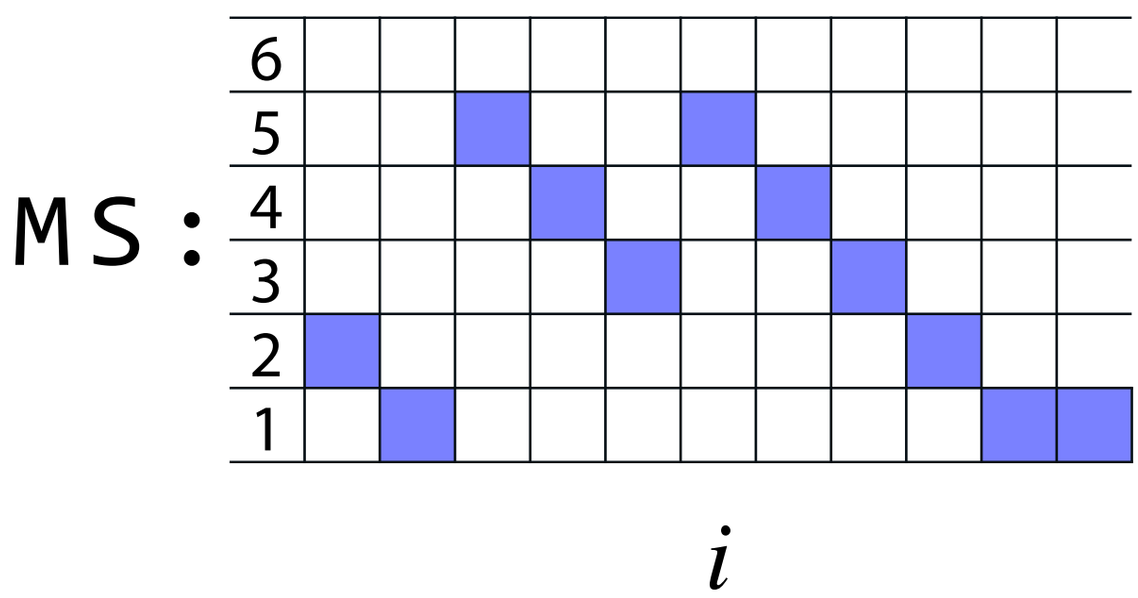
Matching statistics on the suffix tree

Ben Langmead

Matching statistics

```
 T:  abracadabradad
 P:  caradadabrd
MS:  2154354321 1
```

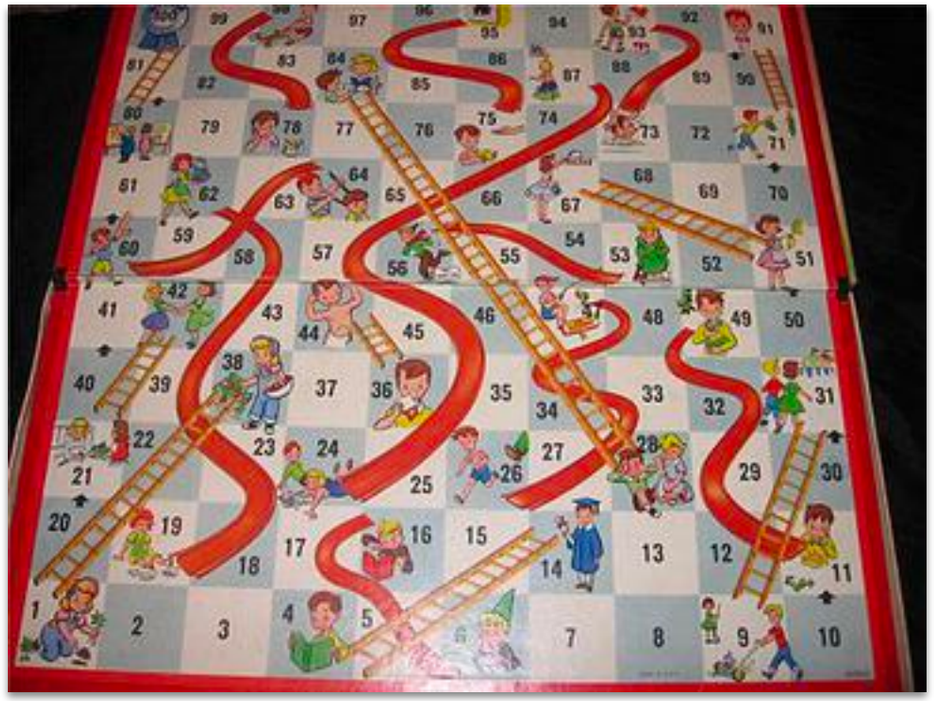


Matching statistics with the suffix tree

As we move along the tree, we use suffix links to carry over partial matches as much possible

Instead of falling off, we **reposition**

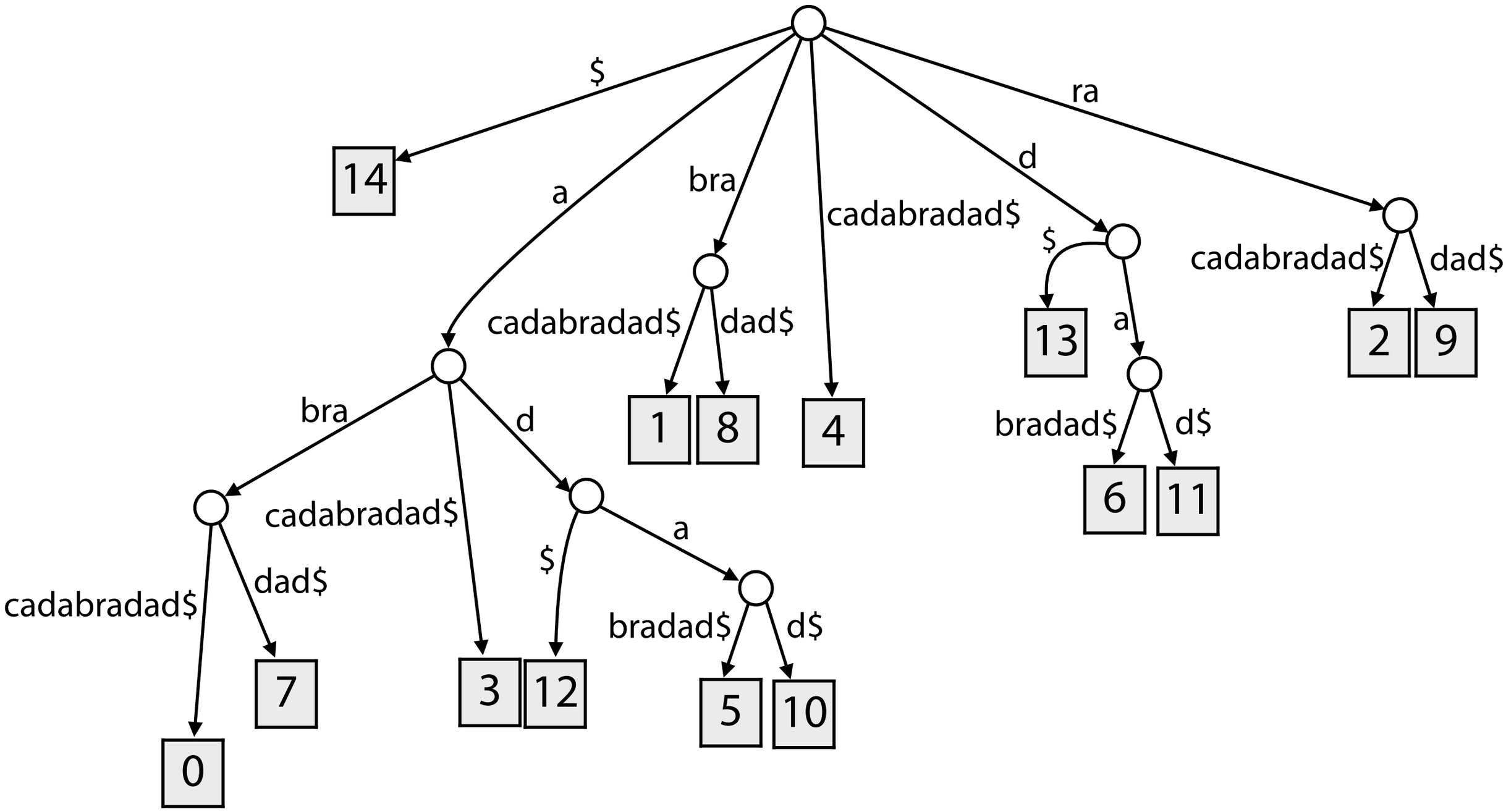
Suffix links are **short cuts**



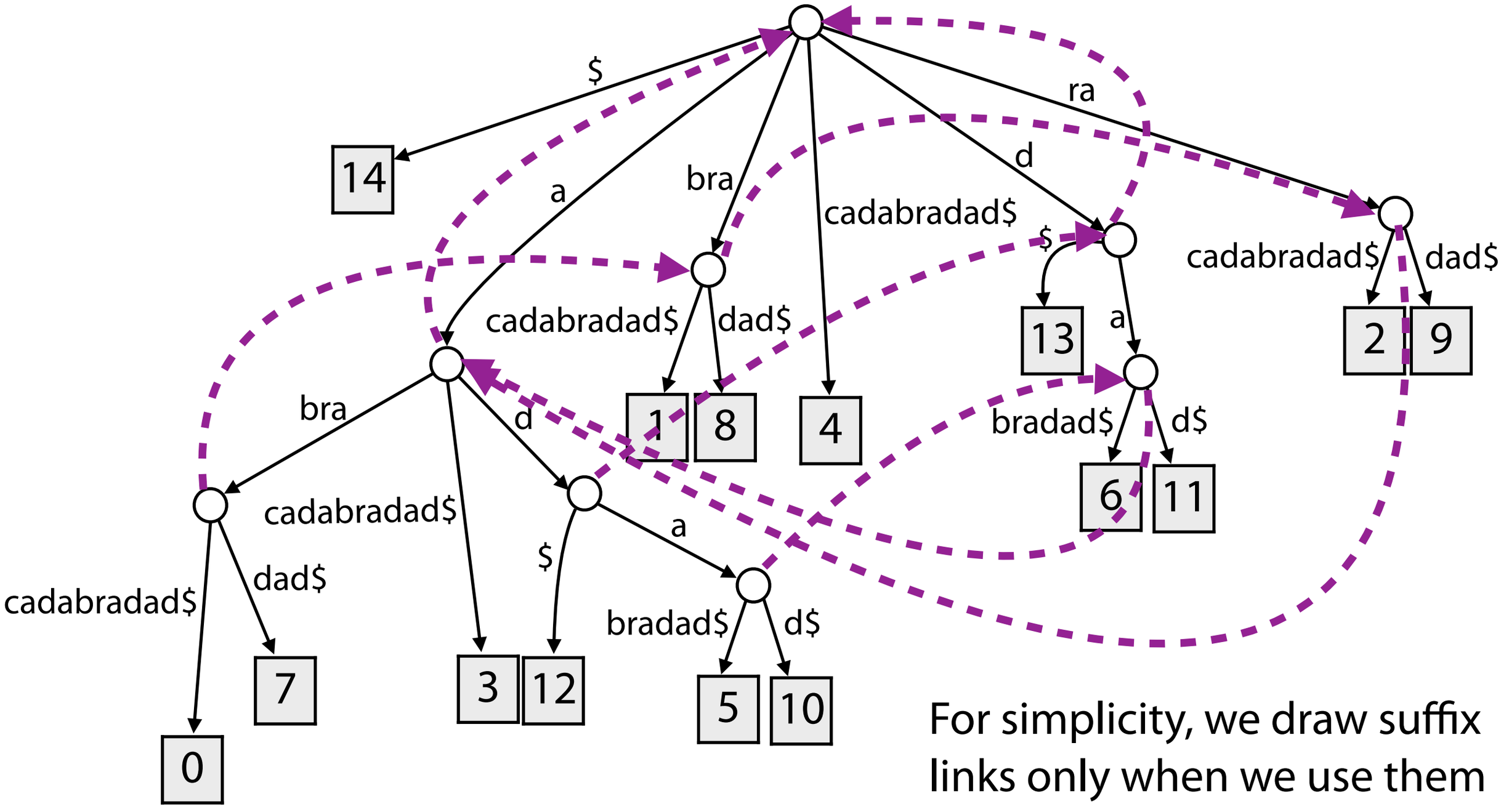
https://en.wikipedia.org/wiki/Snakes_and_ladders

Matching statistics

T = abracadabradad$

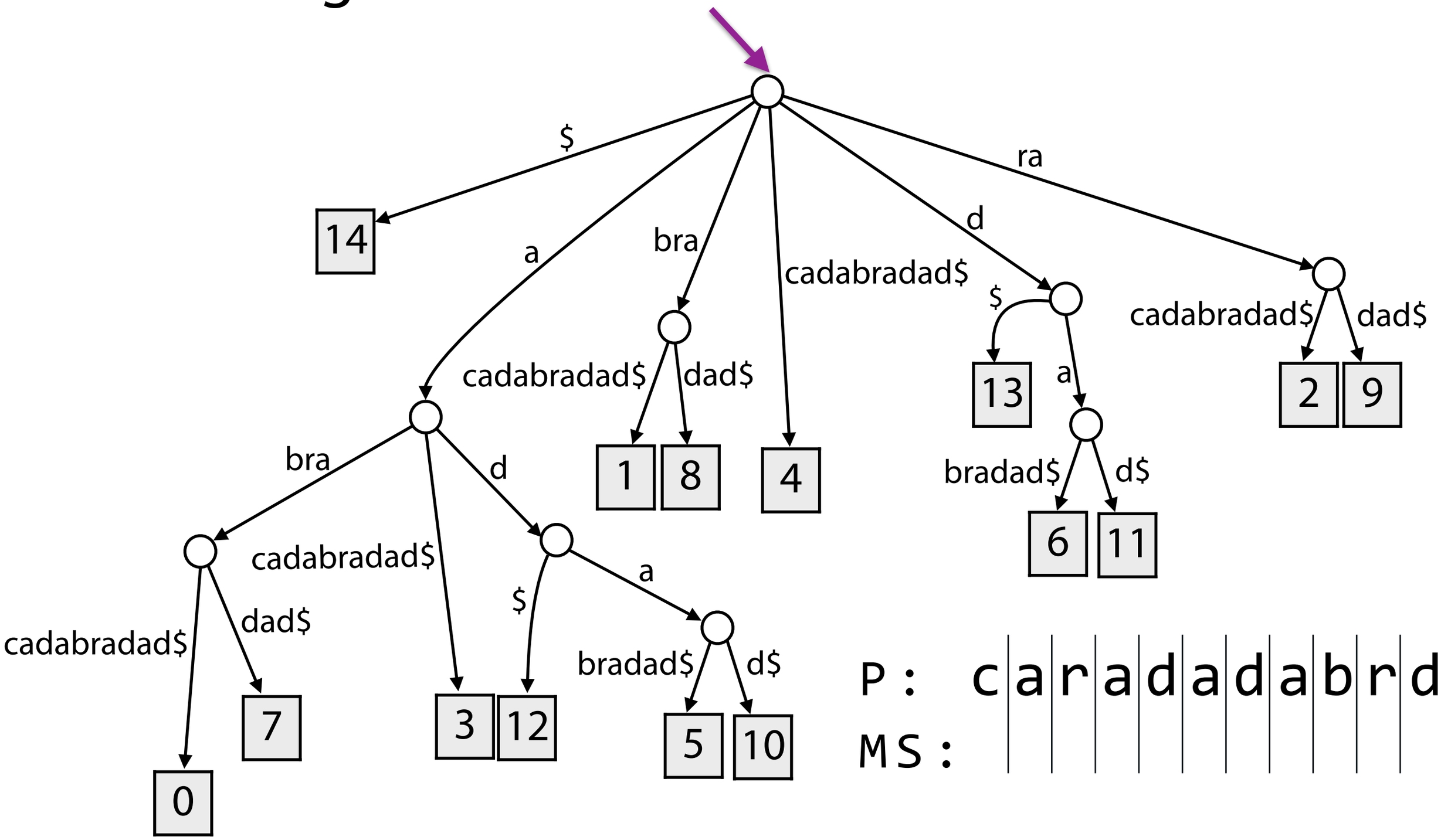


Matching statistics



For simplicity, we draw suffix links only when we use them

Matching statistics

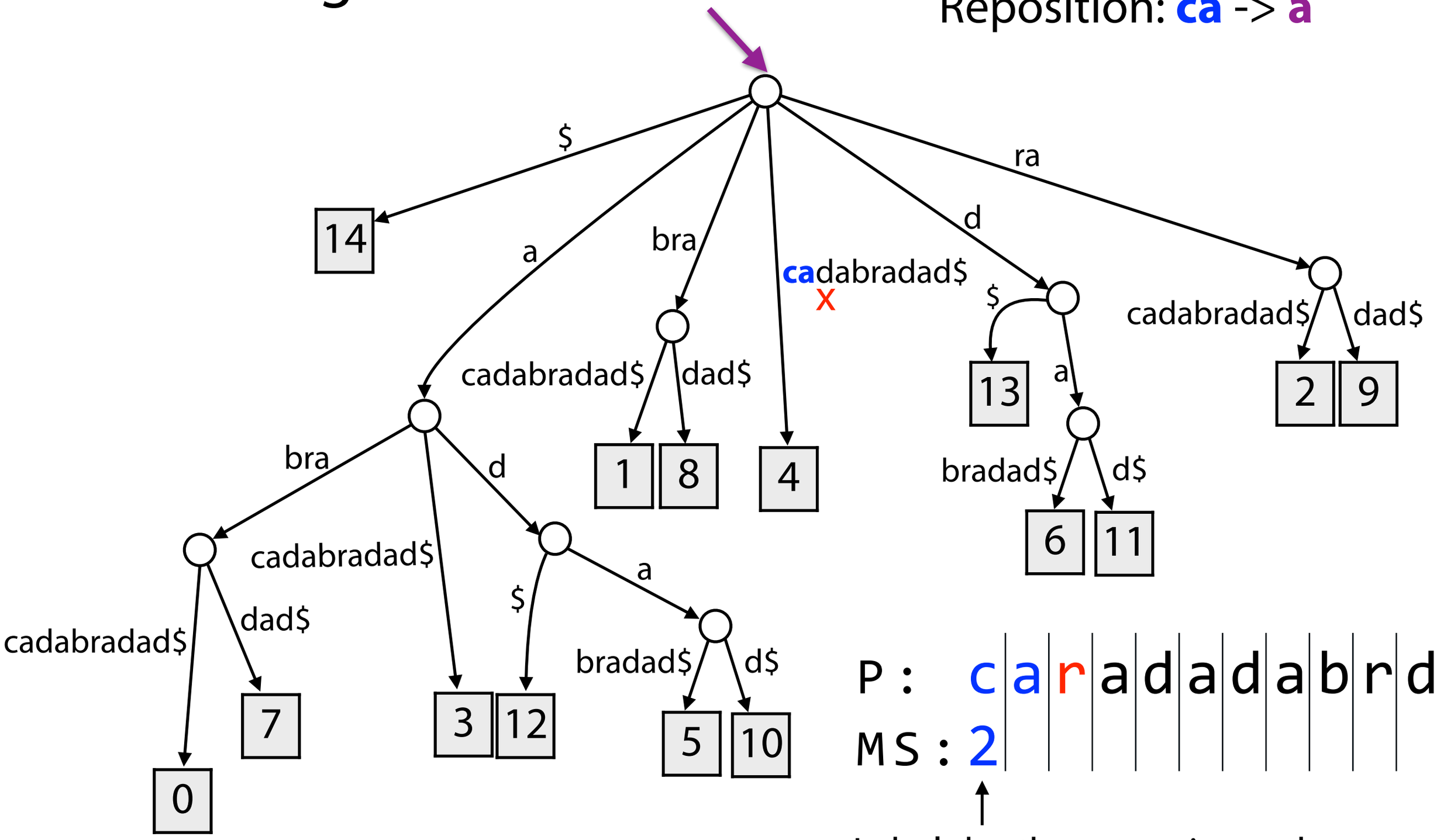


P: c a r a d a d a b r d

MS:

Matching statistics

Reposition: **ca** -> **a**



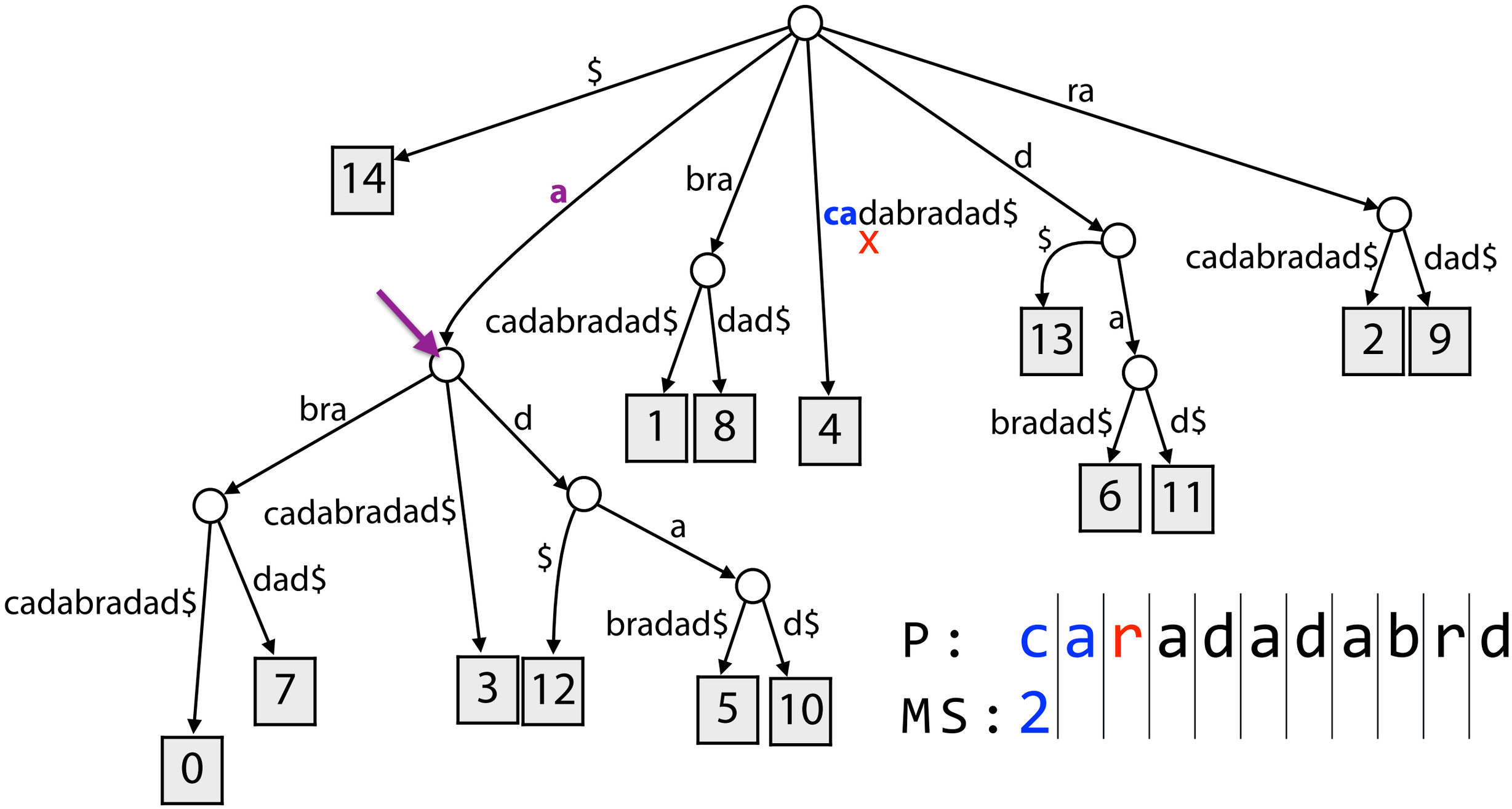
P: c a r a d a d a b r d

MS: 2

Label depth up to mismatch

Matching statistics

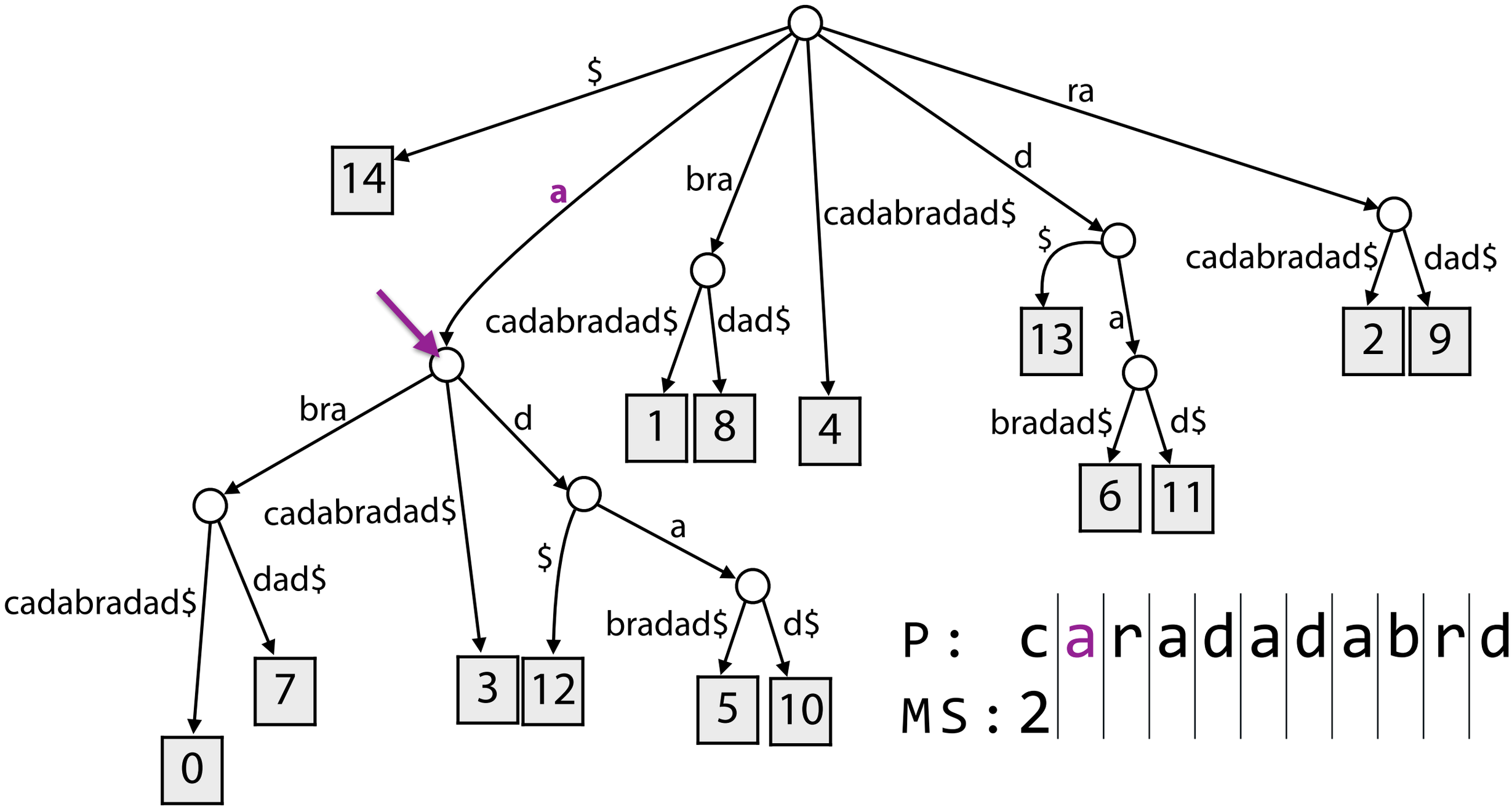
Reposition: **ca** -> **a**



P: c a r a d a d a b r d

MS: 2

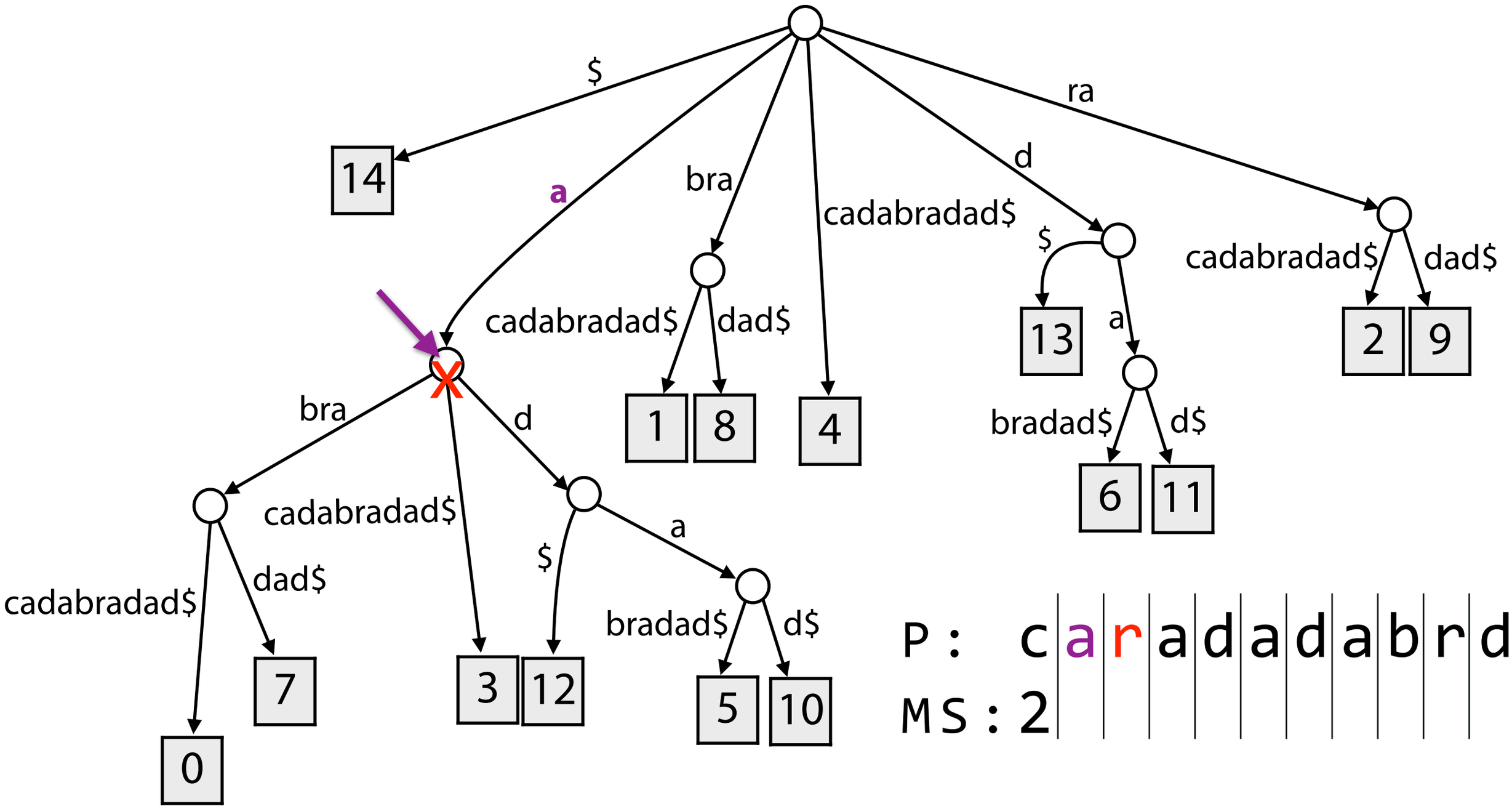
Matching statistics



P: c a r a d a d a b r d

MS: 2

Matching statistics

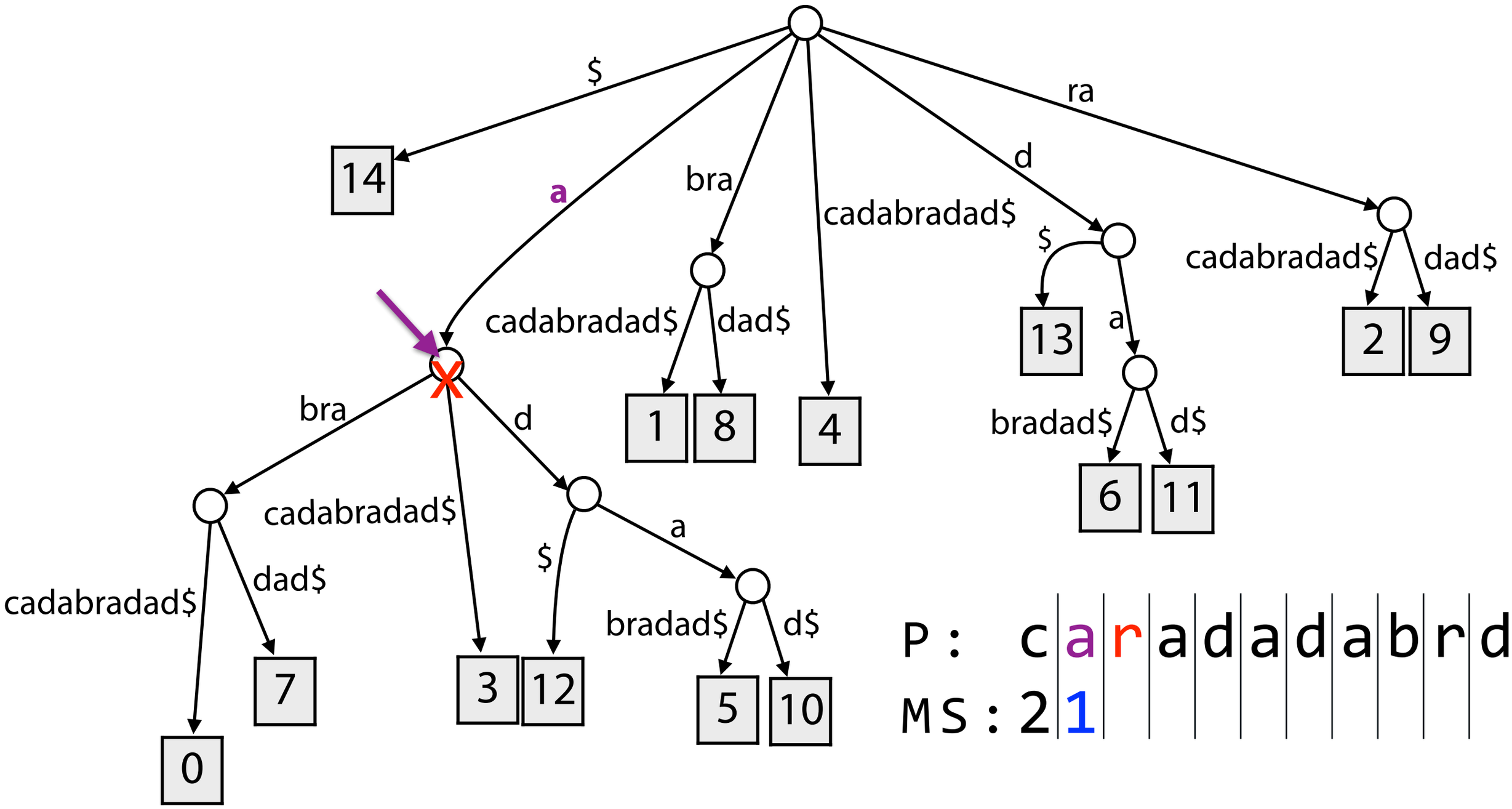


P: c a r a d a d a b r d

MS: 2

Matching statistics

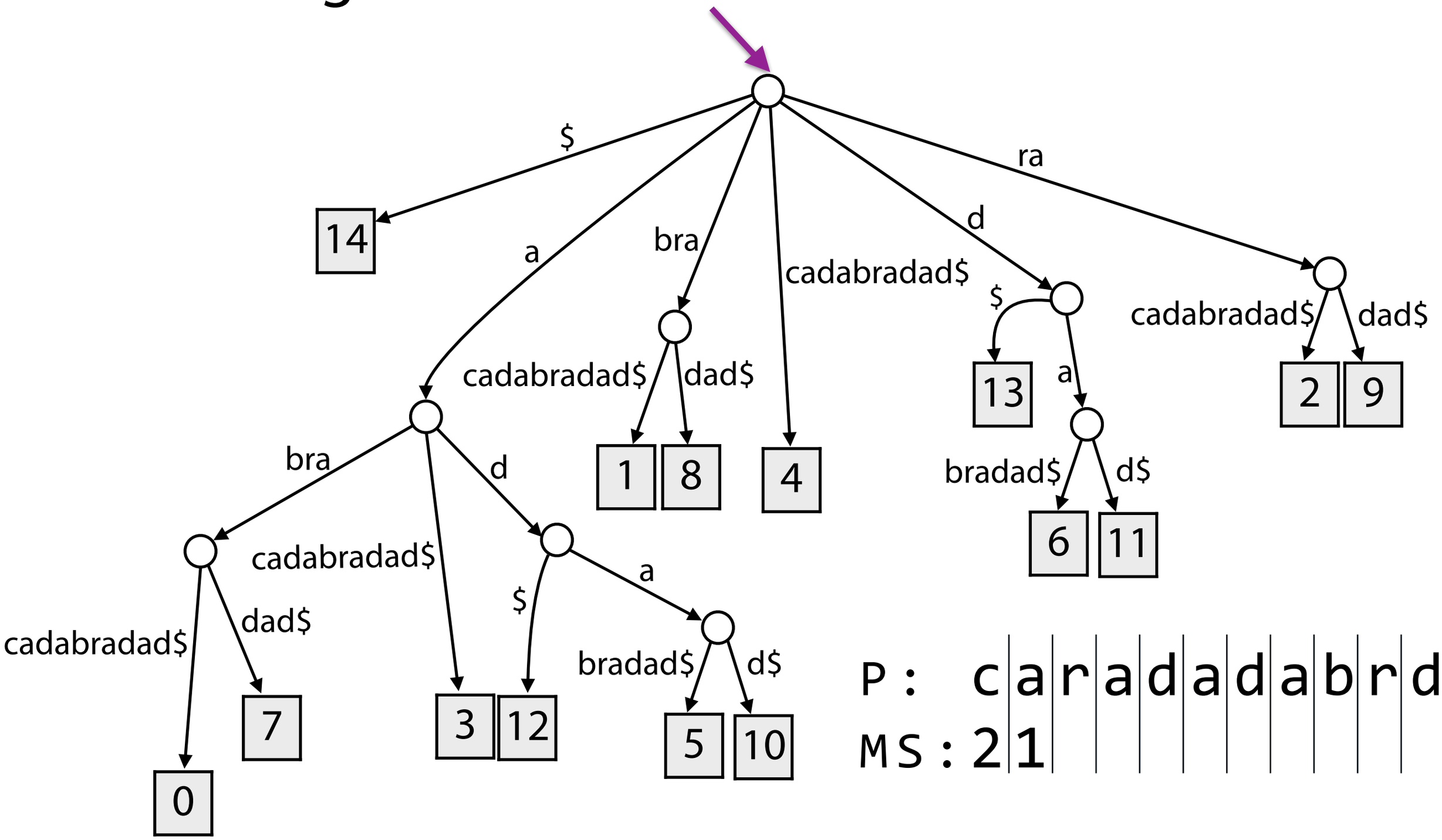
Reposition: **a** -> ϵ



P: c a r a d a d a b r d

MS: 2 1

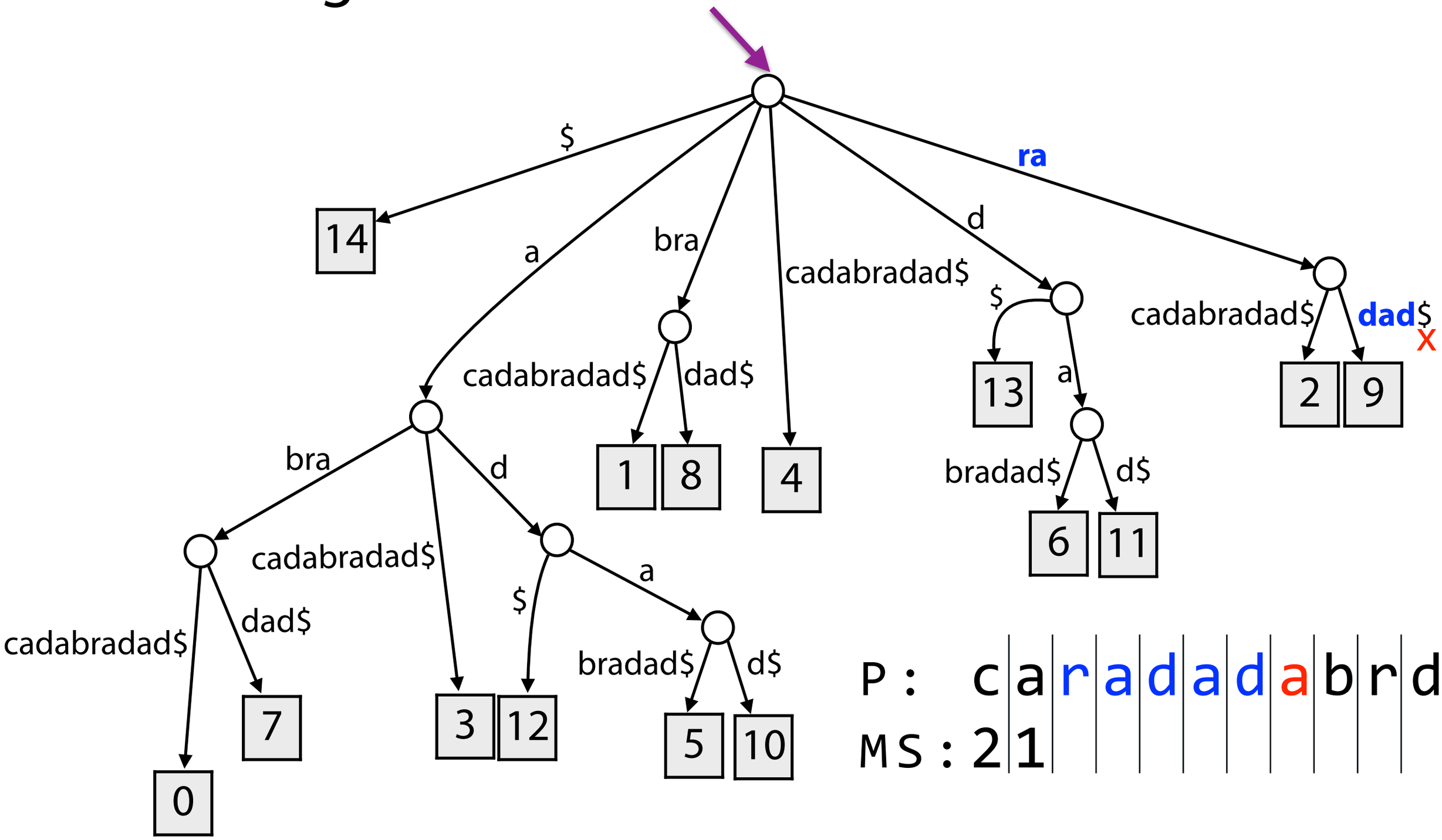
Matching statistics



P: c a r a d a d a b r d

MS: 2 1

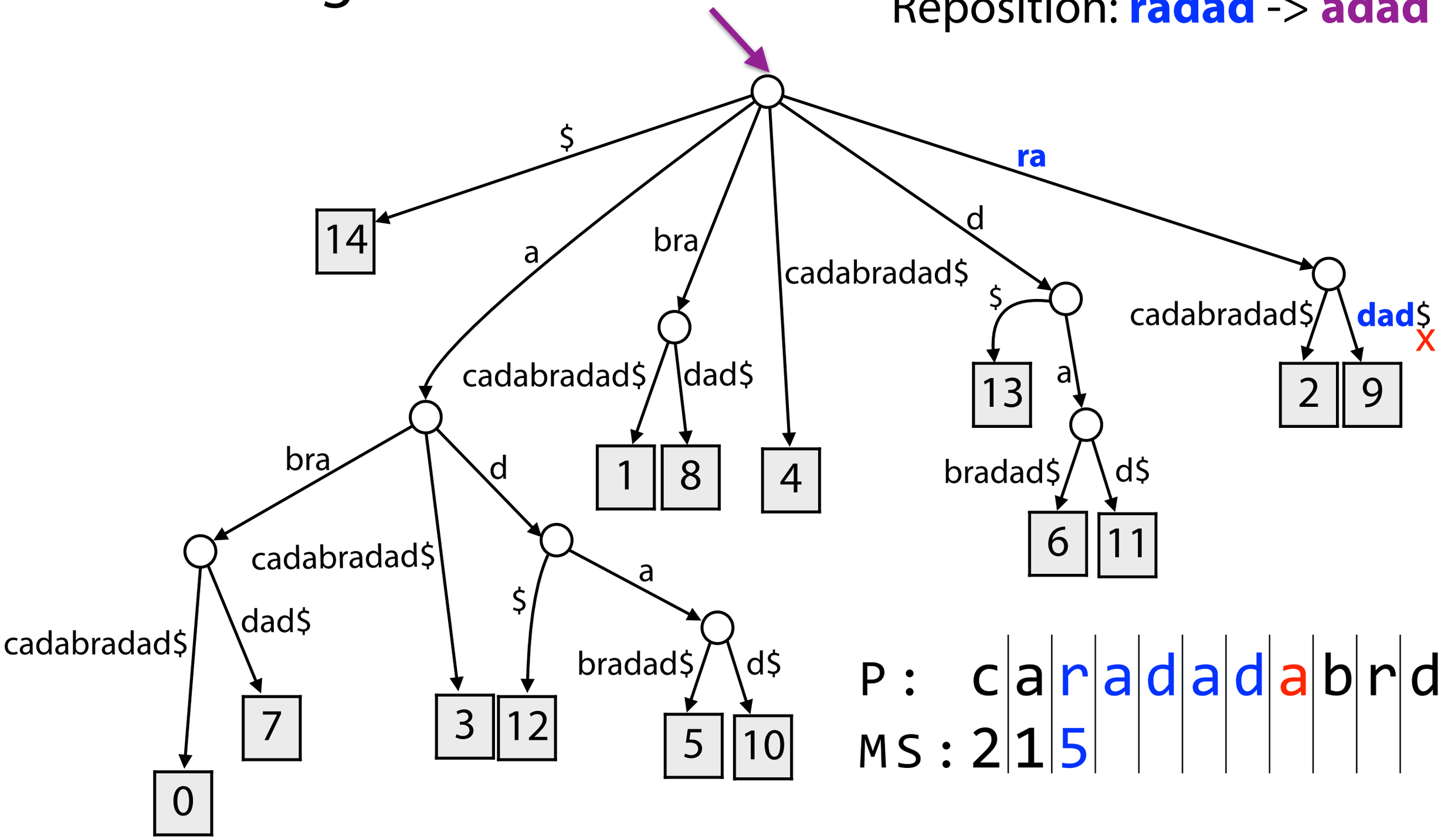
Matching statistics



P:	c	a	r	a	d	a	d	a	b	r	d
MS:	2	1									

Matching statistics

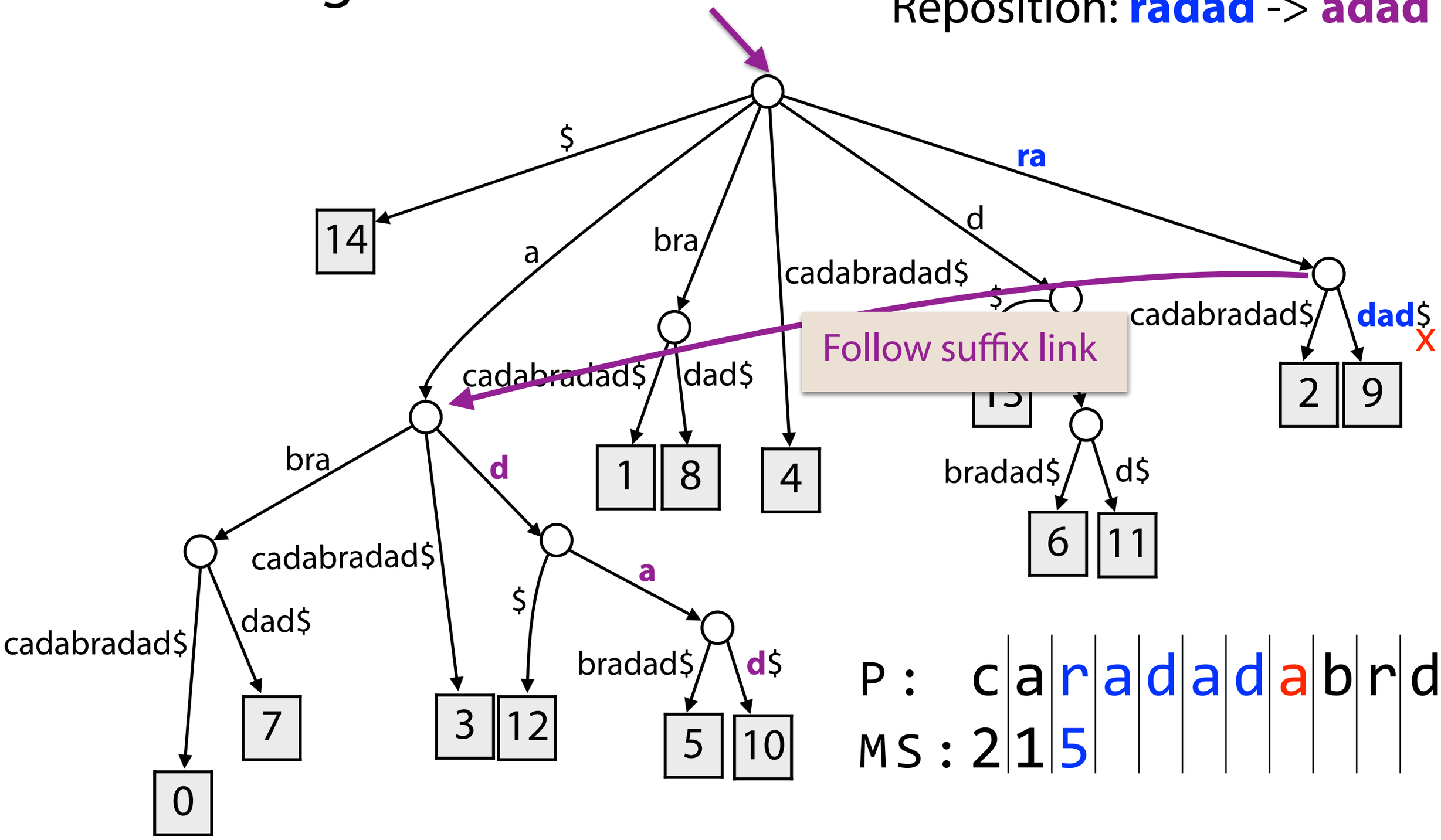
Reposition: **radad** -> **adad**



P:	c	a	r	a	d	a	d	a	b	r	d
MS:	2	1	5								

Matching statistics

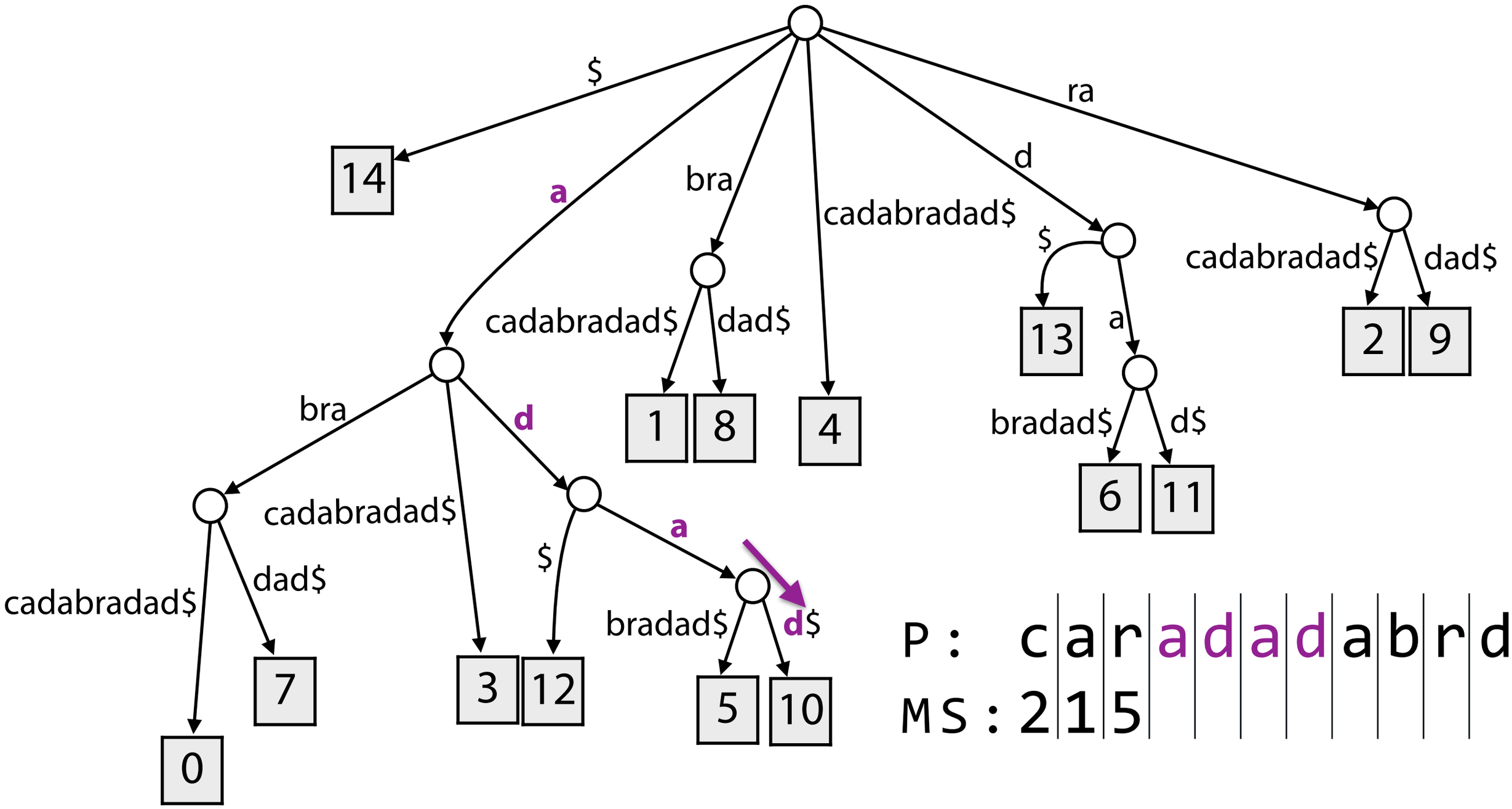
Reposition: **radad** -> **adad**



P: c a r a d a d a b r d

MS: 2 1 5

Matching statistics

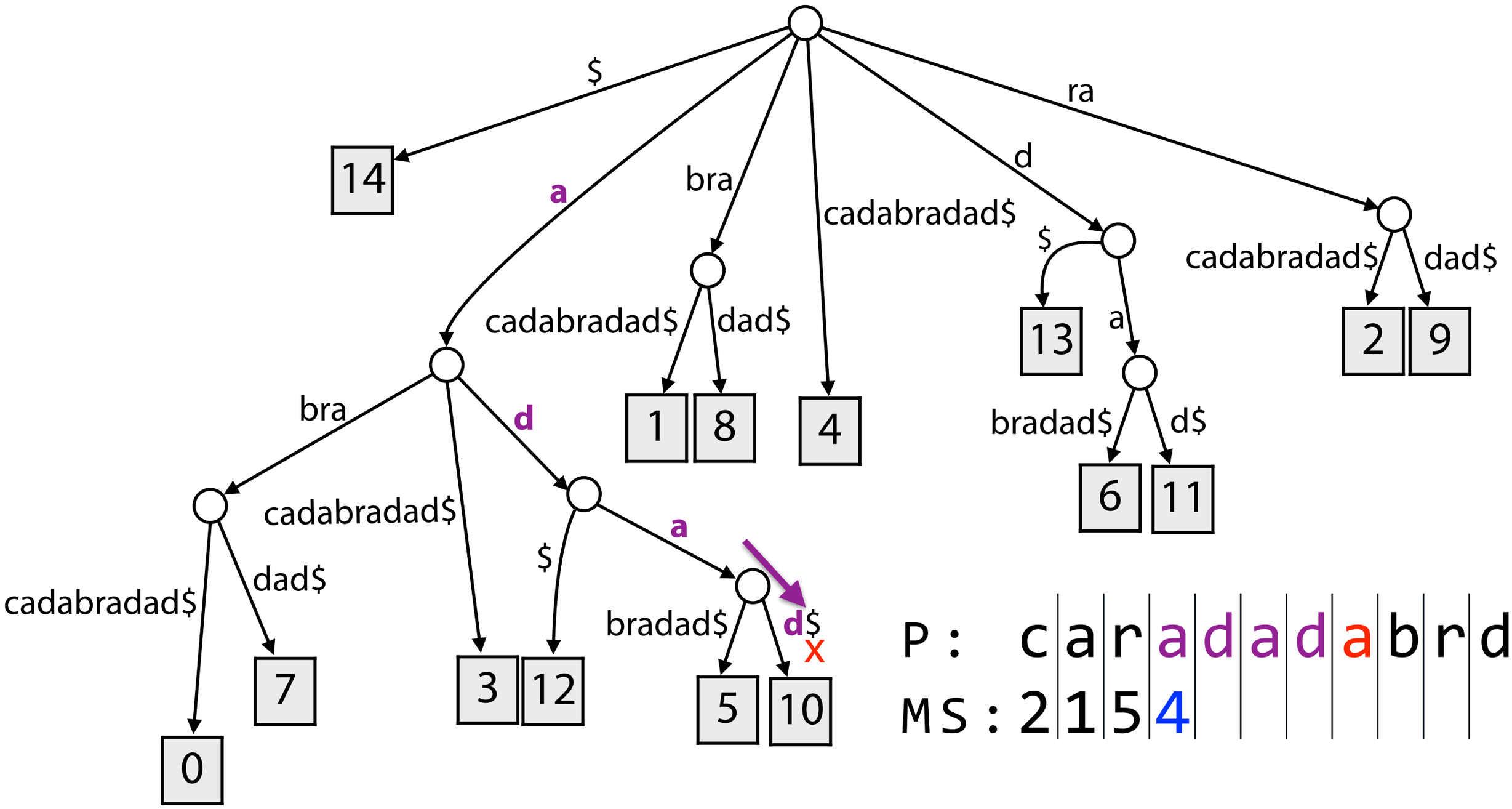


P: c a r a d a d a b r d

MS: 2 1 5

Matching statistics

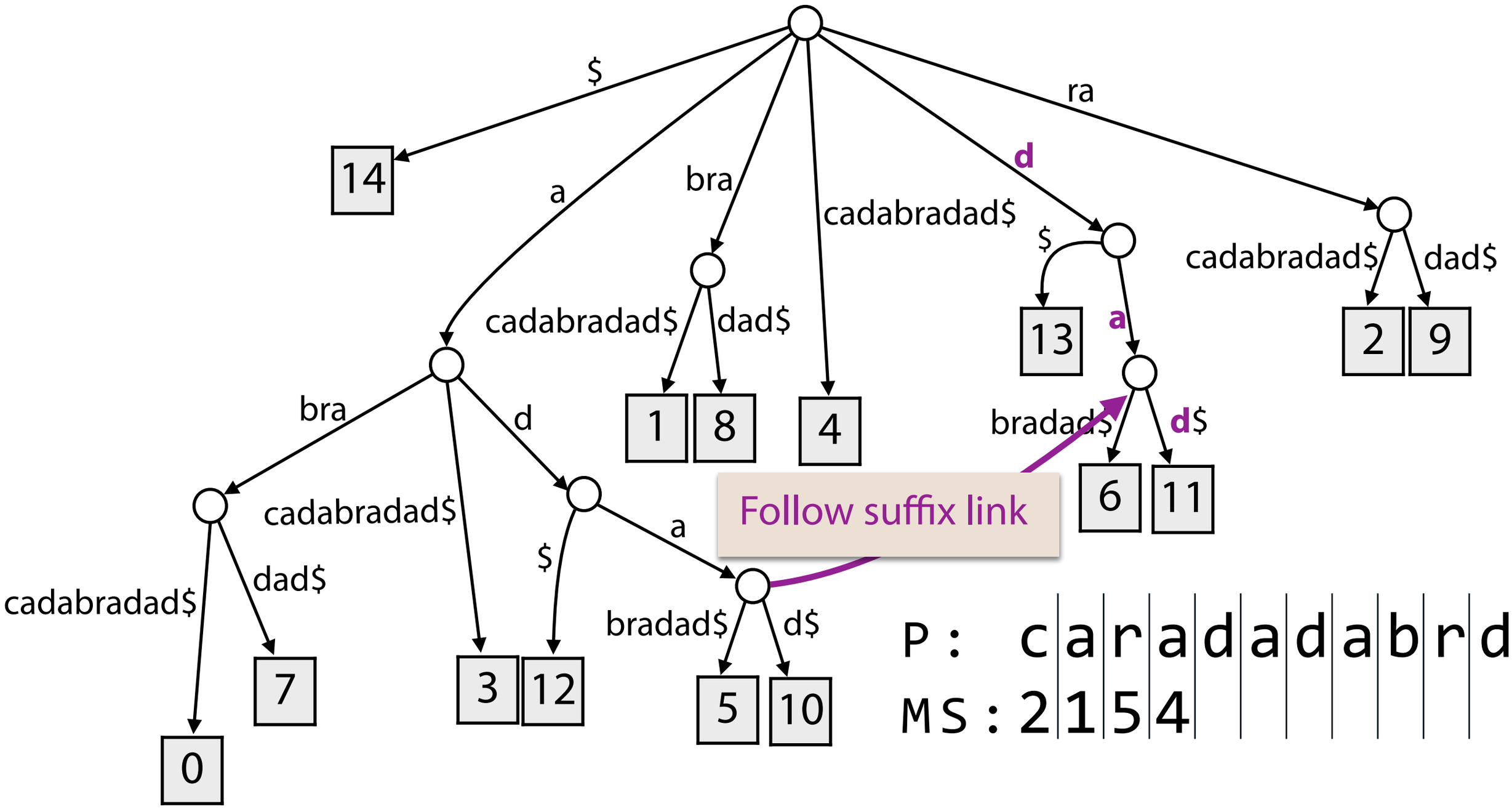
Reposition: **adad** -> **dad**




```
P:  c a r a d a d a b r d
MS: 2 1 5 4
```

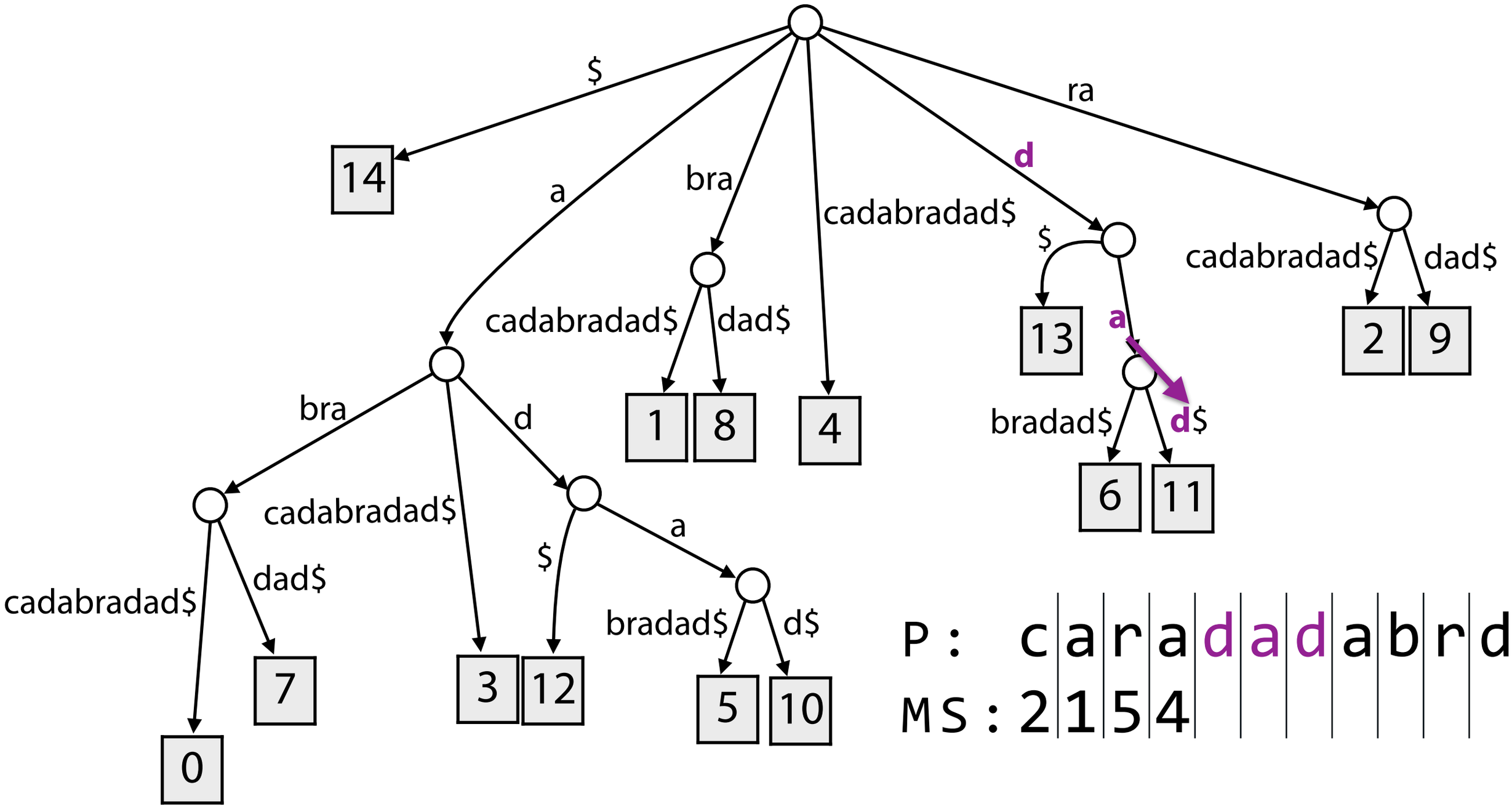
Matching statistics

Reposition: **adad** -> **dad**



P:	c	a	r	a	d	a	d	a	b	r	d
MS:	2	1	5	4							

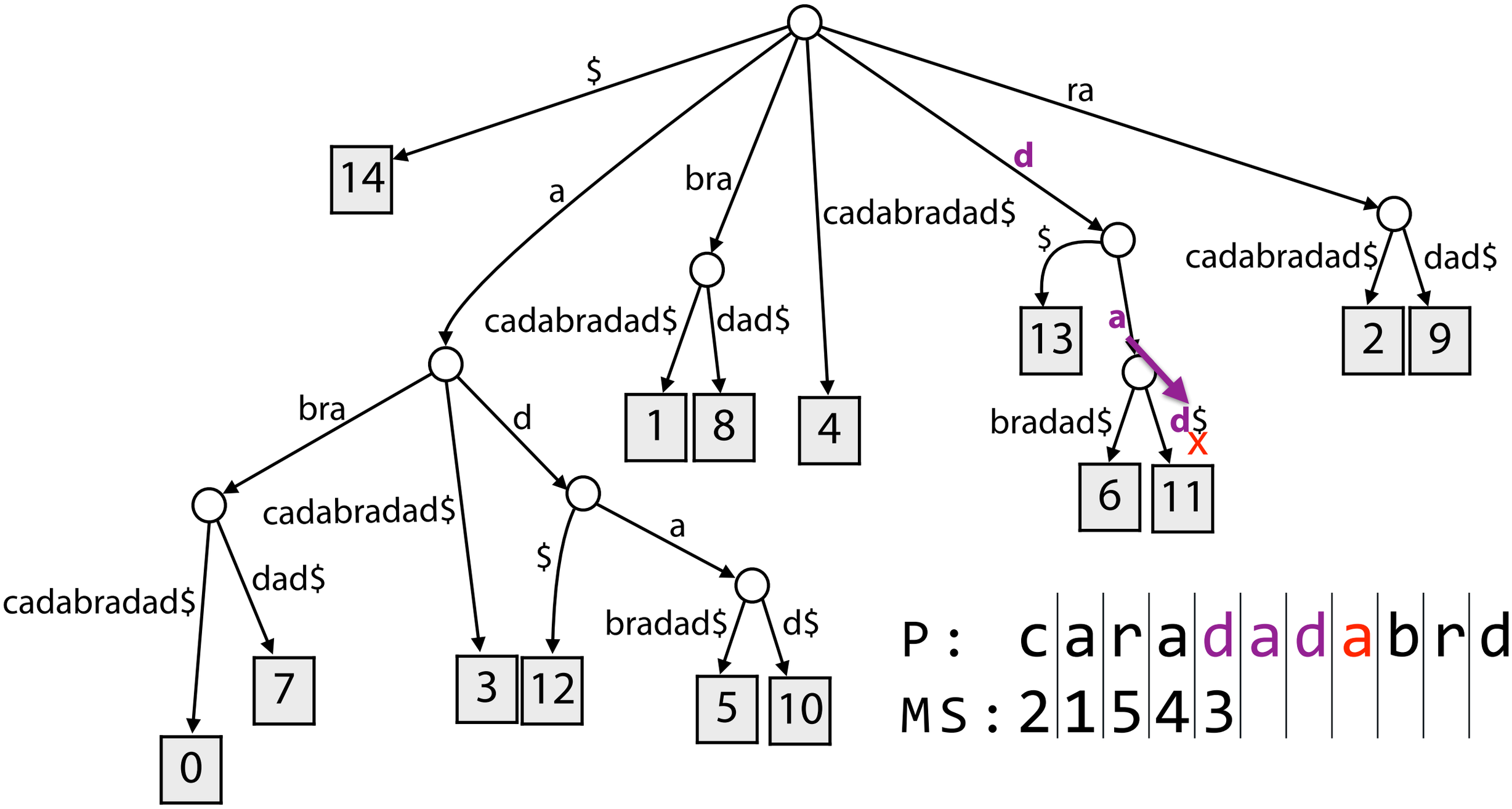
Matching statistics



P:	c	a	r	a	d	a	d	a	b	r	d
MS:	2	1	5	4							

Matching statistics

Reposition: **dad** -> **ad**

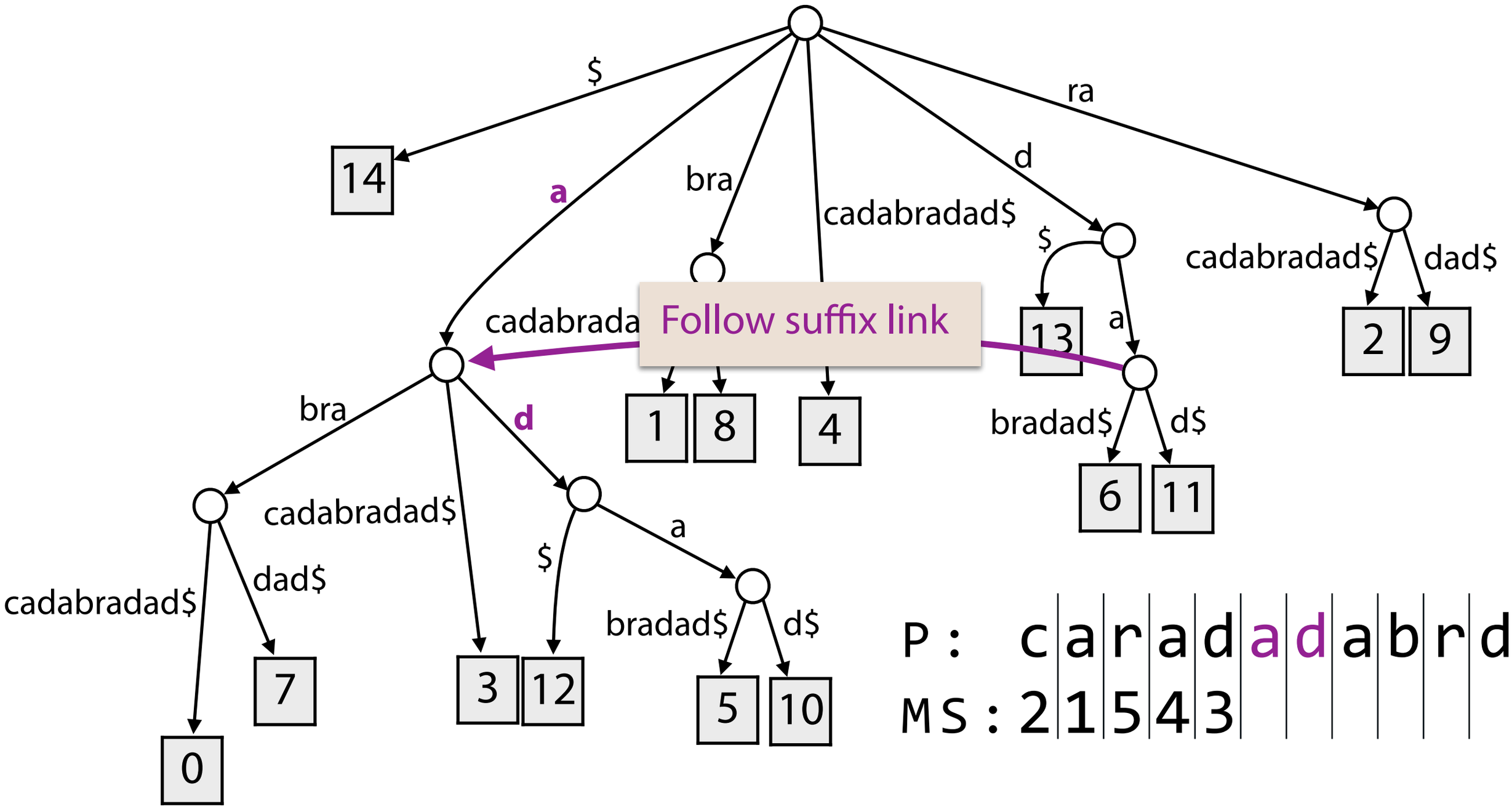


P: c a r a d a d a b r d

MS: 2 1 5 4 3

Matching statistics

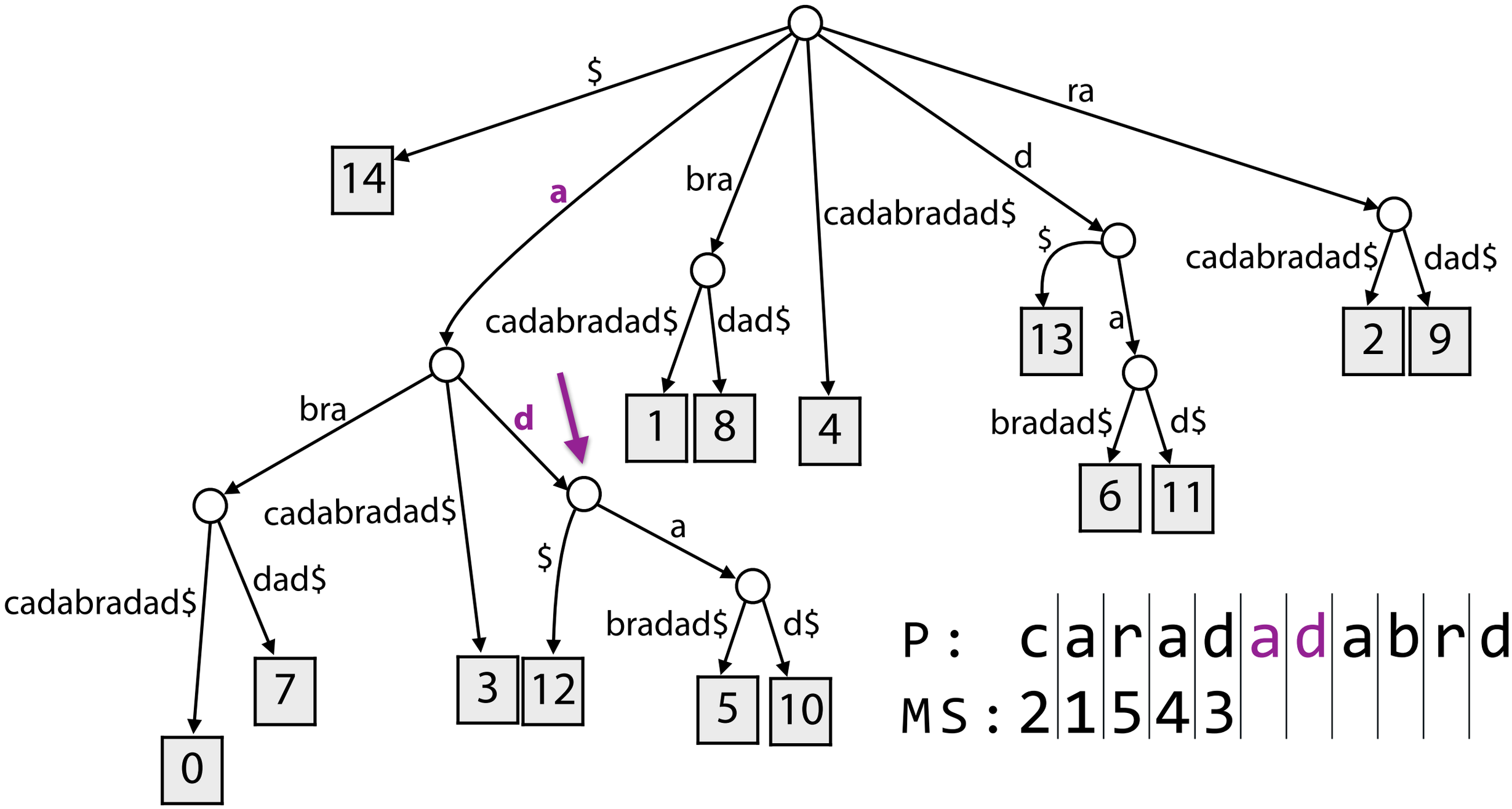
Reposition: **dad** -> **ad**



P: c a r a d a d a b r d

MS: 2 1 5 4 3

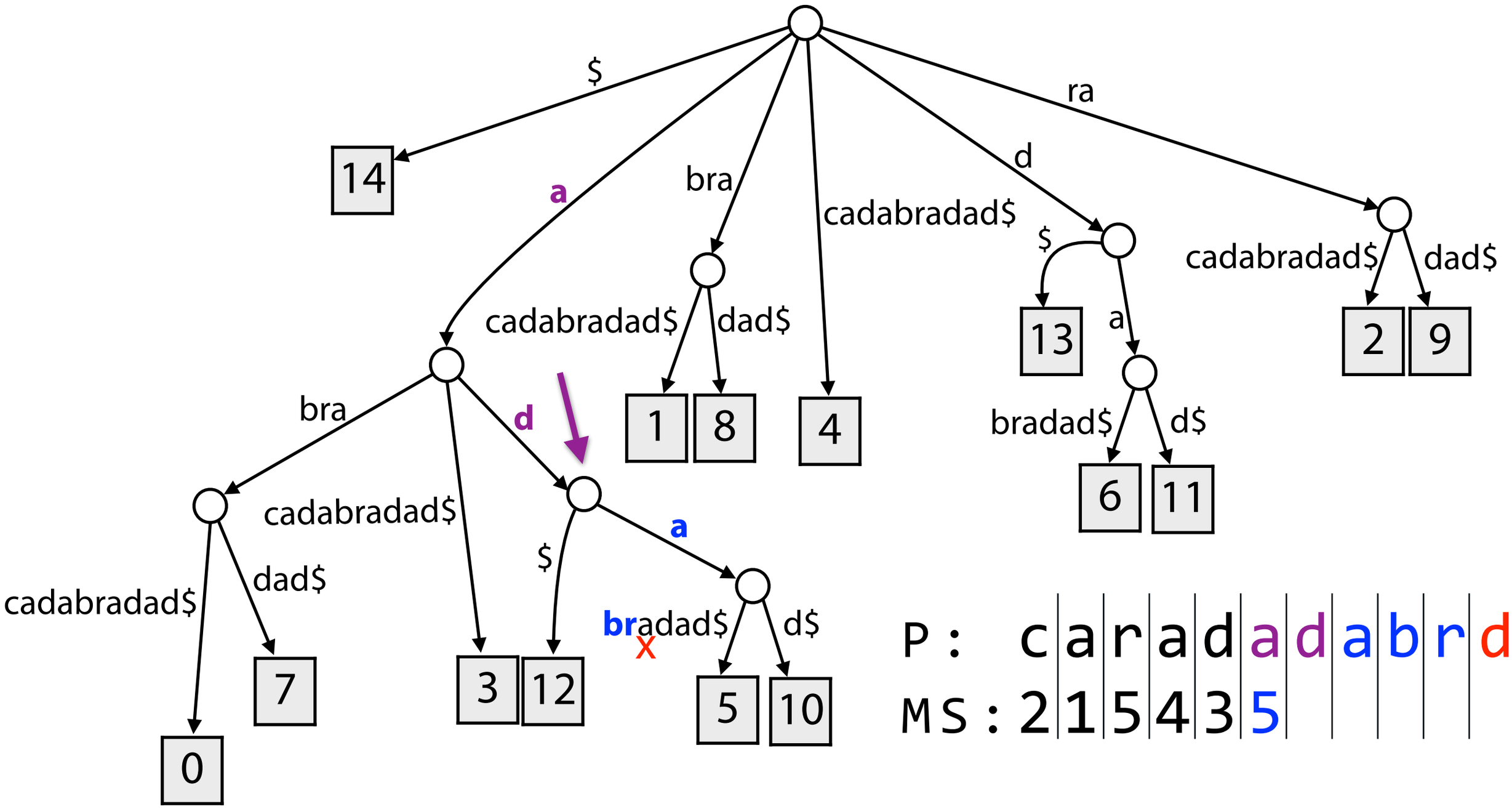
Matching statistics



P: c a r a d a d a b r d

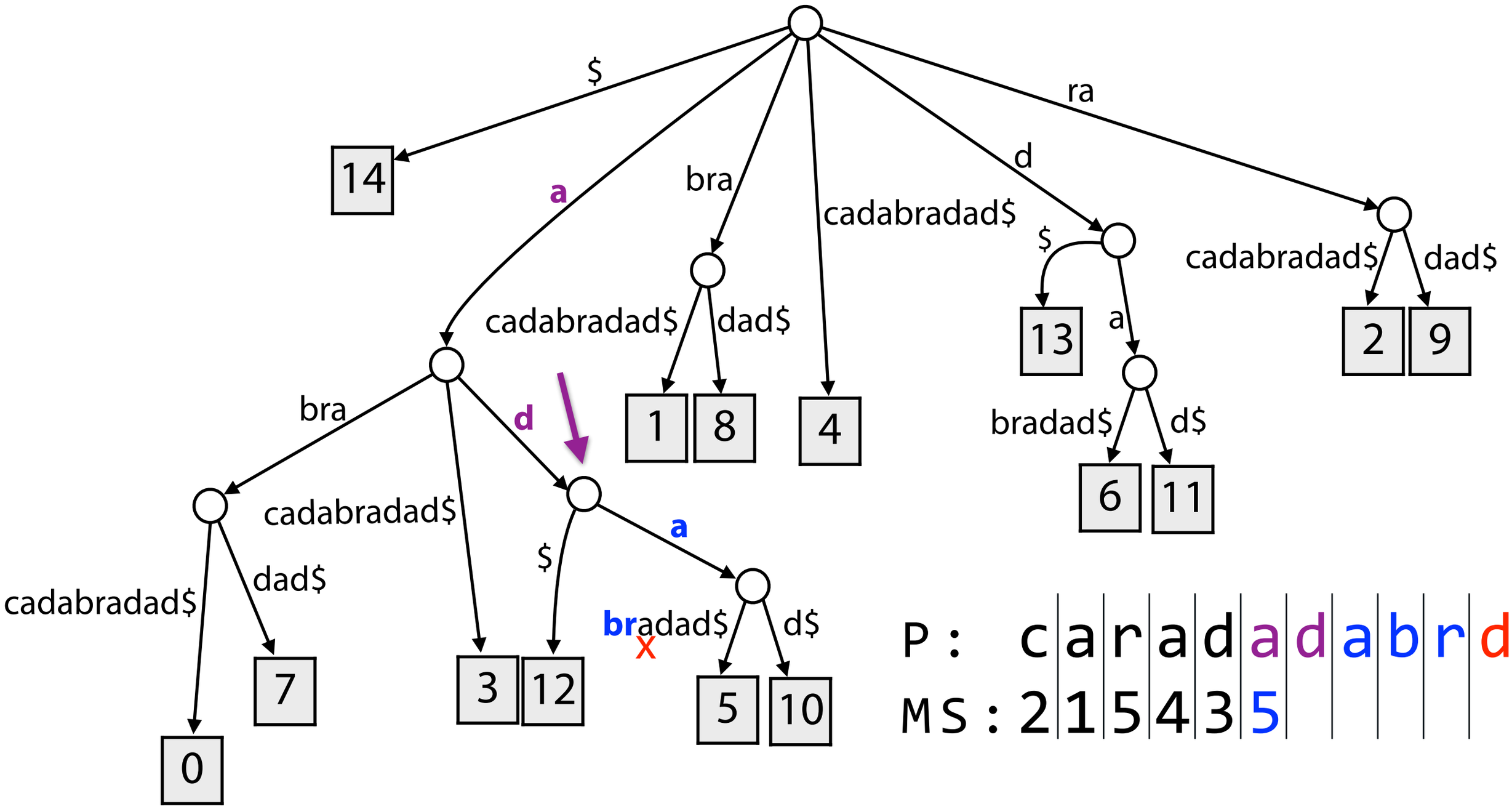
MS: 2 1 5 4 3

Matching statistics



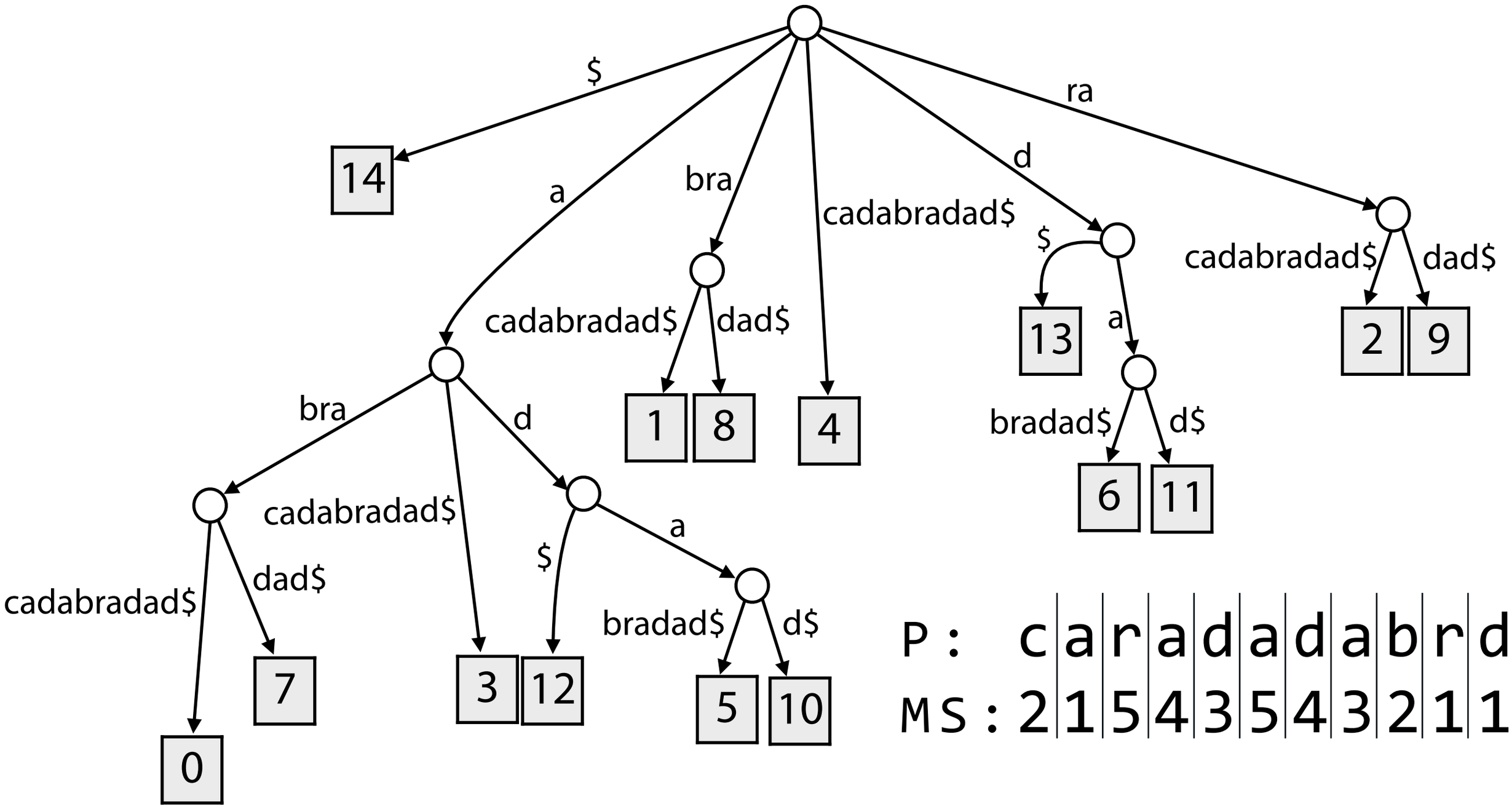
P:	c	a	r	a	d	a	d	a	b	r	d
MS:	2	1	5	4	3	5					

Matching statistics



P:	c	a	r	a	d	a	d	a	b	r	d
MS:	2	1	5	4	3	5					

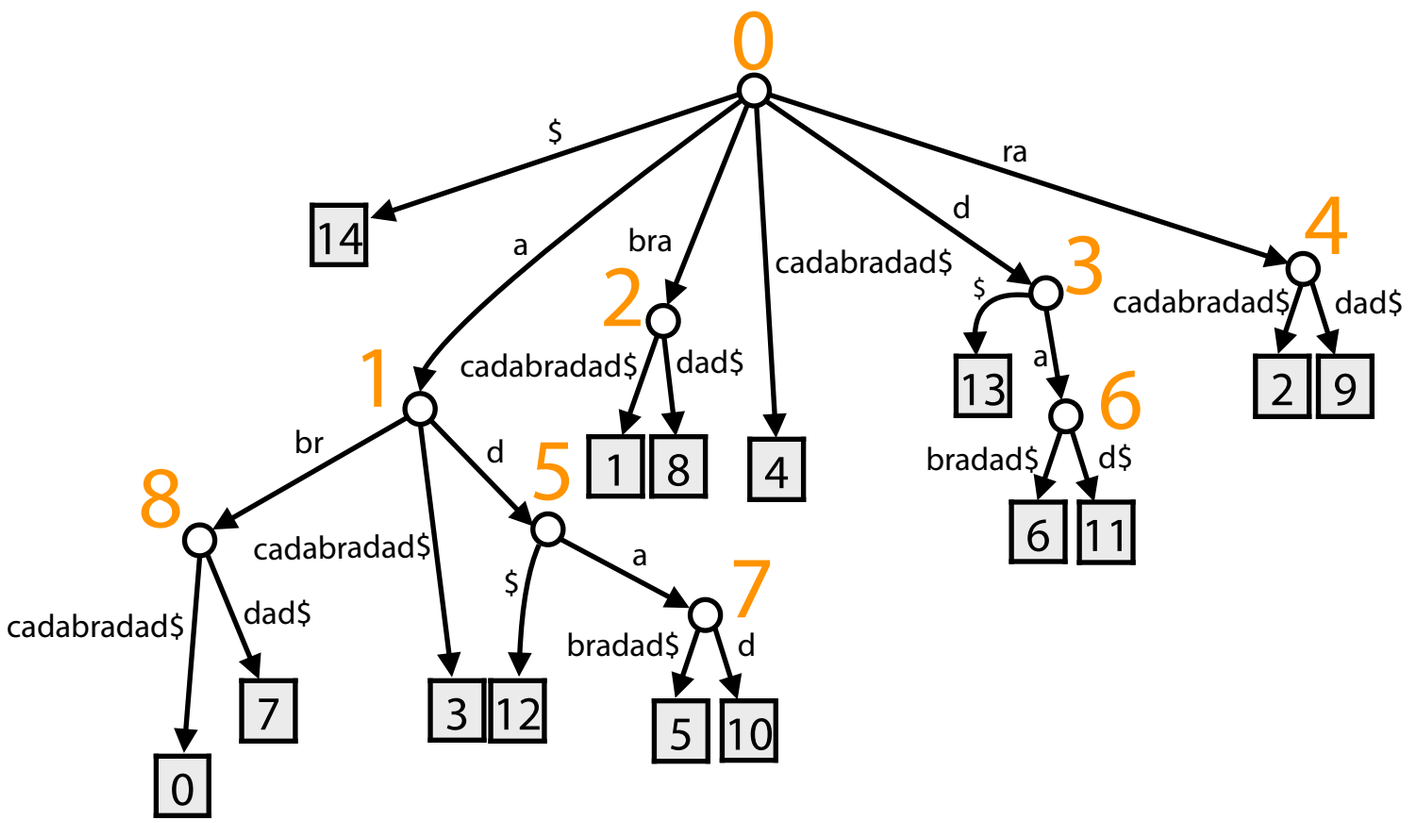
Matching statistics



P:	c	a	r	a	d	a	d	a	b	r	d
MS:	2	1	5	4	3	5	4	3	2	1	1

Can we tally the work done?

Matching statistics



Green: requires char read from P

Blue: match

Red: mismatch

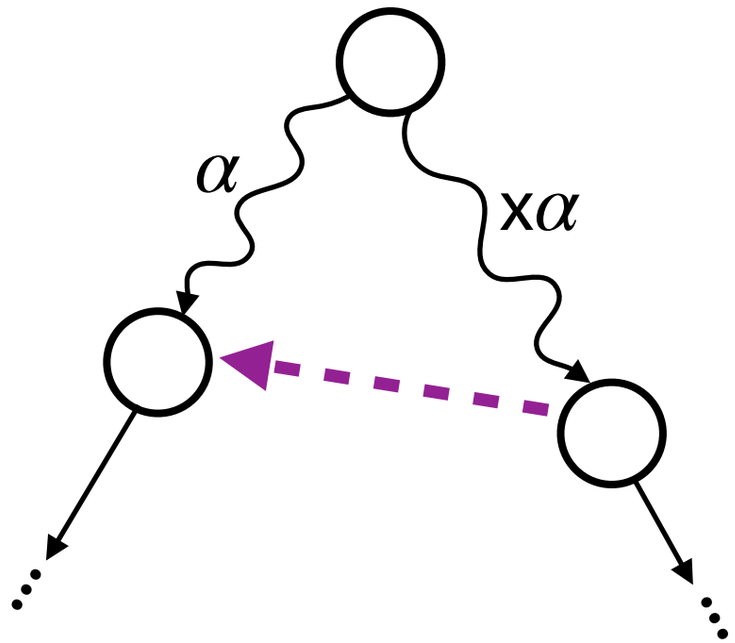
Gray: not read, skipped at beginning of repo

Plum: not read, skipped at end of repo

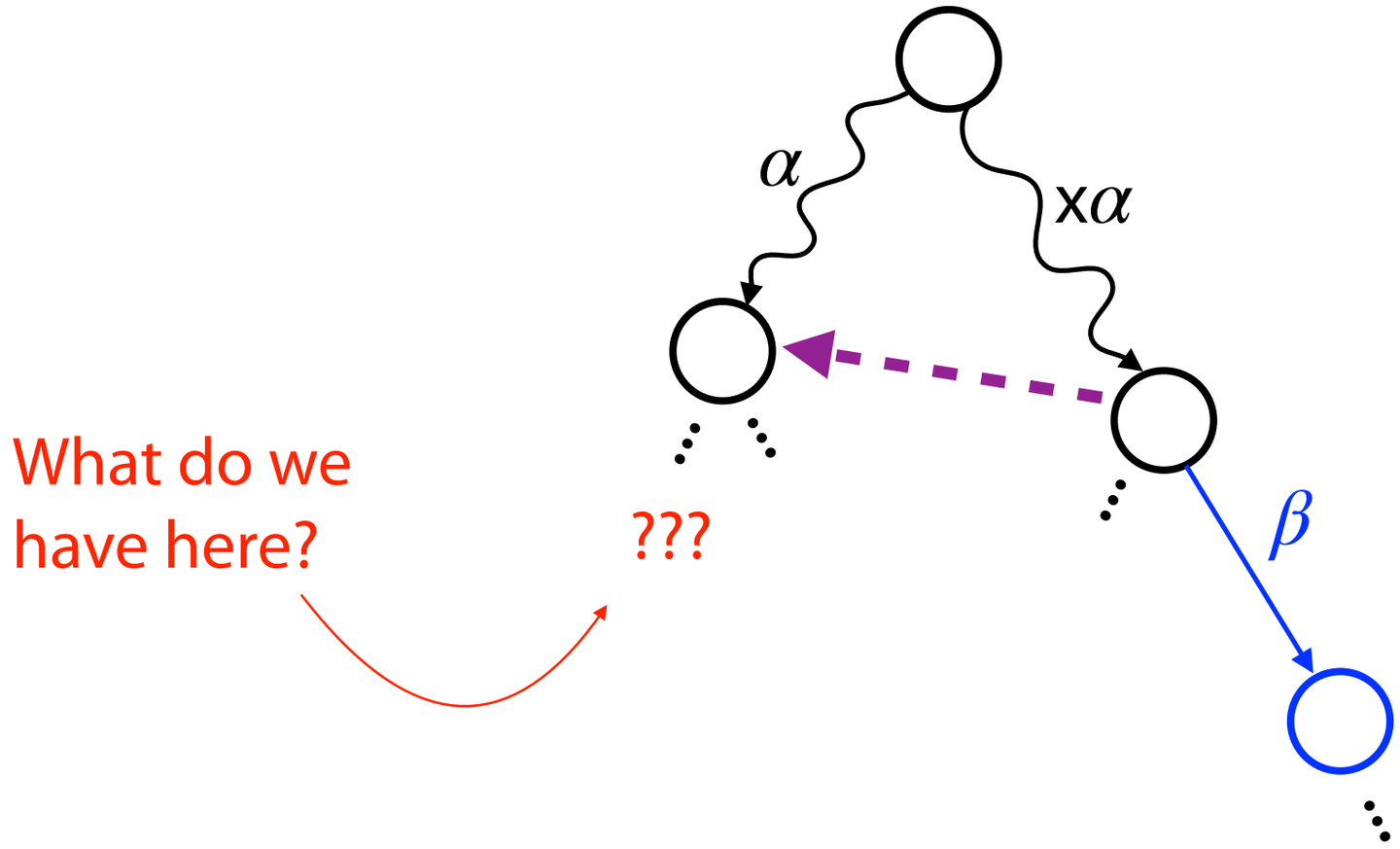
	caradadabrd	jump	statistics
1:	car	0	2
	a		
2:	r	0	1
3:	radada	1	5
	adad		
4:	a	6	4
	dad		
5:	a	1	3
	ad		
6:	abrd	6	5
	dabr		
7:	d	1	4
	abr		
8:	d	0	3
	br		
9:	d	0	2
	r		
10:	d	0	1
11:	d		1

Matching statistics

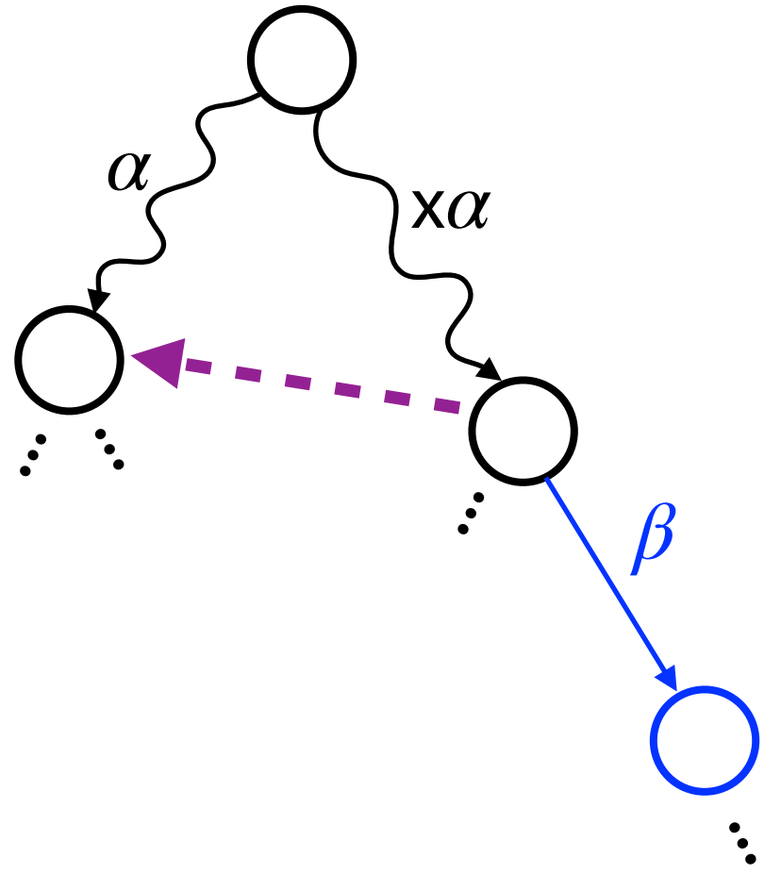
Consider a portion of the tree



Matching statistics

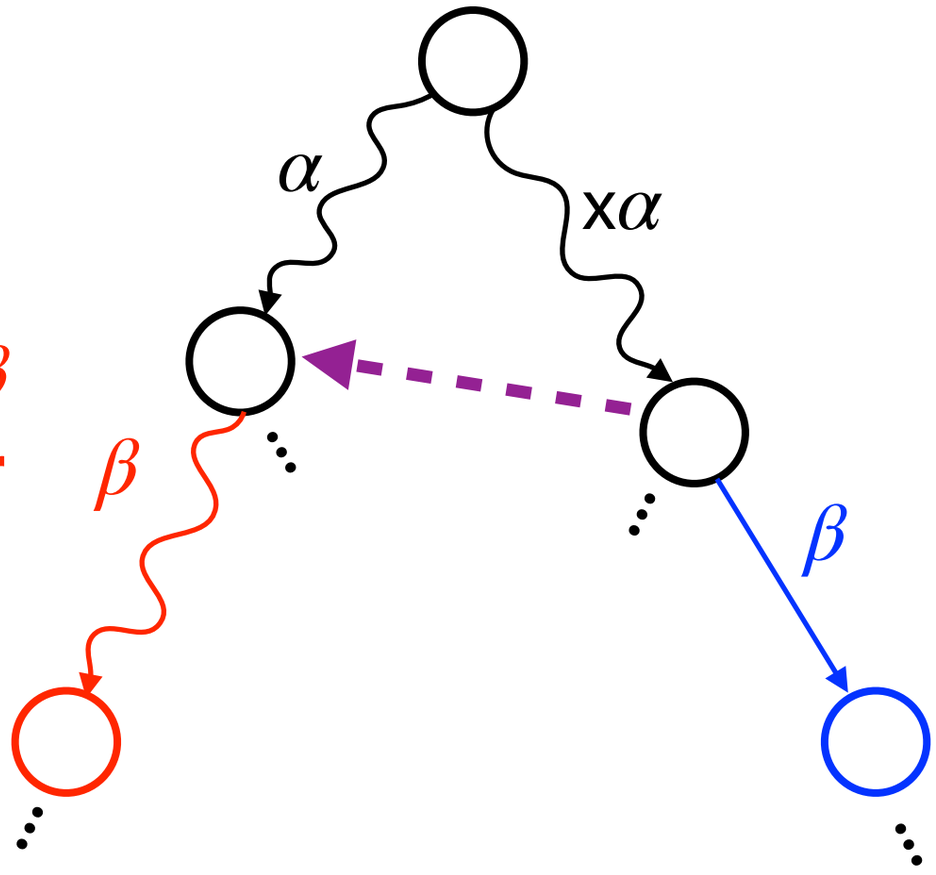


Matching statistics



Matching statistics

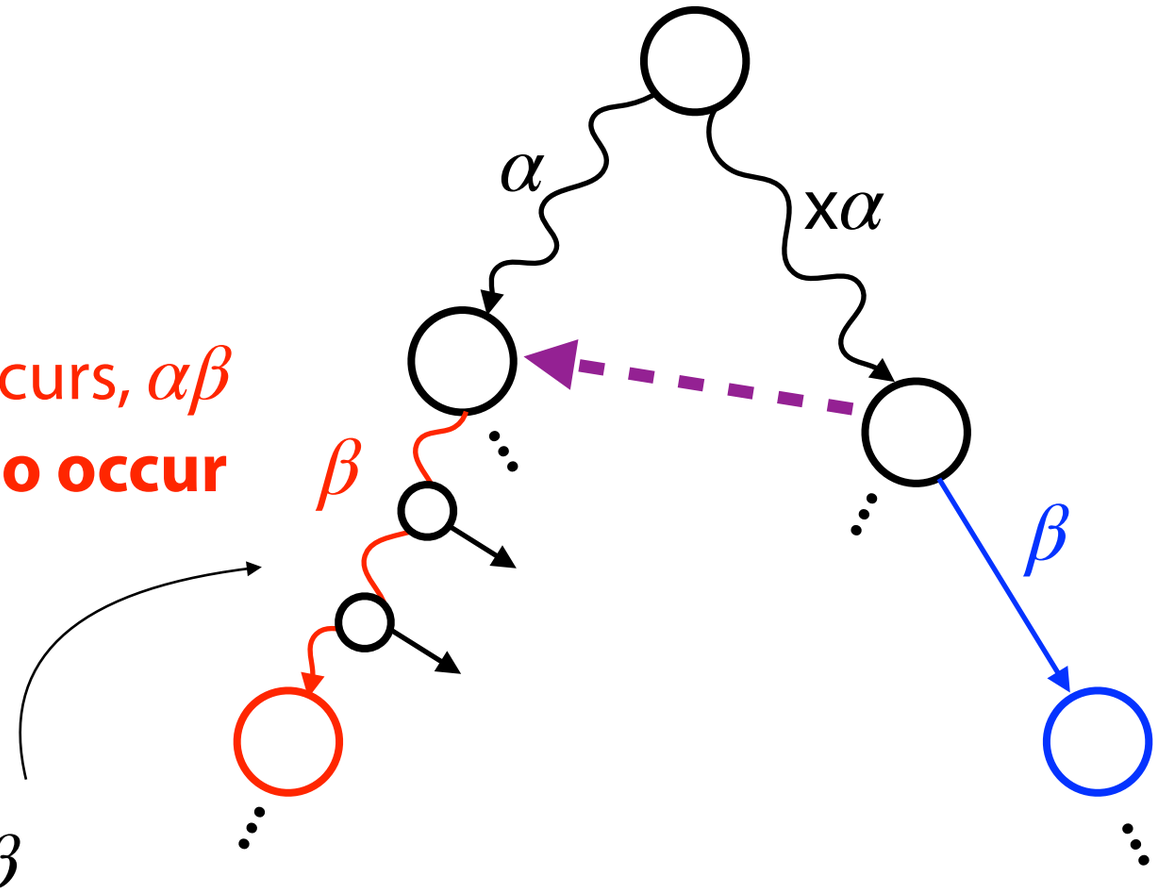
If x$\alpha\beta$ occurs, $\alpha\beta$
must also occur



Matching statistics

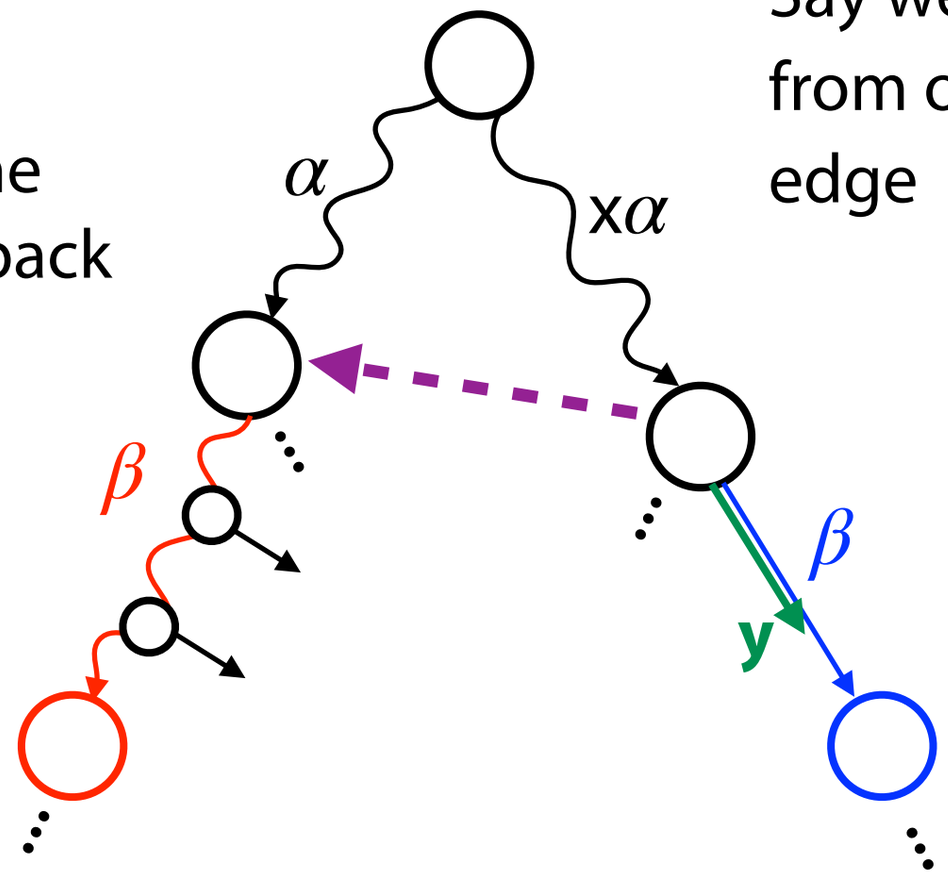
If x$\alpha\beta$ occurs, $\alpha\beta$ **must also occur**

There may be intermediate nodes on the $\alpha\beta$ path not present on the $x\alpha\beta$ path



Matching statistics

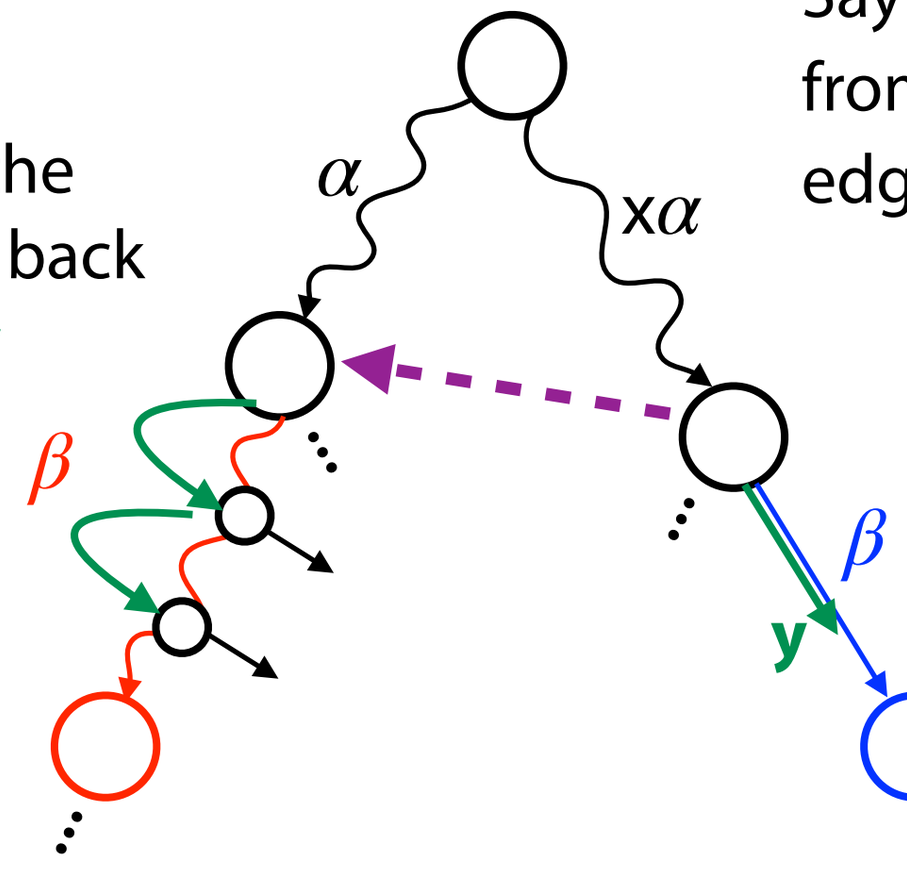
We jump up, follow the suffix link, then walk back down according to **y**



Say we are **repositioning** from offset **y** on the β-labeled edge

Matching statistics

We jump up, follow the suffix link, then walk back down according to **y**



Say we are **repositioning** from offset **y** on the β-labeled edge

We know **y** is there! Following the **y** path requires only **node-to-node** jumps, not character-to-character.

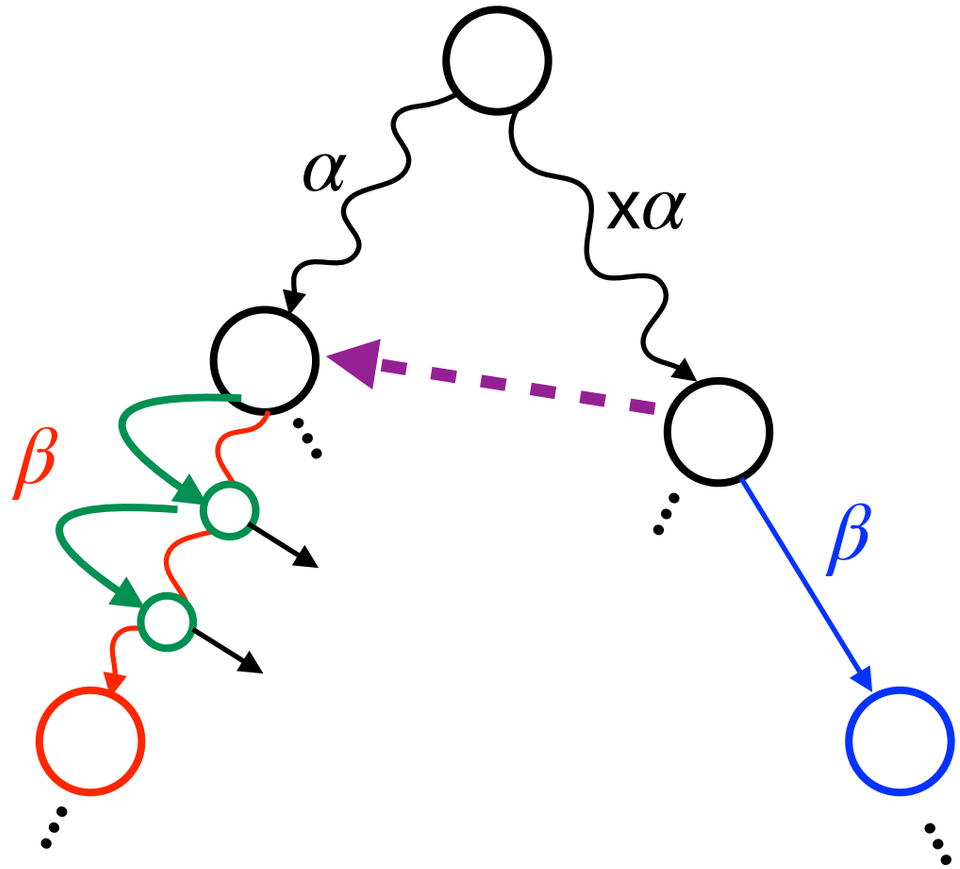
When we enter a node, we decide how to exit it by finding the edge beginning with the corresponding character in **y**

Matching statistics

Final observation!

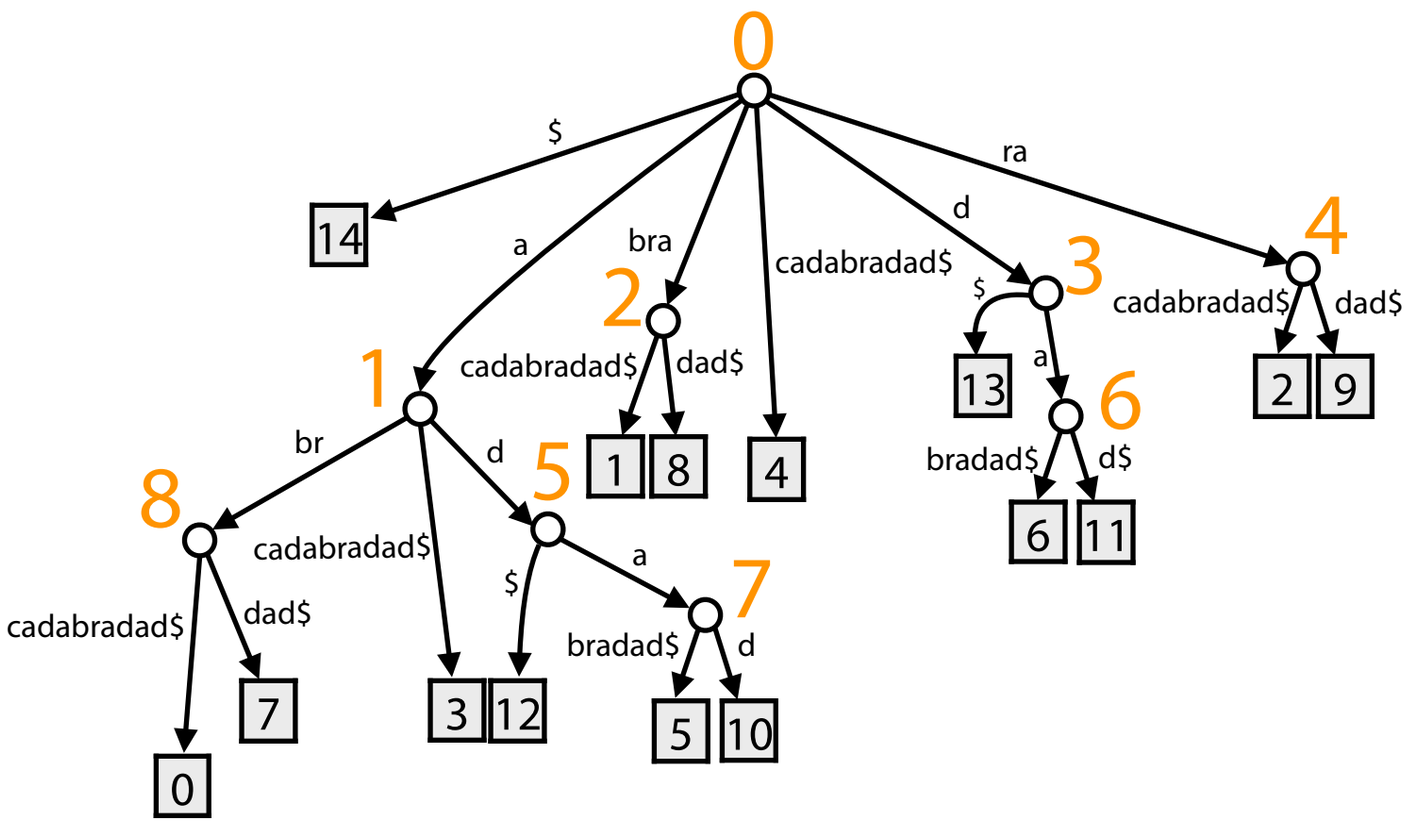
When we jump past a node during repositioning, we will never jump using that character from **y** again

The next suffix link traversal will start at or below that node



Total # of such jumps can't exceed |P|

Matching statistics



Green: requires char read from P

Blue: match

Red: mismatch

Gray: not read, skipped at beginning of repo

Plum: not read, skipped at end of repo

	caradadabrd	jump	statistics
1:	car	0	2
	a		
2:	r	0	1
3:	radada	1	5
	adad		
4:	a	6	4
	dad		
5:	a	1	3
	ad		
6:	abrd	6	5
	dabr		
7:	d	1	4
	abr		
8:	d	0	3
	br		
9:	d	0	2
	r		
10:	d	0	1
11:	d		1

Matching statistics

Green: requires char read from P
Blue: match
Red: mismatch
Gray: not read, skipped at beginning of repo
Plum: not read, skipped at end of repo

```
    caradadabrd
1:  car
      a
2:    r
3:   radada
       adad
4:       a
       dad
5:       a
        ad
6:       abrd
        dabr
7:         d
        abr
8:         d
       br
9:         d
      r
10:        d
11:        d
```

Total # of **jumps** can't exceed |P|

A **matched** character never has to be re-matched; total # matches can't exceed |P|

Each iteration encounters at most 1 **mismatch**; total # mismatches can't exceed |P|

The **Gray** and **Plum** operations are "free" skips

Overall $O(n)$ where $n = |P|$