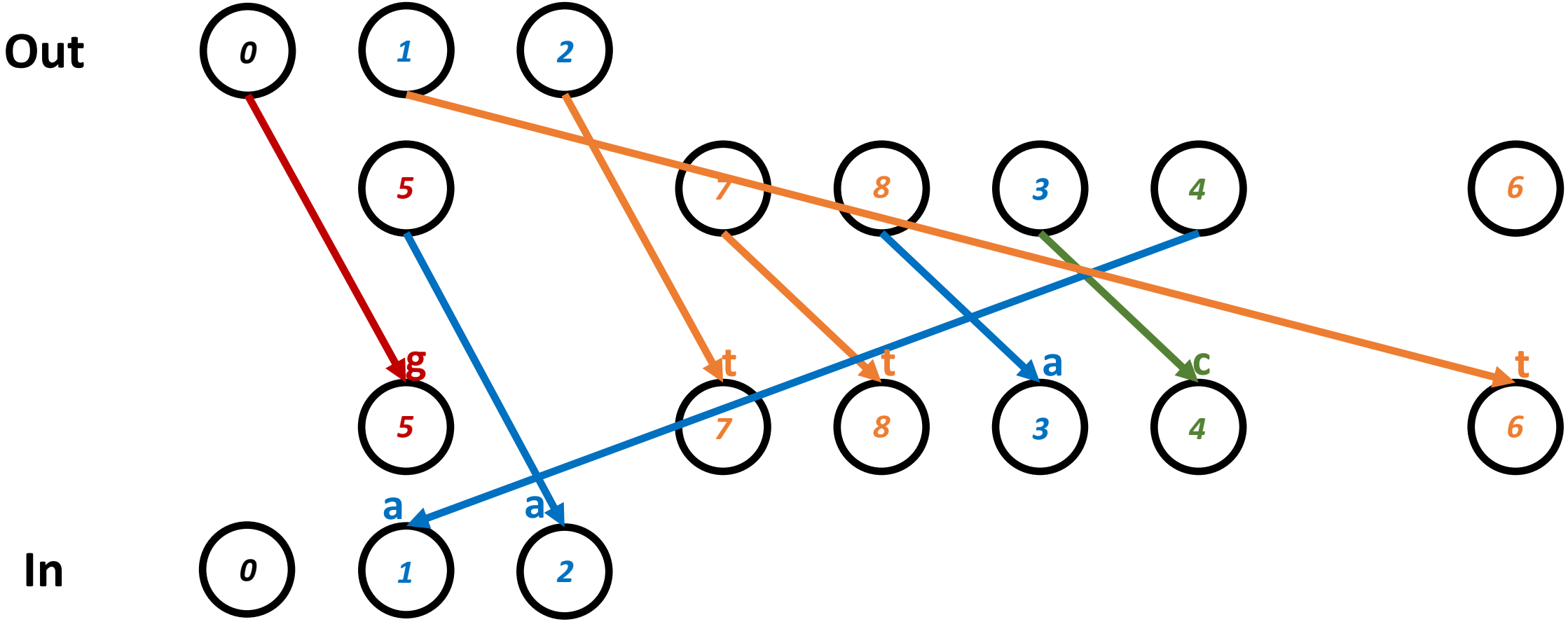
Visualizer: Bipartite WG Representation

T: ϵgattacat



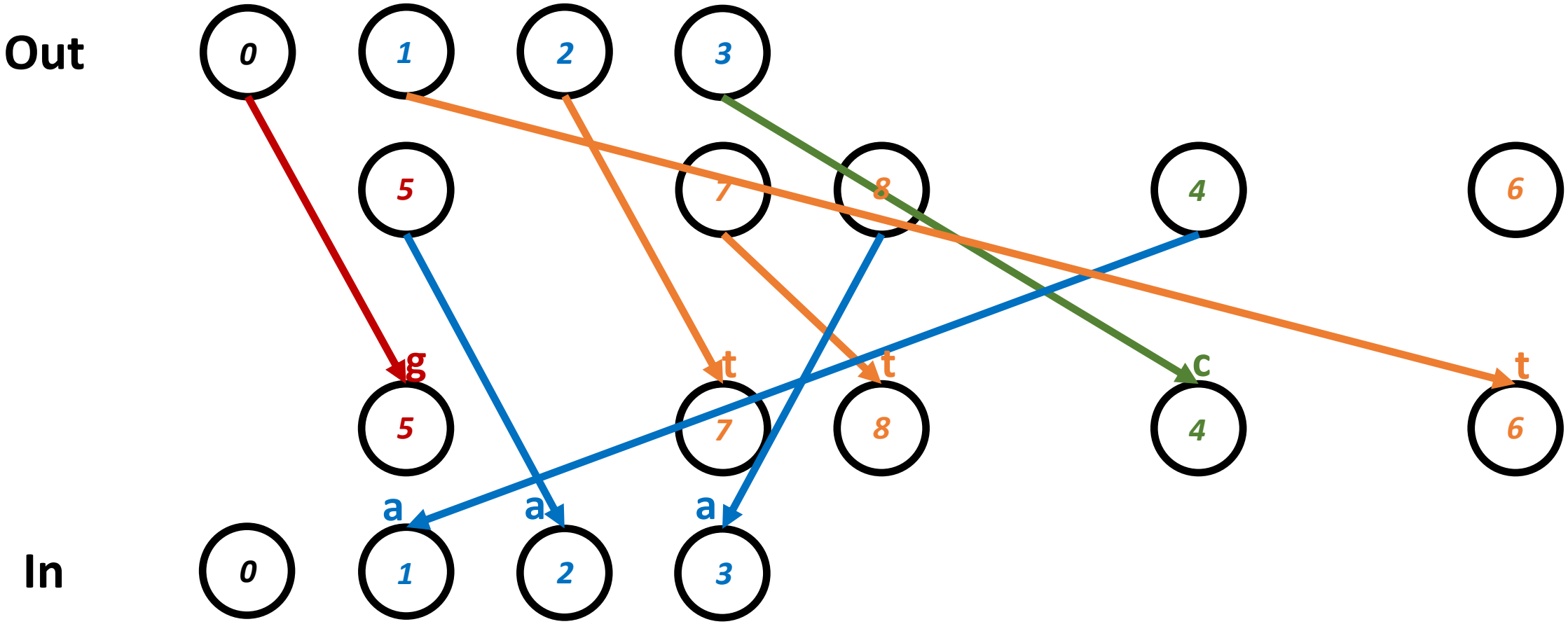
Visualizer: Bipartite WG Representation

T: ϵgattacat



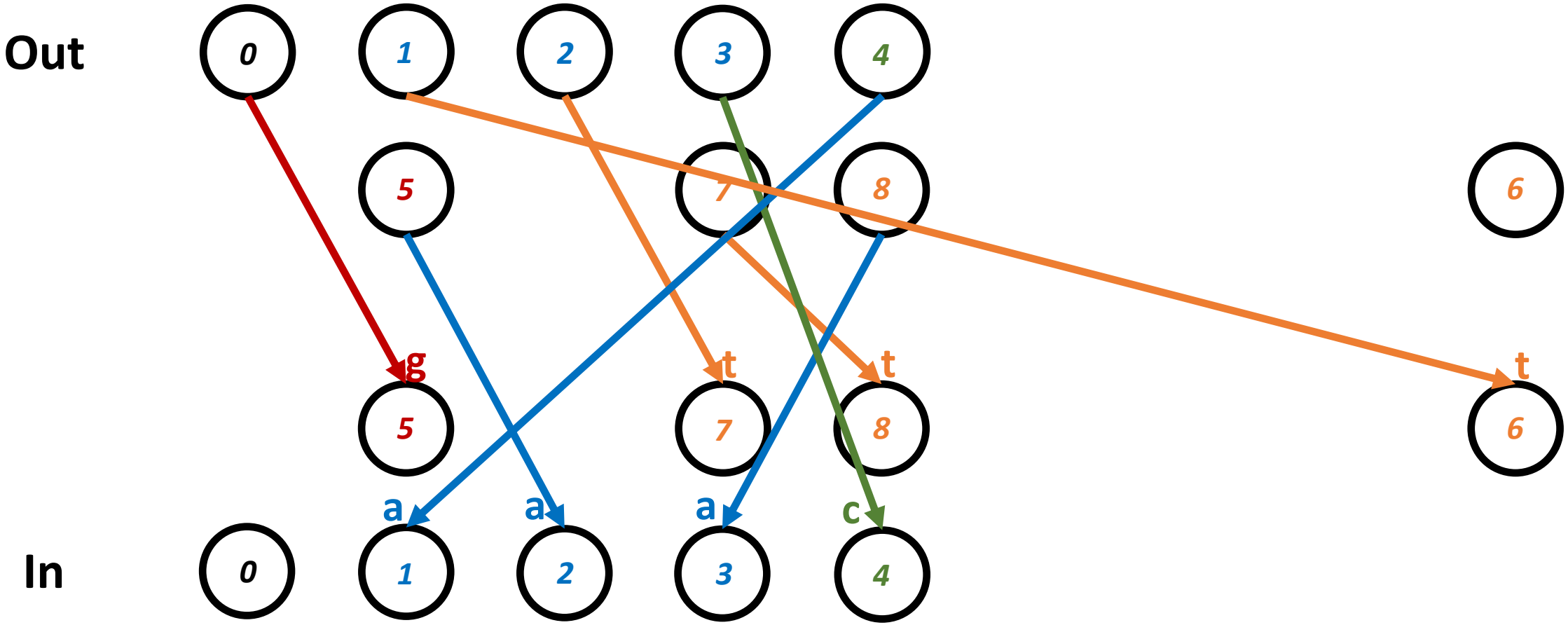
Visualizer: Bipartite WG Representation

T: ϵgattacat

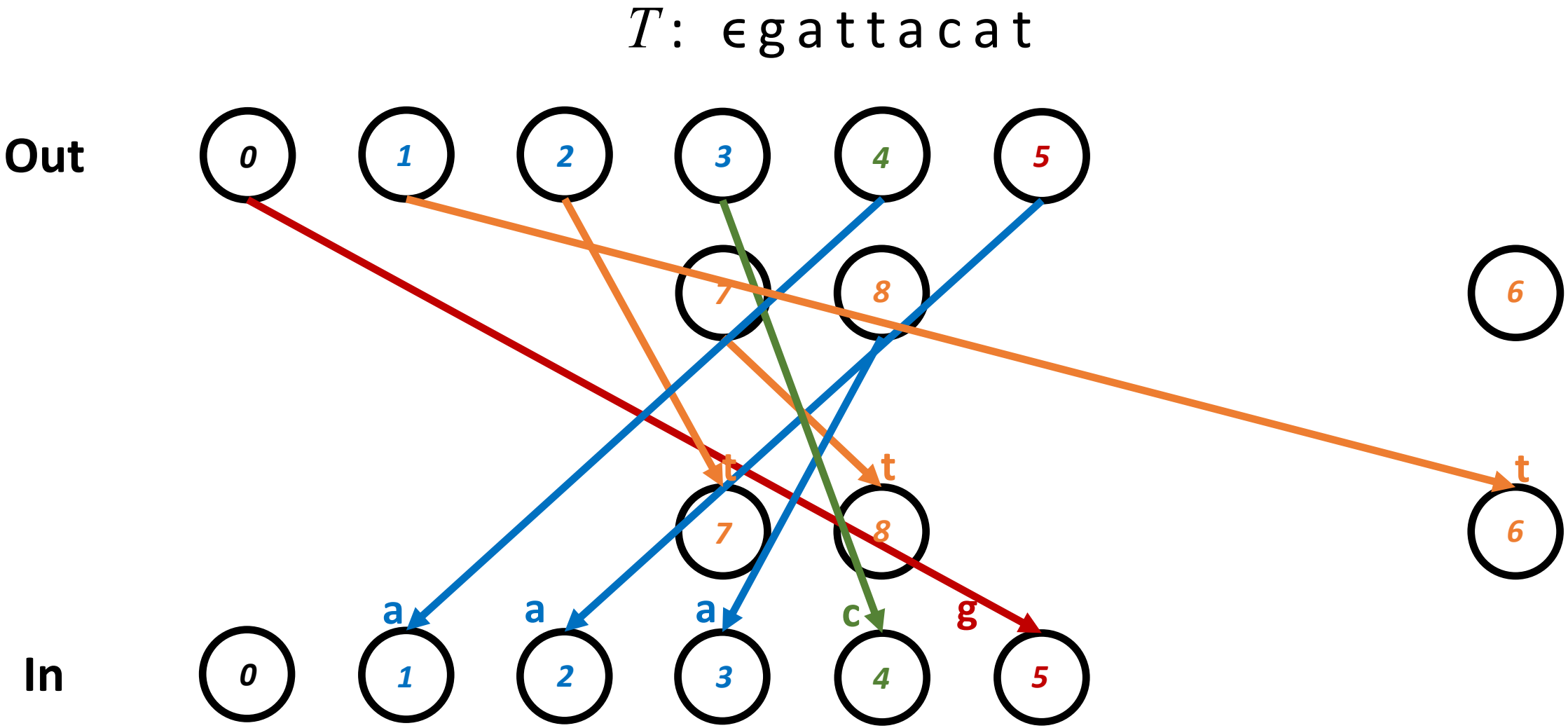


Visualizer: Bipartite WG Representation

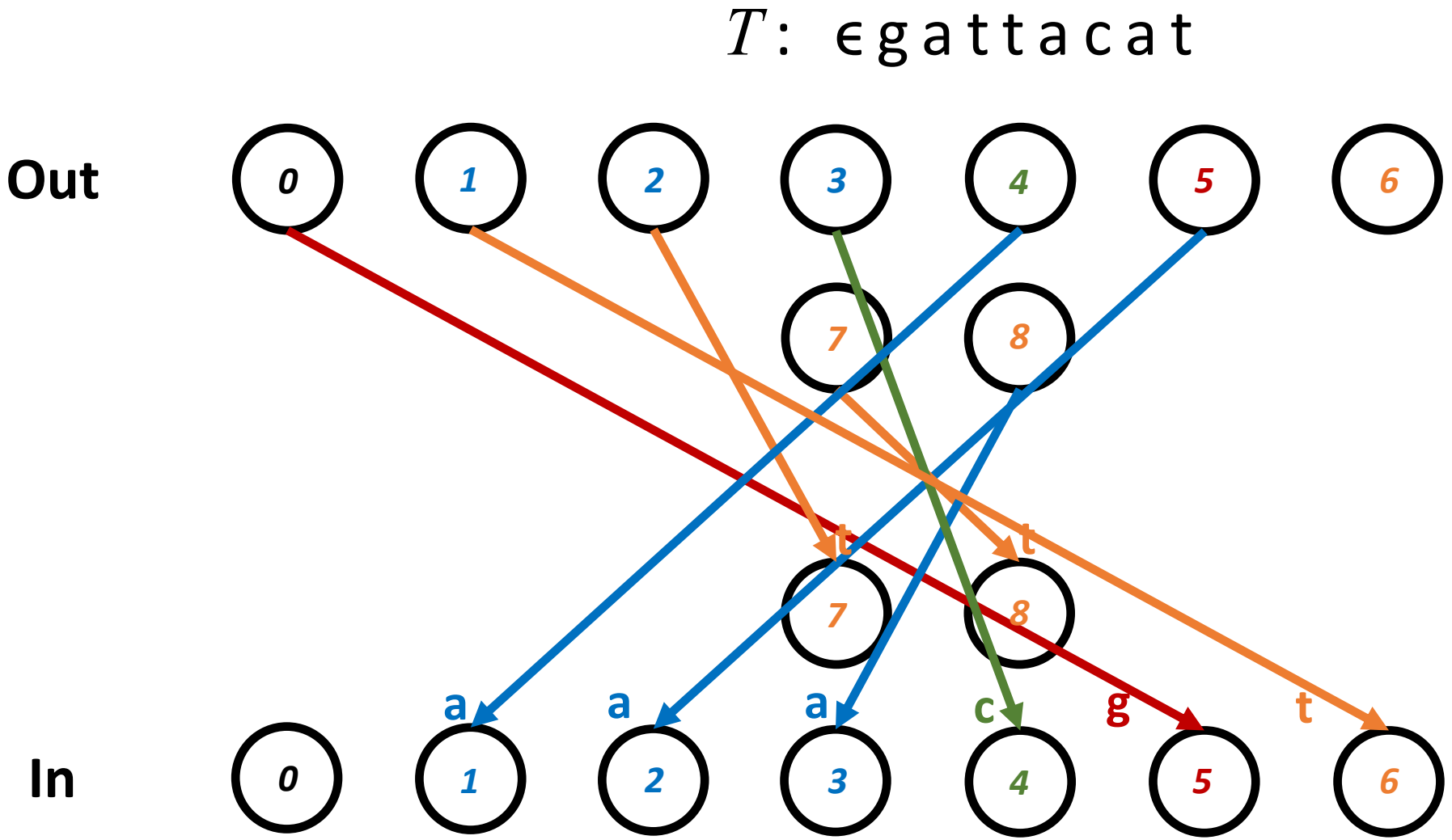
T: ϵgattacat



Visualizer: Bipartite WG Representation

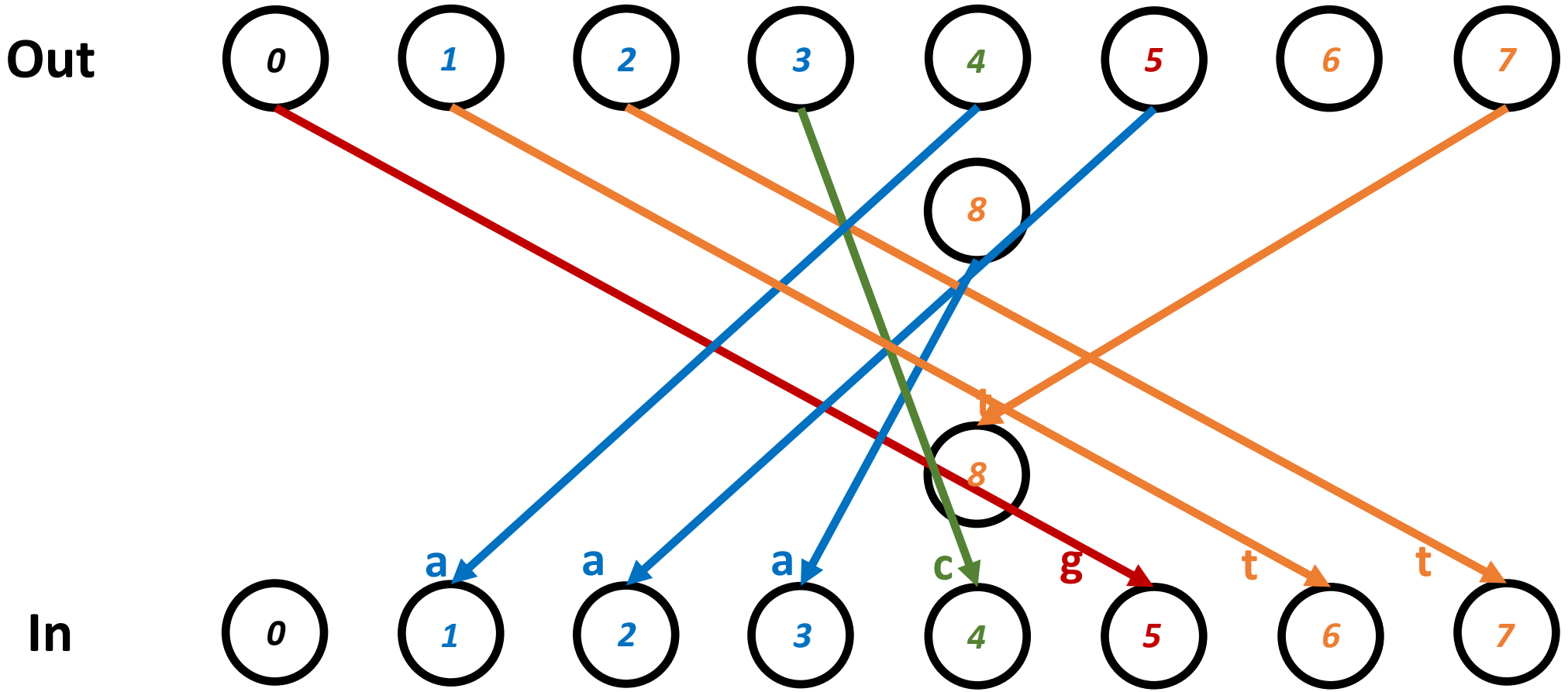


Visualizer: Bipartite WG Representation



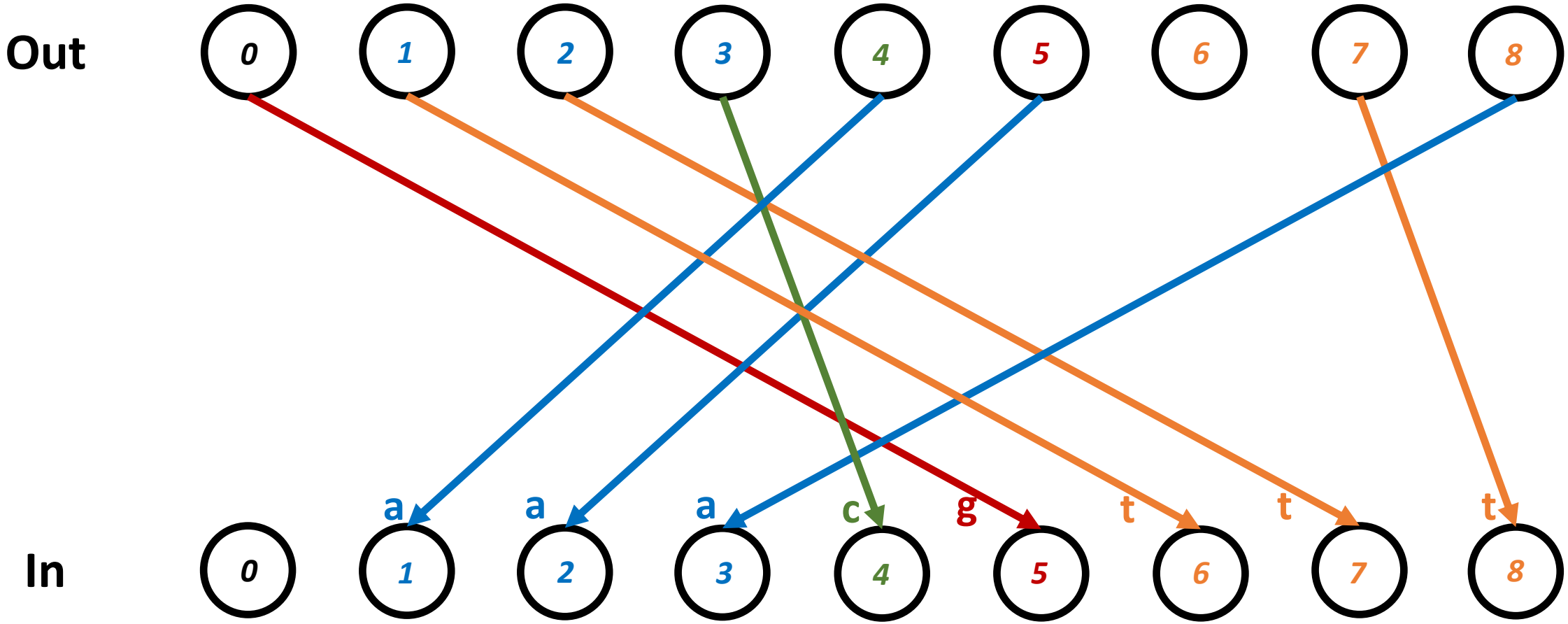
Visualizer: Bipartite WG Representation

T: ϵgattacat

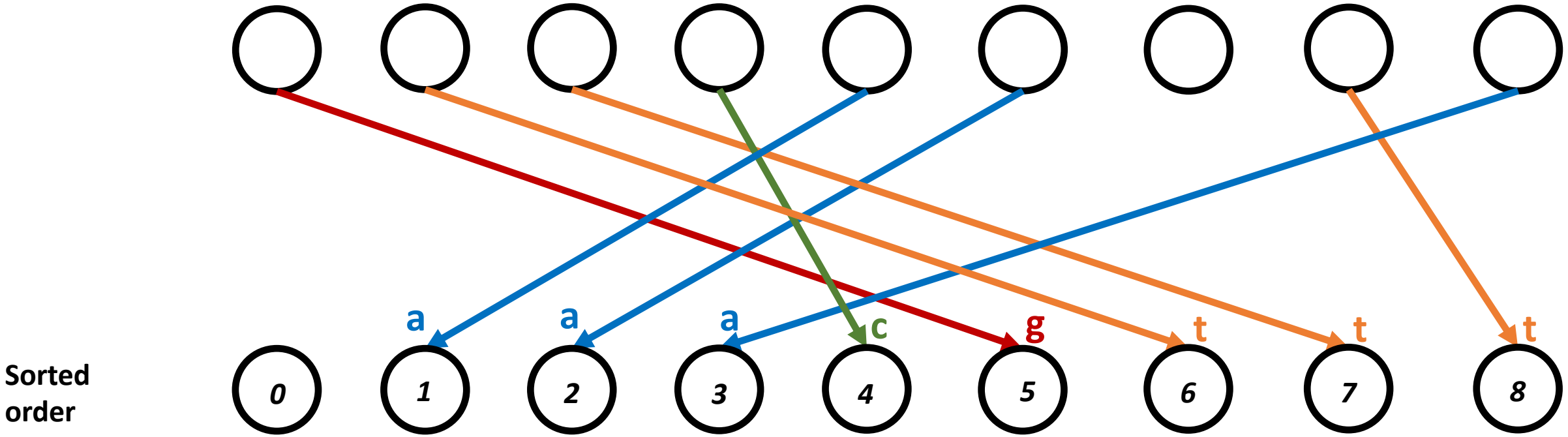


Visualizer: Bipartite WG Representation

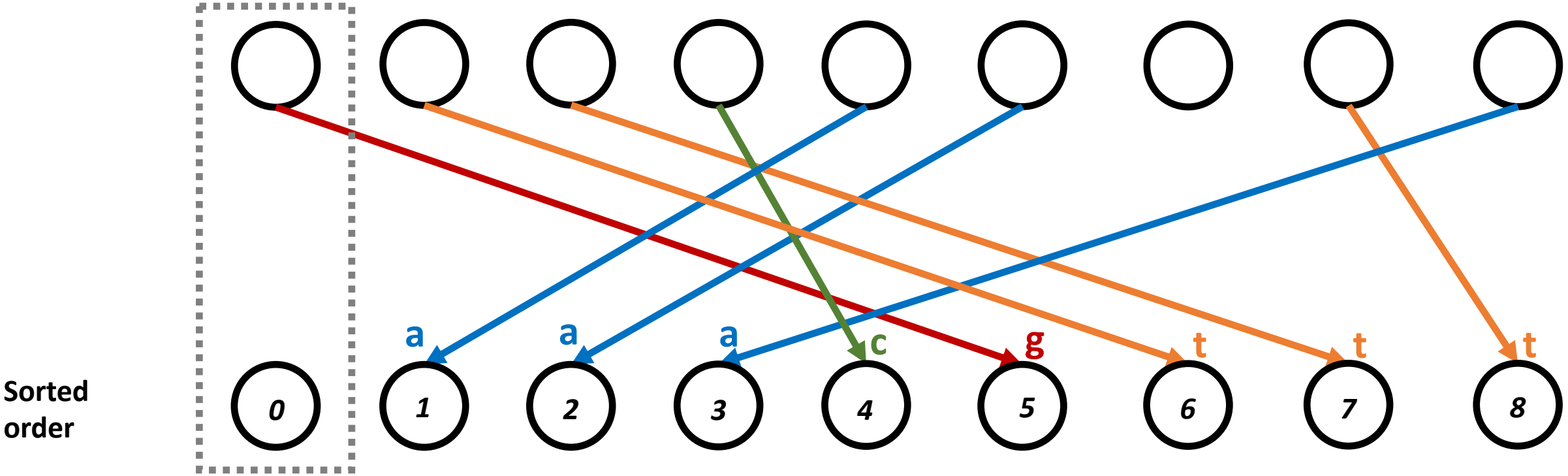
T: ϵ g a t t a c a t



Wheeler Graph Definition

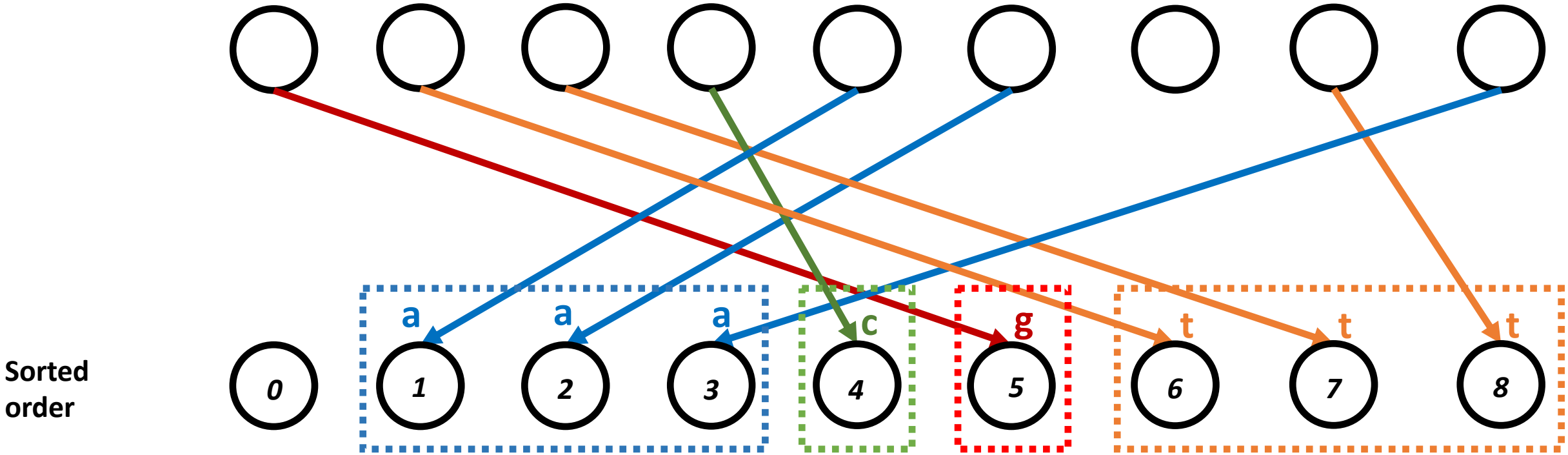


Wheeler Graph Definition

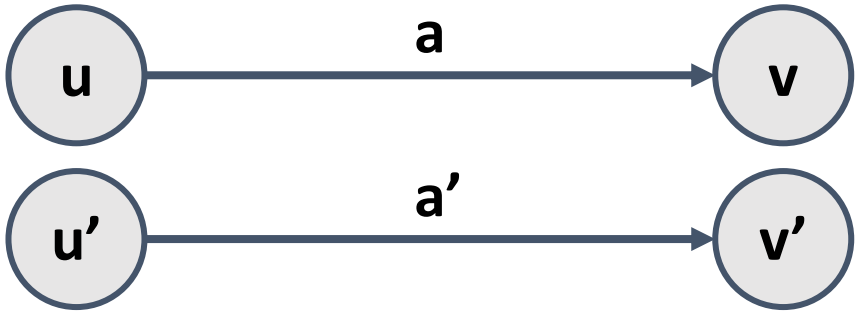


- 1. Node with indegree 0 comes before every other nodes.

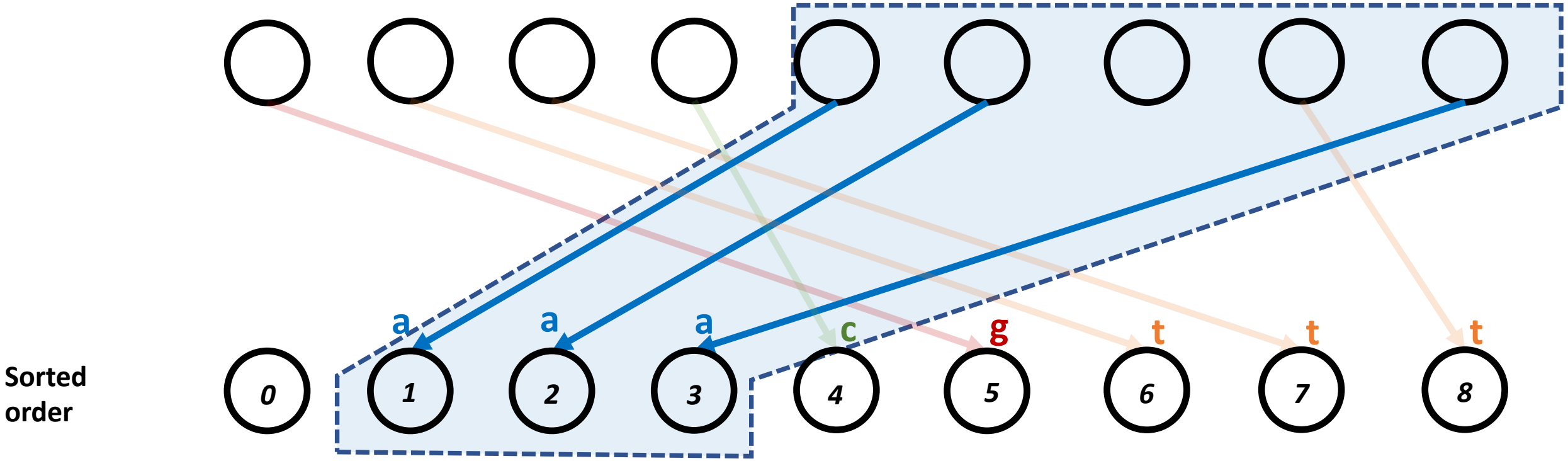
Wheeler Graph Definition



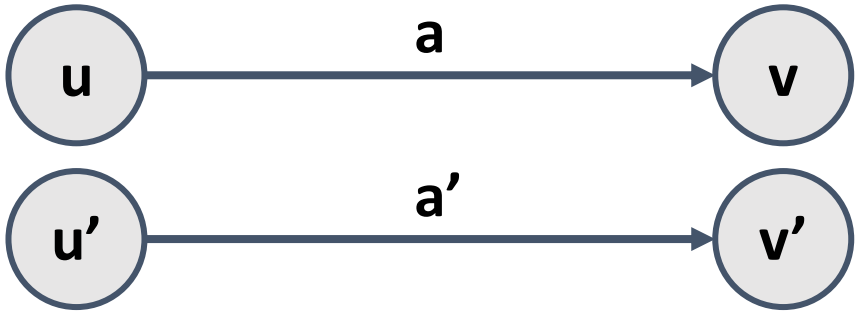
1. Node with indegree 0 comes before every other nodes.
2. $a \prec a' \implies v < v'$



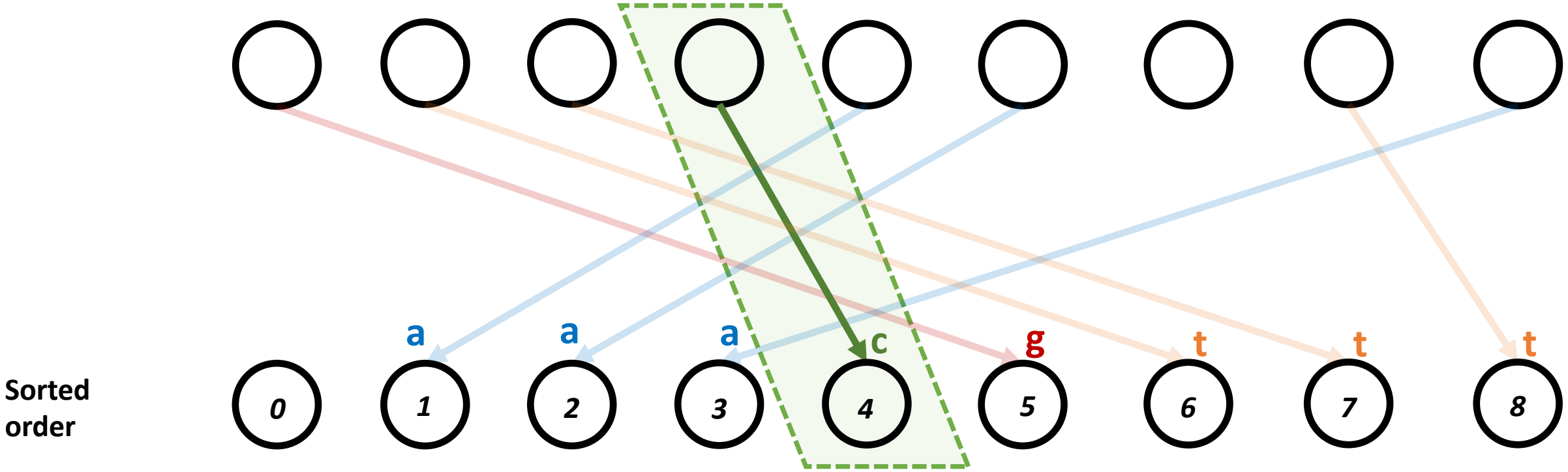
Wheeler Graph Definition



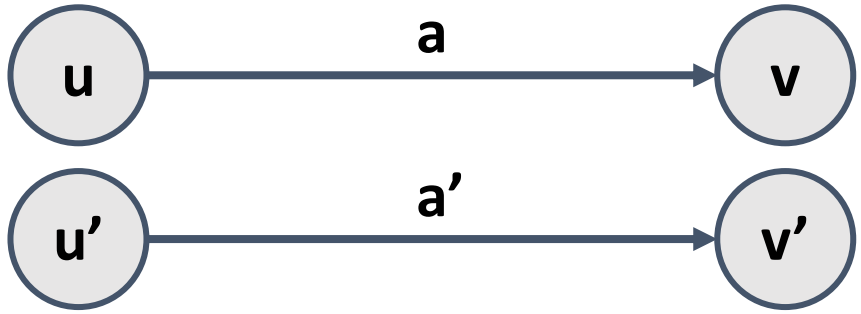
1. Node with indegree 0 comes before every other nodes.
2. $a \prec a' \implies v < v'$
3. $(a = a') \wedge (u < u') \implies v \leq v'$



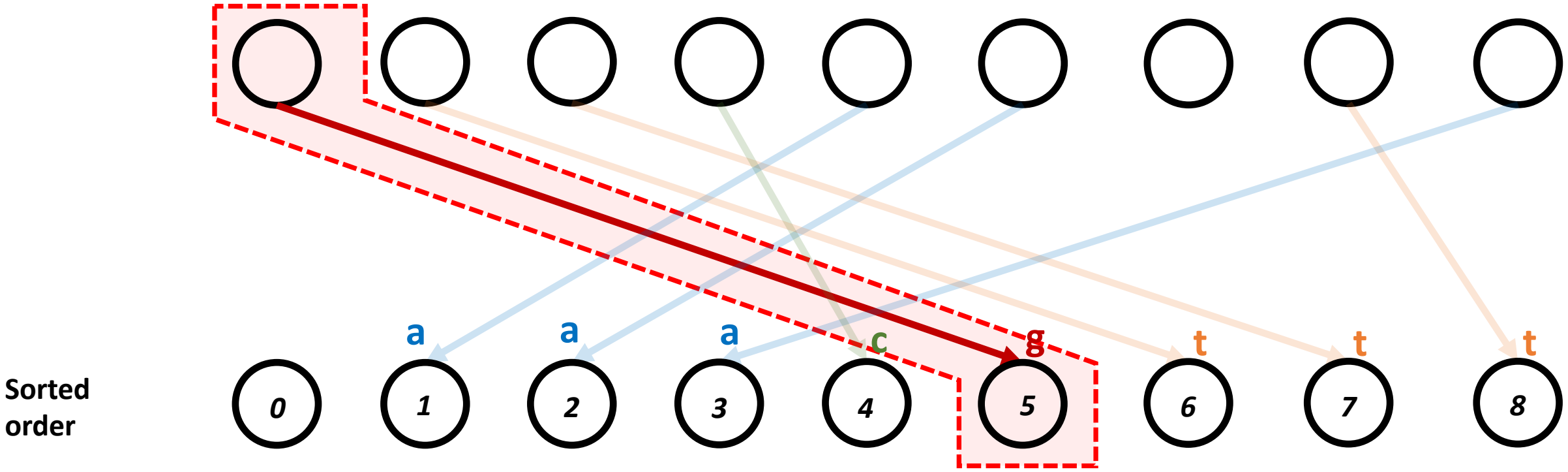
Wheeler Graph Definition



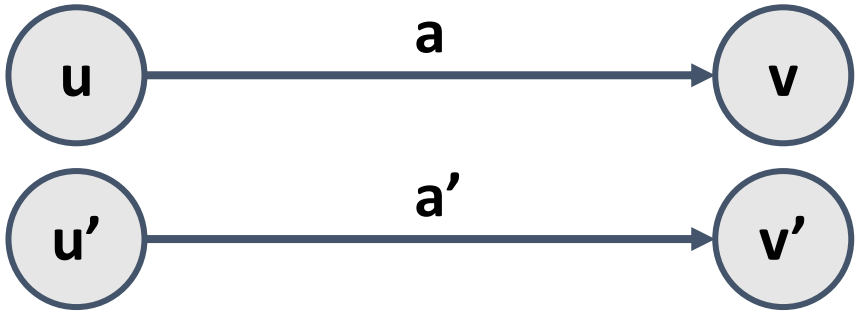
1. Node with indegree 0 comes before every other nodes.
2. $a \prec a' \implies v < v'$
3. $(a = a') \wedge (u < u') \implies v \leq v'$



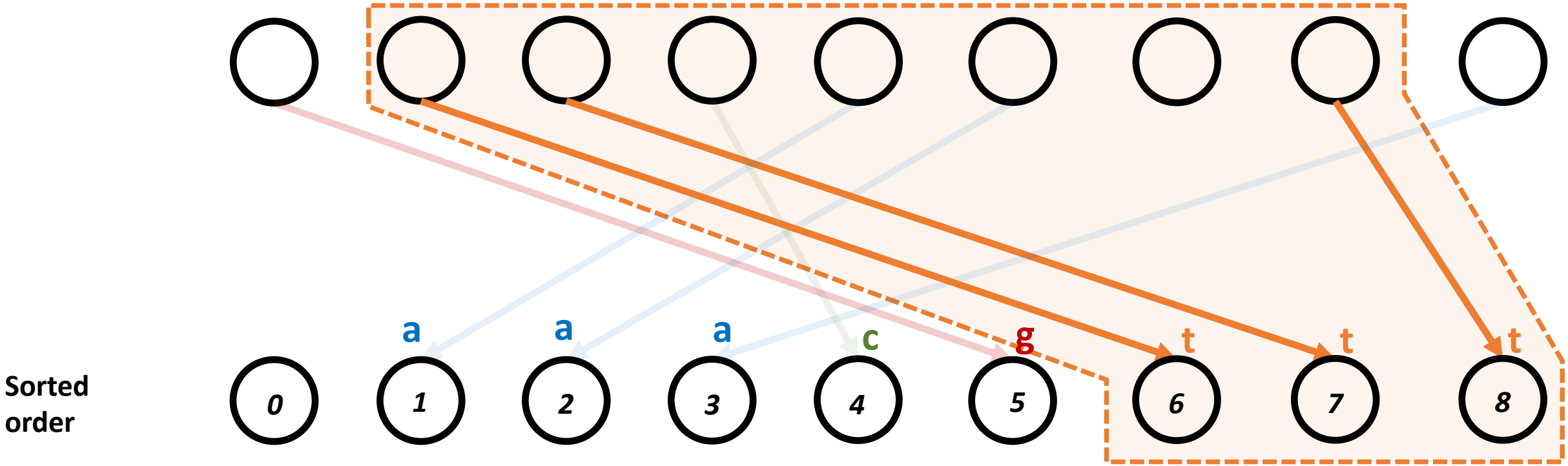
Wheeler Graph Definition



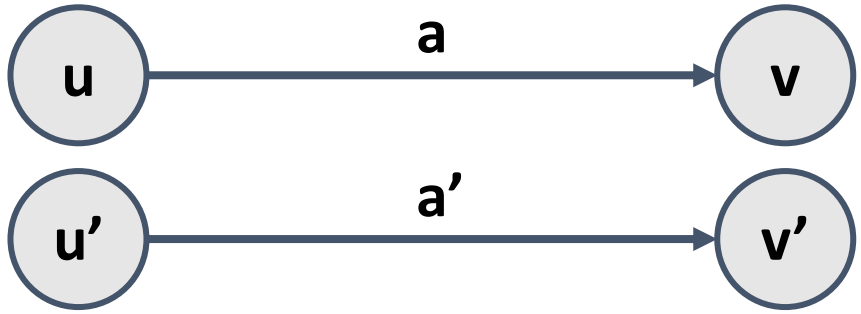
1. Node with indegree 0 comes before every other nodes.
2. $a \prec a' \implies v < v'$
3. $(a = a') \wedge (u < u') \implies v \leq v'$



Wheeler Graph Definition

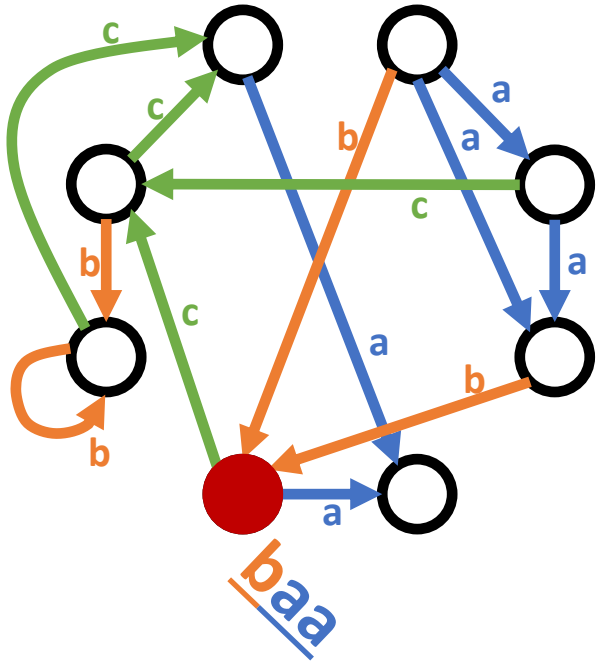


1. Node with indegree 0 comes before every other nodes.
2. $a \prec a' \implies v < v'$
3. $(a = a') \wedge (u < u') \implies v \leq v'$



Wheeler Graph: beyond Definition

1. Node with indegree 0 comes before every other nodes.
2. $$a \prec a' \implies v < v'$$
3. $$(a = a') \wedge (u < u') \implies v \leq v'$$



- Mathematical rules of
  - whether we can **sort all traversed paths in a graph** (the suffix of T^R)

ELSEVIER

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

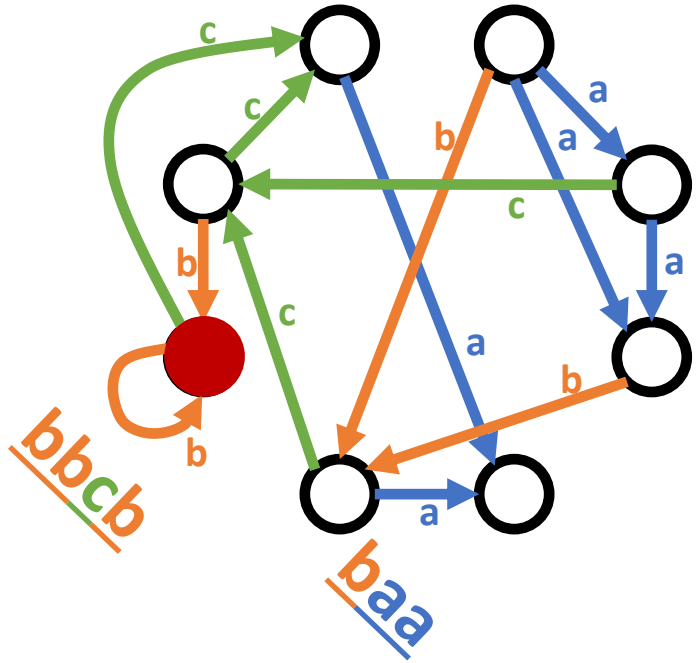
Wheeler graphs: A framework for BWT-based data structures ☆

Travis Gagie [a], Giovanni Manzini [b,c,*], Jouni Sirén [d]

2017

Wheeler Graph: beyond Definition

1. Node with indegree 0 comes before every other nodes.
2. $$a \prec a' \implies v < v'$$
3. $$(a = a') \wedge (u < u') \implies v \leq v'$$



- Mathematical rules of
  - whether we can **sort all traversed paths in a graph** (the suffix of T^R)

ELSEVIER

Contents lists available at ScienceDirect

Theoretical Computer Science

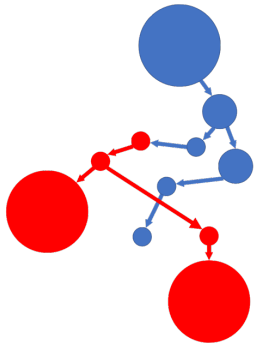
www.elsevier.com/locate/tcs

Wheeler graphs: A framework for BWT-based data structures ☆

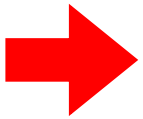
Travis Gagie [a], Giovanni Manzini [b,c,*], Jouni Sirén [d]

2017

Our Contribution



WGT
Wheeler Graph Toolkit

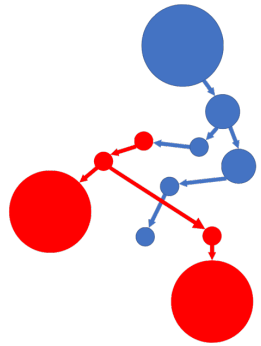


- ***A Wheeler graph visualizer: bipartite representation***

- Implemented the first Wheeler graph *recognizer*

- Wheeler graph *generators*: Random WG / Trie / DBG / RDG

Our Contribution



WGT
Wheeler Graph Toolkit

- A Wheeler graph visualizer: bipartite representation

- ***Implemented the first Wheeler graph recognizer***

- Wheeler graph *generators*: Random WG / Trie / DBG / RDG

Potential application of Wheeler graph ?

```
G A C G T A - C T G
G A C G T A - - - G
G A T G T A - C T G
G A C - T A C C T G
```

IEEE/ACM TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS, VOL. 11, NO. 2, MARCH/APRIL 2014

Indexing Graphs for Path Queries with Applications in Genome Research

Jouni Sirén, Niko Välimäki, and Veli Mäkinen 2014

De Bruijn Graph

=> All are WG ✅

GCSA Reverse deterministic graph

=> Not always WG ⛔

(1) genome coordinates are collapsed.

De Bruijn Graph

K = 4

GACGTA-CTG

GACGTA---G

GATGTA-CTG

GAC-TACCTG

De Bruijn Graph

K = 4

```
GACGTA-CTG
GACGTA---G
GATGTA-CTG
GAC-TACCTG
```

Remove gap &
append $
at the end of the alignment

K = 4

De Bruijn Graph

GACGTACTG$

GACGTAG$

GATGTACTG$

GACTACCTG$

Remove gap &
append $
at the end of the alignment

De Bruijn Graph

K = 4

GACGTACTG$

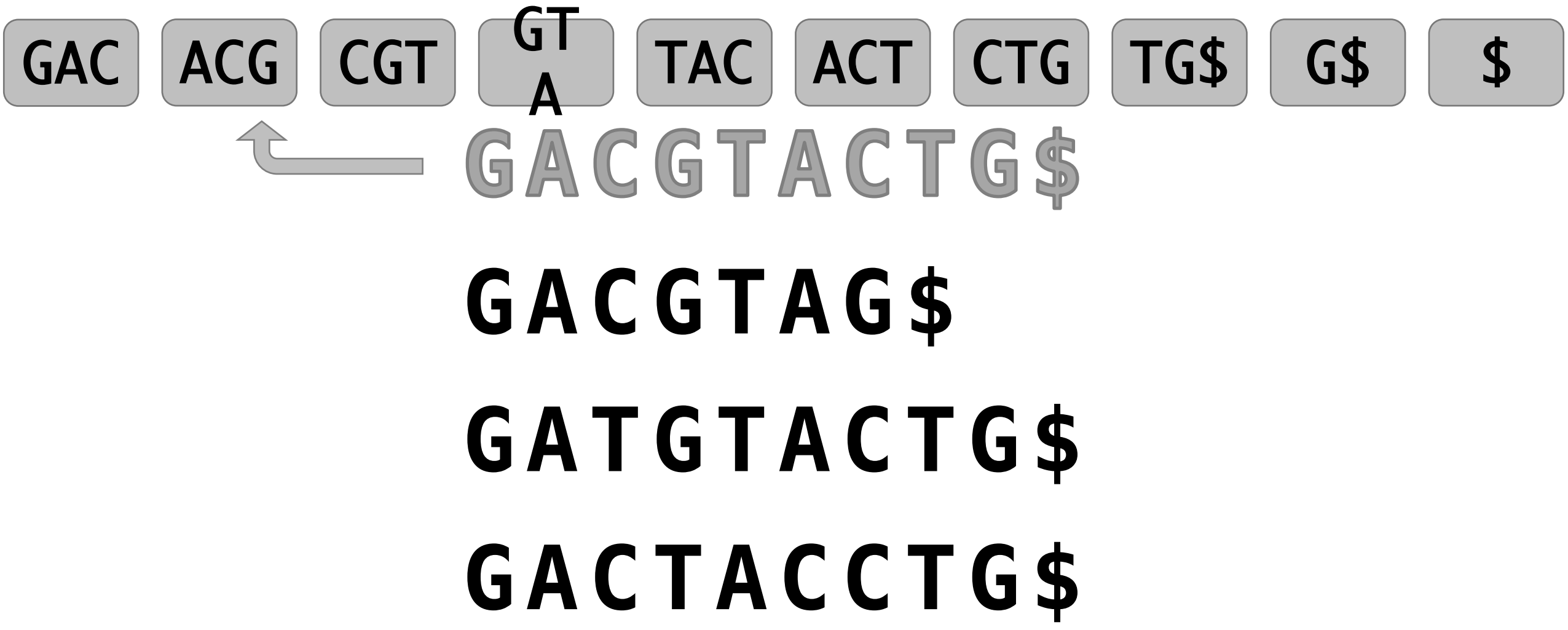
GACGTAG$

GATGTACTG$

GACTACCTG$

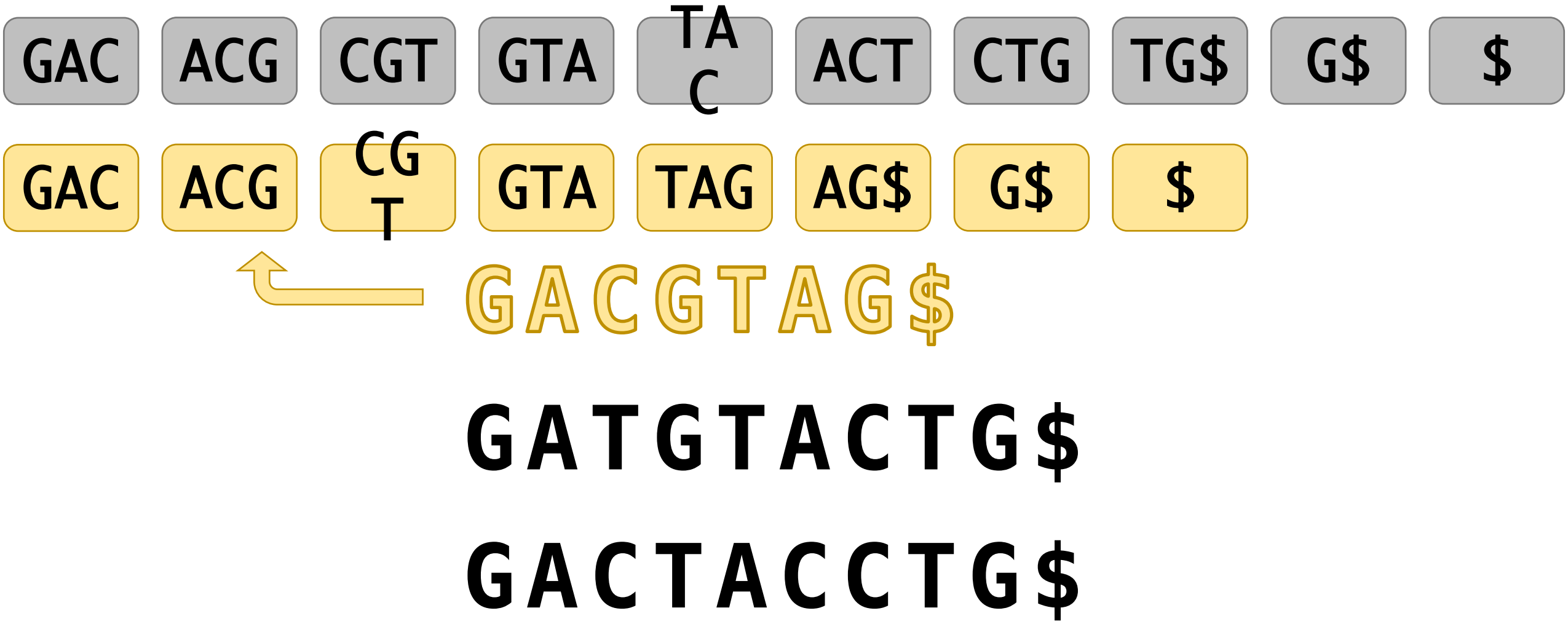
K = 4

De Bruijn Graph



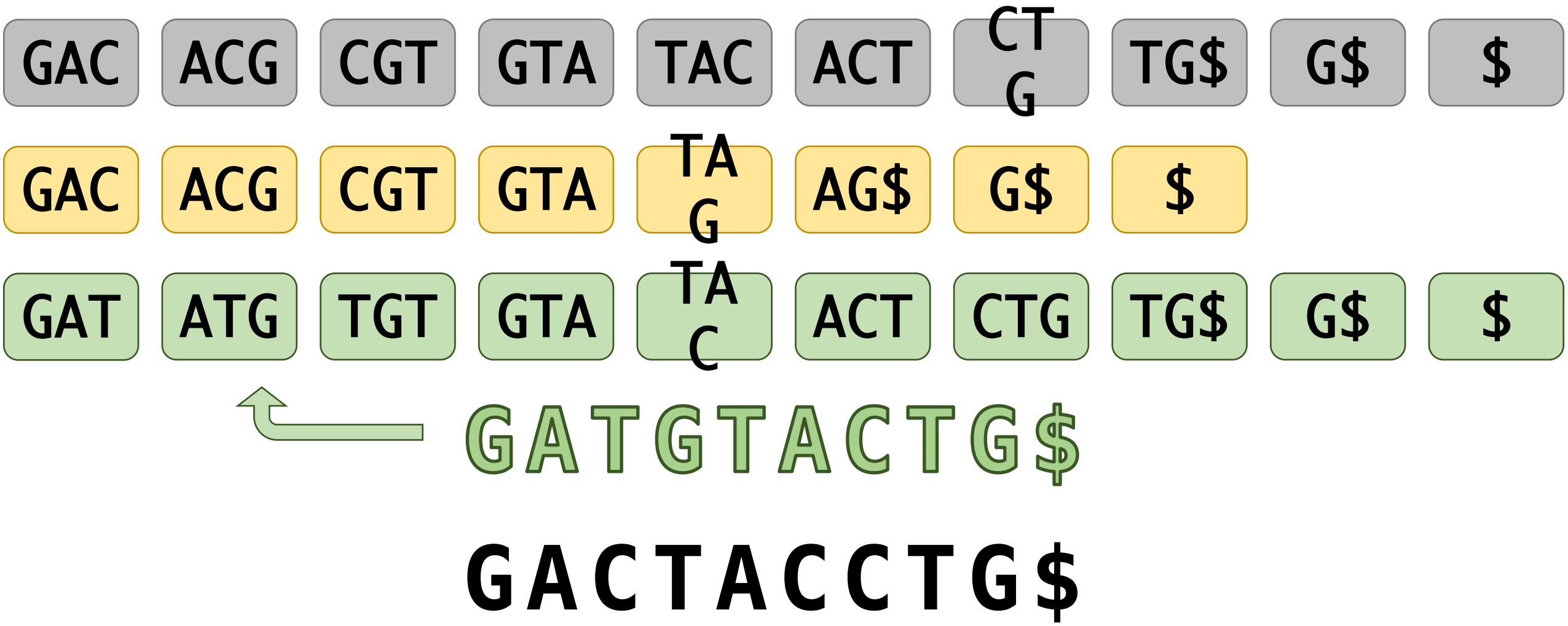
K = 4

De Bruijn Graph



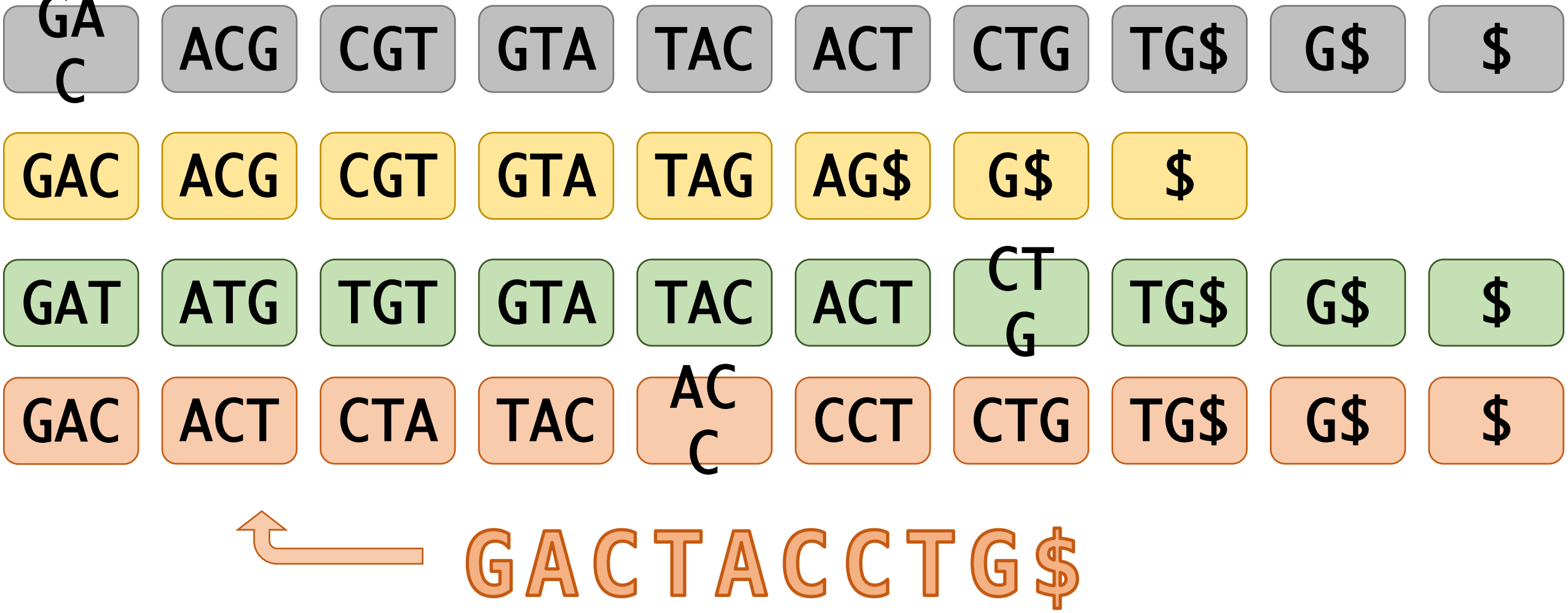
K = 4

De Bruijn Graph



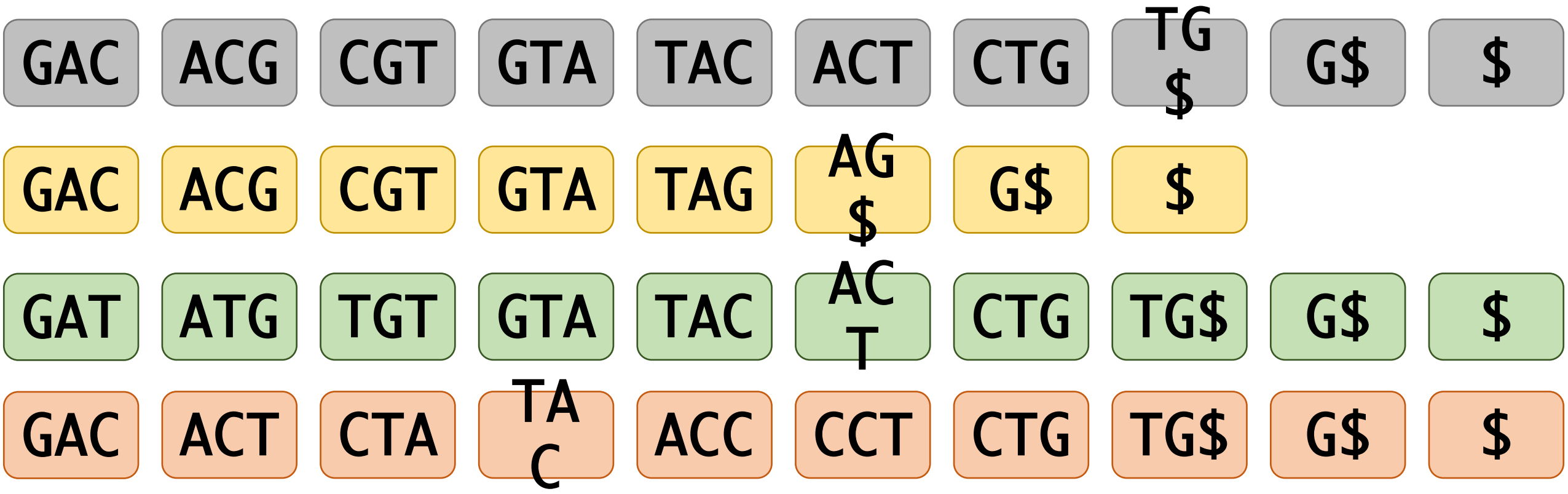
K = 4

De Bruijn Graph



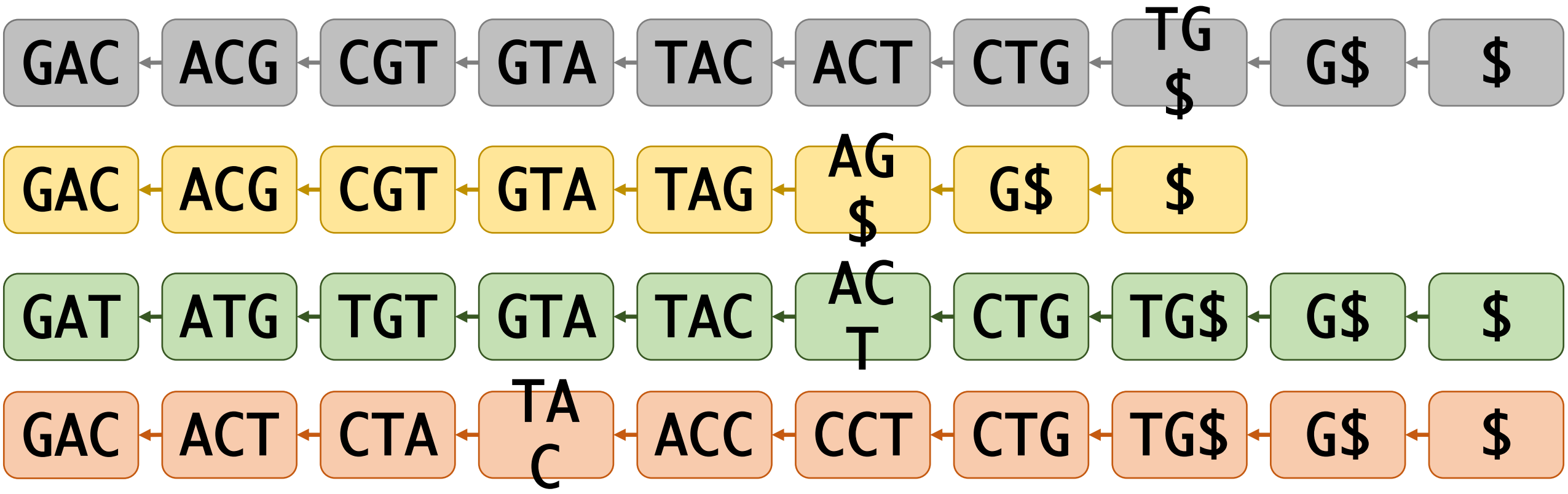
K = 4

De Bruijn Graph



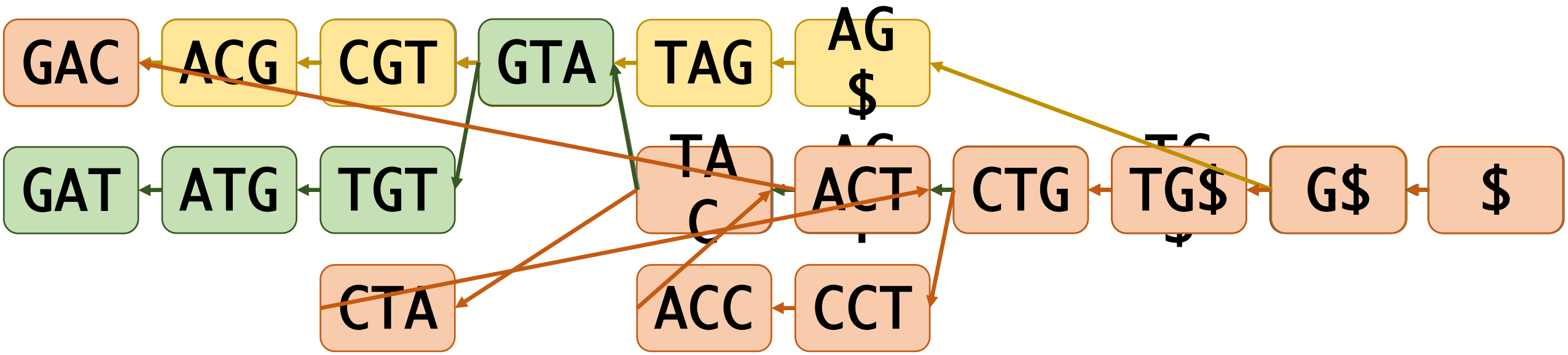
K = 4

De Bruijn Graph



K = 4

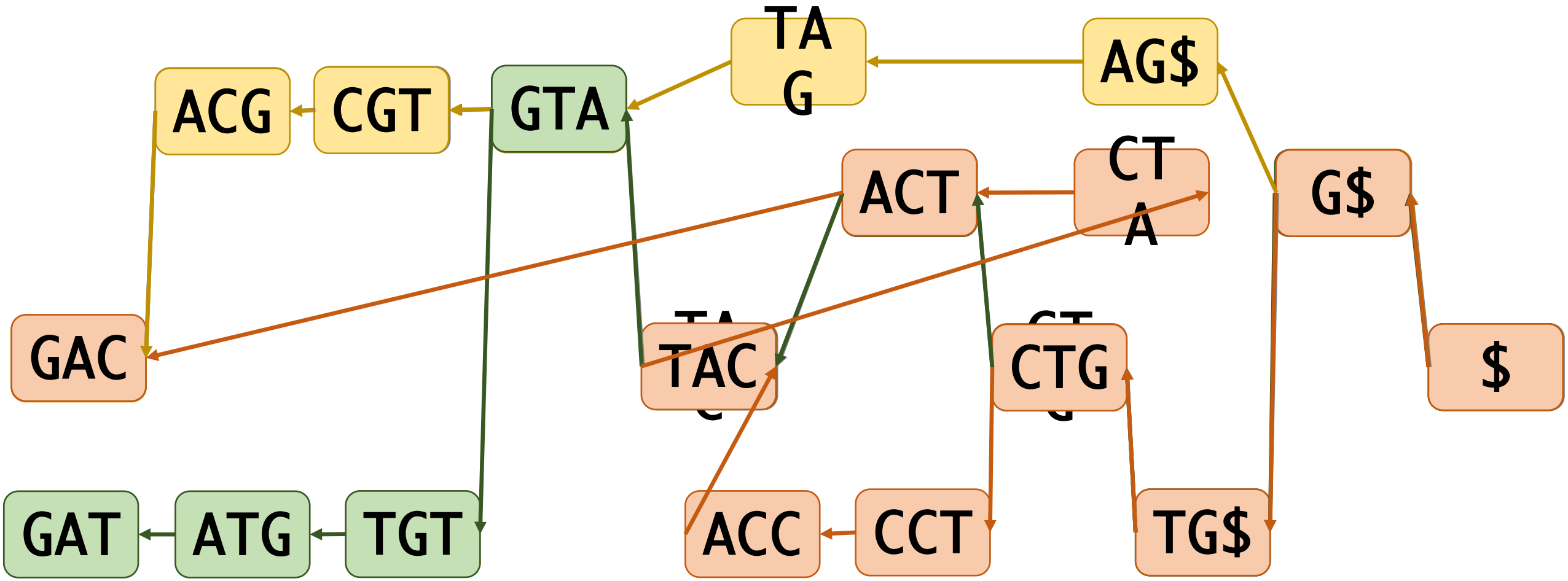
De Bruijn Graph



Genome coordinates are collapsed.

K = 4

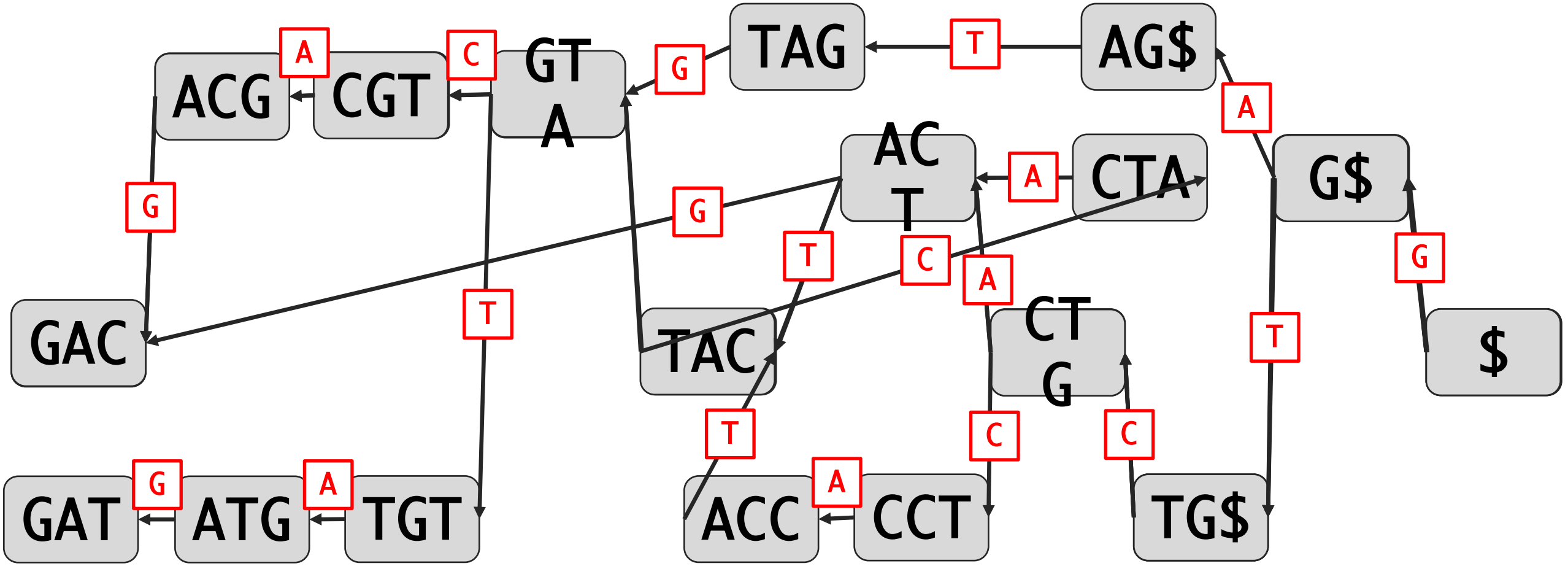
De Bruijn Graph



Genome coordinates are collapsed.

K = 4

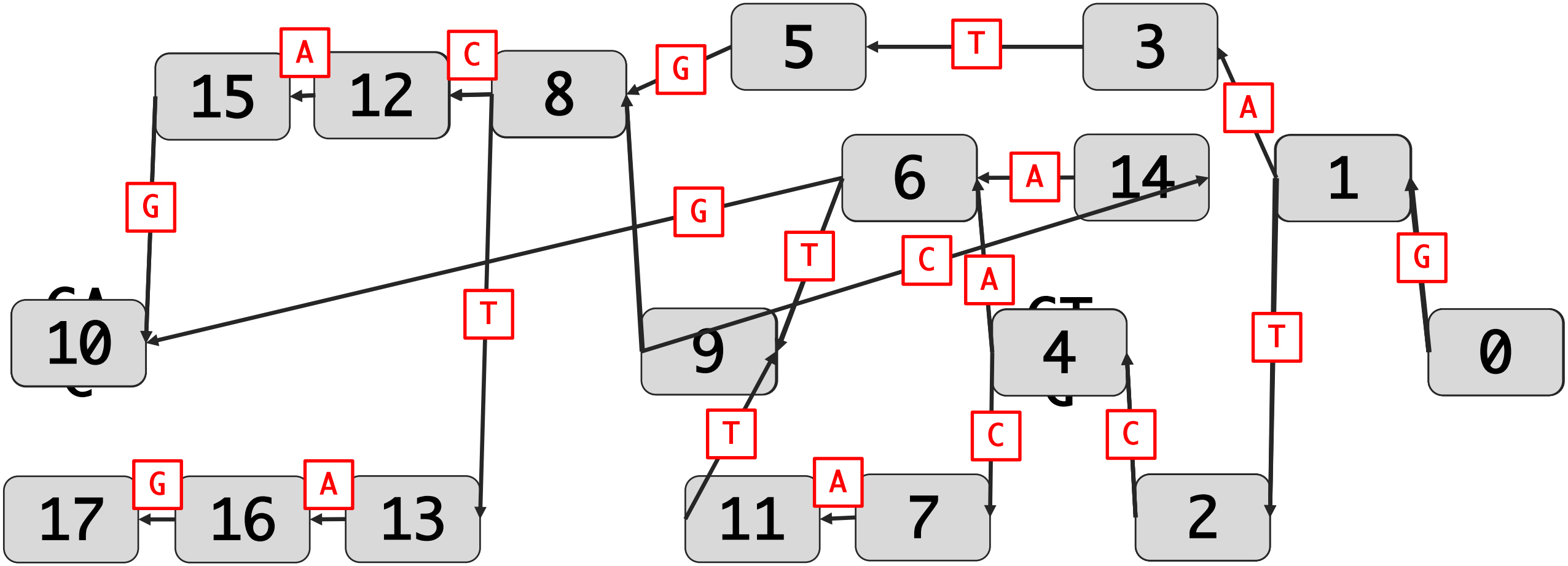
De Bruijn Graph



Genome coordinates are collapsed.

K = 4

De Bruijn Graph



Genome coordinates are collapsed.

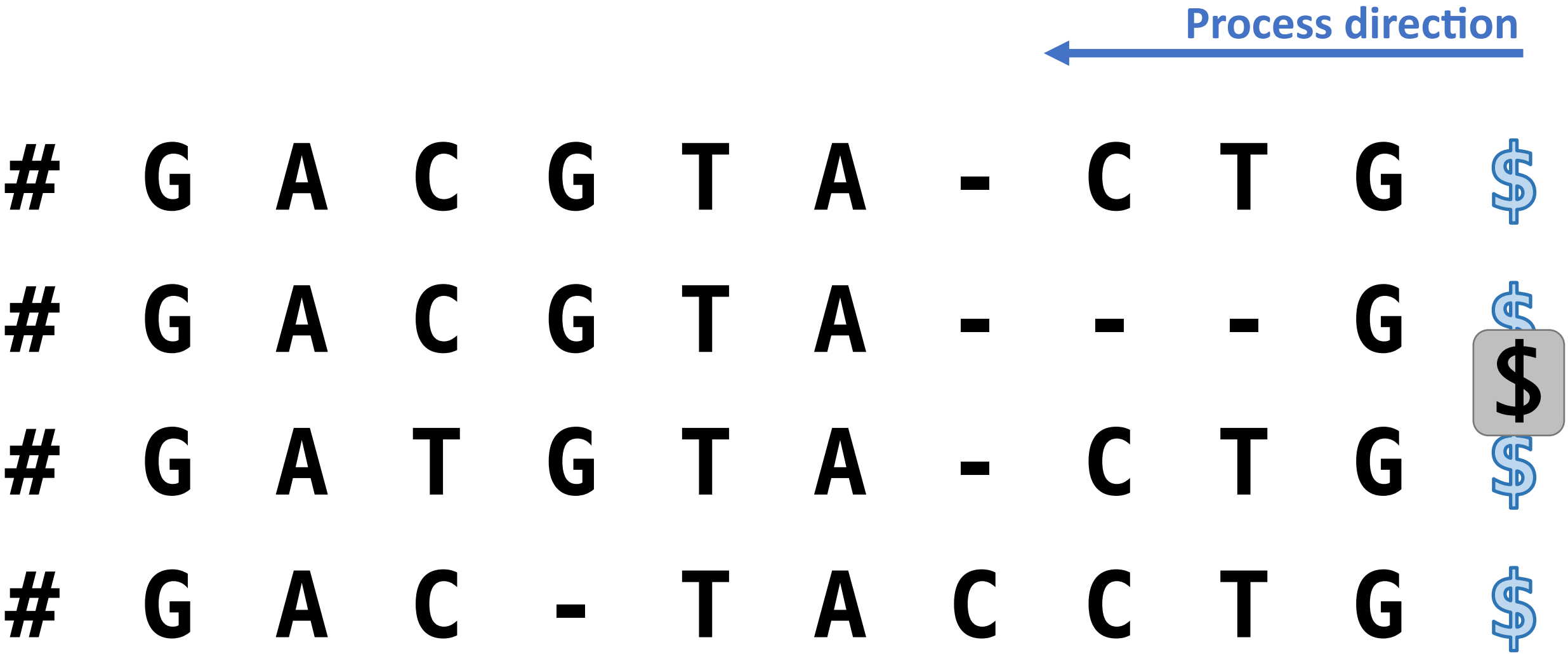
GCSA Reverse Deterministic Graph

```
G A C G T A - C T G
G A C G T A - - - G
G A T G T A - C T G
G A C - T A C C T G
```

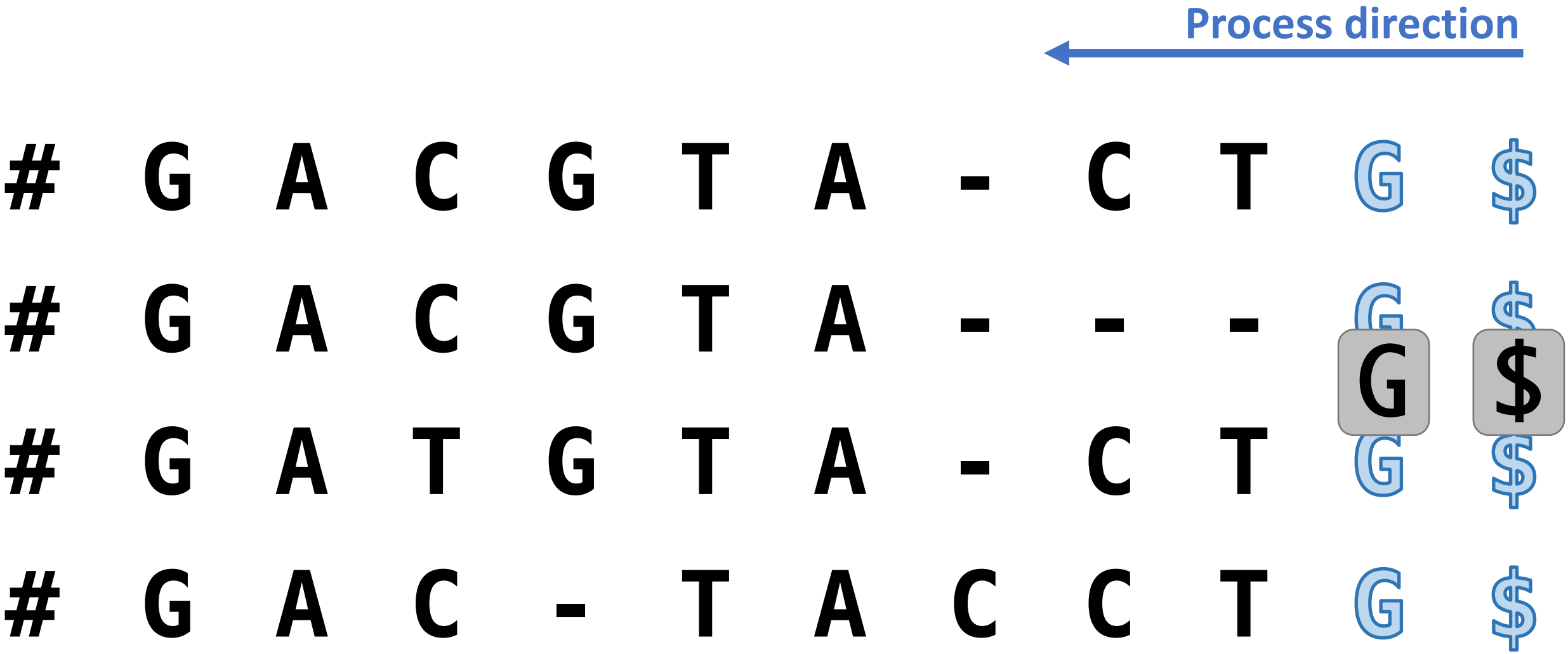
GCSA Reverse Deterministic Graph


```
#  G  A  C  G  T  A  -  C  T  G  $
#  G  A  C  G  T  A  -  -  -  G  $
#  G  A  T  G  T  A  -  C  T  G  $
#  G  A  C  -  T  A  C  C  T  G  $
```

GCSA Reverse Deterministic Graph



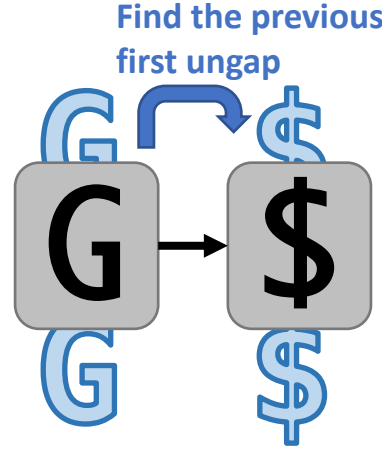
GCSA Reverse Deterministic Graph



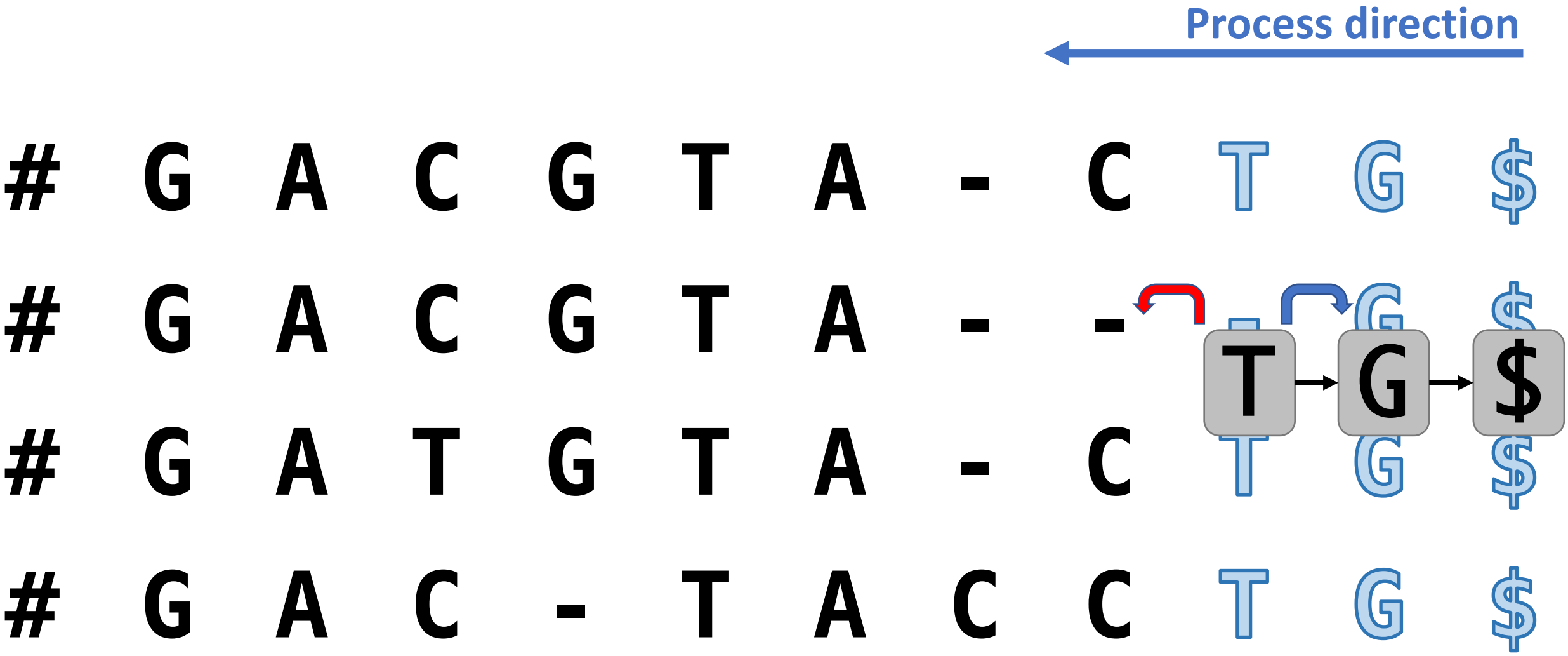
GCSA Reverse Deterministic Graph

Process direction

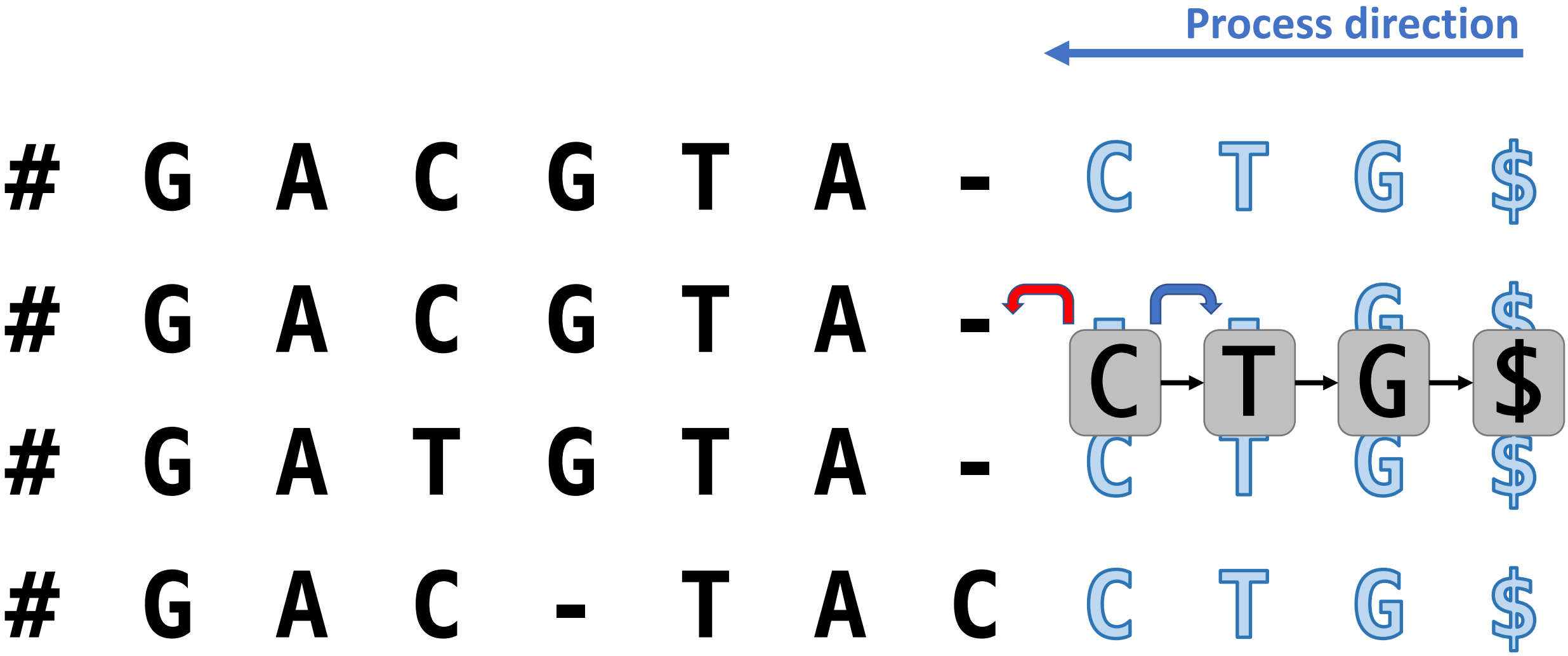
```
# G A C G T A - C T G $
# G A C G T A - - - G $
# G A T G T A - C T G $
# G A C - T A C C T G $
```



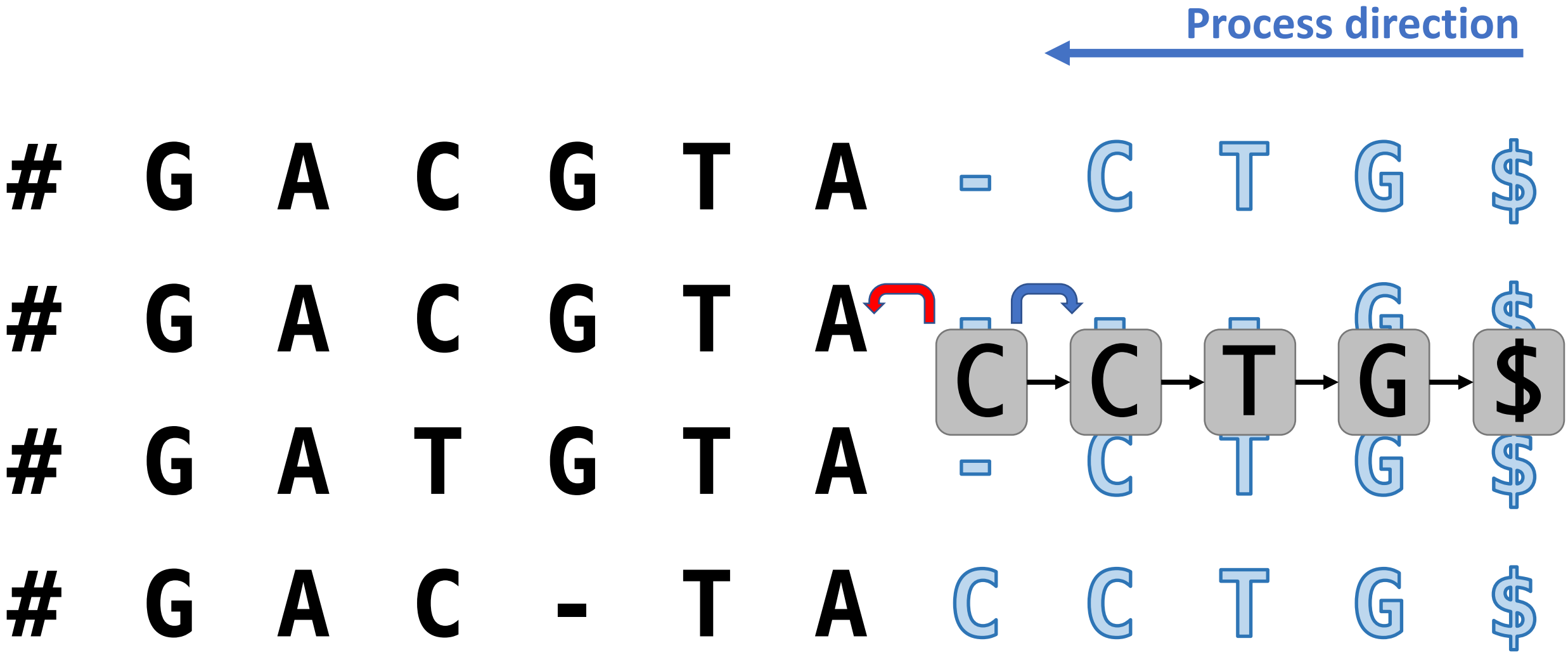
GCSA Reverse Deterministic Graph



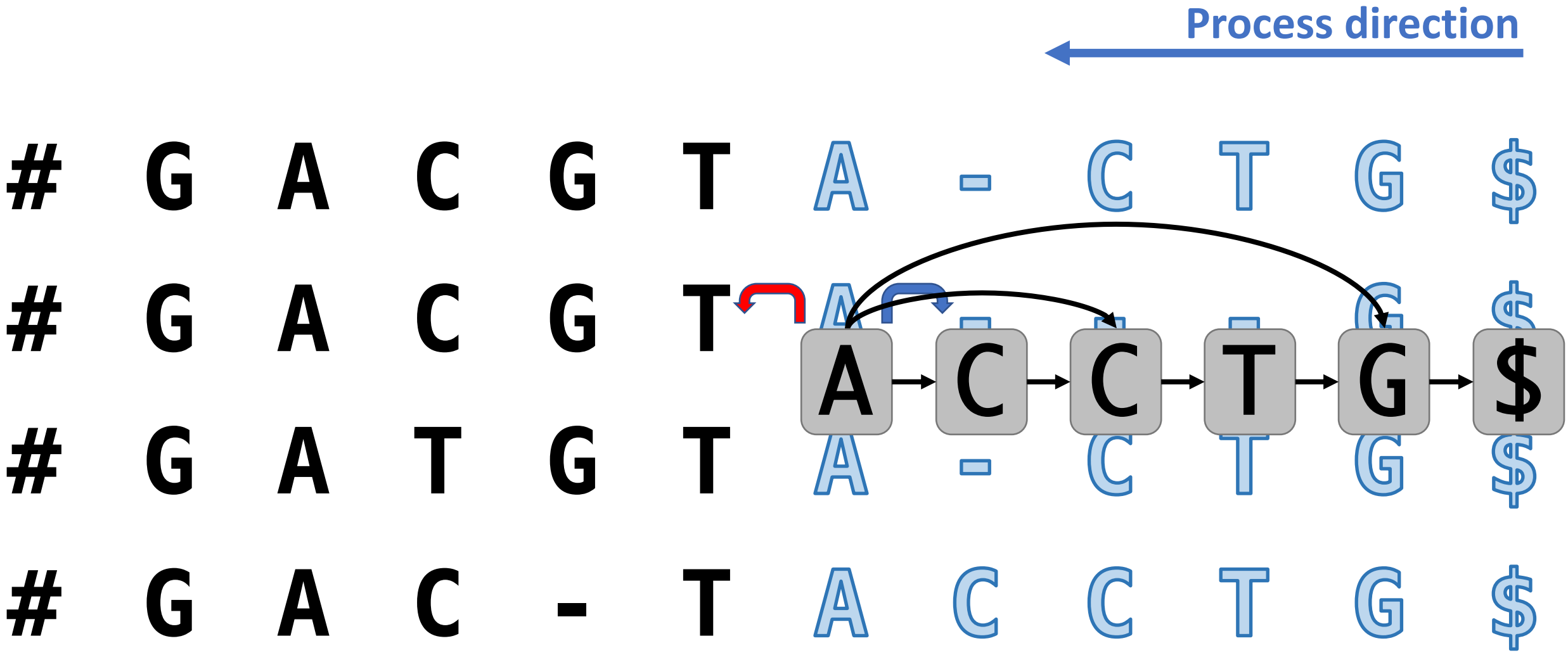
GCSA Reverse Deterministic Graph



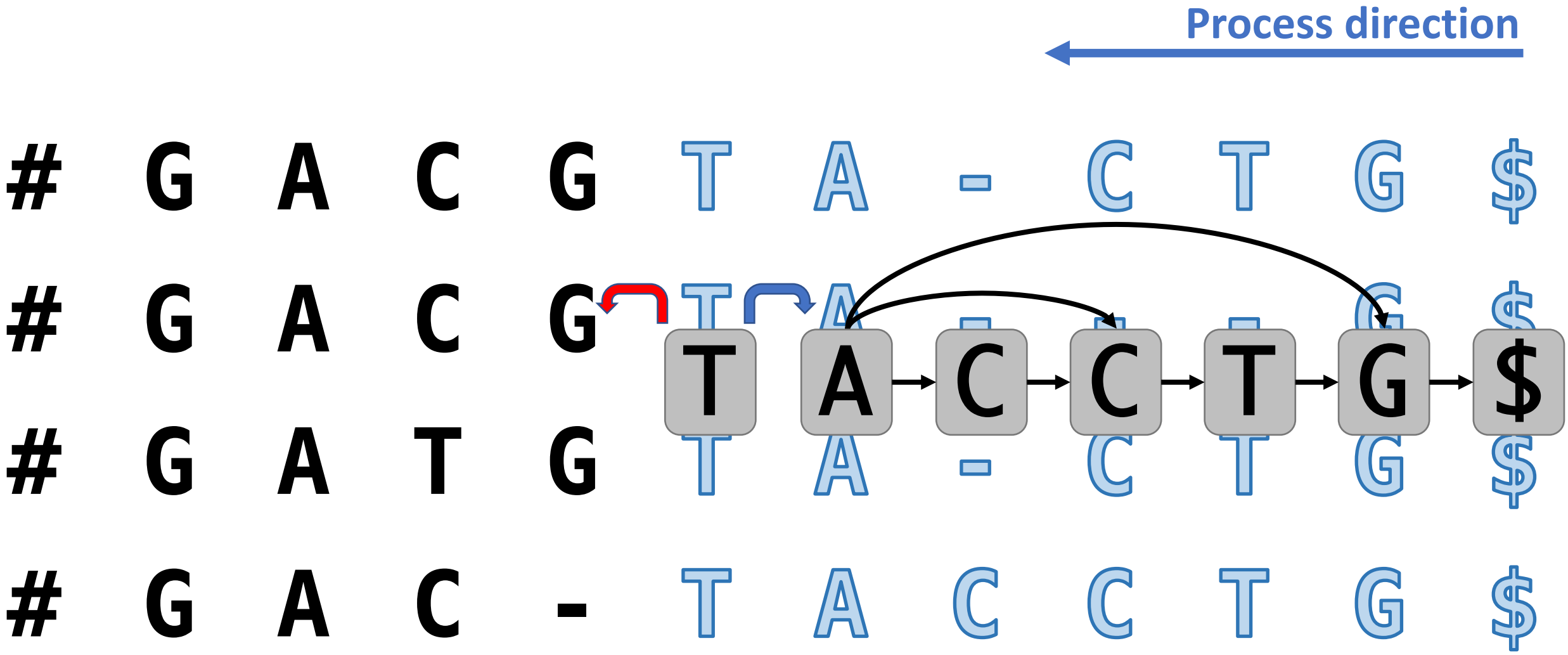
GCSA Reverse Deterministic Graph



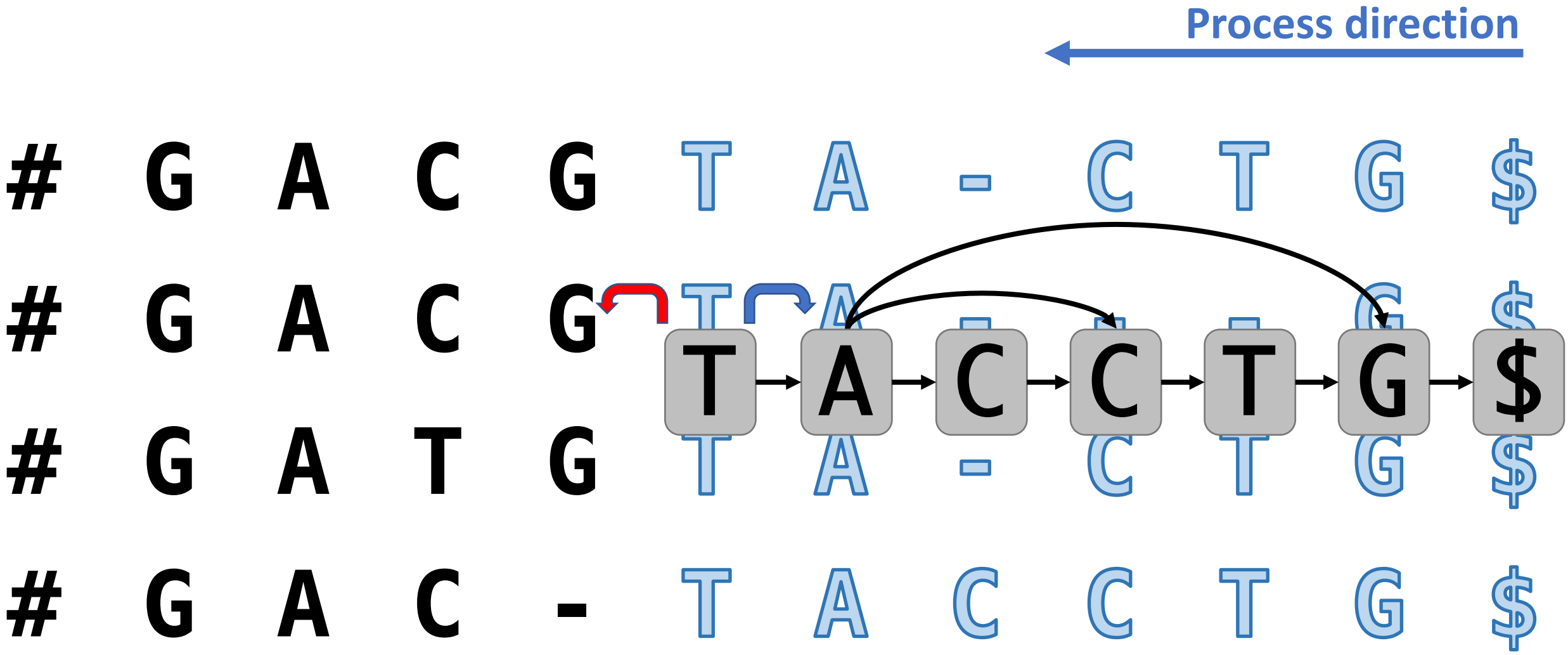
GCSA Reverse Deterministic Graph



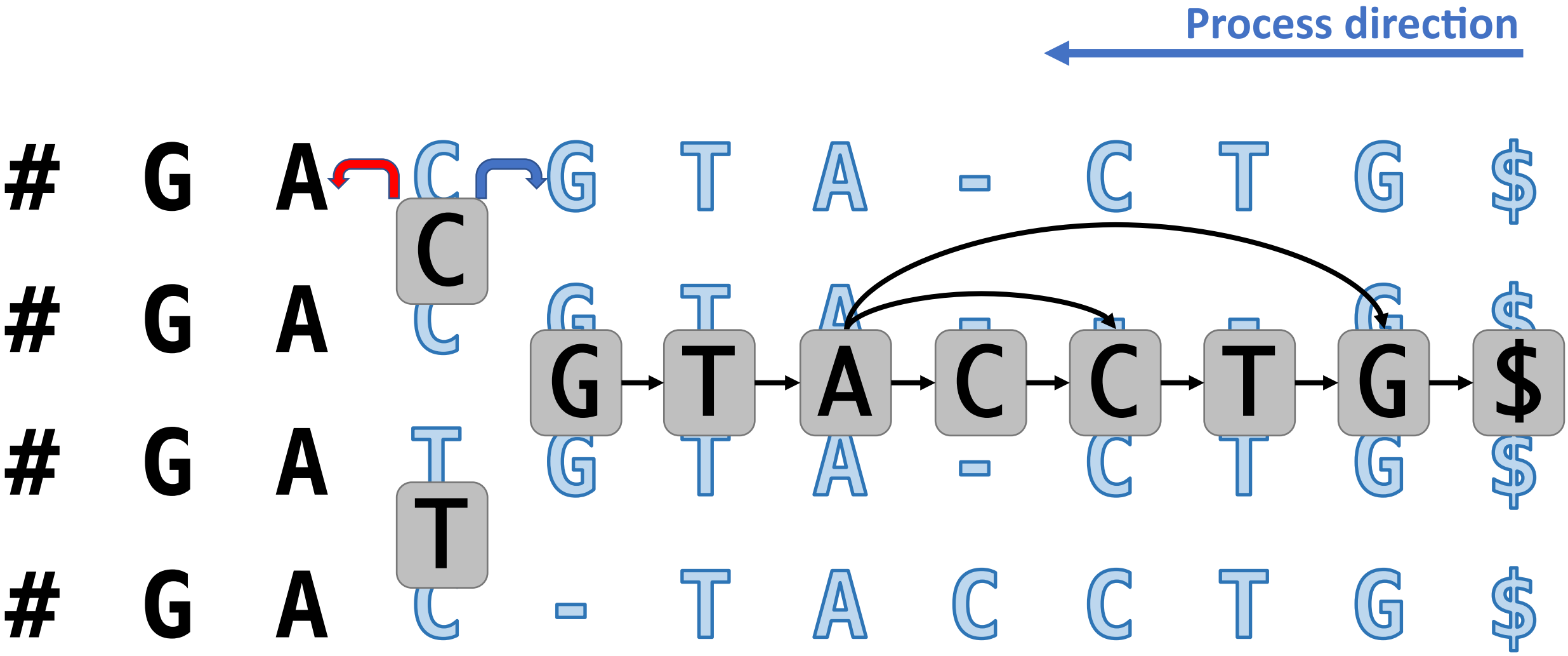
GCSA Reverse Deterministic Graph



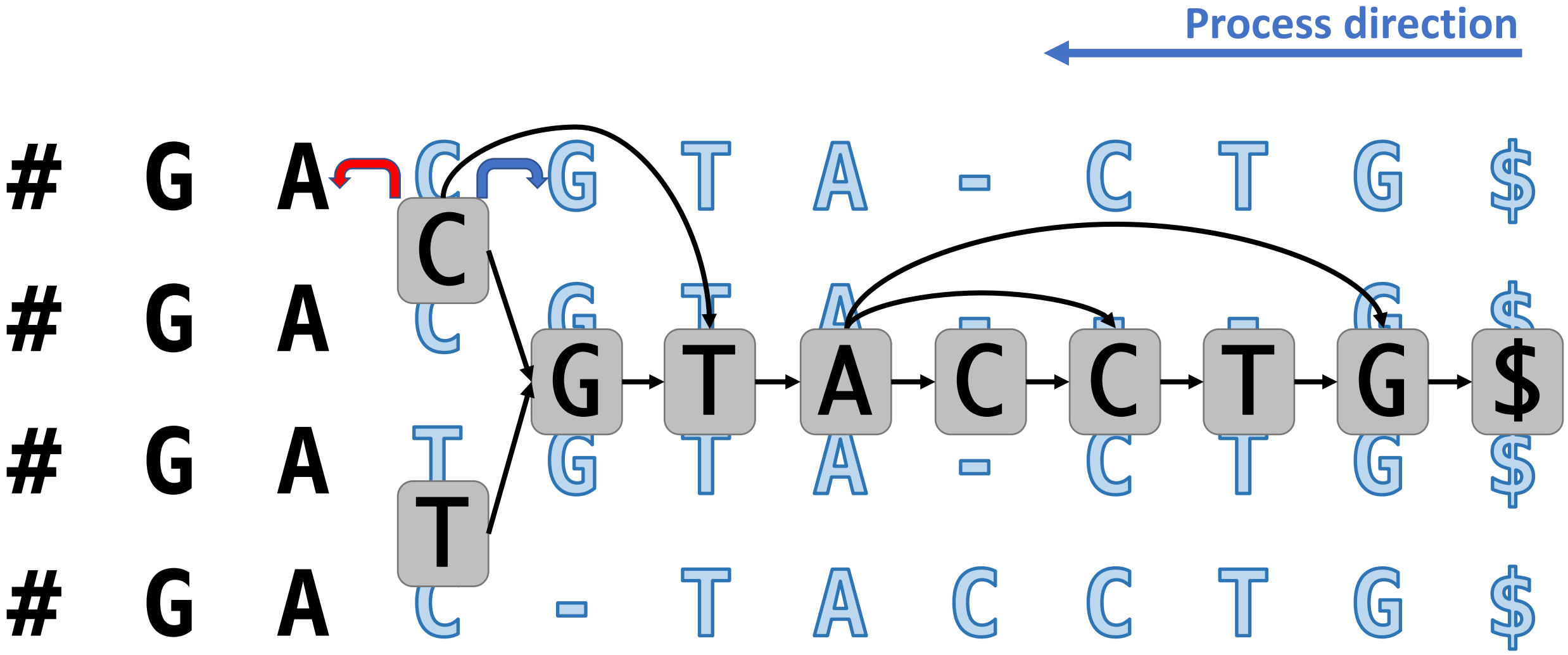
GCSA Reverse Deterministic Graph



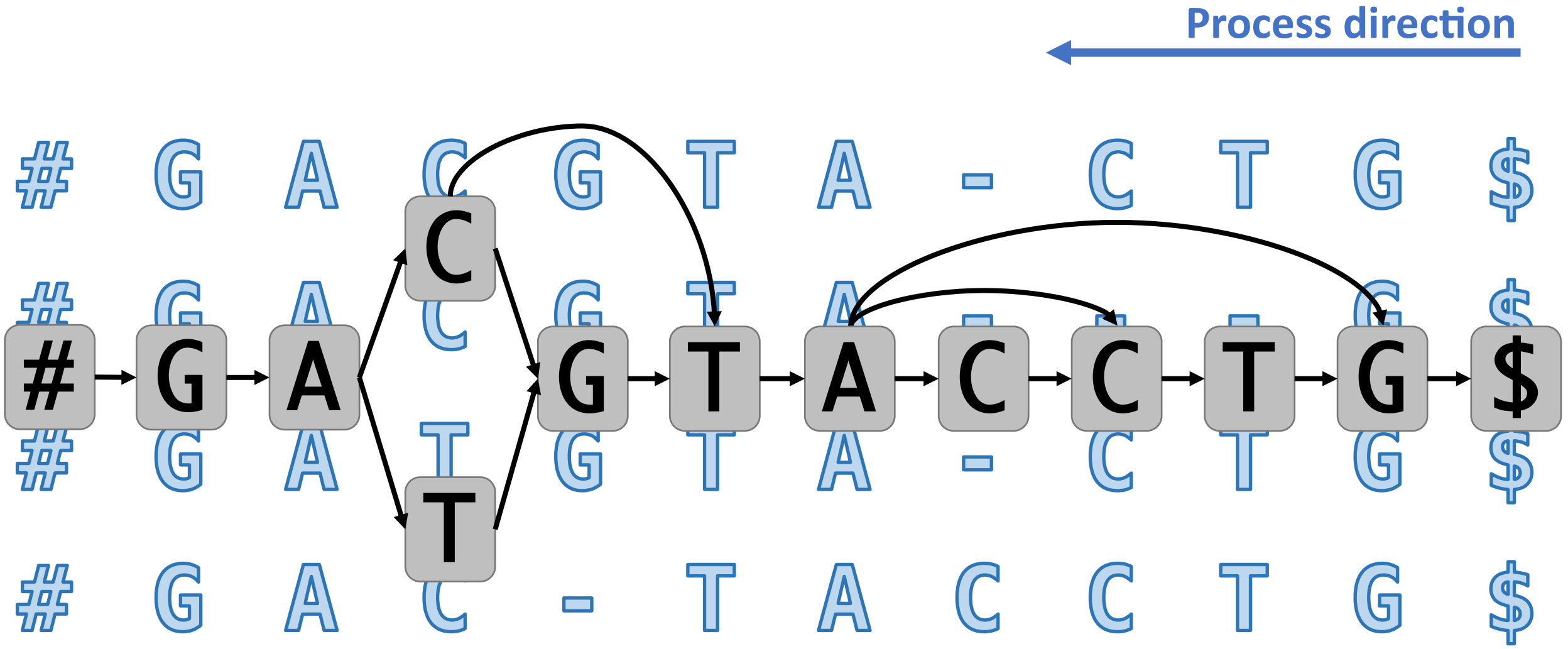
GCSA Reverse Deterministic Graph



GCSA Reverse Deterministic Graph

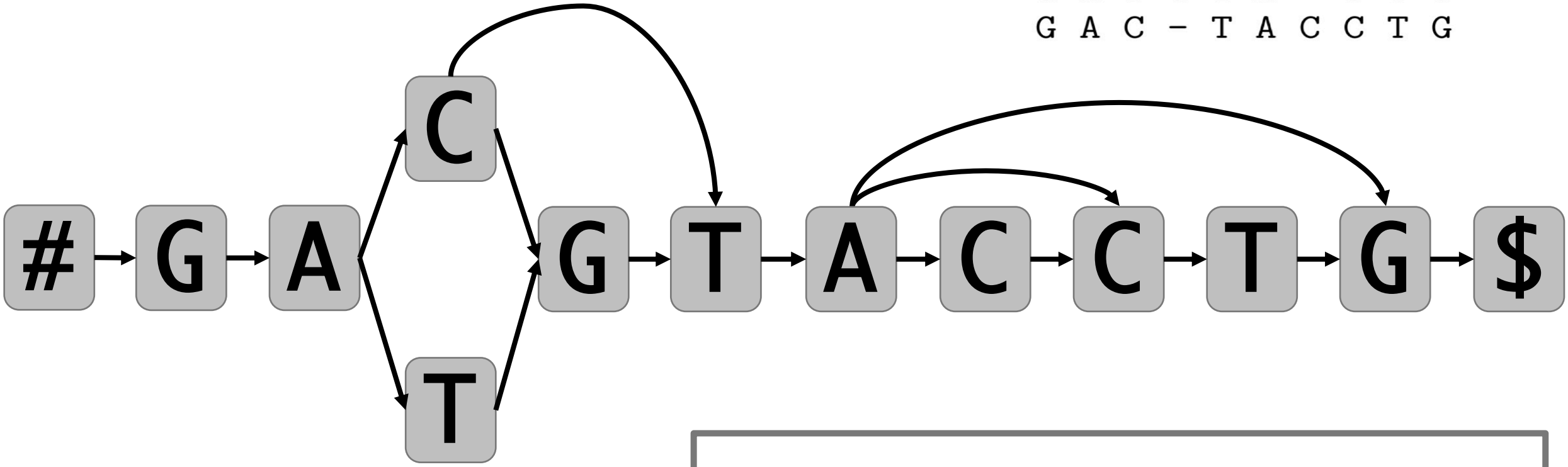


GCSA Reverse Deterministic Graph



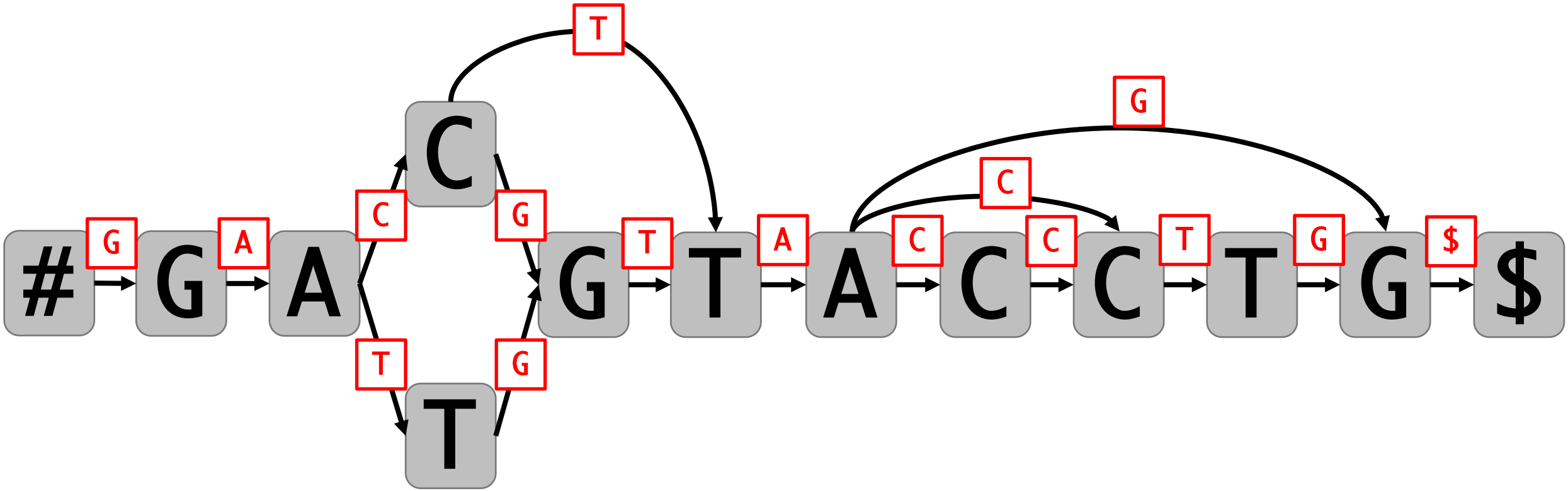
GCSA Reverse Deterministic Graph

```
G A C G T A - C T G
G A C G T A - - - G
G A T G T A - C T G
G A C - T A C C T G
```



Automaton $\mathbf{A}$ is reverse deterministic if, for every node $v \in V$ and every character $c \in \Sigma \cup \{\#, \$\}$, there exists at most one node u such $label(u) = c$ and $(u, v) \in E$

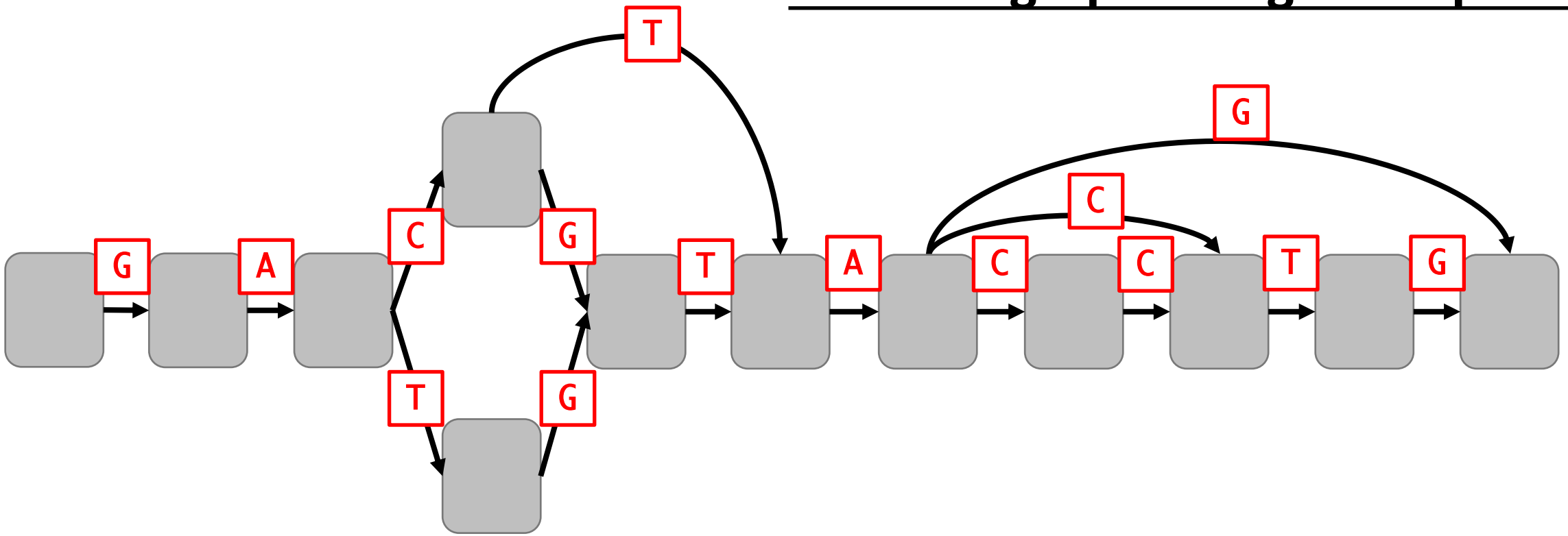
GCSA Reverse Deterministic Graph



Label the edges by the labels of their children.

GCSA Reverse Deterministic Graph

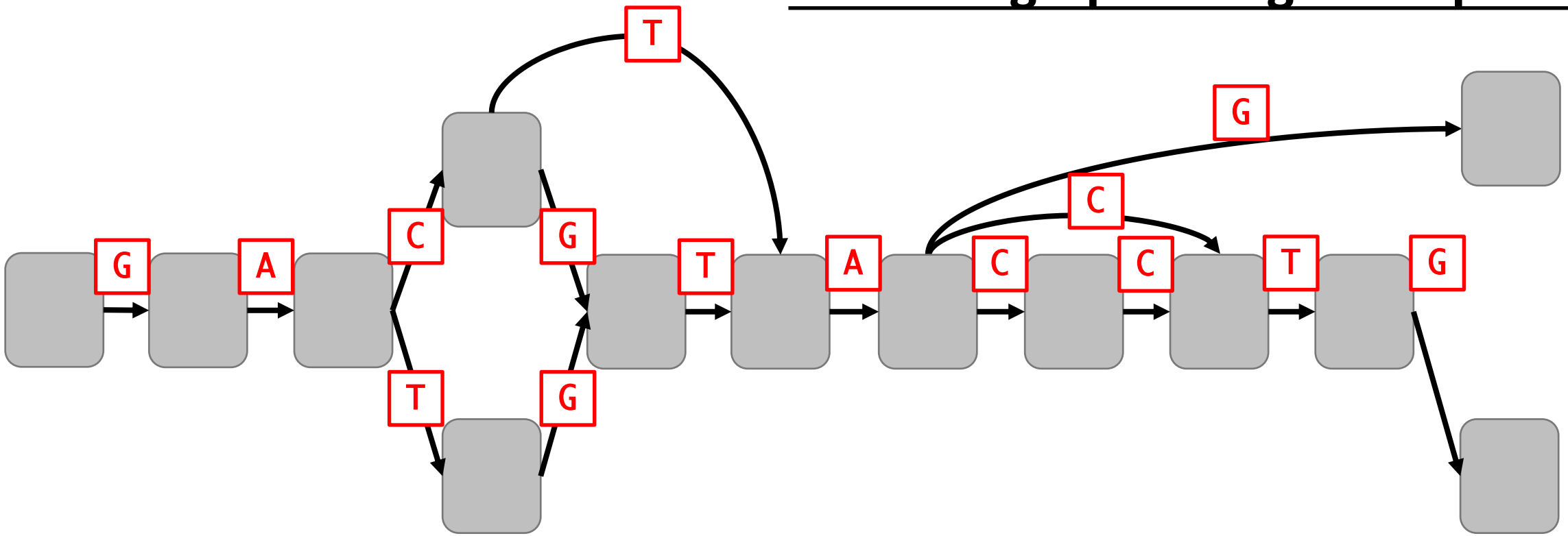
Wheeler graph recognition problem



Is it a Wheeler graph? => ✗ **Wheeler graph.**

GCSA Reverse Deterministic Graph

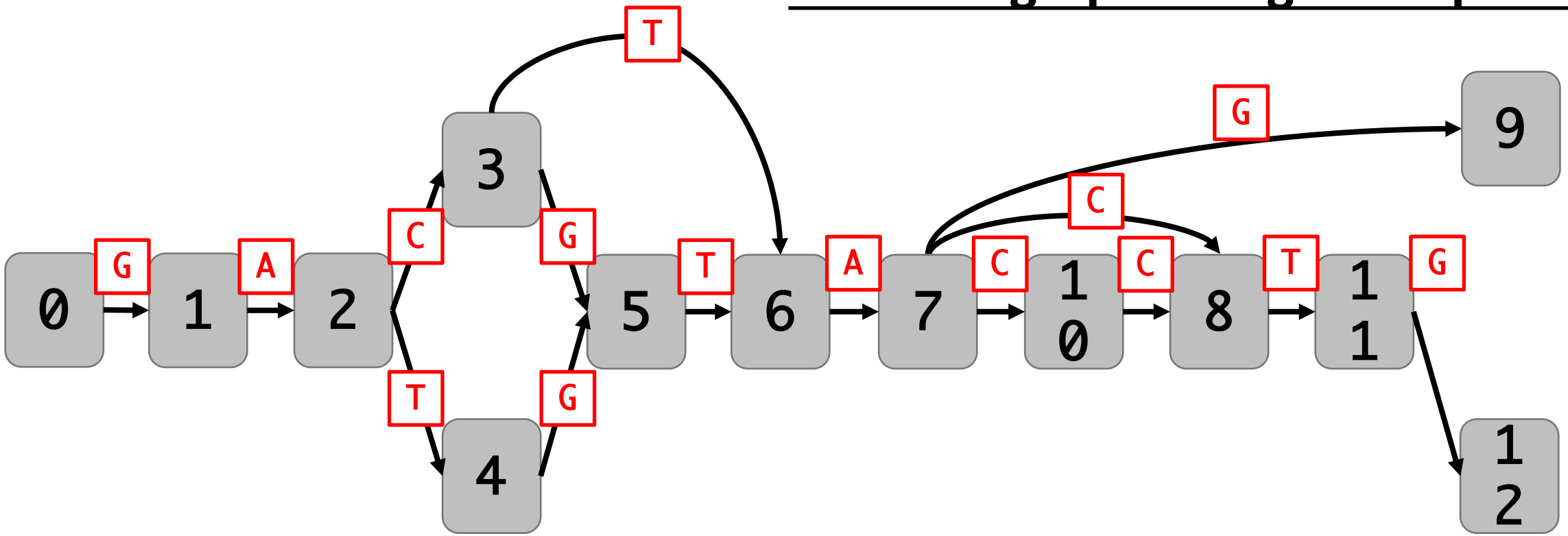
Wheeler graph recognition problem



Is it a Wheeler graph? => ✗ **Wheeler graph.**

GCSA Reverse Deterministic Graph

Wheeler graph recognition problem



Is it a Wheeler graph? => ✅ **Wheeler graph.**

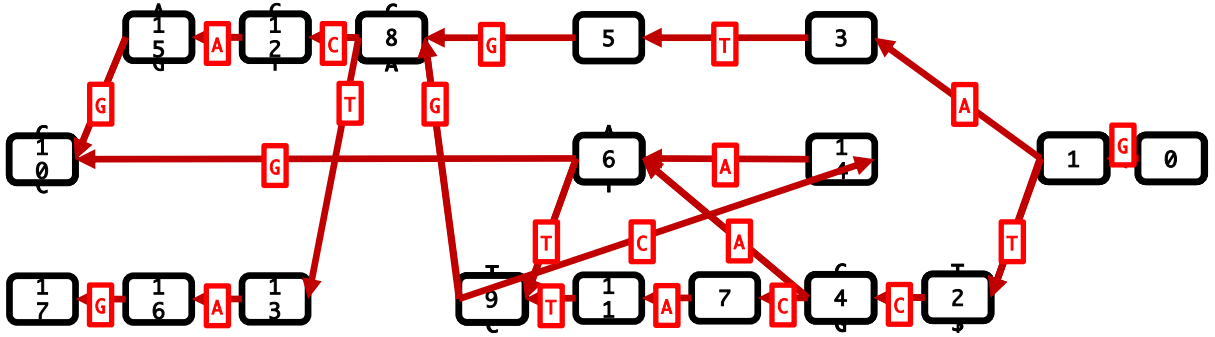
Potential application of Wheeler graph ?

```
G A C G T A - C T G
G A C G T A - - - G
G A T G T A - C T G
G A C - T A C C T G
```

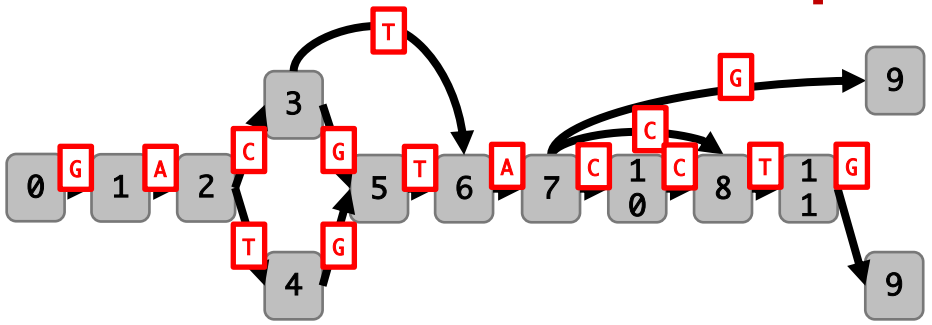
nodes: 18
edges: 21

nodes: 13
edges: 14

De Bruijn Graph

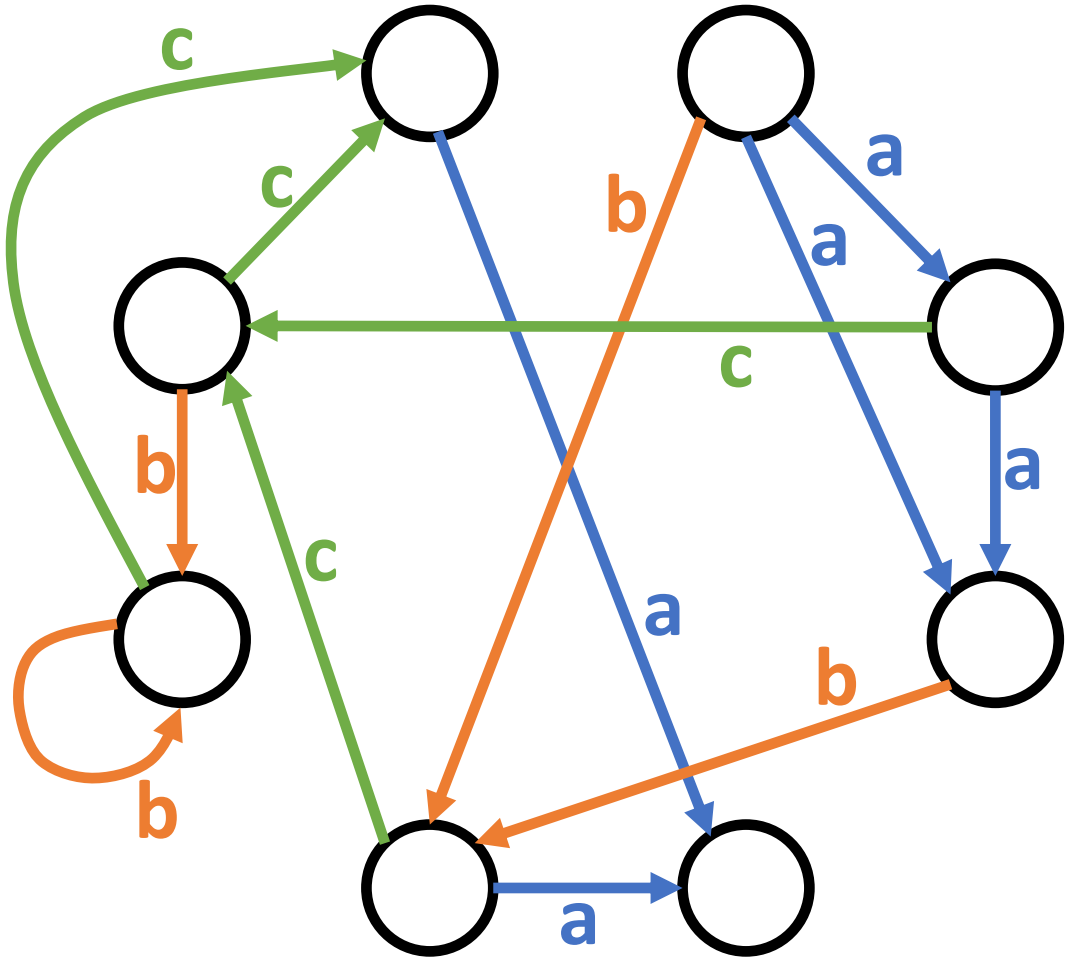


GCSA Reverse Deterministics Graph

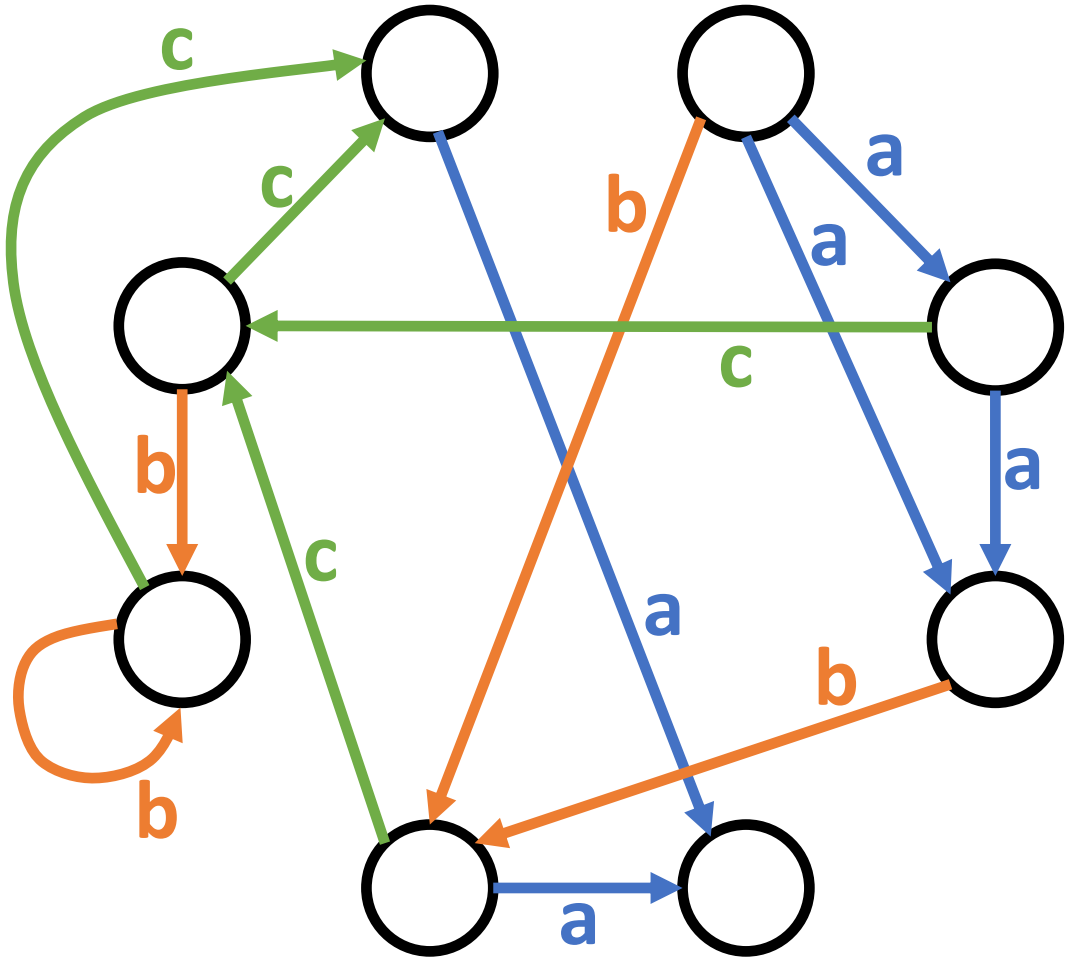


(2) Depending on K, DBG might be bigger than GSCA

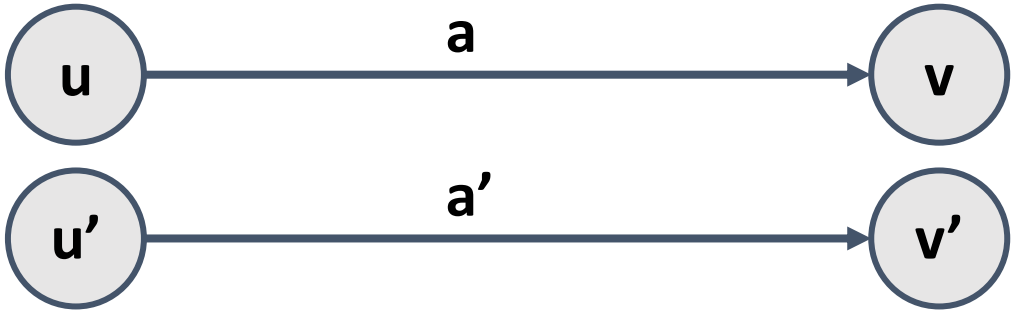
Given an edge-labelled graph, is it a Wheeler graph?



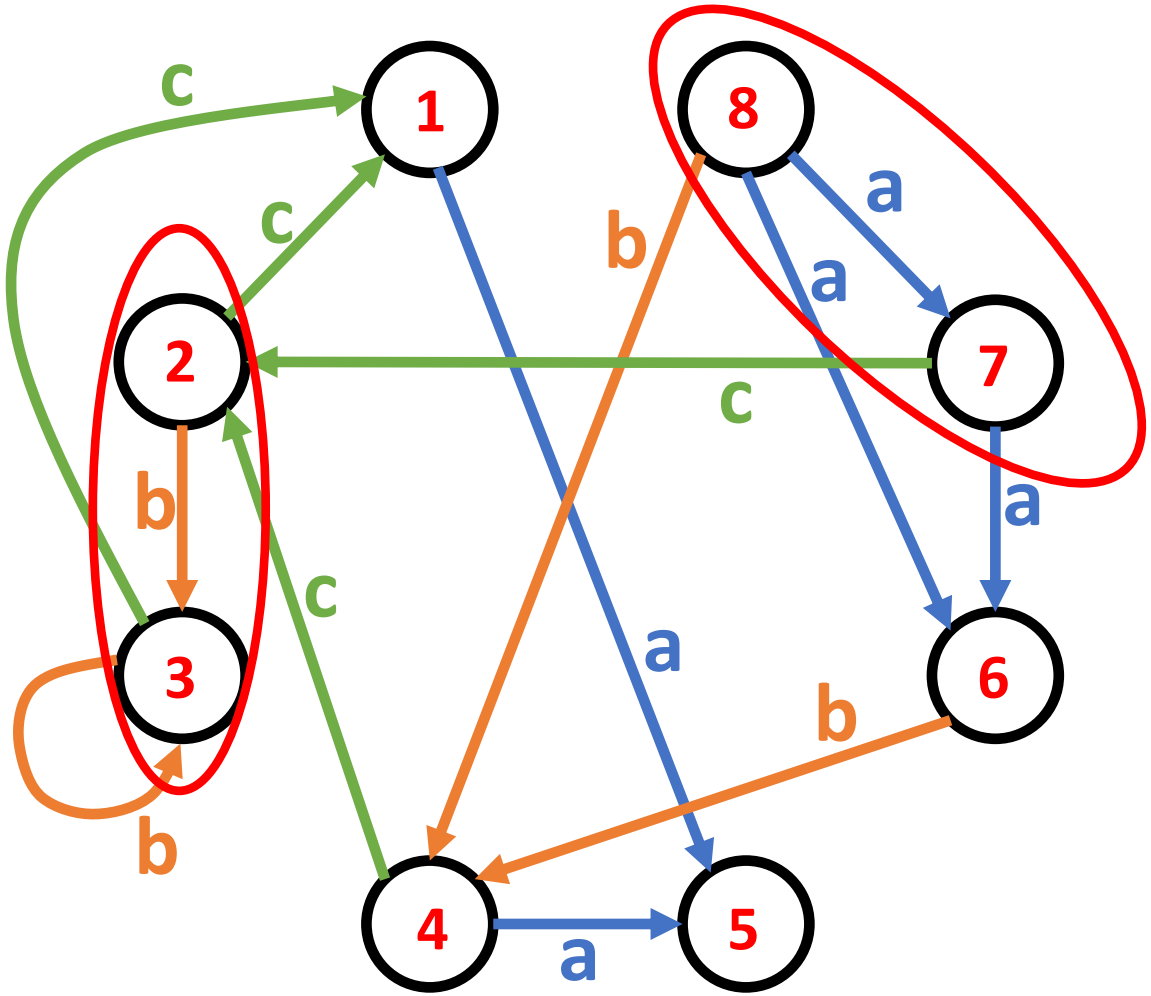
Given an edge-labelled graph, is it a Wheeler graph?



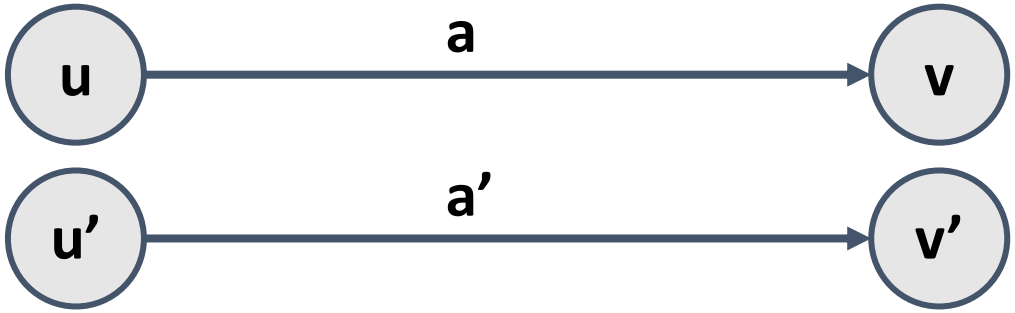
1. Node with indegree 0 comes before every other nodes.
2. $$a \prec a' \implies v < v'$$
3. $$(a = a') \wedge (u < u') \implies v \leq v'$$



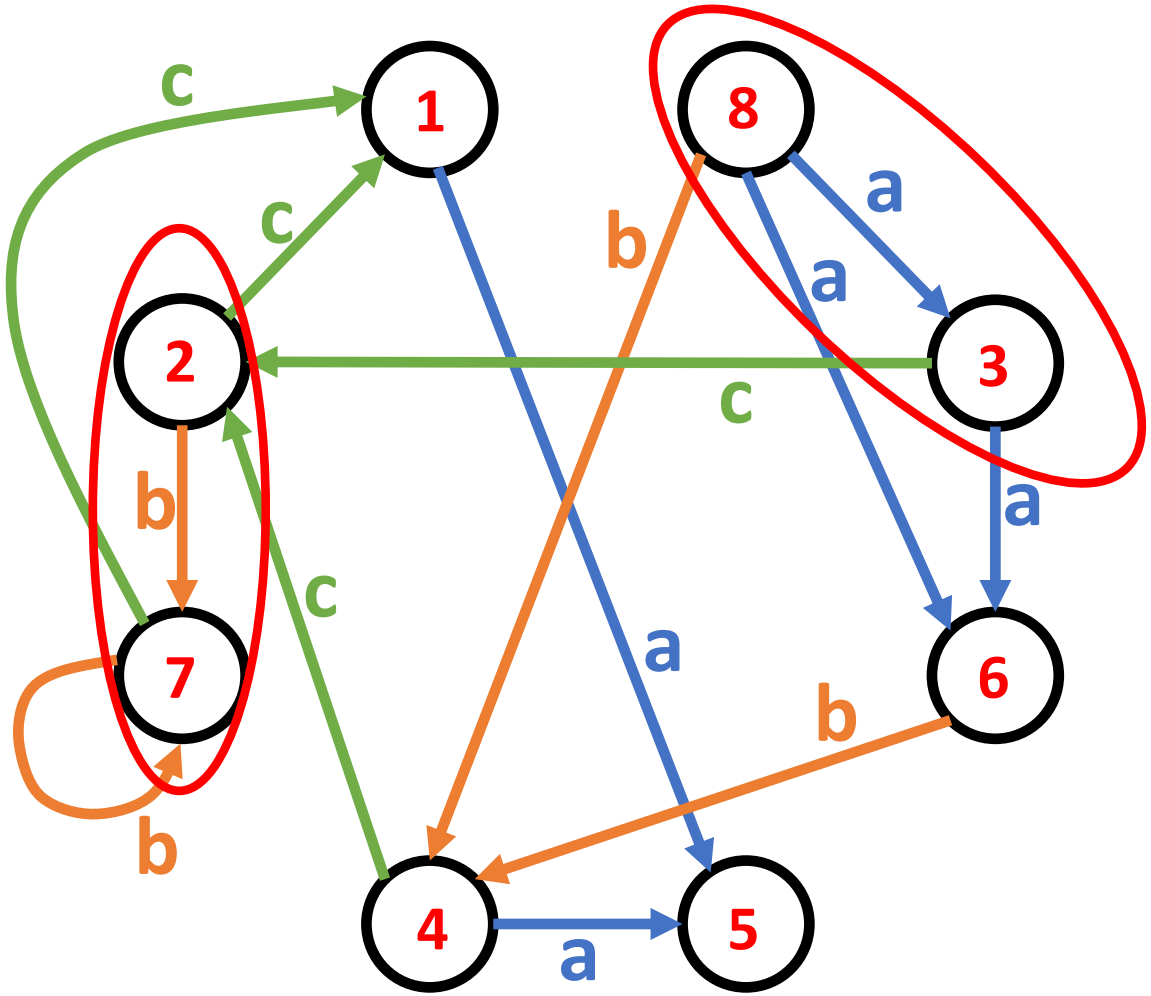
Given an edge-labelled graph, is it a Wheeler graph?



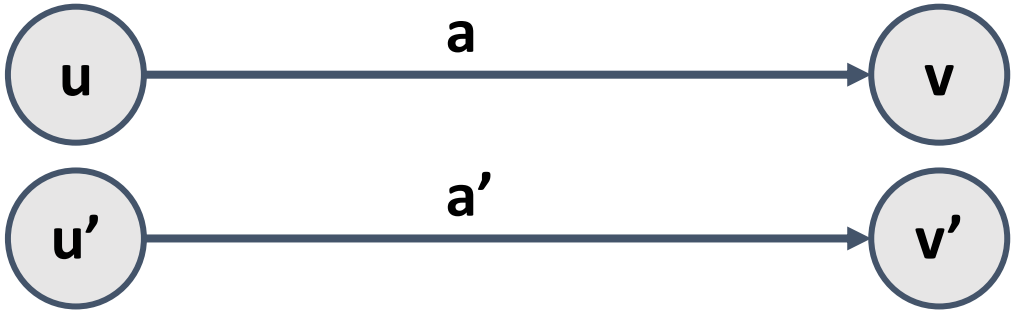
1. Node with indegree 0 comes before every other nodes.
2. $$a \prec a' \implies v < v'$$
3. $$(a = a') \wedge (u < u') \implies v \leq v'$$



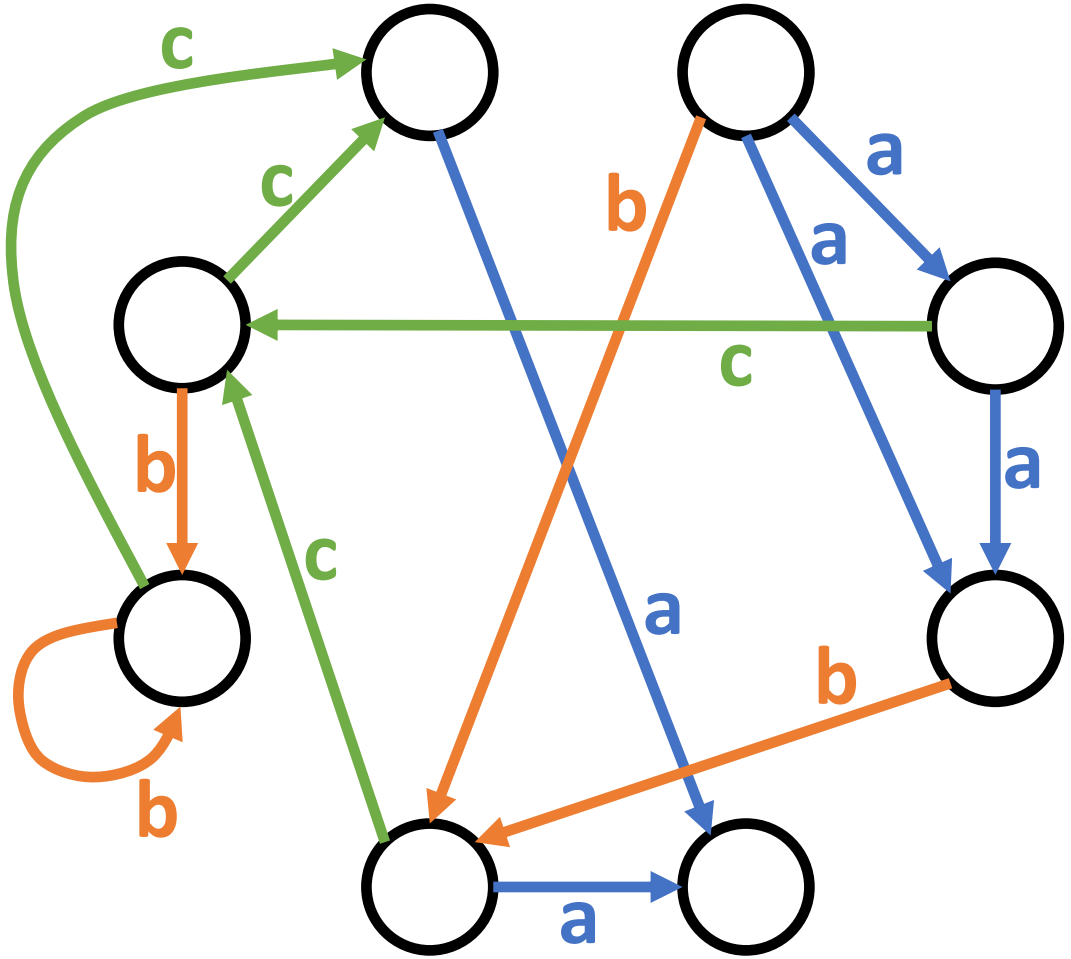
Given an edge-labelled graph, is it a Wheeler graph?



1. Node with indegree 0 comes before every other nodes.
2. $$a \prec a' \implies v < v'$$
3. $$(a = a') \wedge (u < u') \implies v \leq v'$$



Given an edge-labelled graph, is it a Wheeler graph?



If we can come up a valid ordering =>

it is a Wheeler Graph

If not =>

it is not a Wheeler Graph

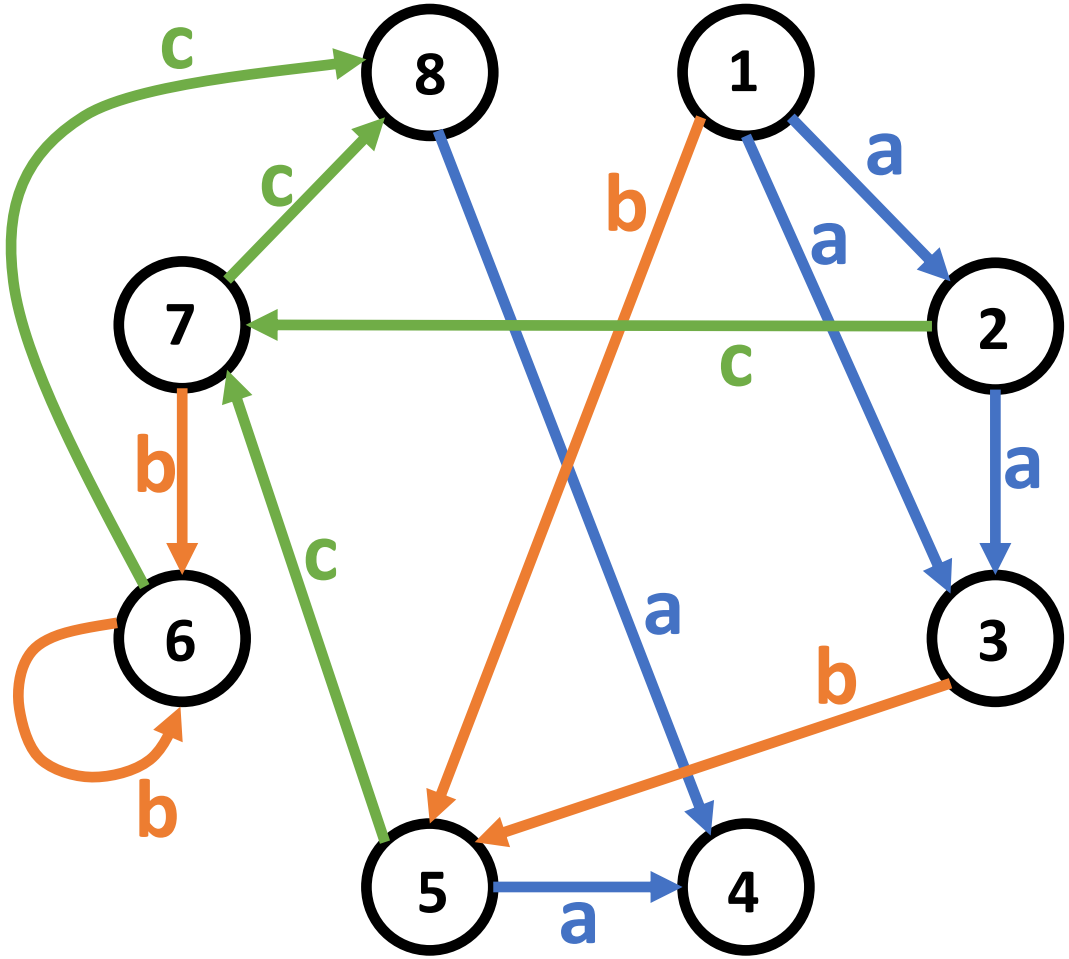
arXiv > cs > arXiv:1902.01960
Search...
Help | Advanc
Computer Science > Computational Complexity
[Submitted on 5 Feb 2019 (v1), last revised 25 Feb 2019 (this version, v2)]

On the Hardness and Inapproximability of Recognizing Wheeler Graphs

Daniel Gibney, Sharma V. Thankachan

2019

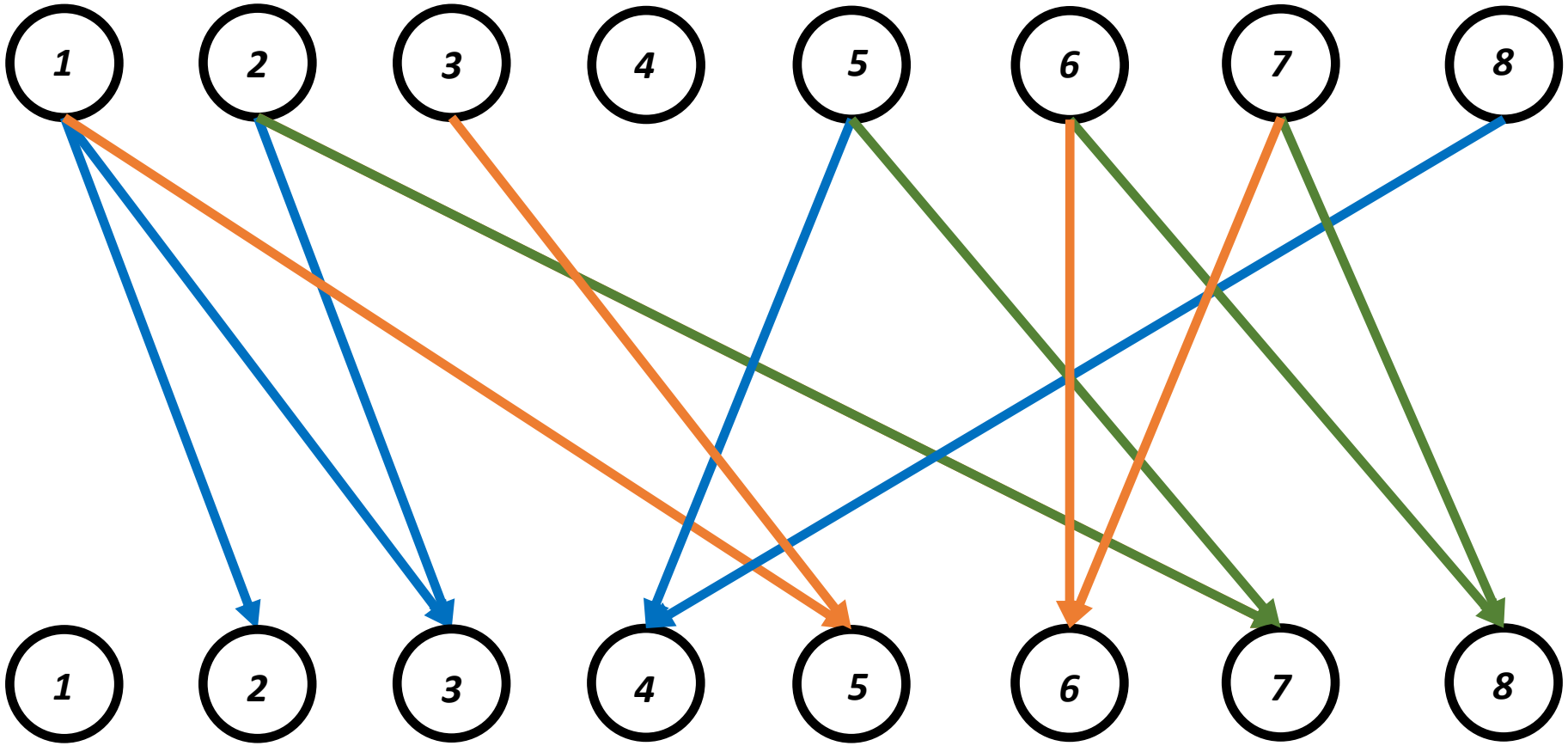
Given an edge-labelled graph, is it a Wheeler graph?



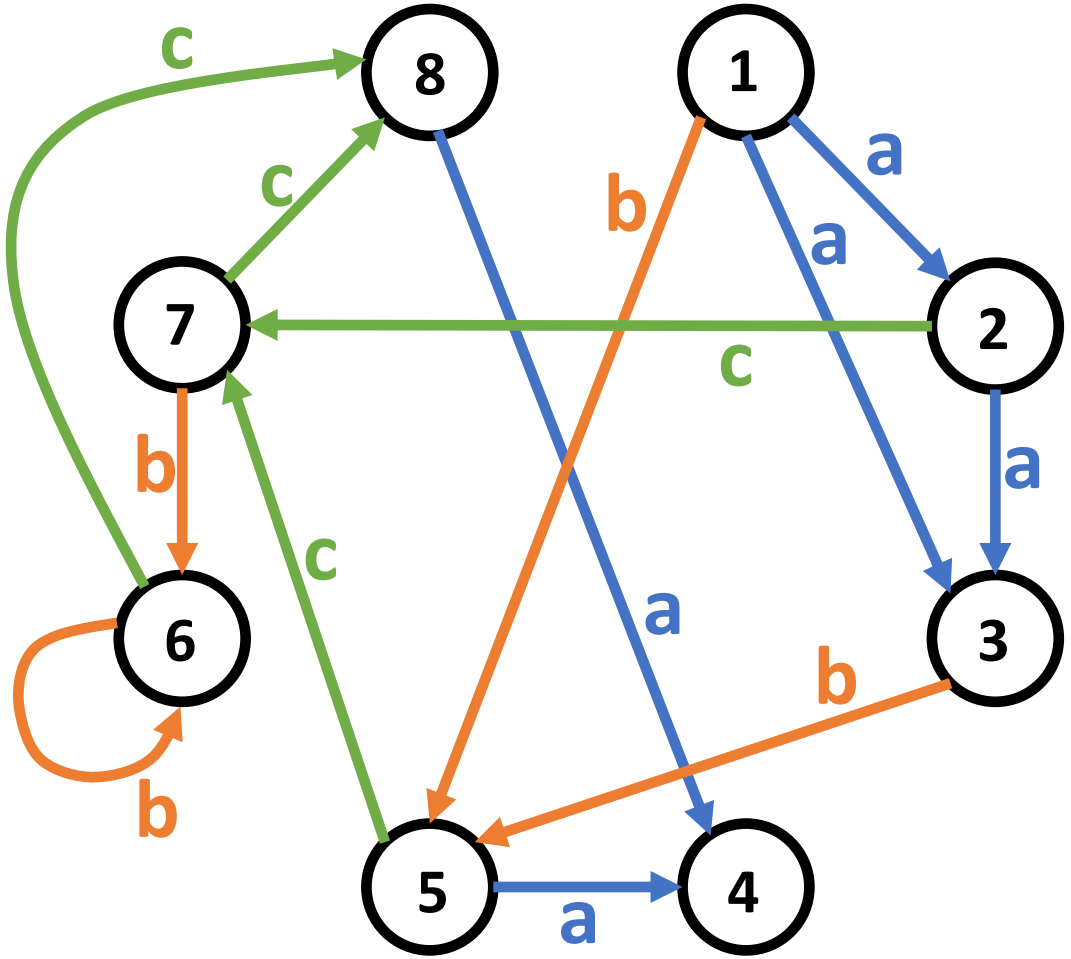
it is a Wheeler Graph

Wheeler graph recognition problem

Given an edge-labelled graph, is it a Wheeler graph?



State-of-the-art WG recognition algorithm



Computer Science > Computational Complexity

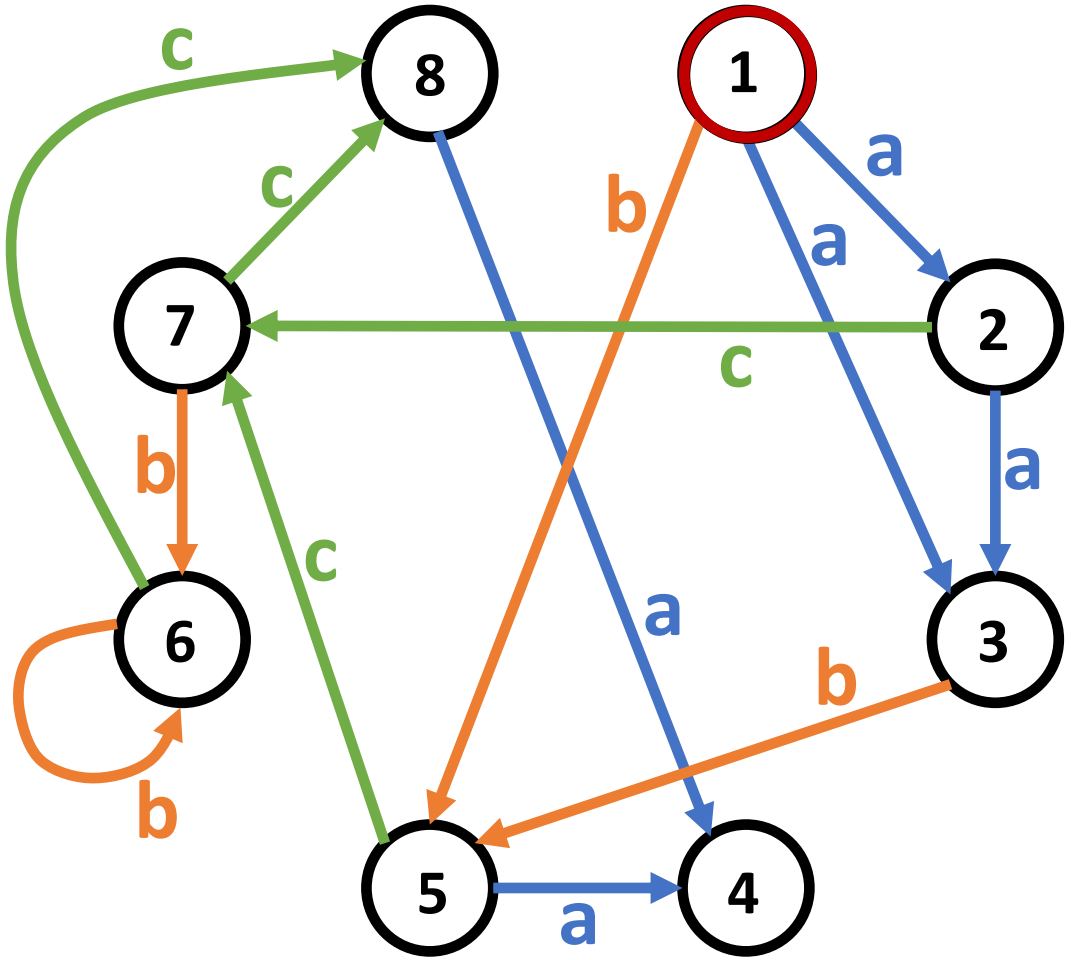
[Submitted on 5 Feb 2019 (v1), last revised 25 Feb 2019 (this version, v2)]

On the Hardness and Inapproximability of Recognizing Wheeler Graphs

Daniel Gibney, Sharma V. Thankachan

2019

How to encode a Wheeler Graph?



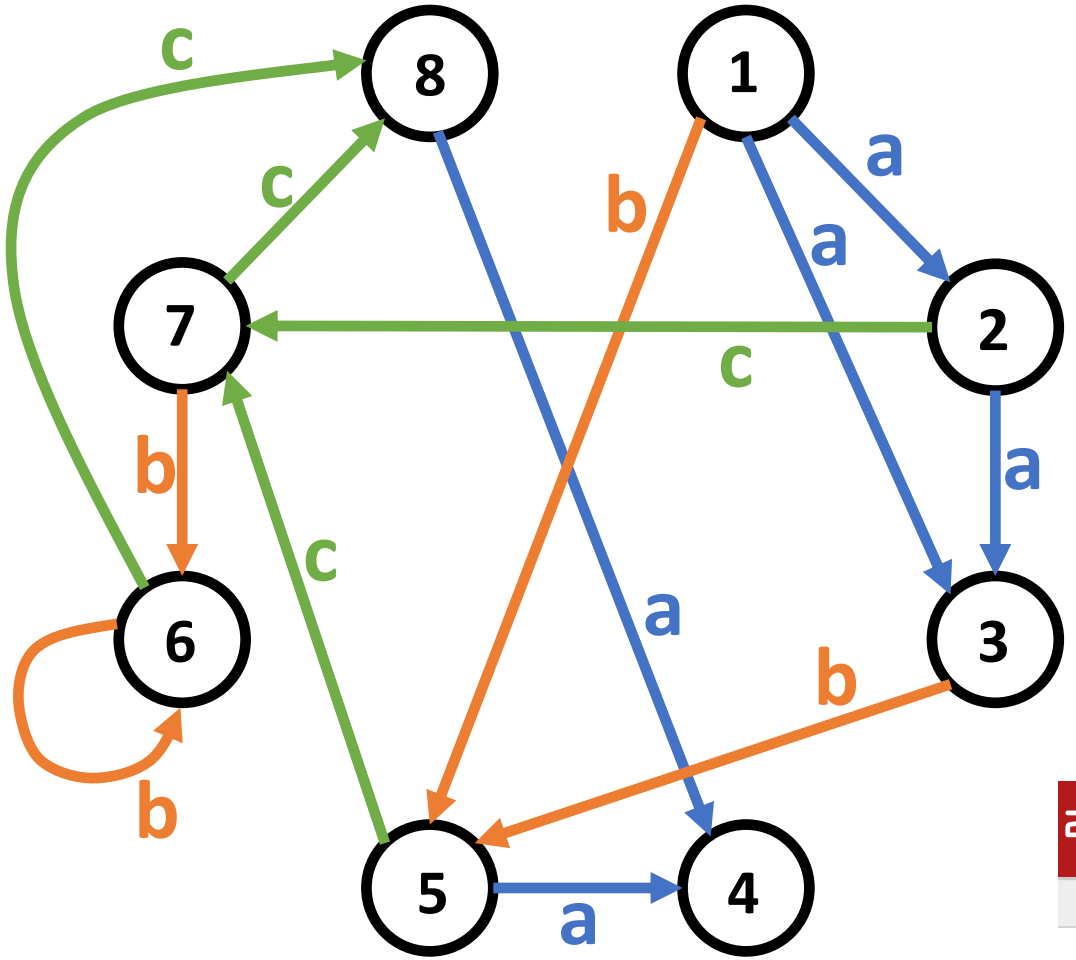
	1	2	3	4	5	6	7	8
O =	0001	001	01	1	001	001	001	01
L =	aab	ac	b		ac	bc	bc	a
I =	1	01	001	001	001	001	001	001

Wheeler graphs: A framework for BWT-based data structures ☆

Travis Gagie [a], Giovanni Manzini [b,c,*], Jouni Sirén [d]

2017

Gibney's & Thankachan's algorithm (G&T)



	1	2	3	4	5	6	7	8
O =	0001	001	01	1	001	001	001	01
L =	aab	ac	b		ac	bc	bc	a
I =	1	01	001	001	001	001	001	001

```
|O| = 13 + 8   (e + n)
|I| = 13 + 8   (e + n)
|L| = 13 * 2   (e * log(sigma))
```

arXiv > cs > arXiv:1902.01960

Search... Help | Advanc

Computer Science > Computational Complexity

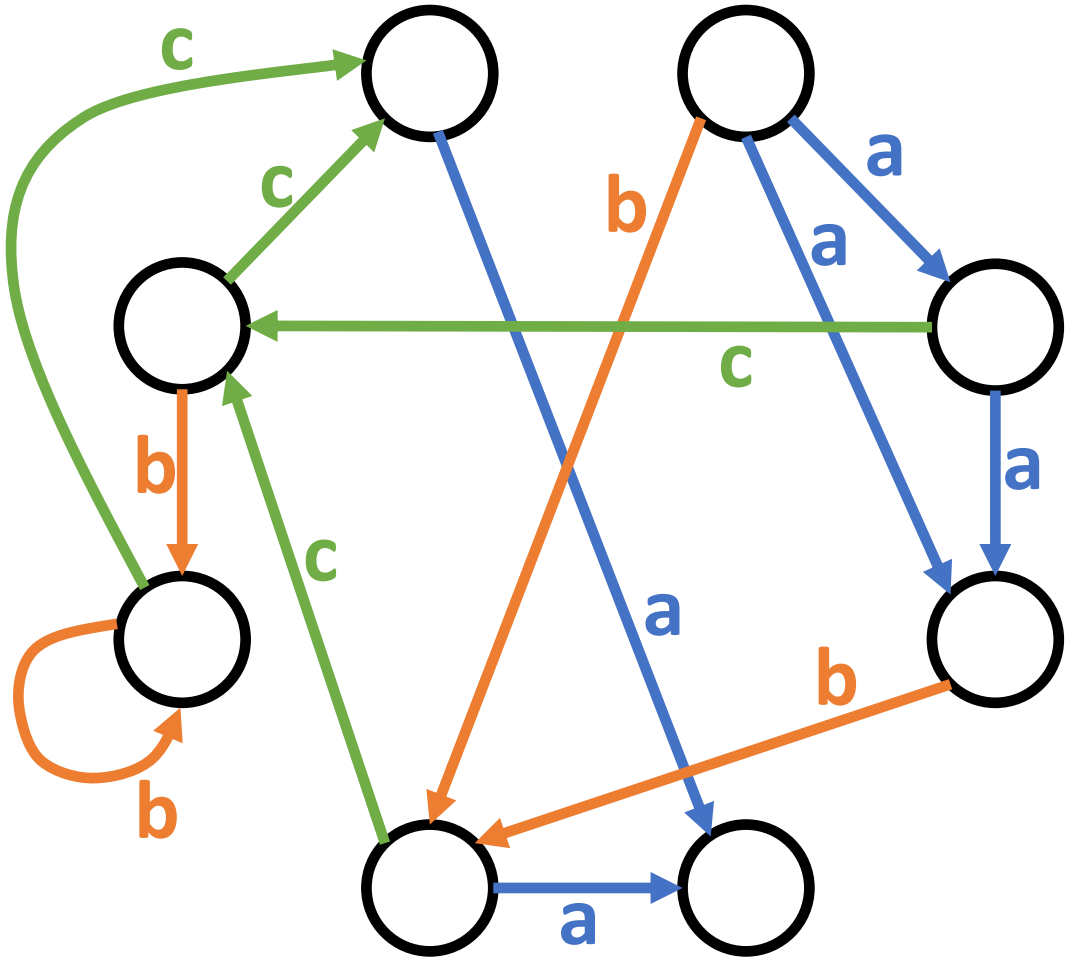
[Submitted on 5 Feb 2019 (v1), last revised 25 Feb 2019 (this version, v2)]

On the Hardness and Inapproximability of Recognizing Wheeler Graphs

Daniel Gibney, Sharma V. Thankachan

2019

Gibney's & Thankachan's algorithm (G&T)



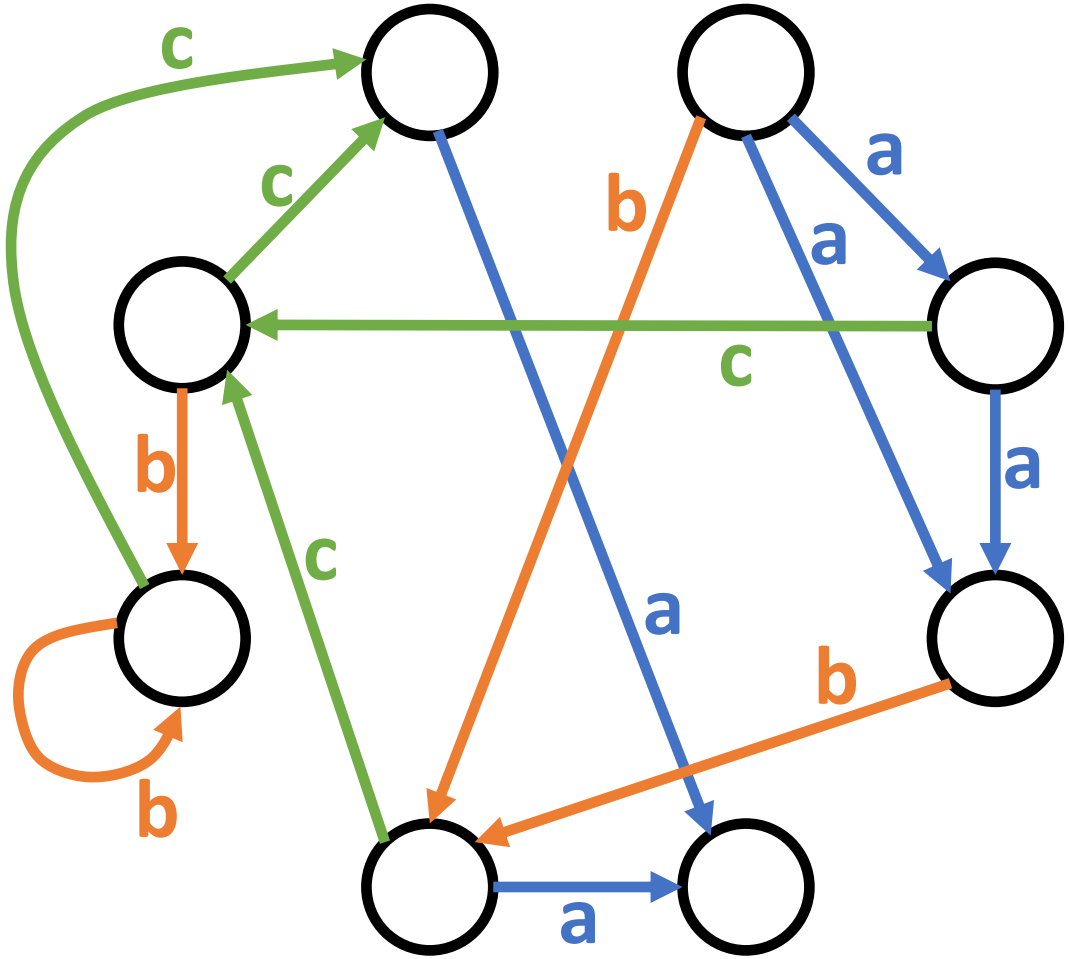

```
|O| = 13 + 8   (e + n)
|I| = 13 + 8   (e + n)
|L| = 13 * 2   (e * log(sigma))
```

Algorithm 1 IdentifyWheelerGraph(G)

for all $(O, I, L) \in S$ **do** $\quad |S| \leq 2^{2(e+n)+e \log \sigma}$
  if (O, I, L) defines a valid wheeler graph G' **then**
    convert G to undirected graph $\alpha(G)$
    convert G' to undirected graph $\alpha(G')$
    if $\alpha(G)$ and $\alpha(G')$ are isomorphic **then**
      return 'Wheeler Graph'
    end if
  end if
end for
return "Not a Wheeler Graph"

$$O(2^{O(e+n)+e \log \sigma})$$

Wheelie: WG recognition algorithm



Renaming heuristic

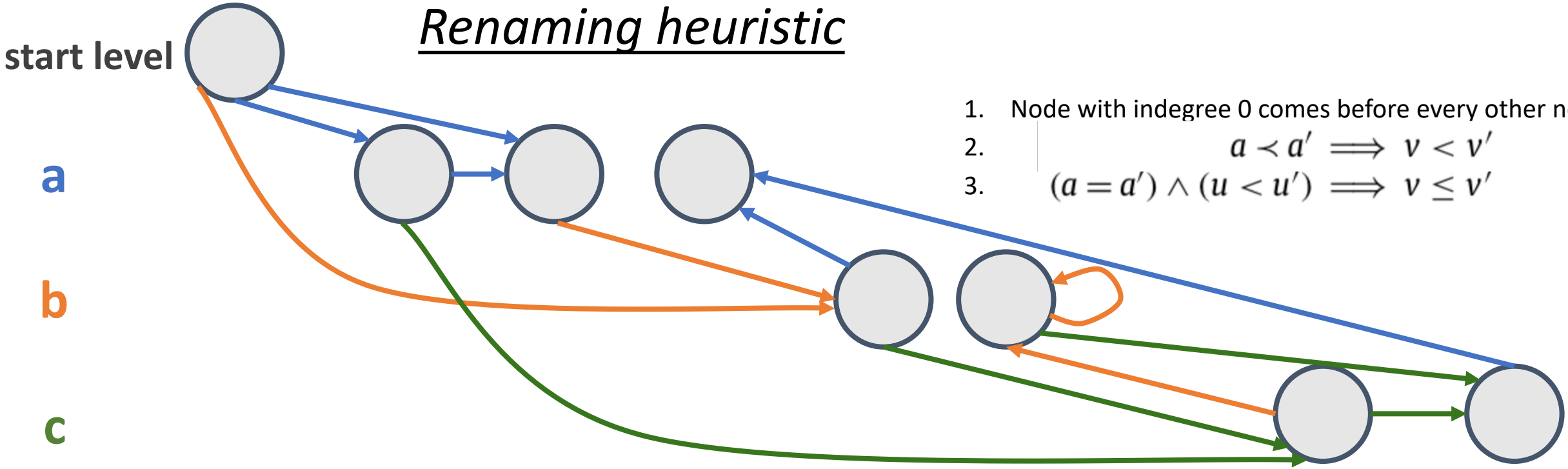
+

Solvers:

Permutation solver
- **Wheelie-PR**

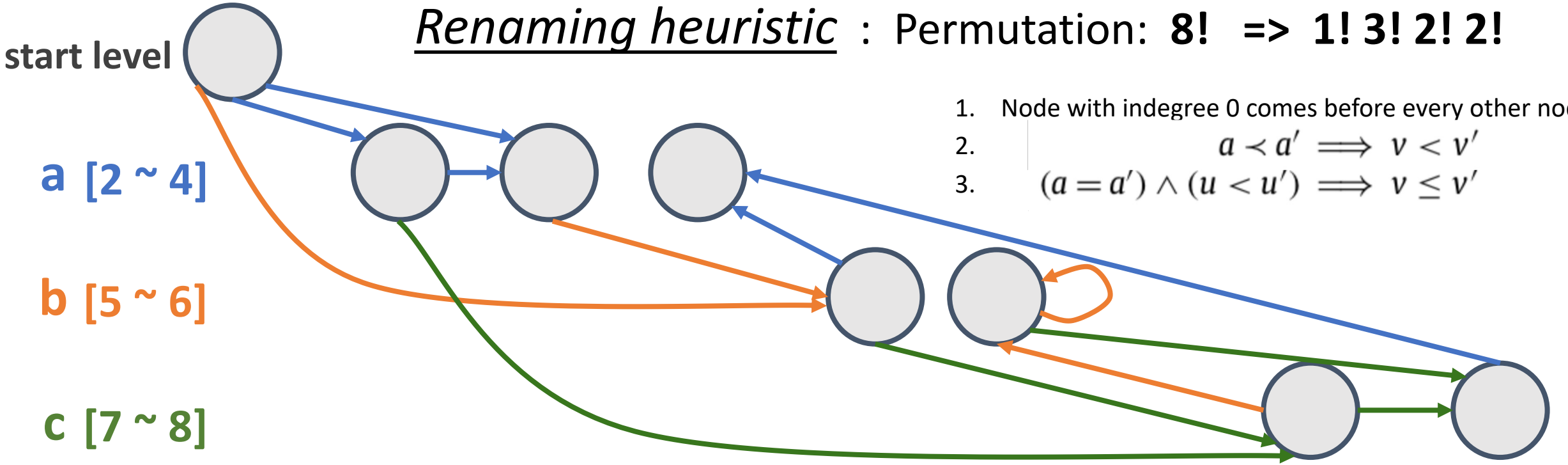
SMT solver:
- **Wheelie-SMT**

Renaming heuristic



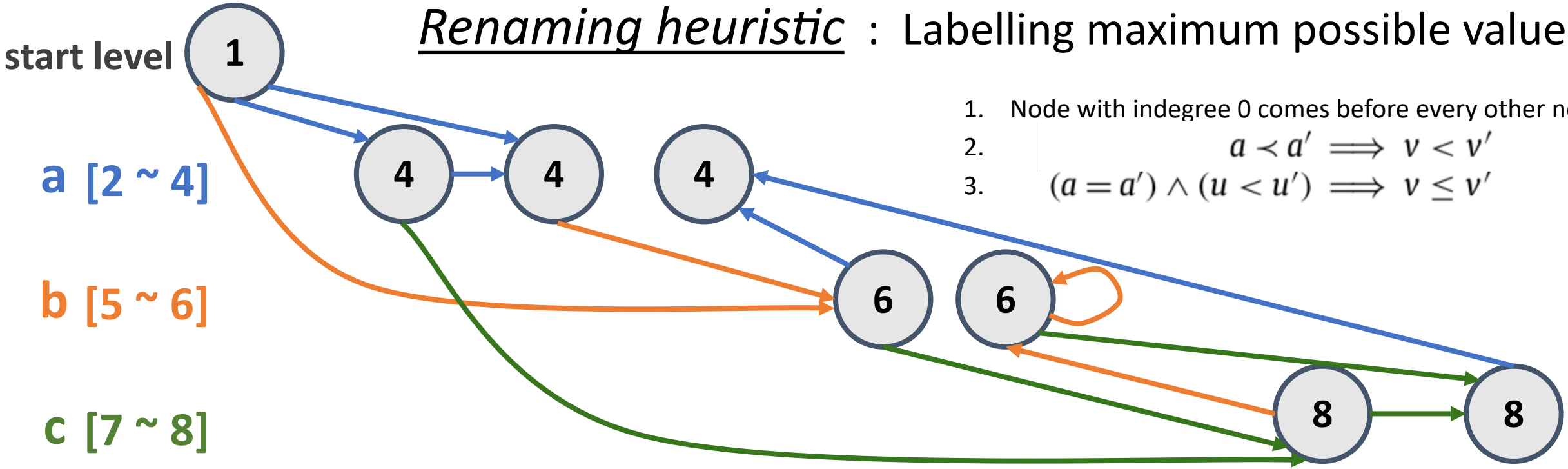
1. Node with indegree 0 comes before every other nodes.
2. $a \prec a' \implies v < v'$
3. $(a = a') \wedge (u < u') \implies v \leq v'$

Renaming heuristic : Permutation: **8! => 1! 3! 2! 2!**



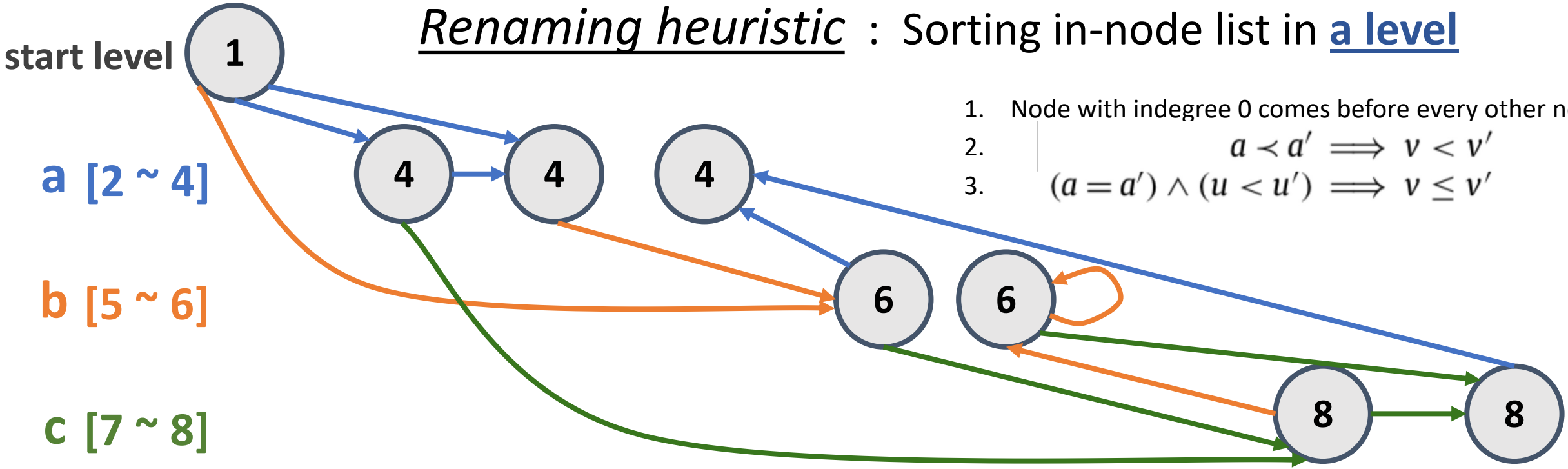
1. Node with indegree 0 comes before every other nodes.
2. $$a \prec a' \implies v < v'$$
3. $$(a = a') \wedge (u < u') \implies v \leq v'$$

Renaming heuristic : Labelling maximum possible value



1. Node with indegree 0 comes before every other nodes.
2. $$a \prec a' \implies v < v'$$
3. $$(a = a') \wedge (u < u') \implies v \leq v'$$

Renaming heuristic : Sorting in-node list in a level



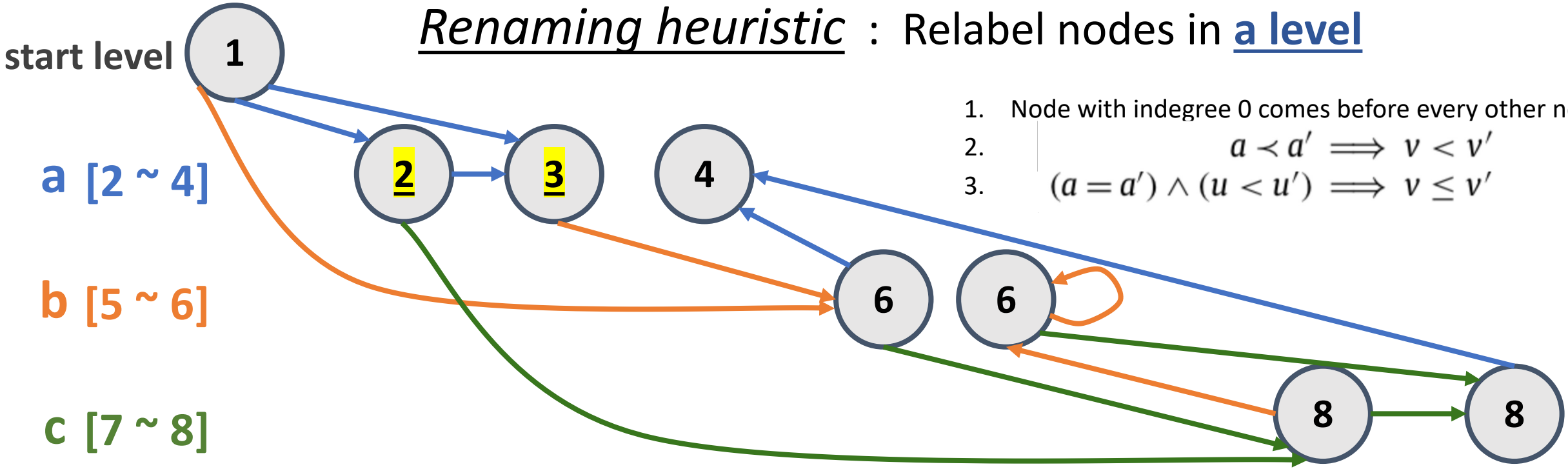
1. Node with indegree 0 comes before every other nodes.
2. $a \prec a' \implies v < v'$
3. $(a = a') \wedge (u < u') \implies v \leq v'$

In-node list:

start level
1: Ø

a level	b level	c level
4: 1	6: 1, 4	8: 4, 6
4: 1, 4	6: 6, 8	8: 6, 8
4: 6, 8		

Renaming heuristic : Relabel nodes in a level



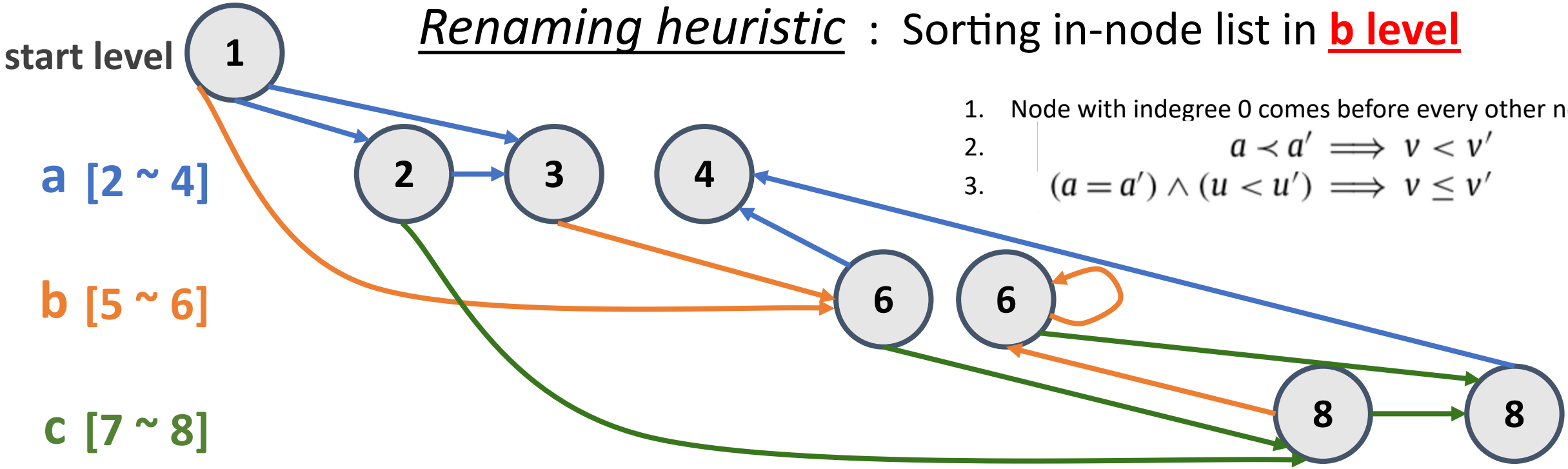
1. Node with indegree 0 comes before every other nodes.
2. $a \prec a' \implies v < v'$
3. $(a = a') \wedge (u < u') \implies v \leq v'$

In-node list:

start level
1: Ø

a level	b level	c level
2: 1	6: 1, 3	8: 2, 6
3: 1, 4	6: 6, 8	8: 6, 8
4: 6, 8		

Renaming heuristic : Sorting in-node list in **b level**

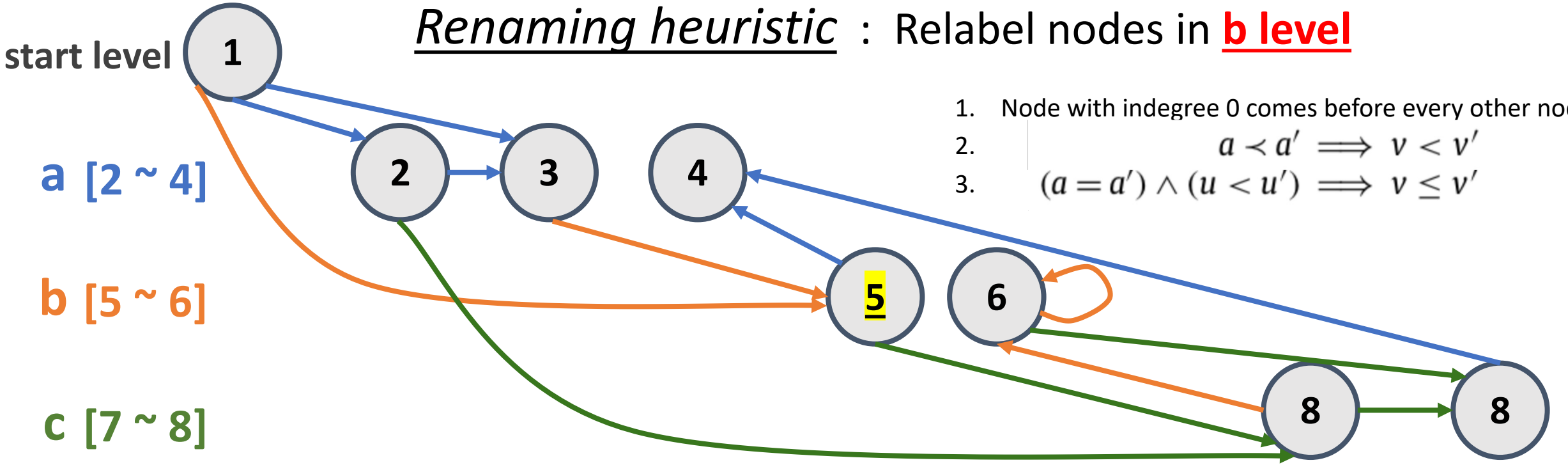


1. Node with indegree 0 comes before every other nodes.
2. $a \prec a' \implies v < v'$
3. $(a = a') \wedge (u < u') \implies v \leq v'$

In-node list:

start level	a level	b level	c level
1: Ø	2: 1	6: 1, 3	8: 2, 6
	3: 1, 4	6: 6, 8	8: 6, 8
	4: 6, 8		

Renaming heuristic : Relabel nodes in **b level**



1. Node with indegree 0 comes before every other nodes.
2. $a \prec a' \implies v < v'$
3. $(a = a') \wedge (u < u') \implies v \leq v'$

In-node list:

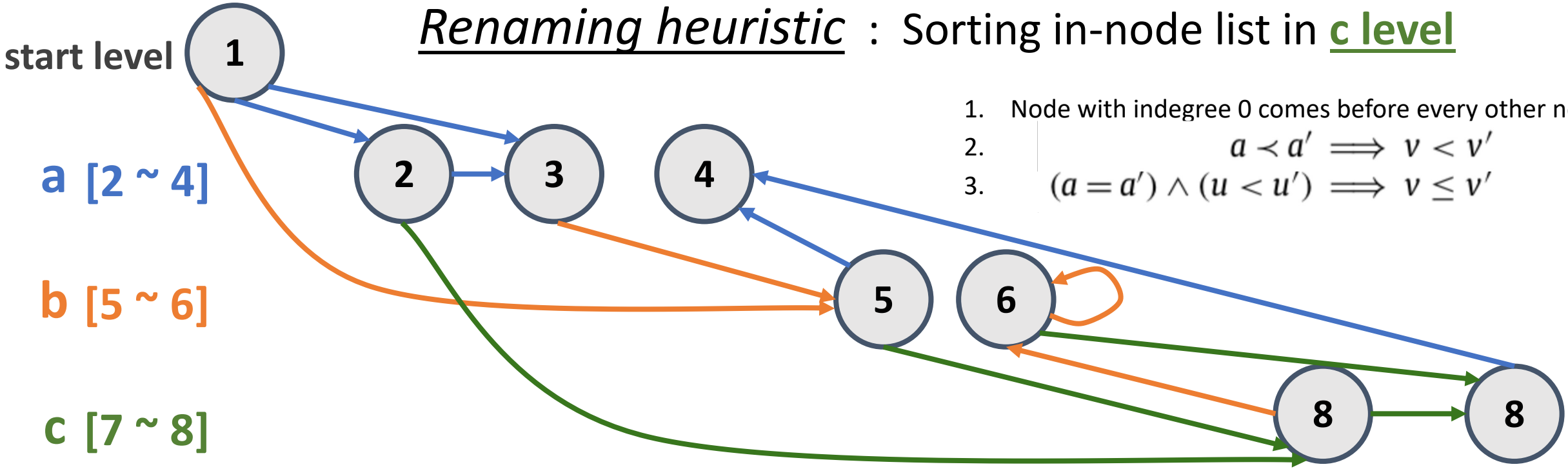
start level
1: Ø

a level
2: 1
3: 1, 4
4: 6, 8

b level
5: 1, 3
6: 6, 8

c level
8: 2, 5
8: 6, 8

Renaming heuristic : Sorting in-node list in c level

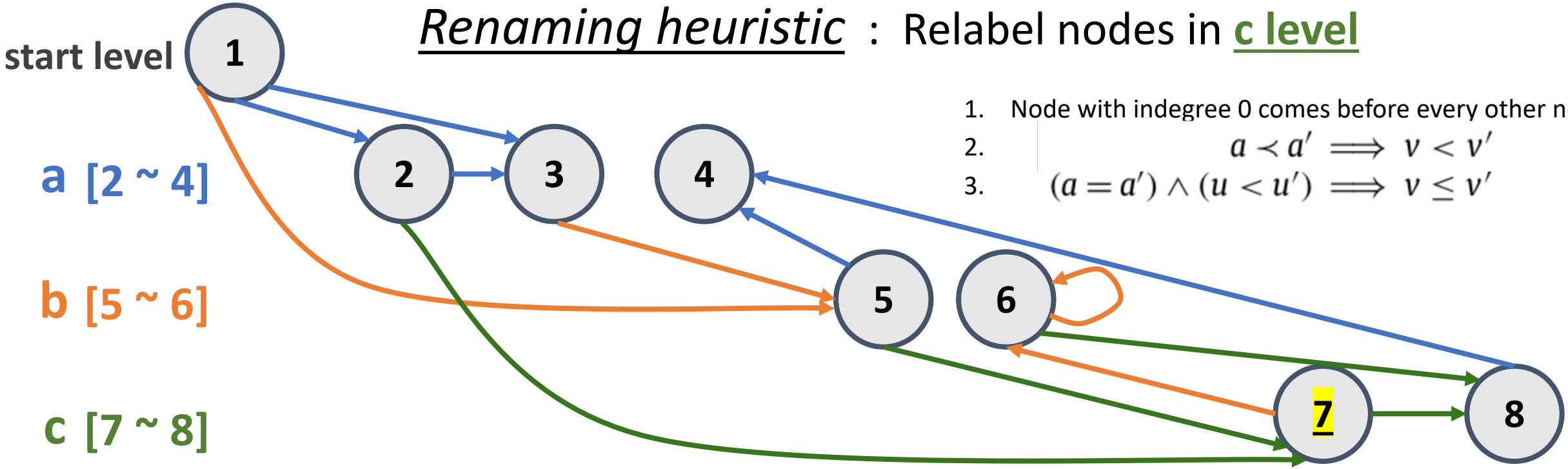


1. Node with indegree 0 comes before every other nodes.
2. $a \prec a' \implies v < v'$
3. $(a = a') \wedge (u < u') \implies v \leq v'$

In-node list:

start level	a level	b level	c level
1: Ø	2: 1	5: 1, 3	8: 2, 6
	3: 1, 4	6: 6, 8	8: 6, 8
	4: 6, 8		

Renaming heuristic : Relabel nodes in c level



1. Node with indegree 0 comes before every other nodes.
2. $a \prec a' \implies v < v'$
3. $(a = a') \wedge (u < u') \implies v \leq v'$

In-node list:

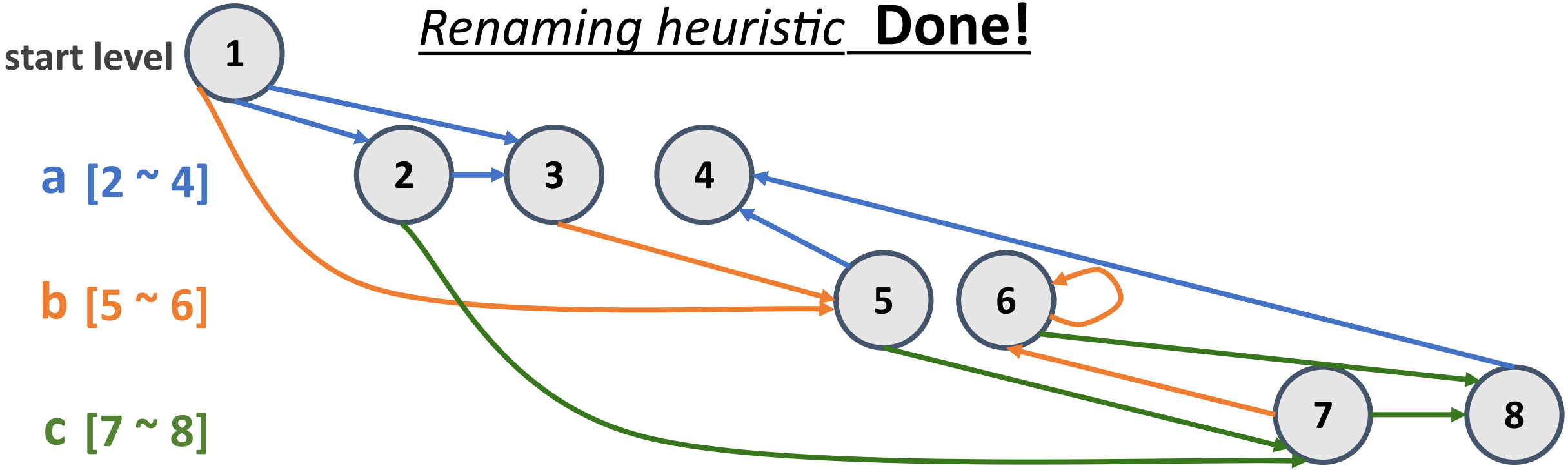
start level
1: Ø

a level
2: 1
3: 1, 4
4: 6, 8

b level
5: 1, 3
6: 7, 8

c level
7: 2, 6
8: 6, 8

Renaming heuristic **Done!**



In-node list:

start level
1: Ø

a level
2: 1
3: 1, 4
4: 6, 8

b level
5: 1, 3
6: 7, 8

c level
7: 2, 6
8: 6, 8

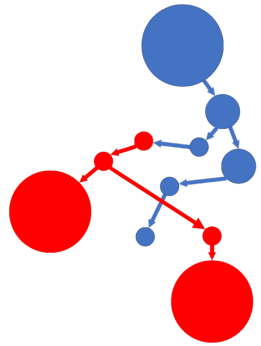
False

All nodes stabled

True

Solvers { Wheelie-PR, Wheelie-SMT

Our Contribution



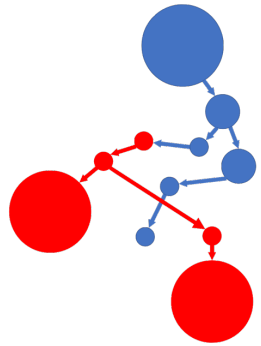
WGT
Wheeler Graph Toolkit

- A Wheeler graph visualizer: bipartite representation

- ***Implemented the first Wheeler graph recognizer***

- Wheeler graph *generators*: Random WG / Trie / DBG / RDG

Our Contribution



WGT
Wheeler Graph Toolkit

- A Wheeler graph visualizer: bipartite representation

- Implemented the first Wheeler graph recognizer

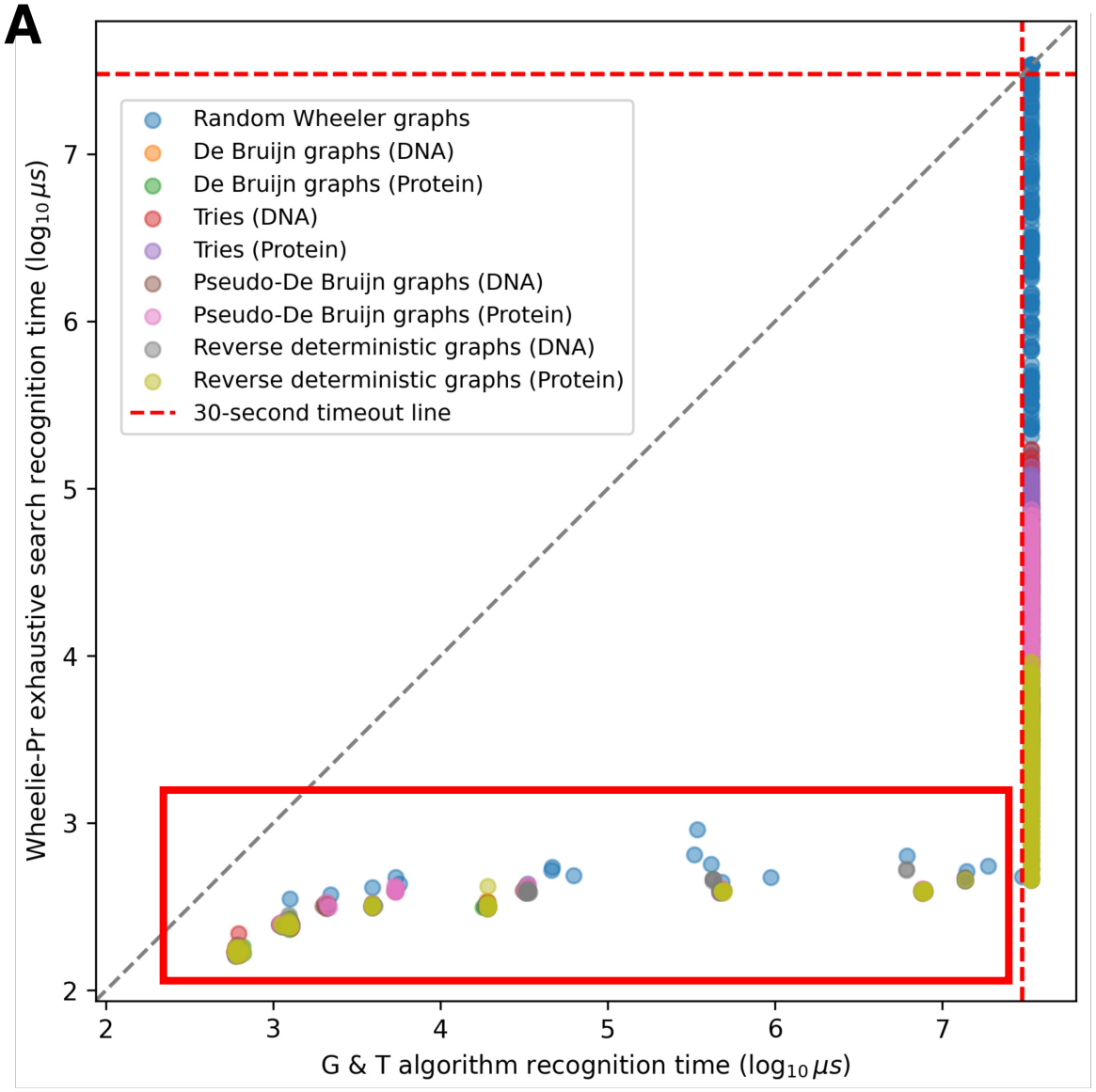
- ***Wheeler graph generators: Random WG / Trie / DBG / RDG***

Comparing Wheelie-PR to G&T's algorithm

Timeout: 30 sec

25 multiple orthologue alignments (GENCODE REST API),

DNA & amino acid (AA) sequences.

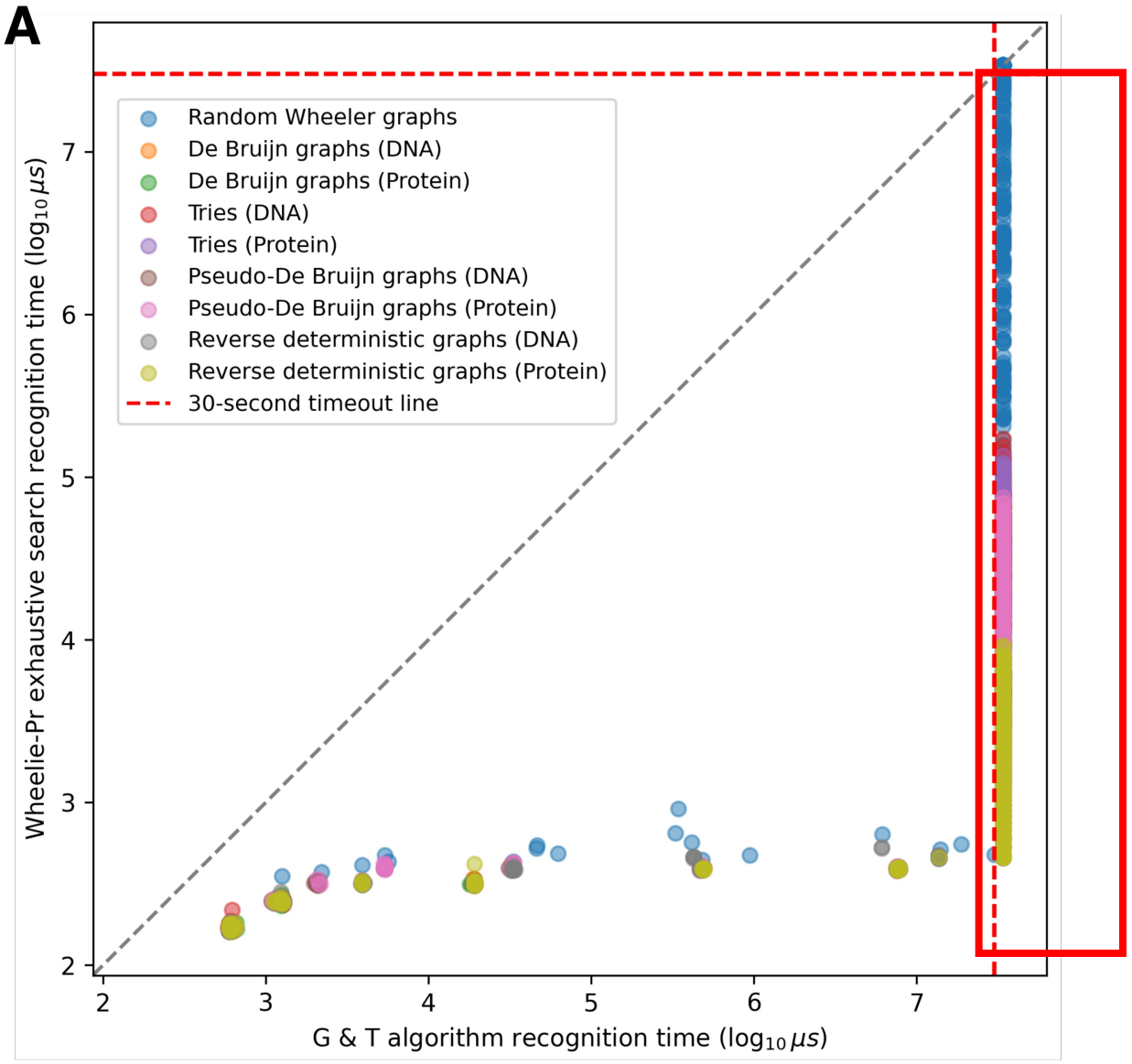


Comparing Wheelie-PR to G&T's algorithm

Timeout: 30 sec

25 multiple orthologue alignments (GENCODE REST API),

DNA & amino acid (AA) sequences.

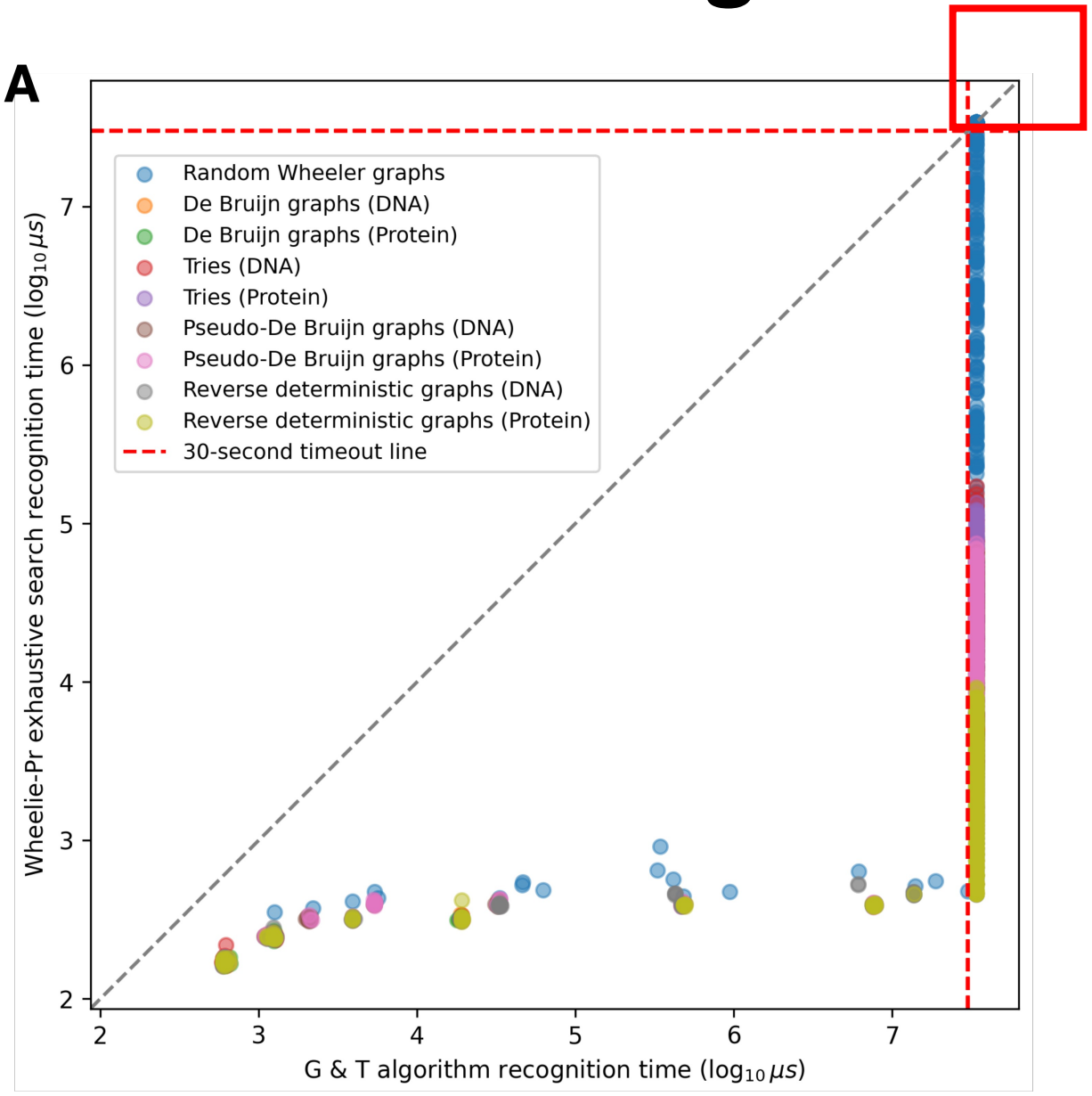


Comparing Wheelie-PR to G&T's algorithm

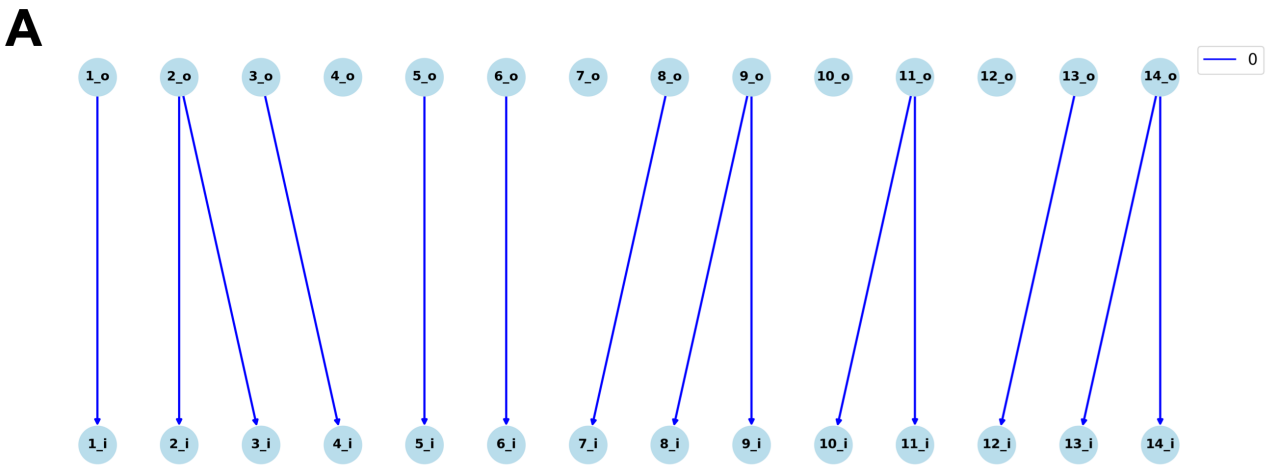
Timeout: 30 sec

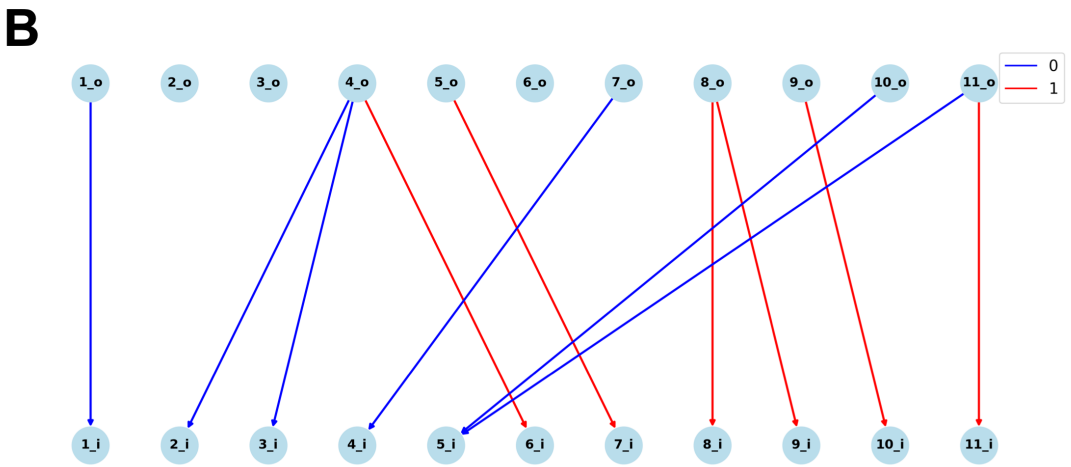
25 multiple orthologue alignments (GENCODE REST API),

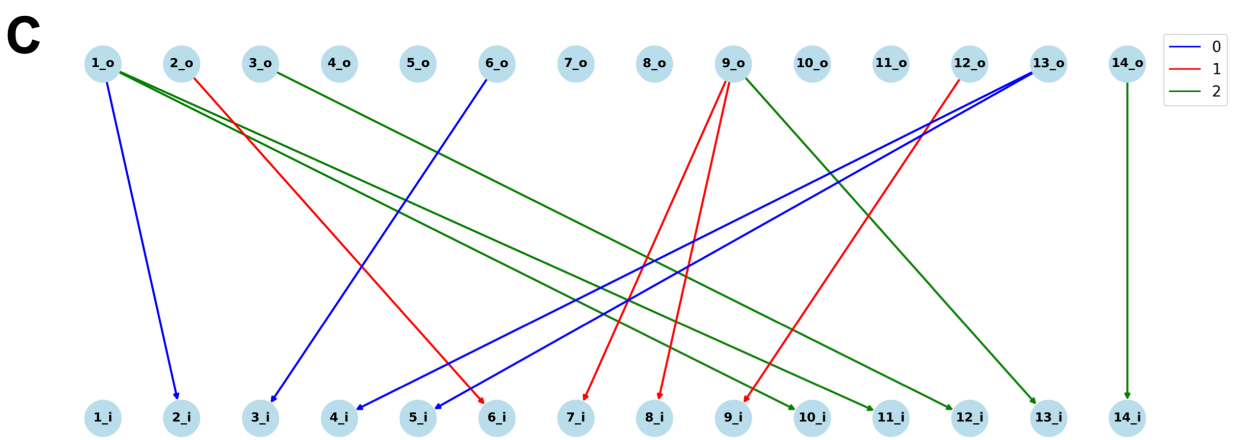
DNA & amino acid (AA) sequences.

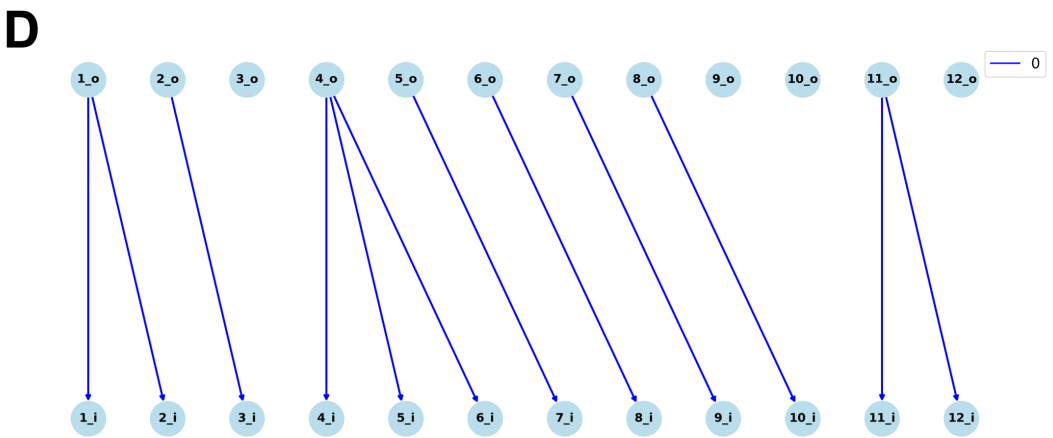


Comparing Wheelie-PR to G&T's algorithm









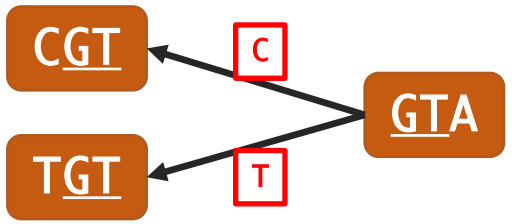
Hardness of Wheeler Graphs: d-NFA

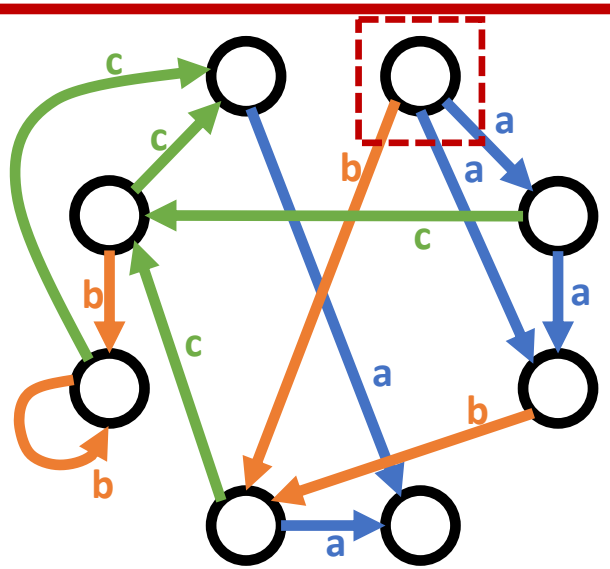
- A linear sequence can always be encoded as a Wheeler graph
  - Sorted Suffix order of reversed T (T^R)
  - It is **1-NFA** (simplest Wheeler Graph)

Definition

d-NFA : describing a Wheeler Graph where all nodes have ≤ d outgoing same-label edges, and at least one node has exactly d outgoing same-label edges.

- Harder graph: d-NFA, d > 1
  - 2 edges labelled as a coming out from the node
- Other 1-NFA Wheeler graphs
  - DeBruijn graph

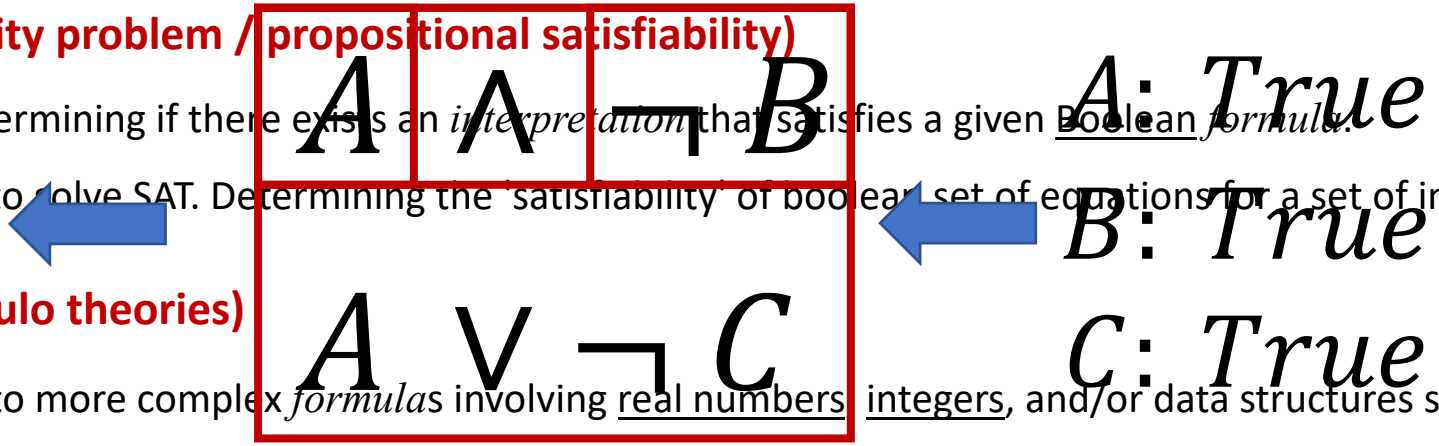




Satisfiability modulo theories (SMT)

- An ***atom*** is a basic unit of a logical expression (predicate) which can be `True` or `False`
- A ***literal*** is either an *atom* or its negation (¬).
- A ***formula*** is a set of *literals (atoms)* connected by Boolean connectives (∧, ∨, ¬),
- A ***theory*** is a set of *formulas* in the formal language
- An ***interpretation*** is an assignment of meanings to a *theory* and predicate symbols within the theory.

- **SAT (Boolean satisfiability problem / propositional satisfiability)**
  - is the problem of determining if there exists an *interpretation* that satisfies a given Boolean *formula*.
  - **SAT solver**: program to solve SAT. Determining the "satisfiability" of boolean set of equations for a set of inputs.
- **SMT (Satisfiability modulo theories)**
  - SMT generalizes SAT to more complex *formulas* involving real numbers, integers, and/or data structures such as lists, arrays, bit vectors, and strings.
  - decides the satisfiability of a first-order *formula* with respect to one or more background *theories*.

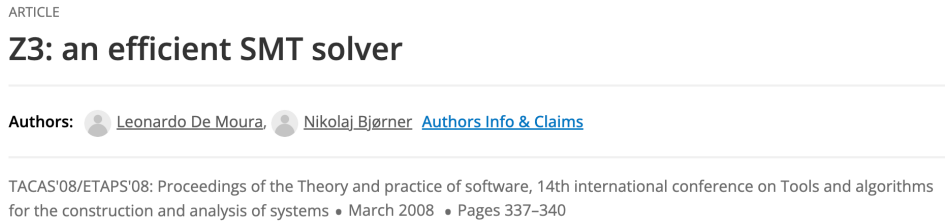


Satisfiability modulo theories (SMT)

- **SAT (Boolean satisfiability problem / propositional satisfiability)**
  - is the problem of determining if there exists an *interpretation* that satisfies a given Boolean *formula*.
  - **SAT solver**: program to solve SAT. Determining the 'satisfiability' of boolean set of equations for a set of inputs.
- **SMT (Satisfiability modulo theories)**
  - SMT generalizes SAT to more complex *formula*s involving real numbers, integers, and/or data structures such as lists, arrays, bit vectors, and strings.
  - decides the satisfiability of a first-order *formula* with respect to one or more background *theories*.

Satisfiability modulo theories (SMT)

- **SAT (Boolean satisfiability problem / propositional satisfiability)**
  - is the problem of determining if there exists an *interpretation* that satisfies a given Boolean *formula*.
  - **SAT solver**: program to solve SAT. Determining the 'satisfiability' of boolean set of equations for a set of inputs.
- **SMT (Satisfiability modulo theories)**
  - SMT generalizes SAT to more complex *formulas* involving real numbers, integers, and/or data structures such as lists, arrays, bit vectors, and strings.
  - decides the satisfiability of a first-order *formula* with respect to one or more background *theories*.
- Only consider the **theory of *Integer Difference Logic (IDL)***,
  - requires *atoms* to be of the form $xi - xj \leq c$
- **A theory solver** decides the satisfiability of a conjunction of *literals*.
  - ***IDL theory solver*** => the Bellman-Ford algorithm (polynomial time).
- **SMT solver = SAT solver + IDL theory solver**
  - Input: a formula
  - Output: an assignment to the variables if the formula is satisfiable / unsatisfiable.

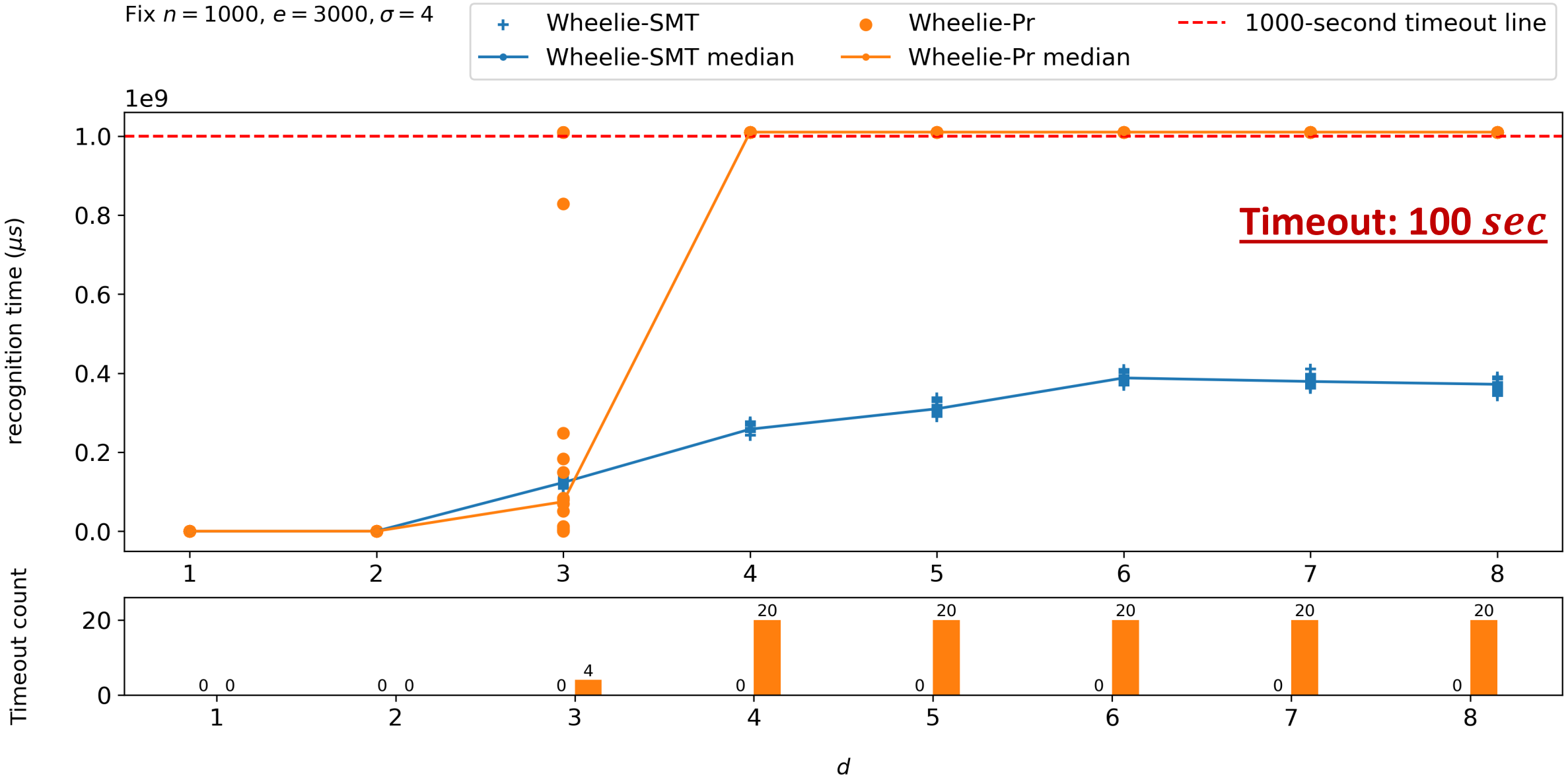

ARTICLE

Z3: an efficient SMT solver

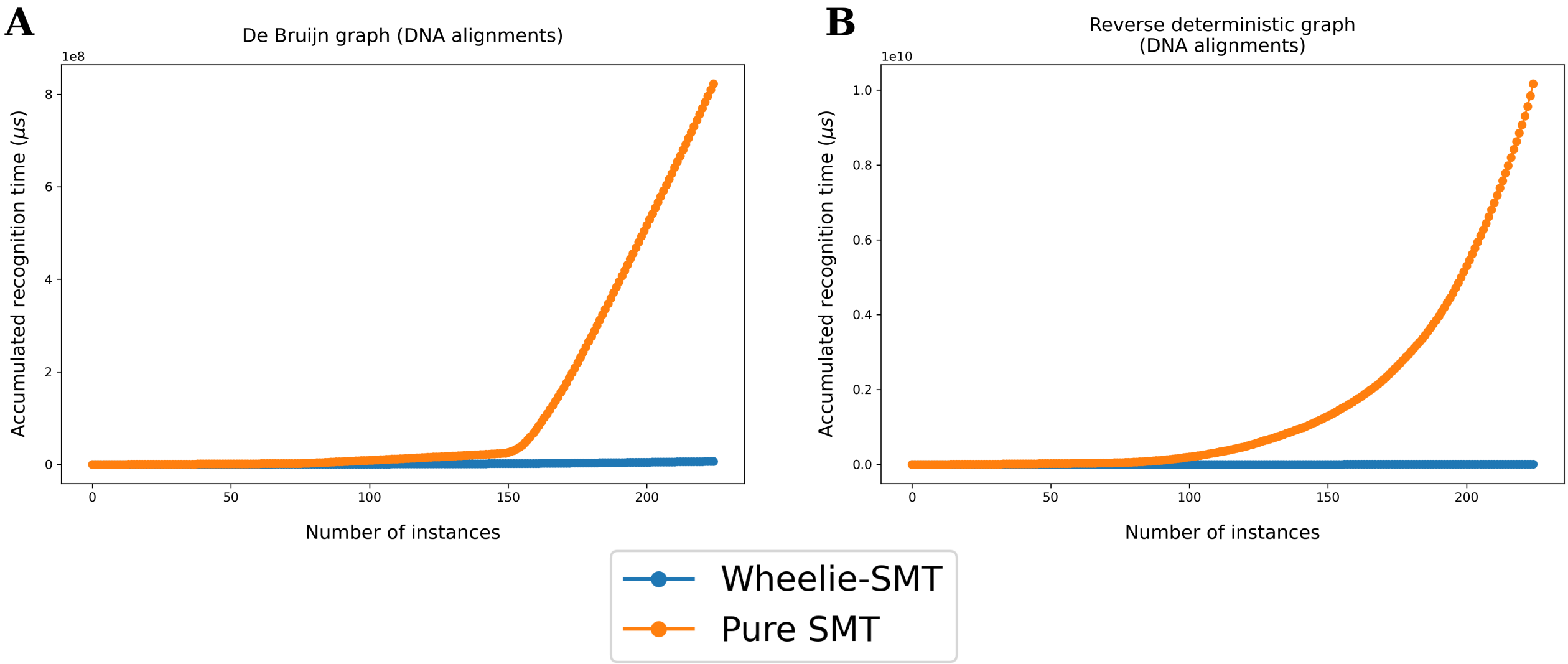
Authors: Leonardo De Moura, Nikolaj Bjørner Authors Info & Claims

TACAS'08/ETAPS'08: Proceedings of the Theory and practice of software, 14th international conference on Tools and algorithms for the construction and analysis of systems • March 2008 • Pages 337–340

Wheelie_SMT vs Wheelie_PR (d-NFA)



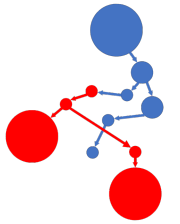
Wheelie_SMT vs Pure SMT



Conclusion

- A new Wheeler graph **recognizer**
  - Renaming heuristic + SMT
- A Wheeler graph **visualizer**: bipartite representation
- Wheeler graph **generators**: Random WG / Trie / DBG / RDG
- the recognizer is effective up to 3-NFA's, xx nodes.

Acknowledge



WGT
Wheeler Graph Toolkit

Project advisors:

Ben Langmead

Sanjit A Seshia

Co-first author:

Pei-Wei Chen

My PhD advisors:

Steven Salzberg

Mihaela Pertea

U.S. National Institutes of Health under grant
R0I-HG006677 (SLS.)
R35GM139602 (MP.)
R01HG011392 (BL.)

U.S. National Science Foundation under grant
DBI-1759518 (MP.)

Berkeley Fellowship (SAS.)