Wheeler graphs, part 1

Ben Langmead

JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

Department of Computer Science

BWT: matching

T : g a t t a c a t $

P : g a t

$	g	a	t	t	a	c	a	t
a	c	a	t	$	g	a	t	t
a	t	$	g	a	t	t	a	c
a	t	t	a	c	a	t	$	g
c	a	t	$	g	a	t	t	a
g	a	t	t	a	c	a	t	$
t	$	g	a	t	t	a	c	a
t	a	c	a	t	$	g	a	t
t	t	a	c	a	t	$	g	a

BWT: matching

T : g a t t a c a t $

P : g a t

```
$ g a t t a c a t
a c a t $ g a t t
a t $ g a t t a c
a t t a c a t $ g
c a t $ g a t t a
g a t t a c a t $
t $ g a t t a c a
t a c a t $ g a t
t t a c a t $ g a
```

BWT: matching

T : g a t t a c a t $

P : g a t

```
$ g a t t a c a t
a c a t $ g a t t
a t $ g a t t a c
a t t a c a t $ g
c a t $ g a t t a
g a t t a c a t $
t $ g a t t a c a
t a c a t $ g a t
t t a c a t $ g a
```

BWT: matching

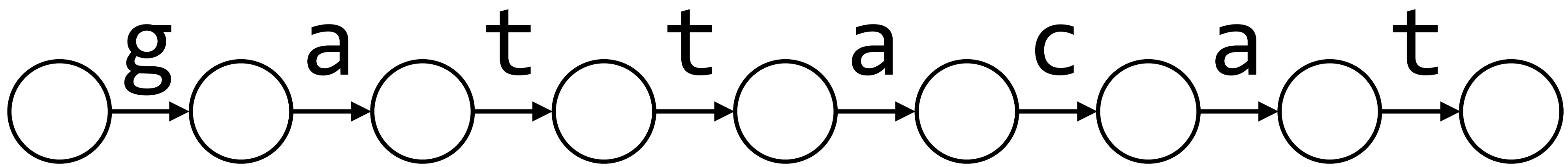
T : g a t t a c a t $

P : g a t

```
$ g a t t a c a t
a c a t $ g a t t
a t $ g a t t a c
a t t a c a t $ g
c a t $ g a t t a
g a t t a c a t $
t $ g a t t a c a
t a c a t $ g a t
t t a c a t $ g a
```

BWT: matching

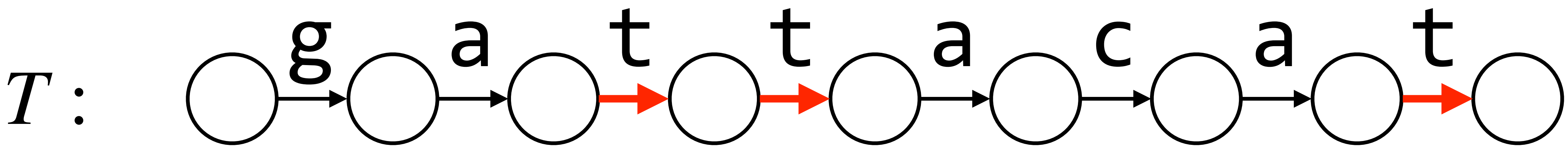
T :



P : g a t

```
$ g a t t a c a t
a c a t $ g a t t
a t $ g a t t a c
a t t a c a t $ g
c a t $ g a t t a
g a t t a c a t $
t $ g a t t a c a
t a c a t $ g a t
t t a c a t $ g a
```

BWT: matching

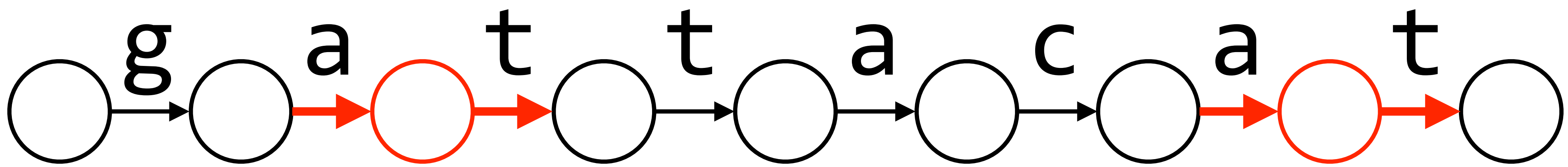


P : g a t

```
$ g a t t a c a t
a c a t $ g a t t
a t $ g a t t a c
a t t a c a t $ g
c a t $ g a t t a
g a t t a c a t $
t $ g a t t a c a
t a c a t $ g a t
t t a c a t $ g a
```

BWT: matching

T :

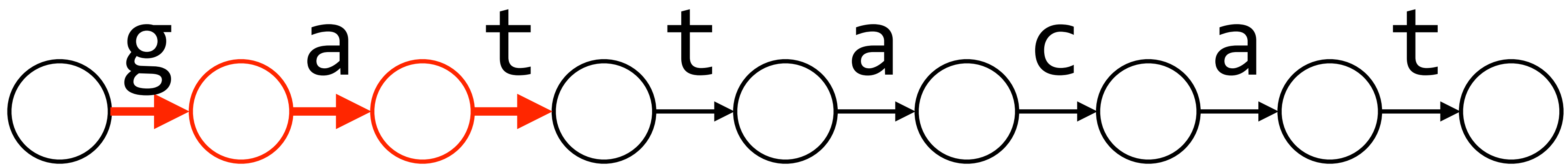


P : g a t

```
$ g a t t a c a t
a c a t $ g a t t
a t $ g a t t a c
a t t a c a t $ g
c a t $ g a t t a
g a t t a c a t $
t $ g a t t a c a
t a c a t $ g a t
t t a c a t $ g a
```

BWT: matching

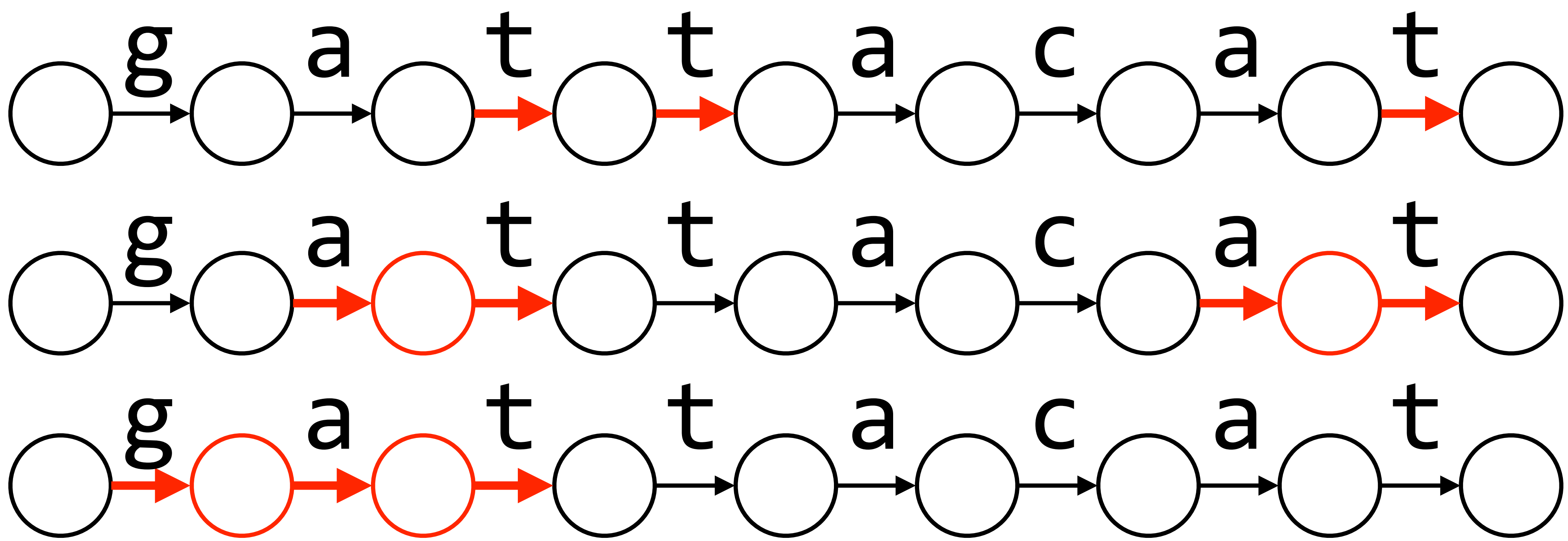
T :



P : g a t

```
$ g a t t a c a t
a c a t $ g a t t
a t $ g a t t a c
a t t a c a t $ g
c a t $ g a t t a
g a t t a c a t $
t $ g a t t a c a
t a c a t $ g a t
t t a c a t $ g a
```

BWT: matching



g a t t a c a t

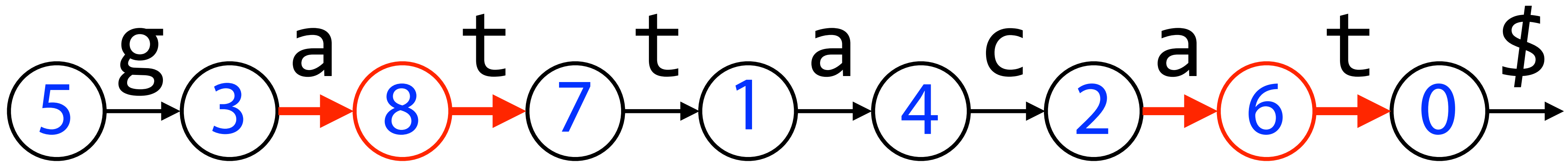
g a t t a c a t

g a t t a c a t

Two interpretations: we're finding matching *substrings* in a *string*, or we're finding matching *paths* in a *graph*

BWT: matching

T :

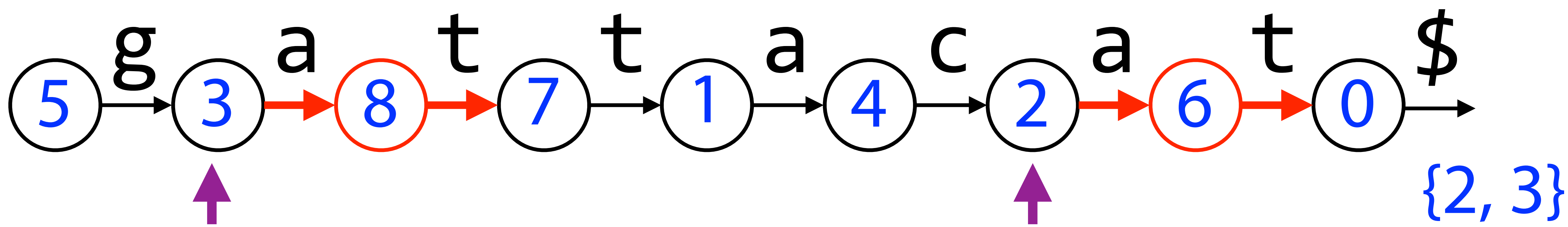


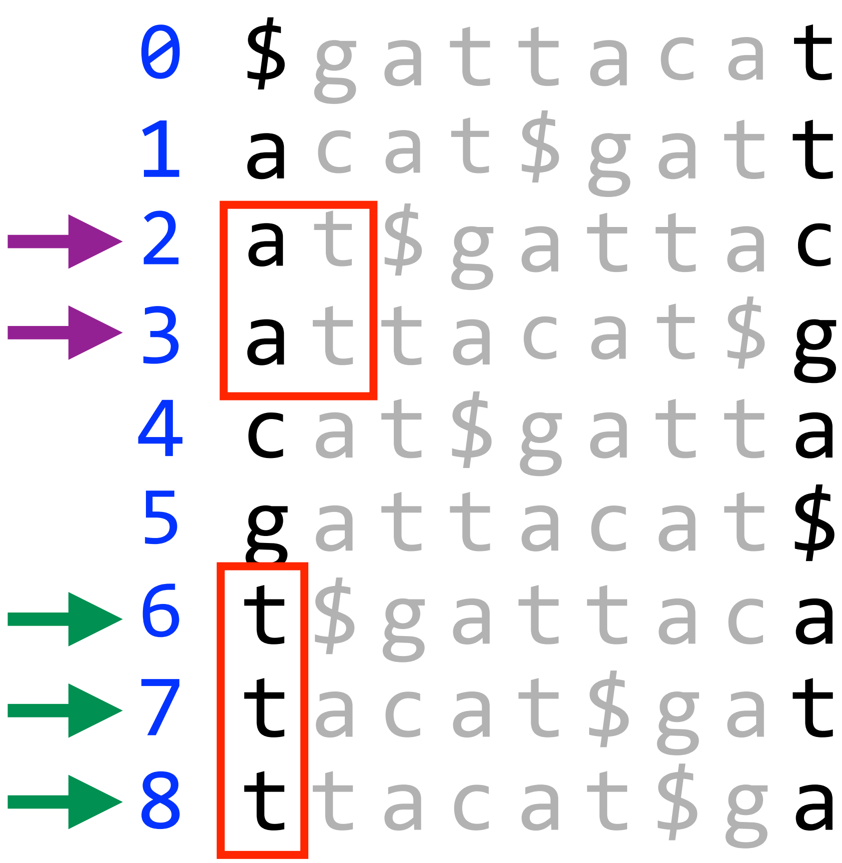

```
0 $ g a t t a c a t
1 a c a t $ g a t t
2 a t $ g a t t a c
3 a t t a c a t $ g
4 c a t $ g a t t a
5 g a t t a c a t $
6 t $ g a t t a c a
7 t a c a t $ g a t
8 t t a c a t $ g a
```

Consecutivity. In BW order, rows with same prefix are consecutive.

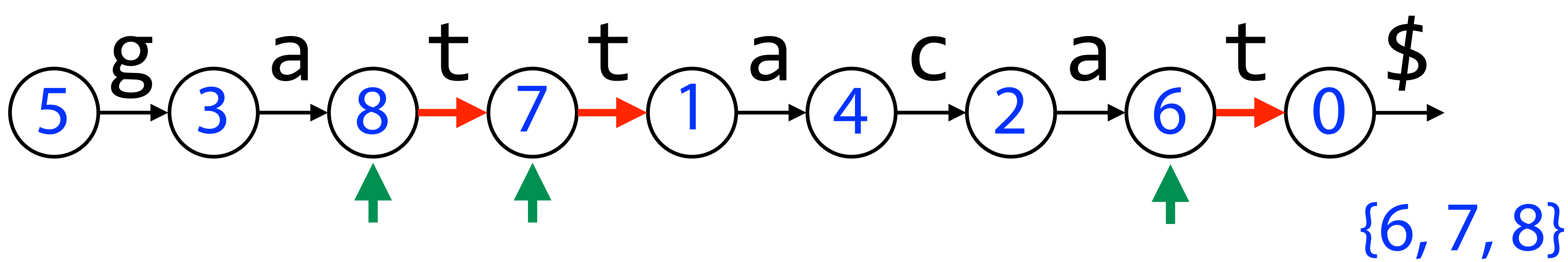
Is this visible in the graph? Let's label nodes with BW order…

BWT: matching

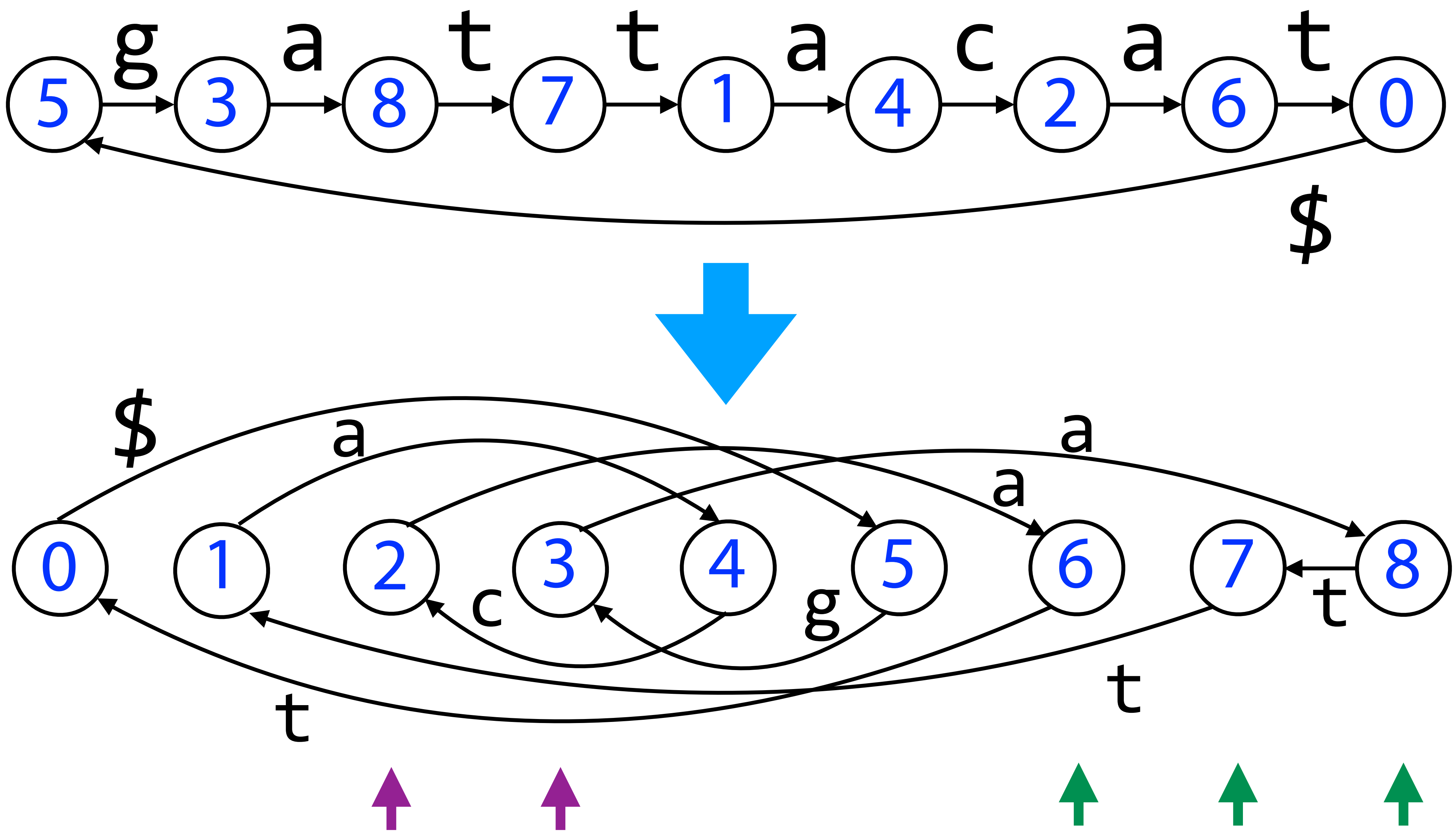




Consecutivity holds for labels of nodes in the BW range; would be clearer if we redrew the graph in BWT(T) order rather than T order



BWT: matching

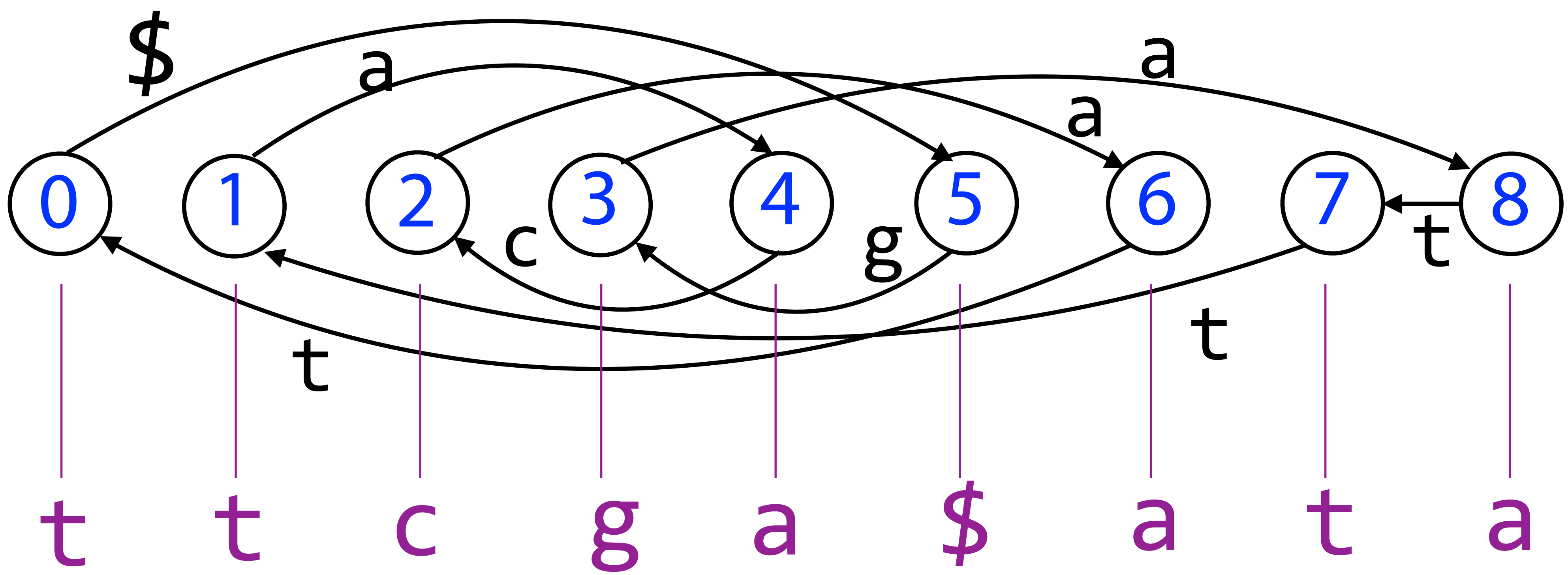


{2, 3}

{6, 7, 8}

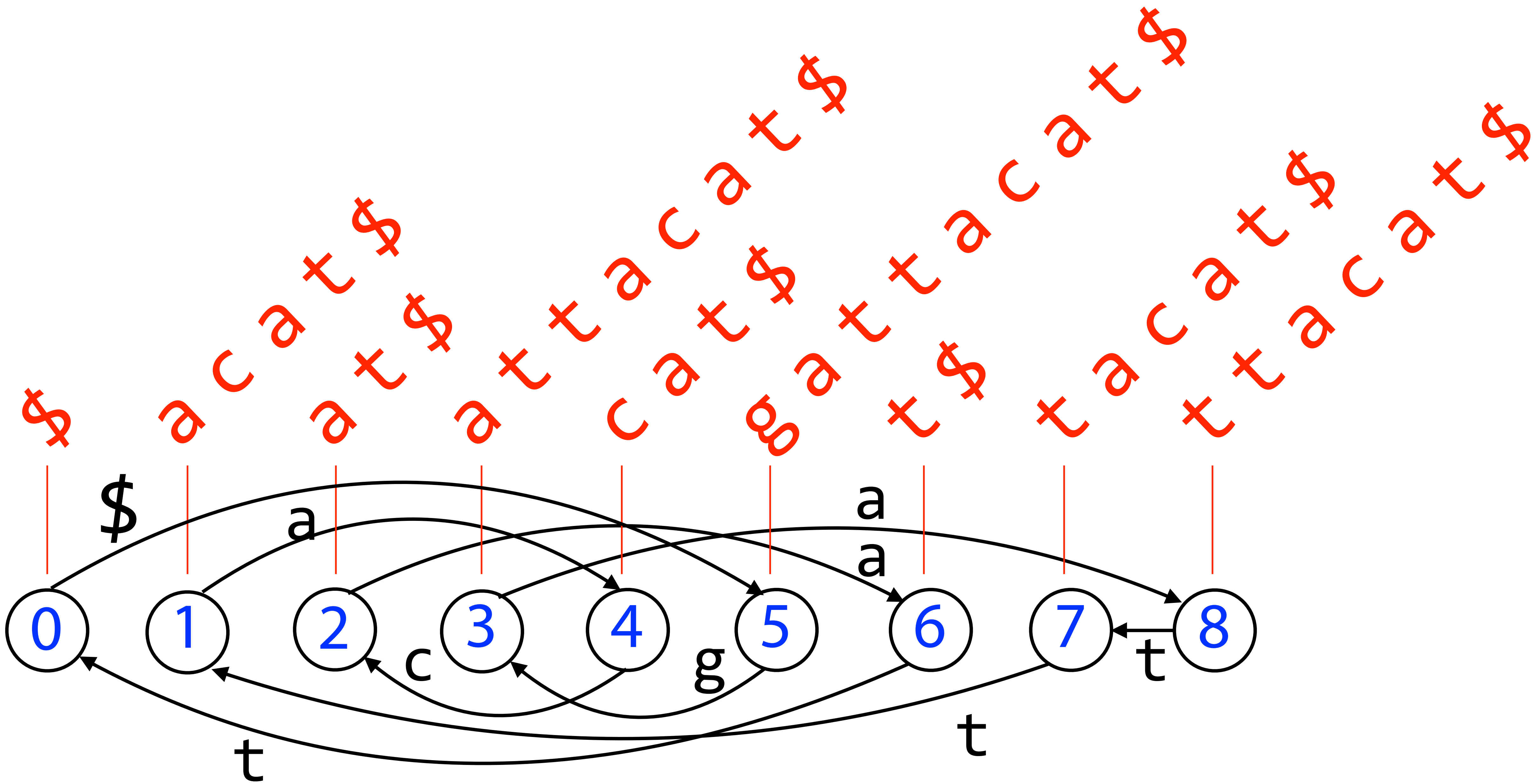
BWT: matching

Nodes can be thought of according to what comes after (outgoing edges) and or just before (incoming)



Incoming edges spell out BWT

BWT: matching



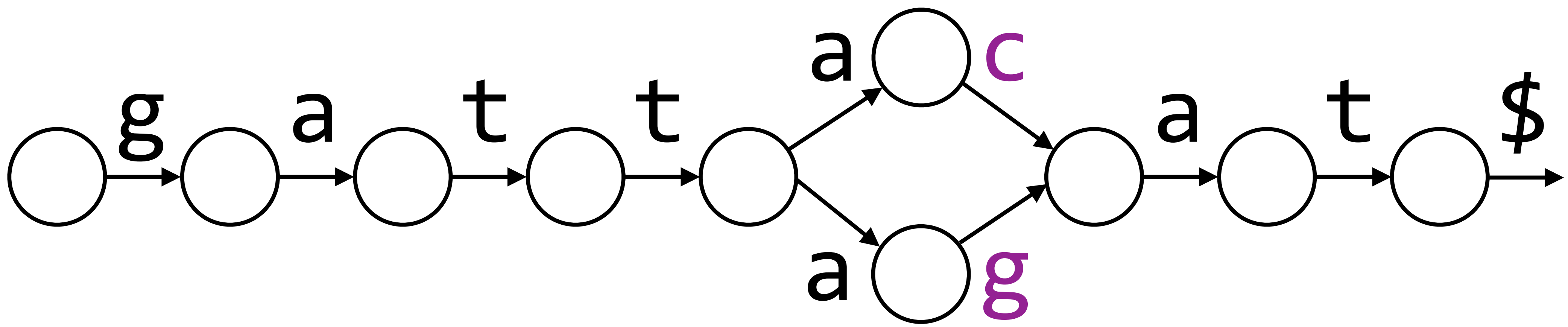
Outgoing paths spell out suffixes/rotations

BWT: matching

Can we go beyond straight-line graphs?

BWT: matching

What does this mean?



g a t t a c a t $

or

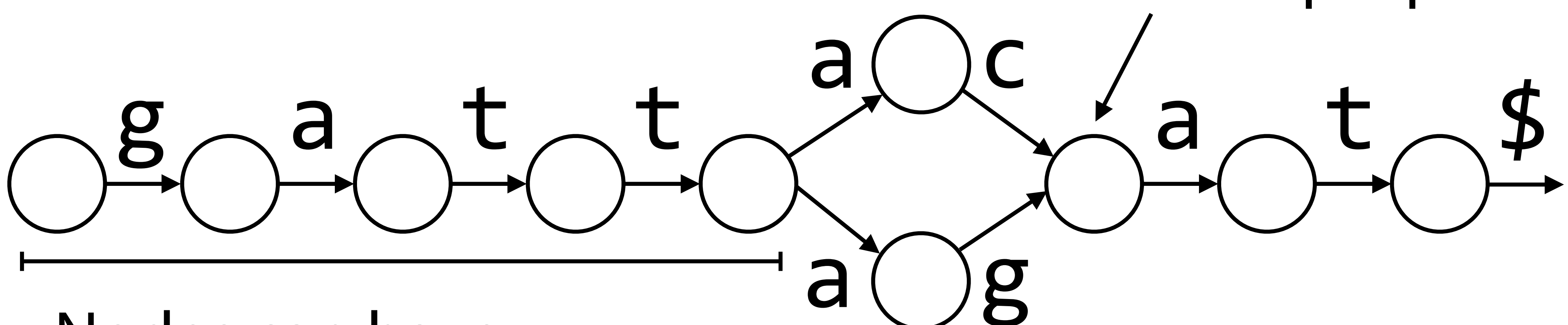
g a t t a g a t $

BWT: matching

Does our way of thinking about nodes still hold?

No:

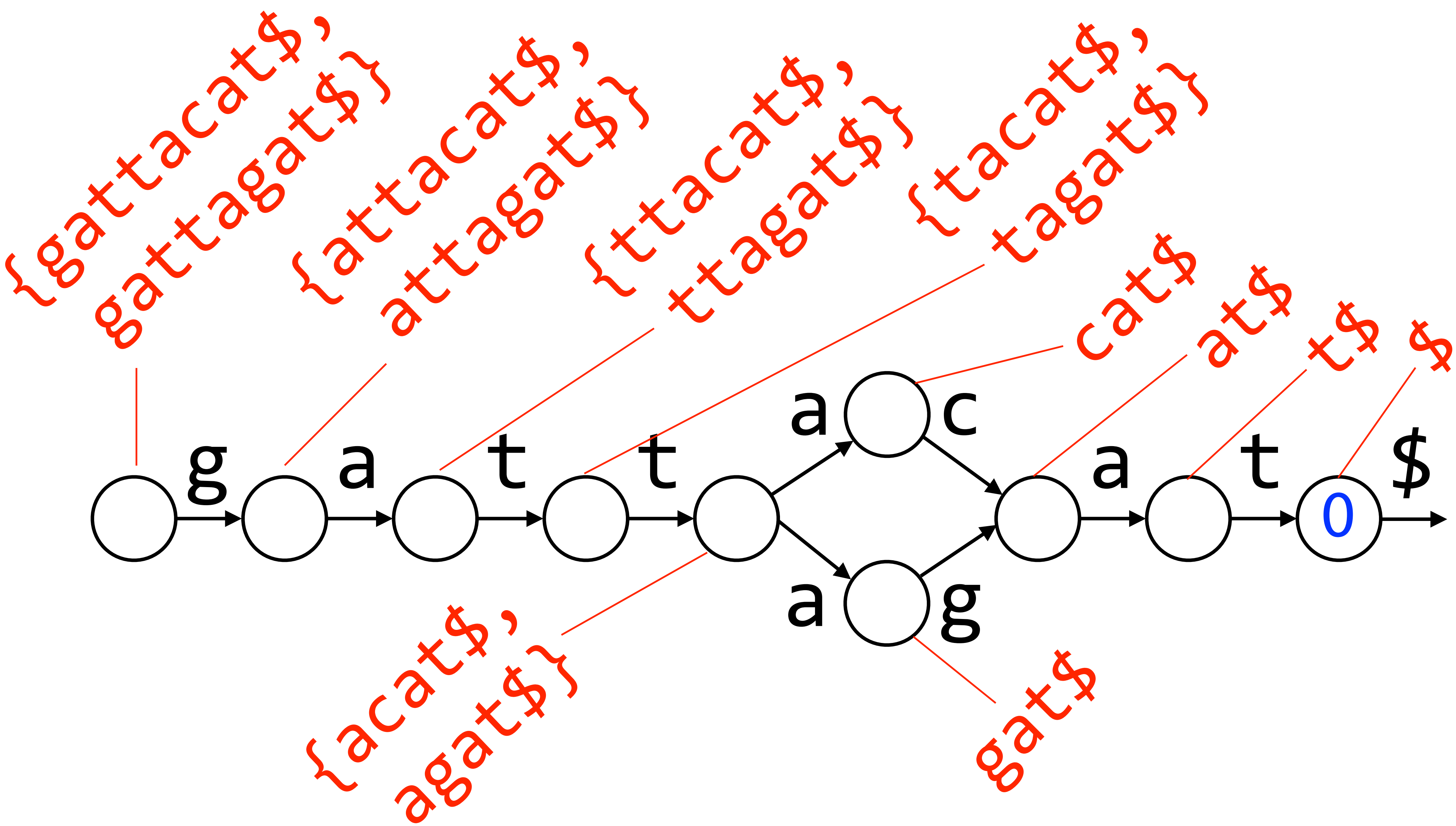
Nodes can have
multiple predecessors



Nodes can have
multiple suffixes
leading out from them

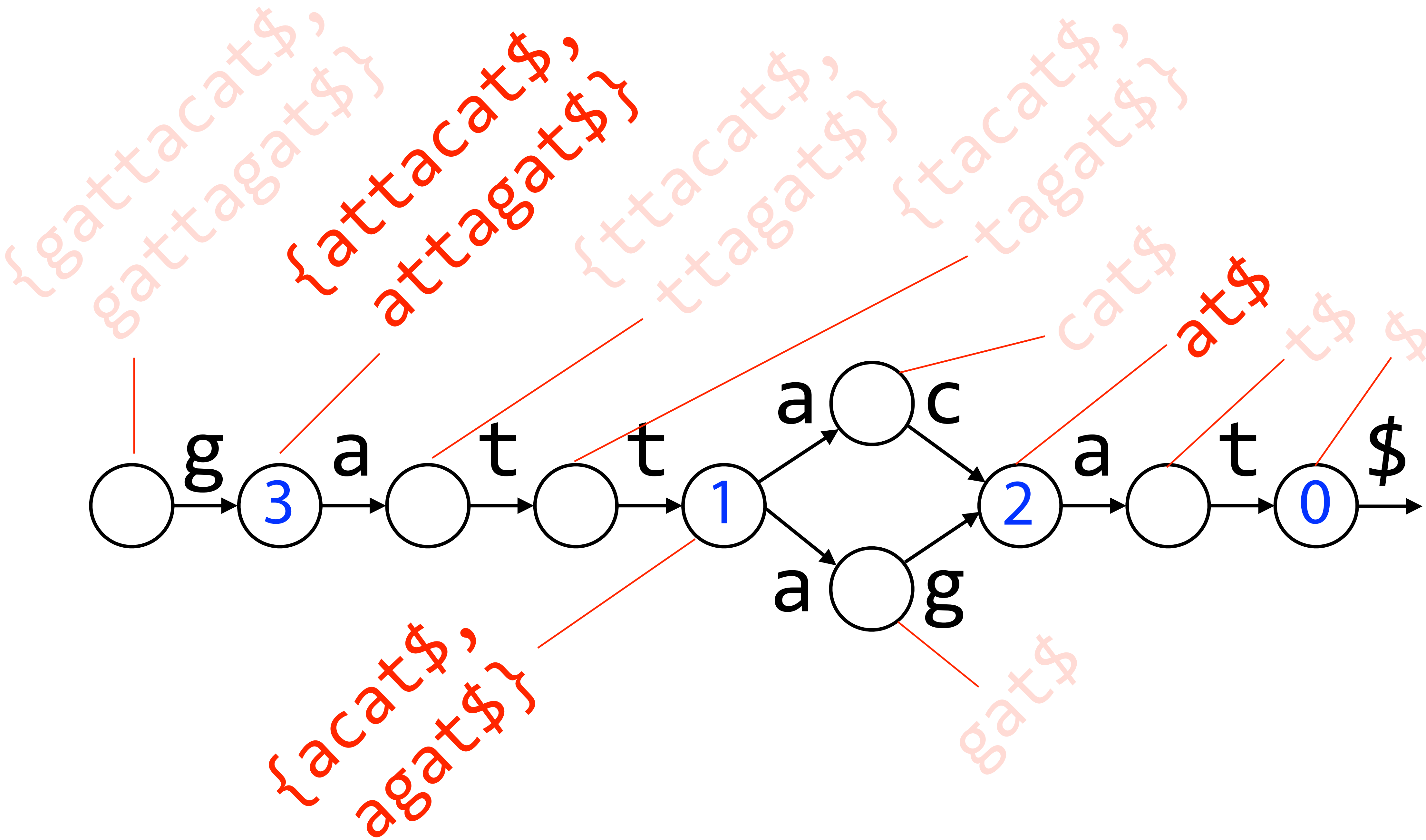
BWT: matching

Can we preserve a total order over outgoing suffixes, even when there's >1 per node?



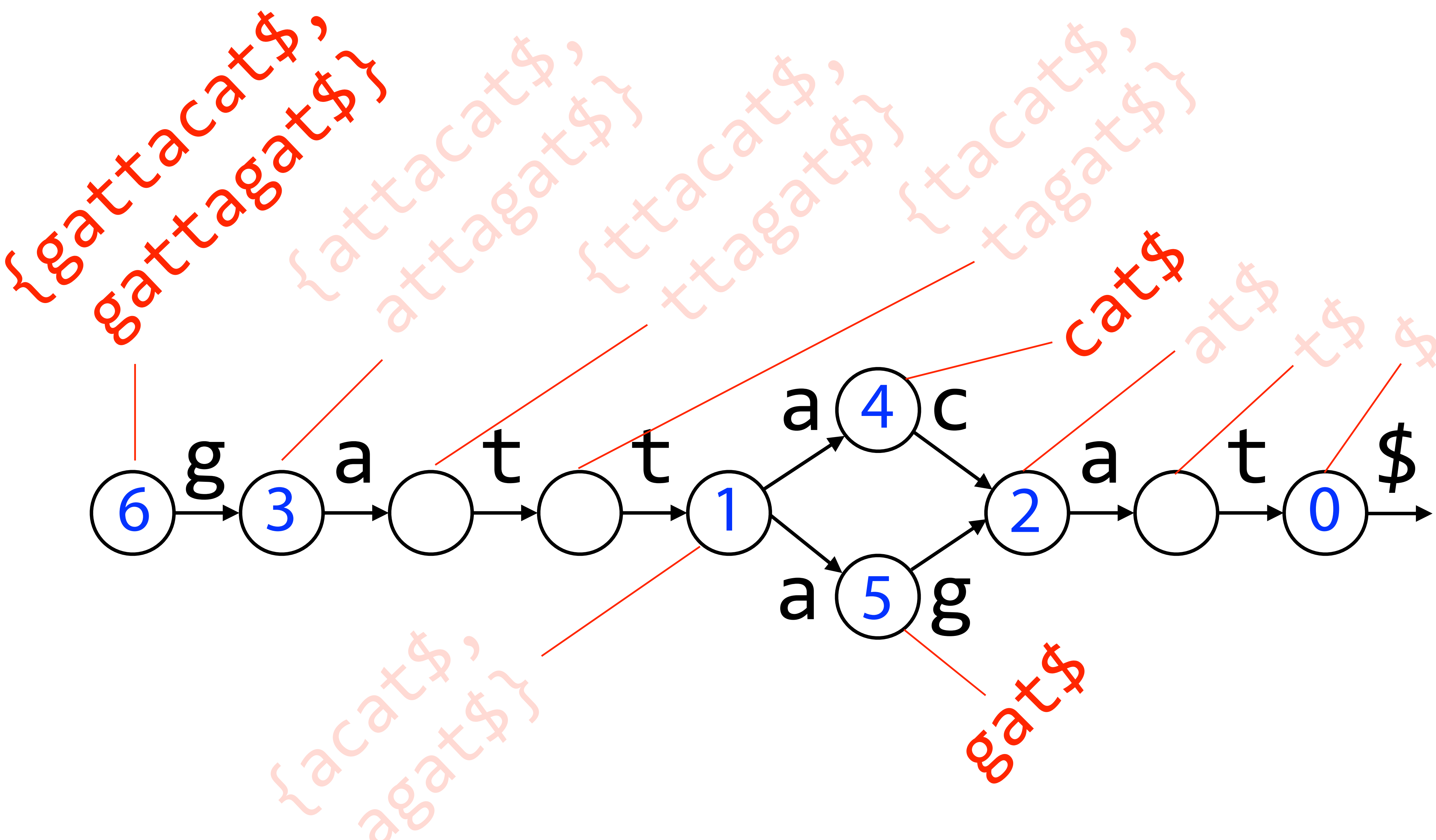
BWT: matching

Can we preserve a total order over outgoing suffixes, even when there's >1 per node?



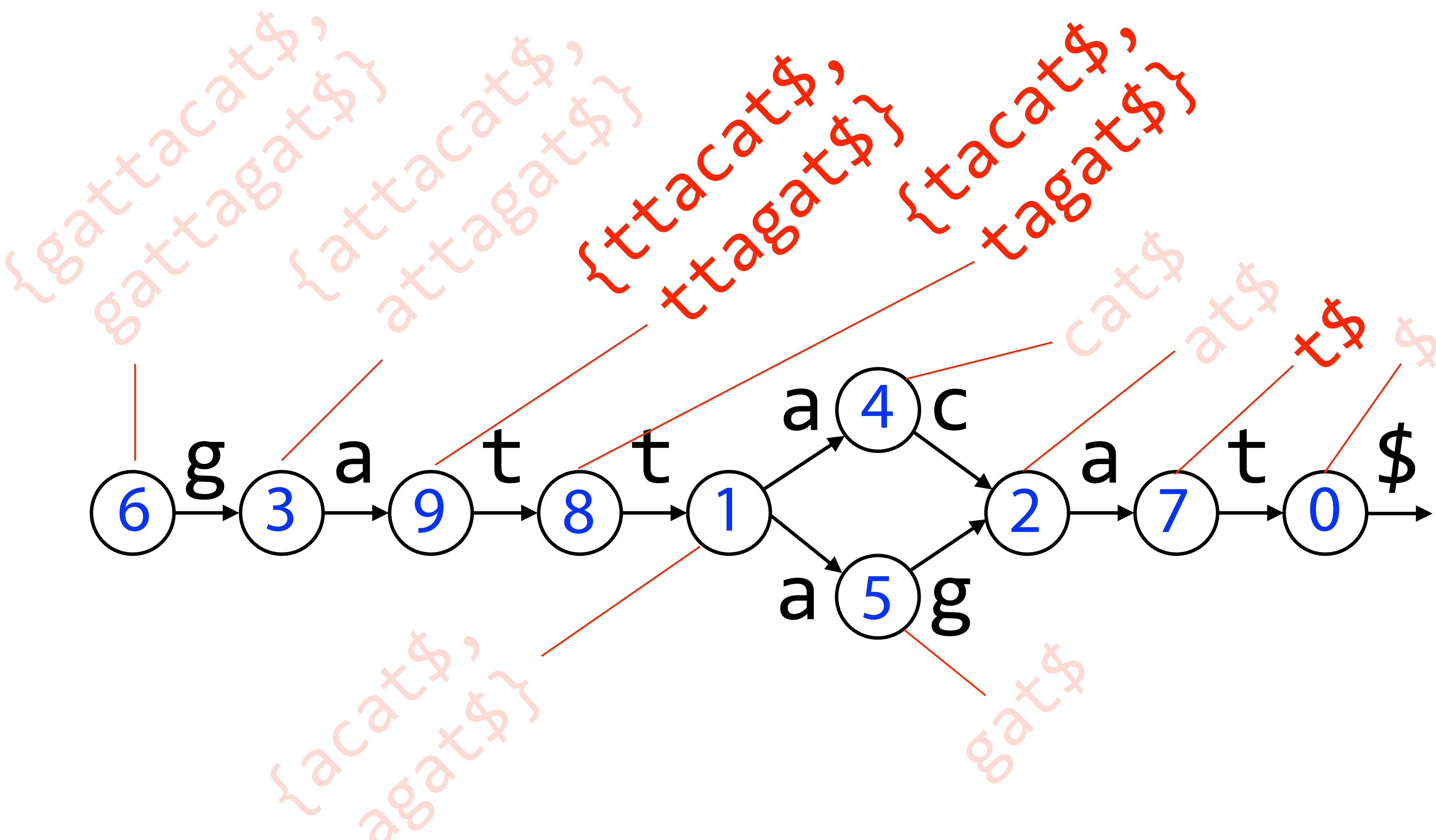
BWT: matching

Can we preserve a total order over outgoing suffixes, even when there's >1 per node?



BWT: matching

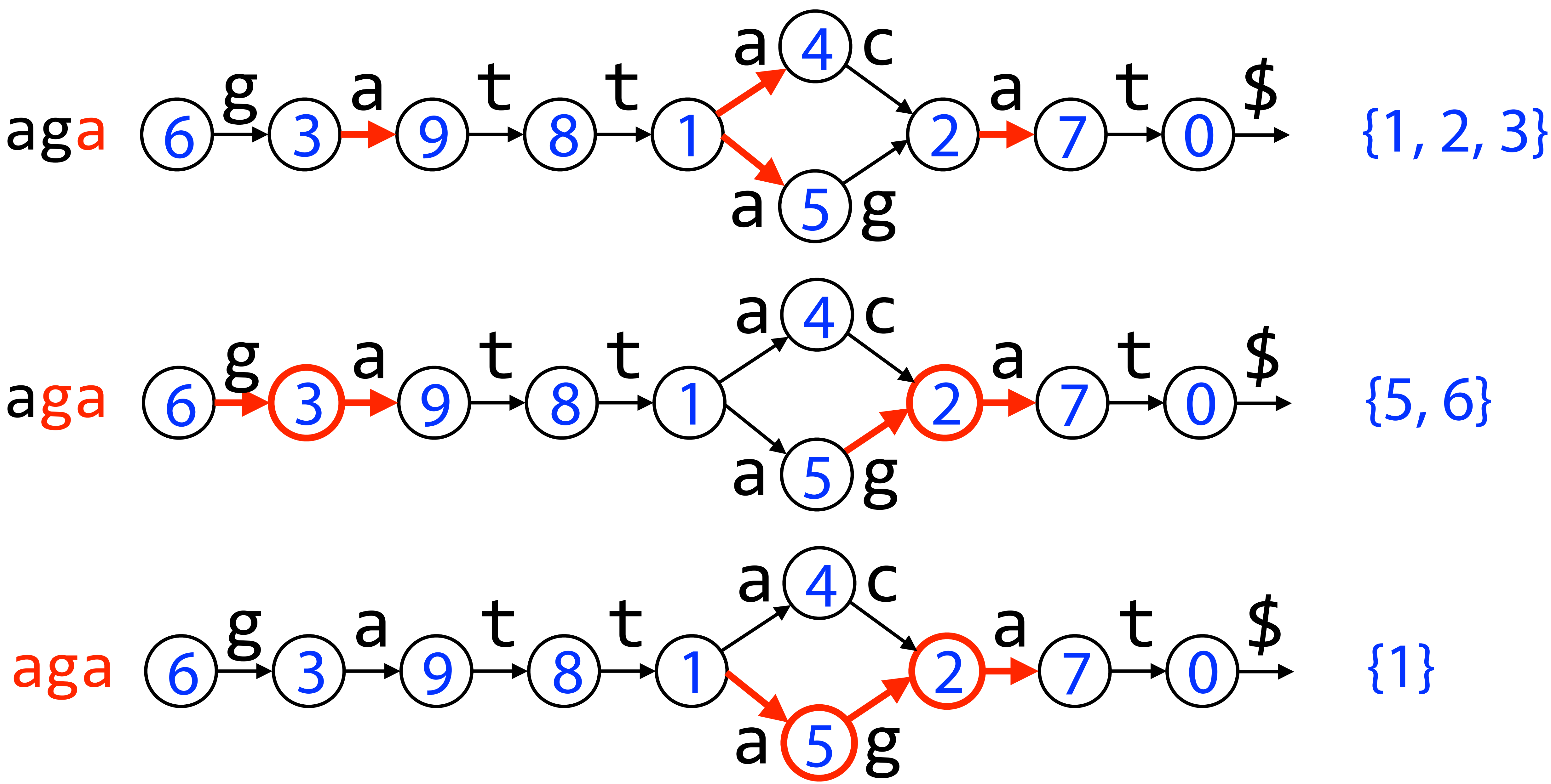
Can we preserve a total order over outgoing suffixes, even when there's >1 per node?



BWT: matching

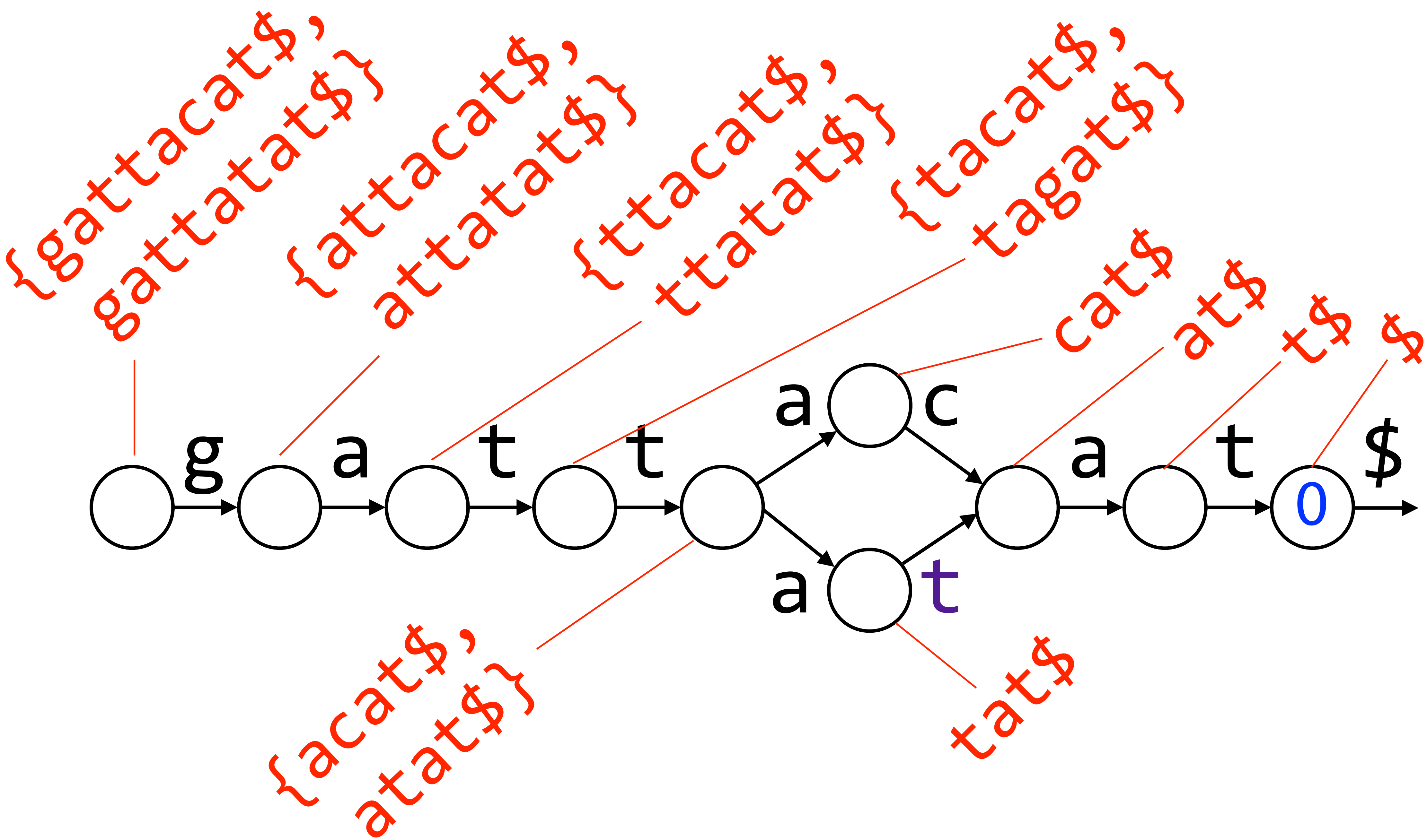
Graph has something like a BW order!

Matching aga, we still have consecutivity.



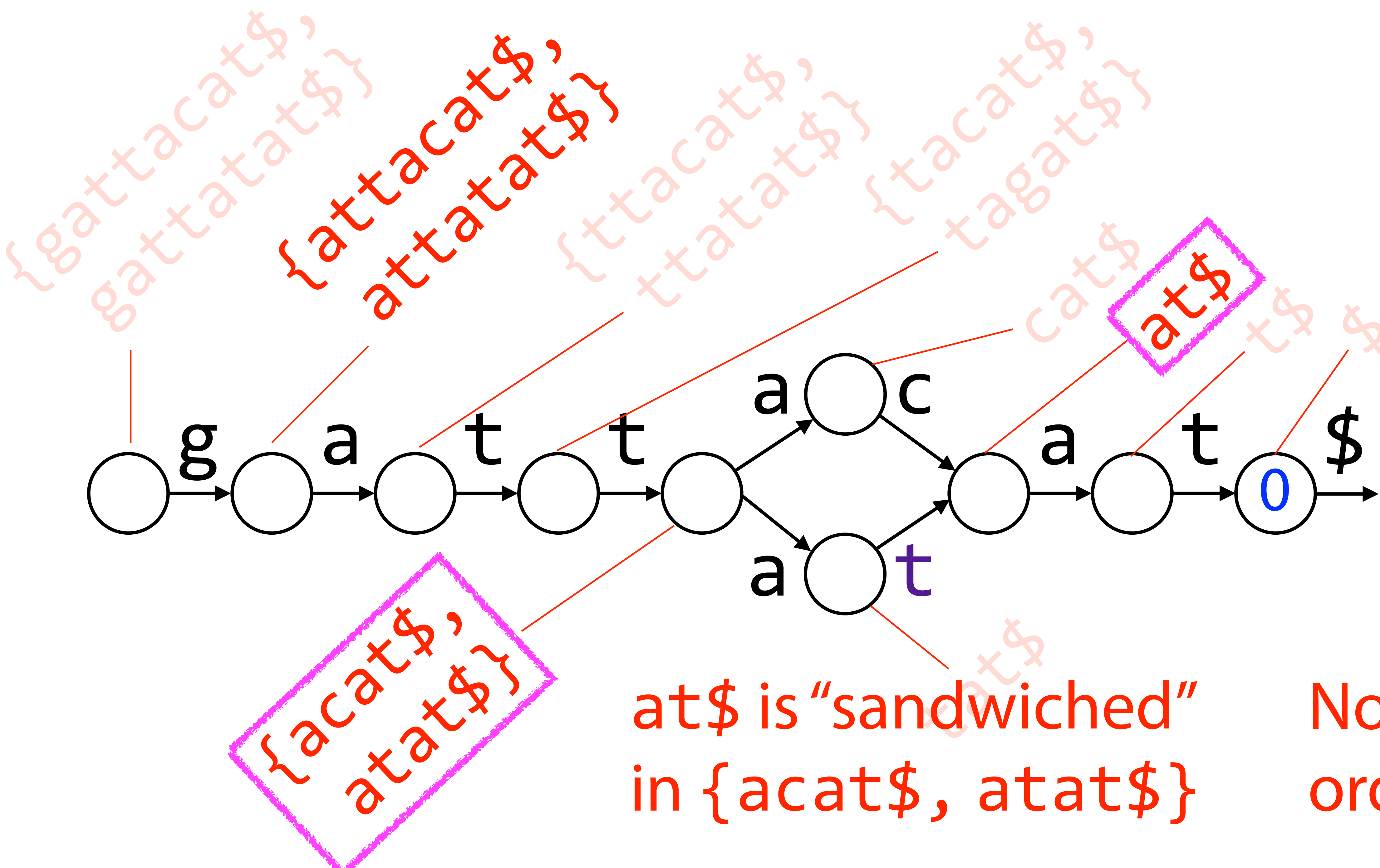
BWT: matching

Does it work for every graph?



BWT: matching

Does it work for every graph?

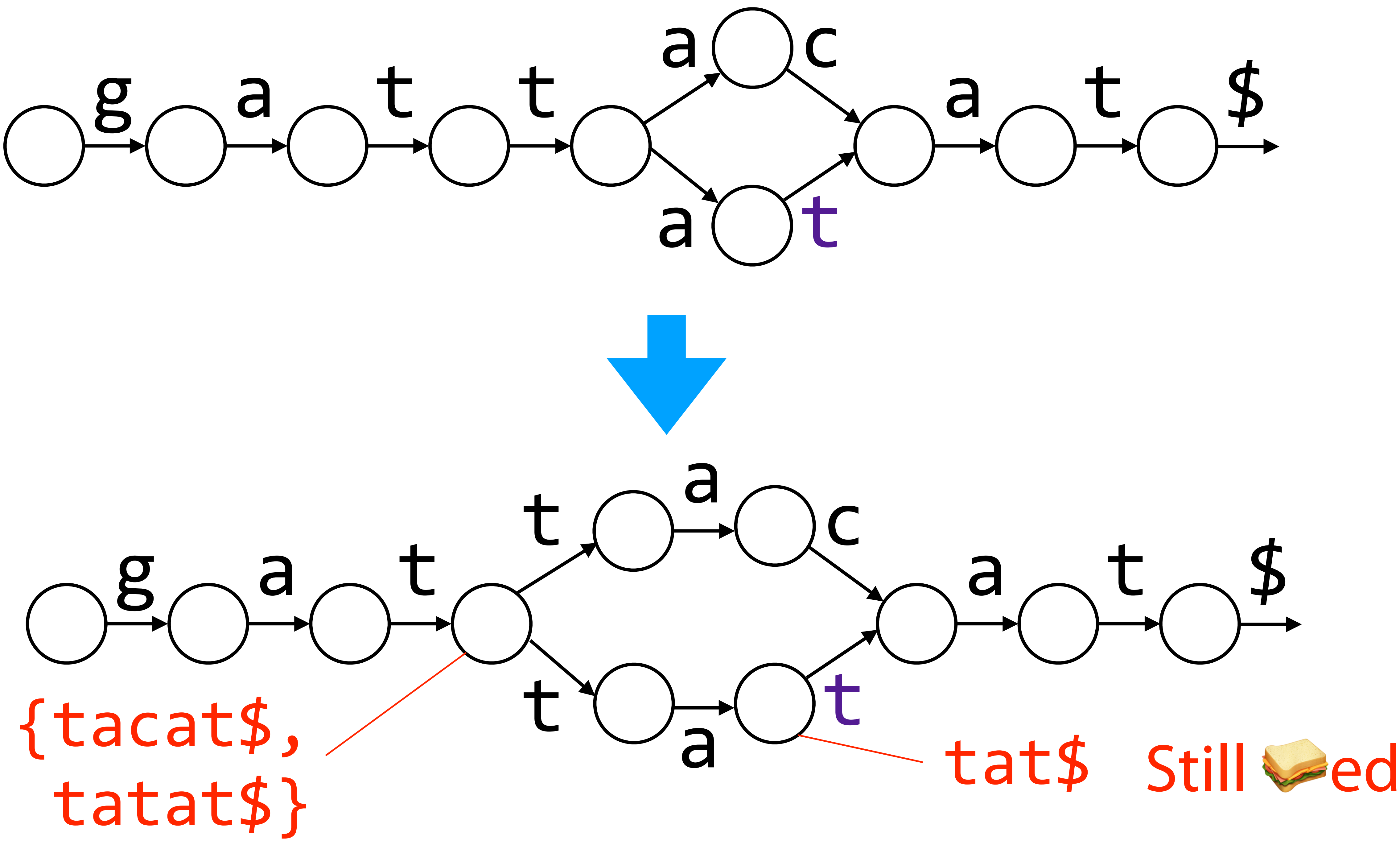


at$ is "sandwiched" in {acat$, atat$}

No total order!

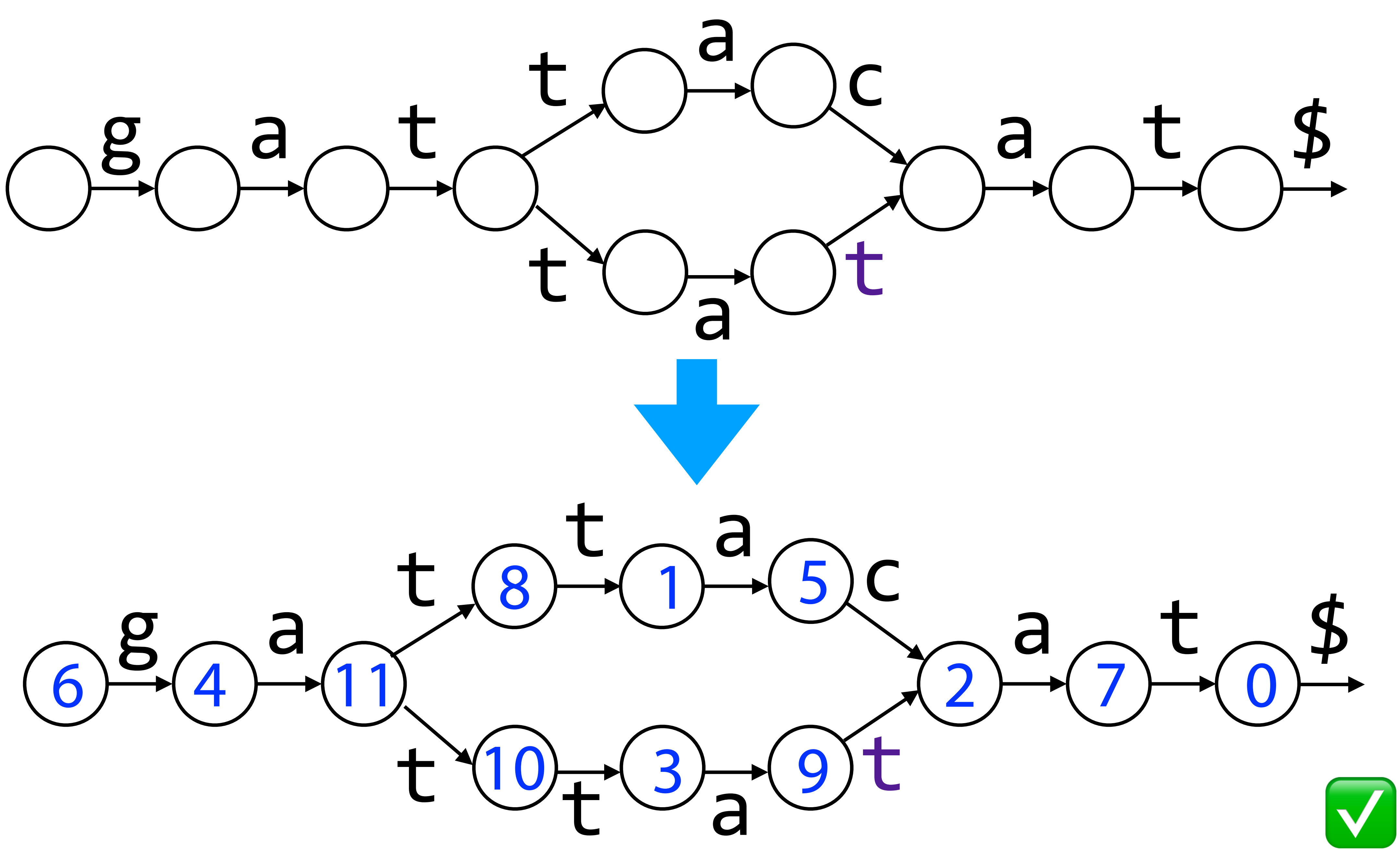
BWT: matching

Can I fix it?



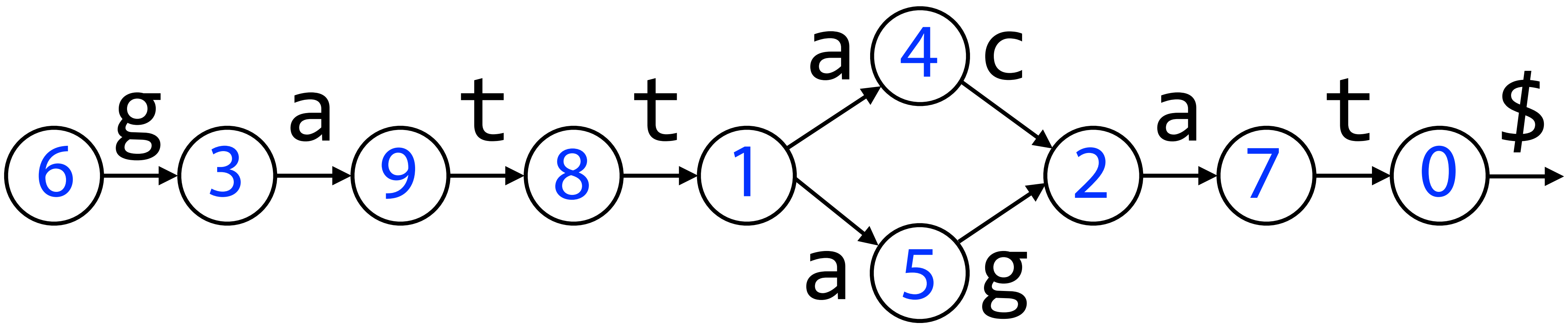
BWT: matching

Can I fix it?

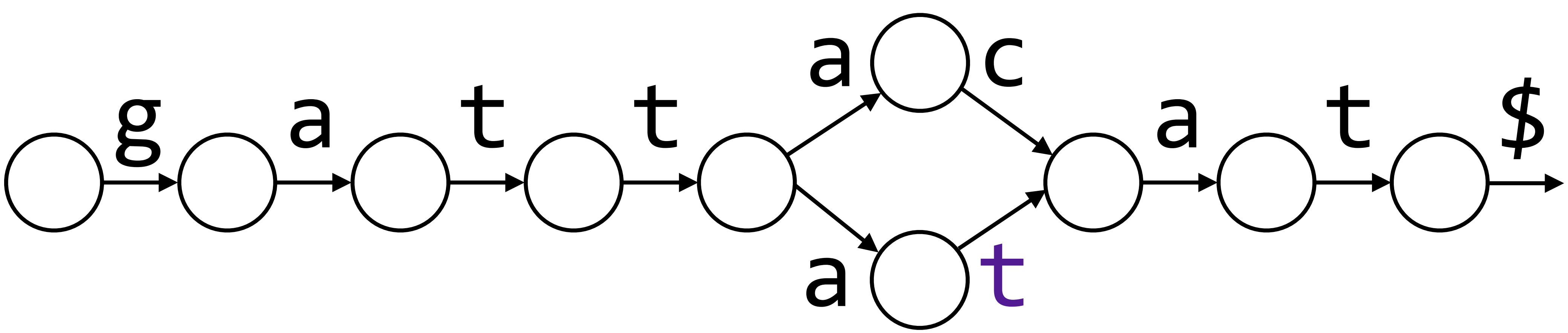


BWT: matching

For some graphs, total order exists



For others, not (but we can “fix" them sometimes)



BWT: matching

Questions:

Which graphs does it work for?

Do these graphs provably have the desired consecutivity property, so we can do matching?

How do we represent and query the graph?

Wheeler graphs

An edge-labeled directed multigraph G is a ***Wheeler Graph*** if nodes can be ordered such that:

1. 0 in-degree nodes come before others
2. For all pairs of edges $e = (u, v), e' = (u', v')$ labeled a, a' respectively, we have:

$$a \prec a' \implies v < v',$$

$$(a = a') \wedge (u < u') \implies v \leq v'.$$

$\prec$ alphabetical, $<$ total order over node labels

Travis Gagie, Giovanni Manzini, and Jouni Sirén. "Wheeler graphs: A framework for BWT-based data structures." *Theoretical computer science* 698 (2017): 67-78.

Wheeler graphs

For each pair of edges:

If edges have different labels, the destination of the edge with the smaller label must come before the destination of the edge with the larger label

Consequence: node cannot have 2 incoming edges with different labels

$$a \prec a' \implies v < v',$$
$$(a = a') \wedge (u < u') \implies v \leq v'.$$

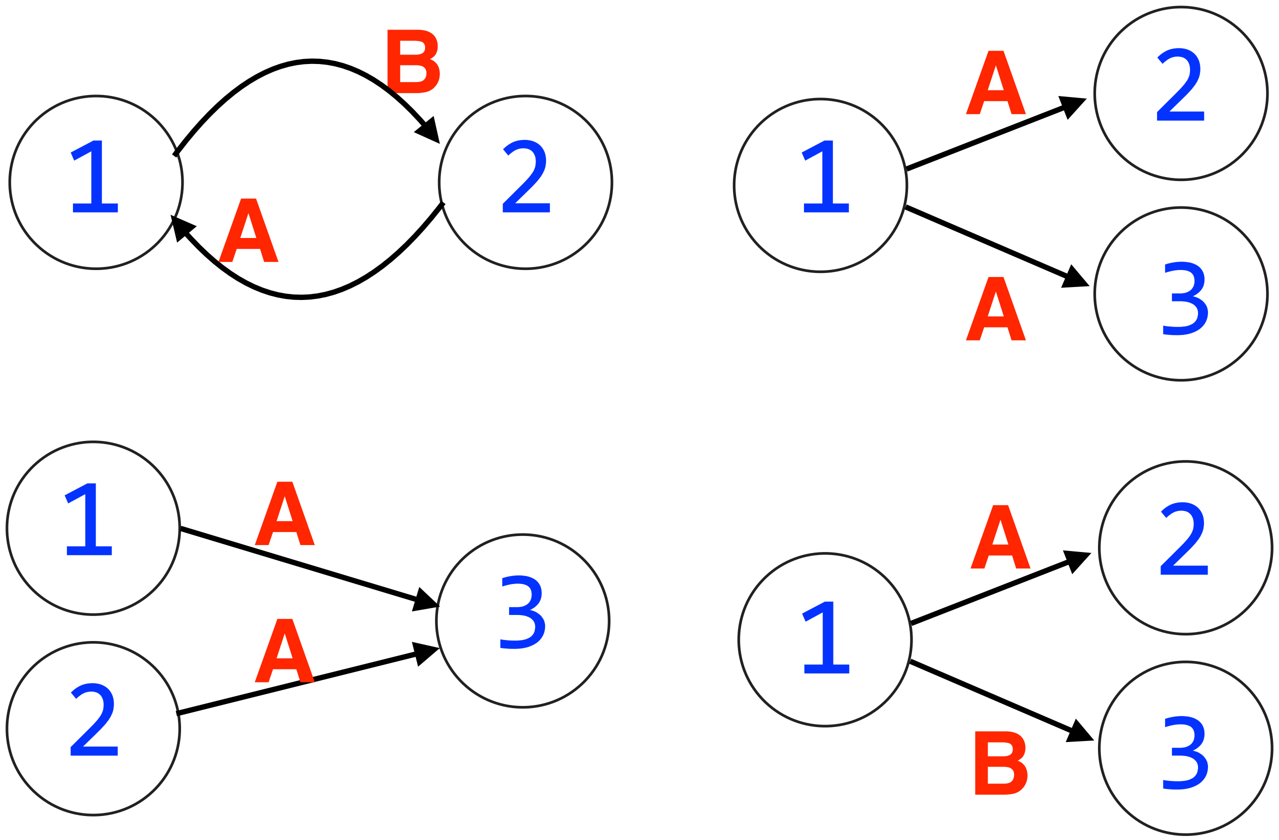
If edges have the same label but different sources, the destination of the edge from the low source must not come after the destination of the edge from the high source

0 in-degree nodes come before others $\quad (1)$

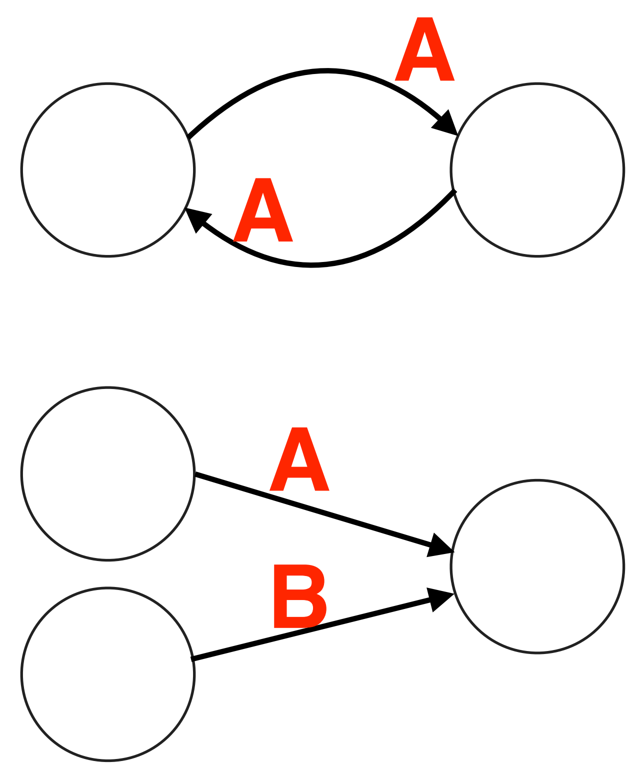
For all pairs of edges

$$\left[\begin{array}{rr} a \prec a' \implies v < v' & (2) \\ (a = a') \wedge (u < u') \implies v \leq v' & (3) \end{array}\right.$$

Wheeler



Not Wheeler



Wheeler graphs

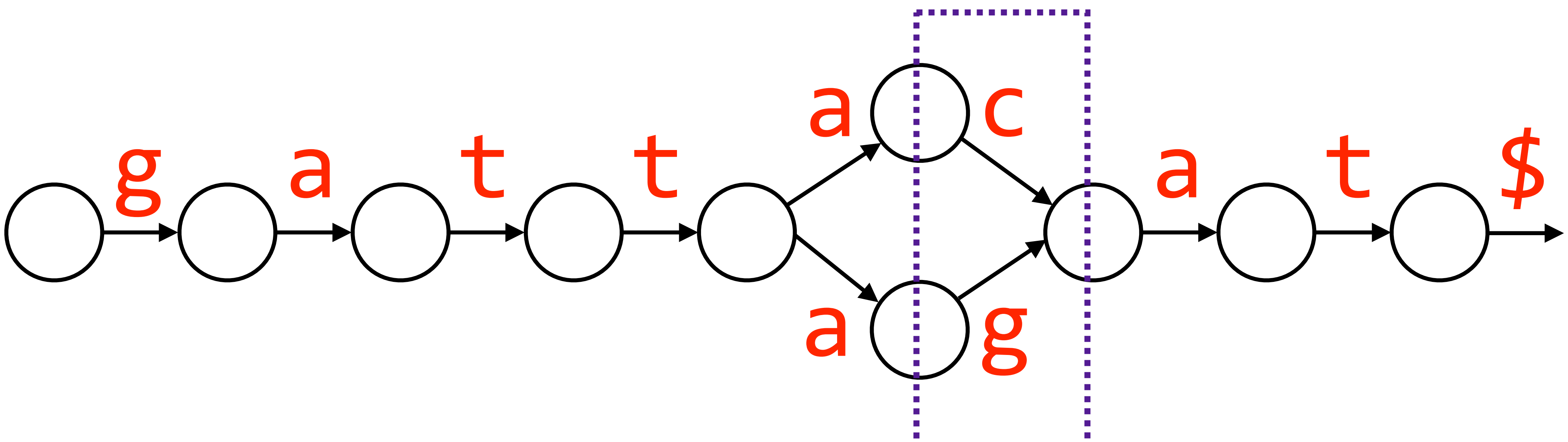
$$\text{0 in-degree nodes come before others} \quad (1)$$

For all pairs of edges

$$a \prec a' \implies v < v' \quad (2)$$

$$(a = a') \wedge (u < u') \implies v \leq v' \quad (3)$$

Is this a Wheeler Graph? **No**



$a \prec a'$ but $v = v'$ (2) cannot hold

Wheeler graphs

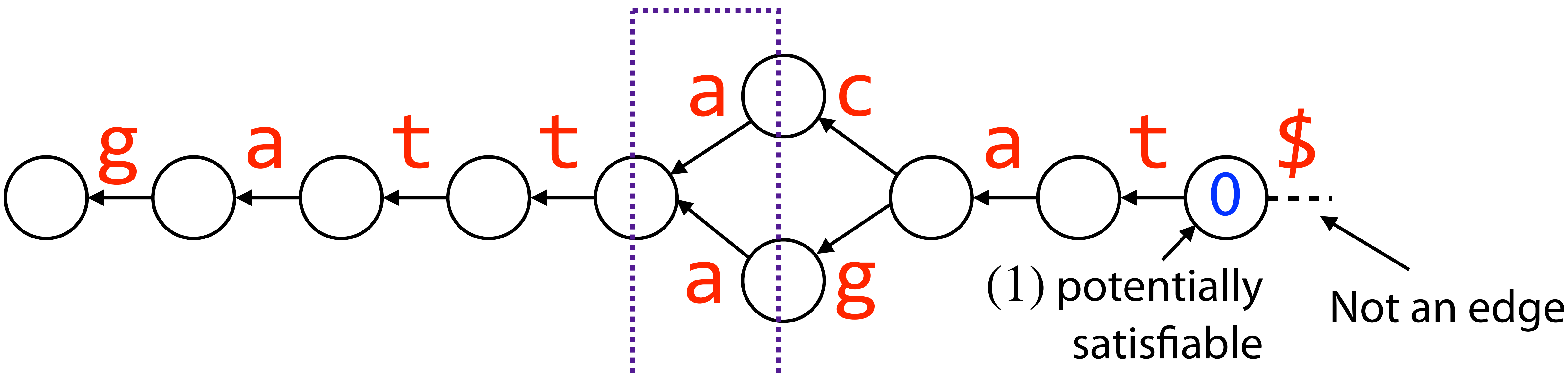
0 in-degree nodes come before others $\quad (1)$

For all pairs of edges

$$a \prec a' \implies v < v' \quad (2)$$

$$(a = a') \wedge (u < u') \implies v \leq v' \quad (3)$$

What if we flip edges to follow the direction of matching?



$a = a'$ and $v = v'$, so (3) is satisfied whether or not $u < u'$

Wheeler graphs

0 in-degree nodes come before others $\quad (1)$

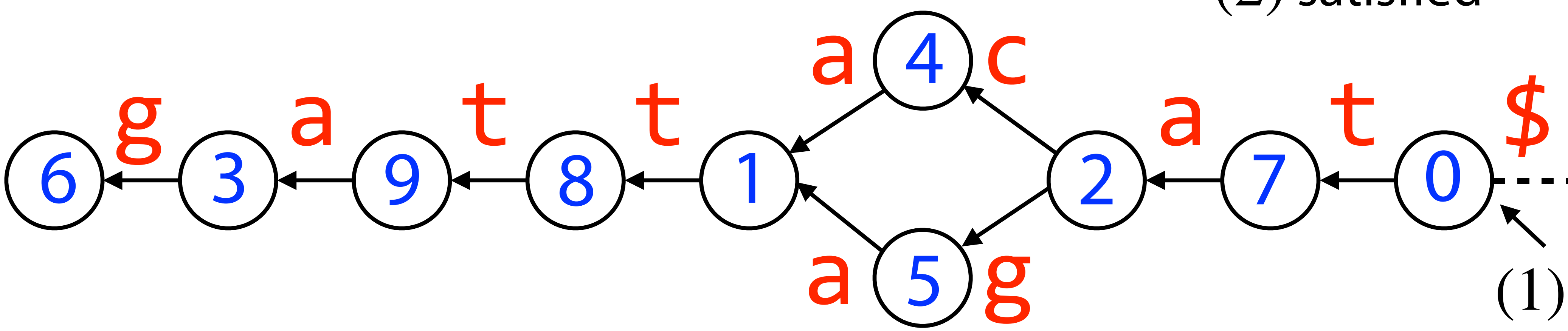
For all pairs of edges

$$a \prec a' \implies v < v' \quad (2)$$

$$(a = a') \wedge (u < u') \implies v \le v' \quad (3)$$

Successors of edges labeled: a : $\{1, 2, 3\}$ g : $\{5, 6\}$ c : $\{4\}$ t : $\{7, 8, 9\}$

(2) satisfied



(1) satisfied

Exercise: prove (3) is satisfied for all pairs of edges

✅ Wheeler