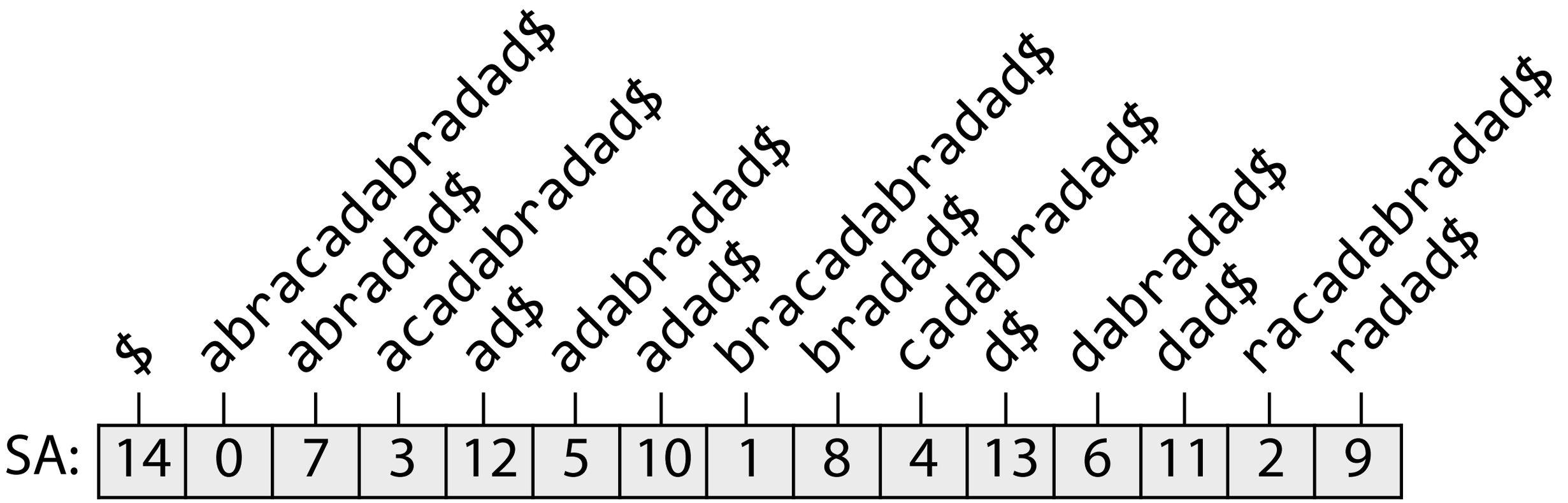
Suffix Arrays: basic queries

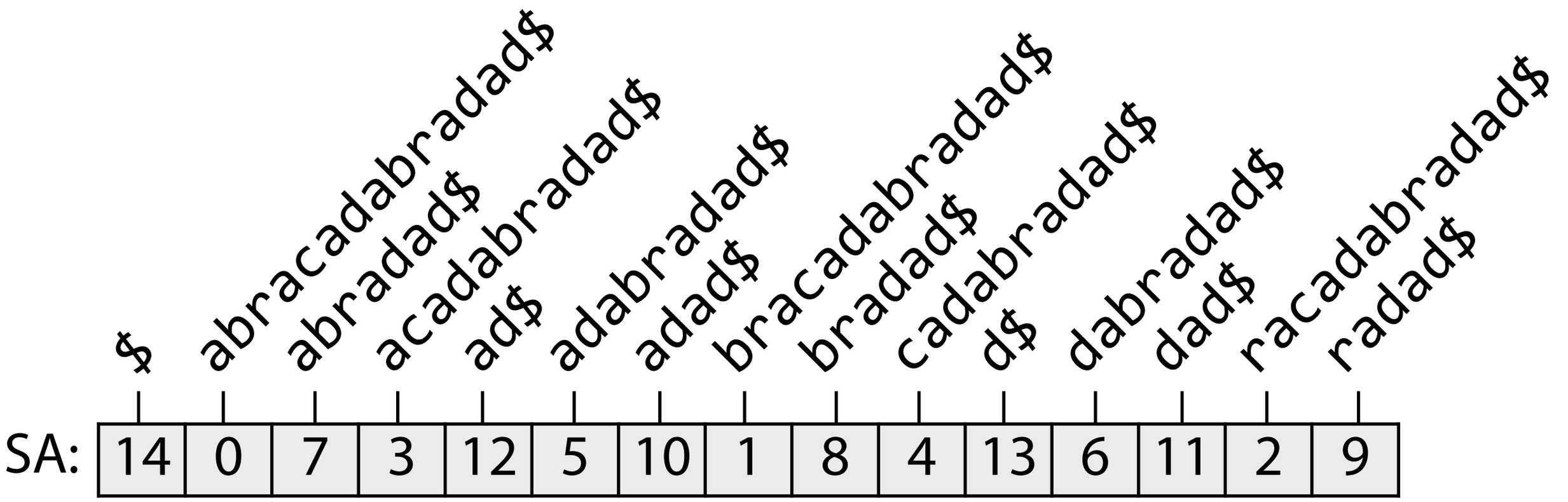
Ben Langmead

Suffix array: querying



Suffix array for T = abracadabradad$

Suffix array: querying



How to check if a query string *P* a substring of *T*?

1. For *P* to be a substring, it must be a **prefix** of ≥1 of *T*'s **suffixes**

2. Suffixes sharing a prefix are **consecutive** in the suffix array

Suffix array: lexicographical comparisons

When comparing two strings, we start at the left end

If characters are equal, we advance to the right in both.
If characters are not equal, we have our answer.

A: abracadabradad

B: abradad

Suffix array: lexicographical comparisons

When comparing two strings, we start at the left end

If characters are equal, we advance to the right in both.
If characters are not equal, we have our answer.

```
A: abracadabradad
   =------
B: abradad
```

Suffix array: lexicographical comparisons

When comparing two strings, we start at the left end

If characters are equal, we advance to the right in both.
If characters are not equal, we have our answer.

```
A: abracadabradad
   ==-----
B: abradad
```

Suffix array: lexicographical comparisons

When comparing two strings, we start at the left end

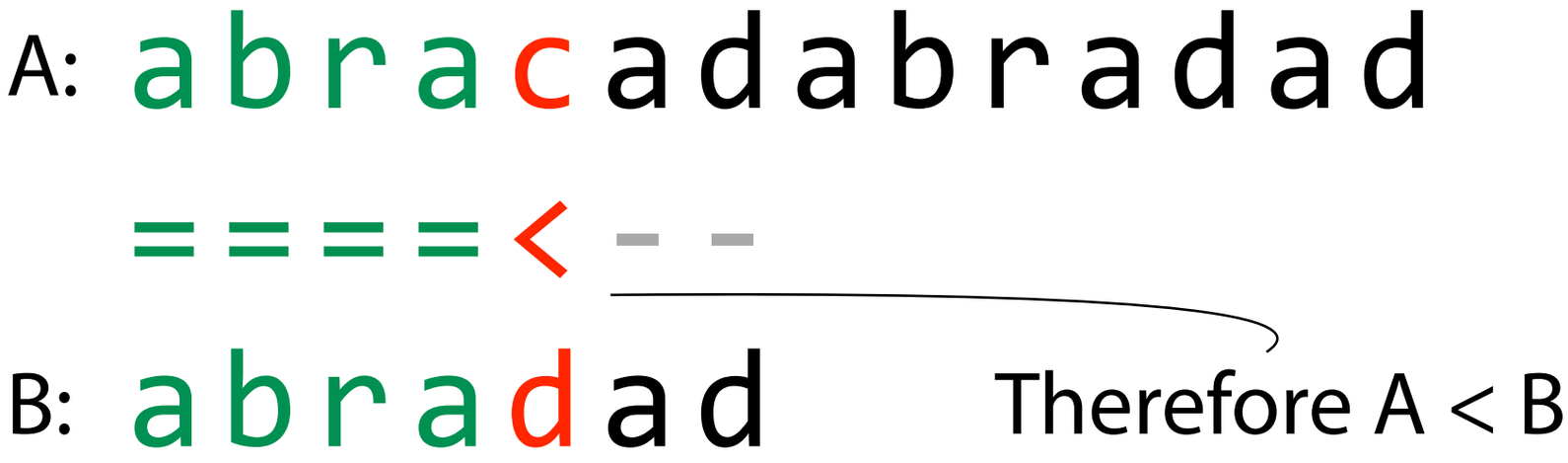
If characters are equal, we advance to the right in both.
If characters are not equal, we have our answer.

A: abracadabradad
   ====---
B: abradad

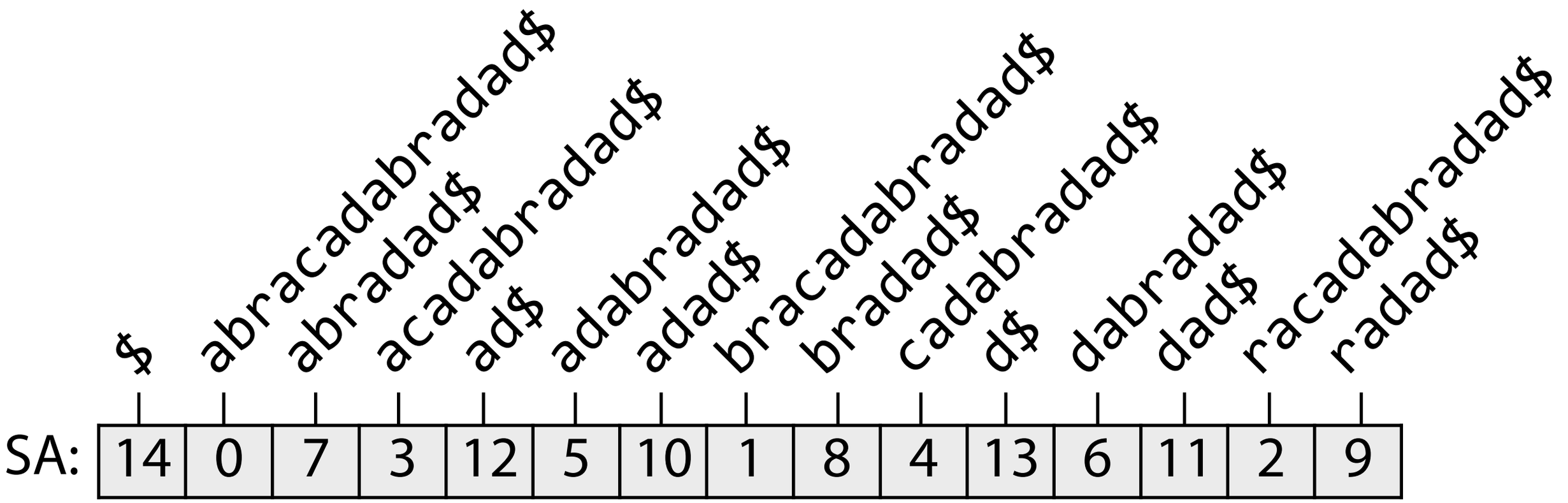
Suffix array: lexicographical comparisons

When comparing two strings, we start at the left end

If characters are equal, we advance to the right in both.
If characters are not equal, we have our answer.



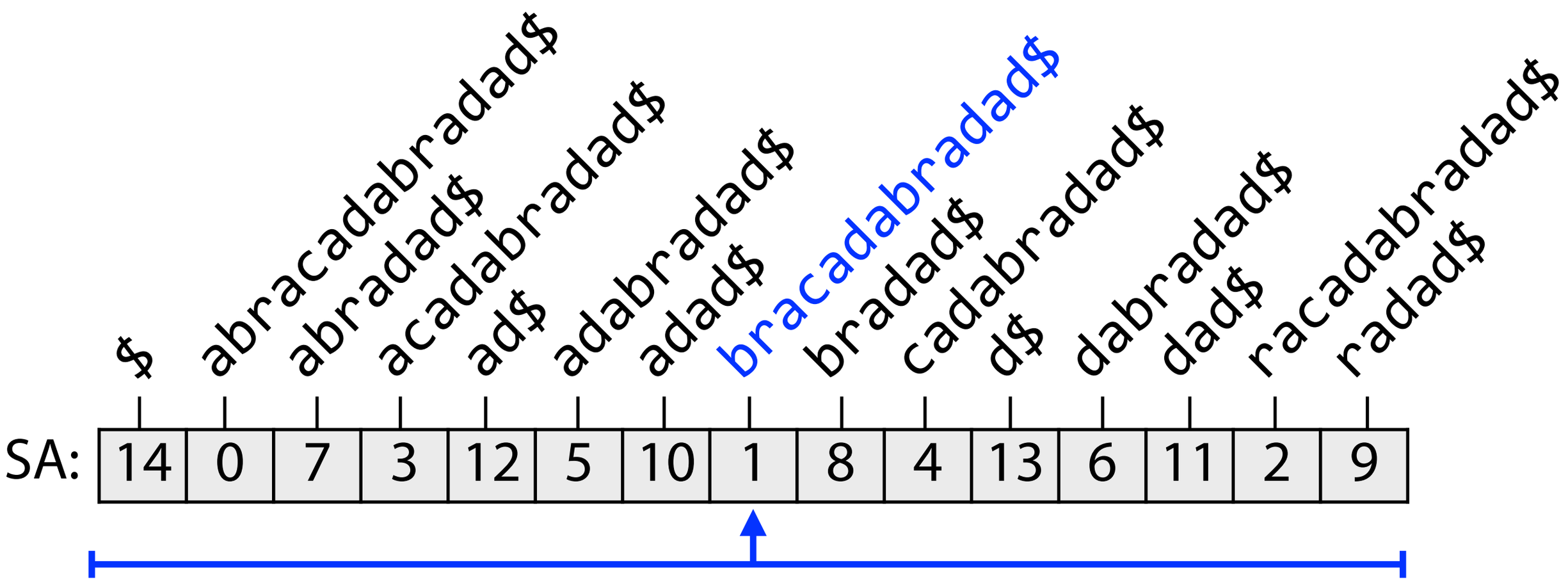
Suffix array: querying



Say query is **dad**

Which suffixes (if any) have **dad** as a prefix? Let's apply binary search.

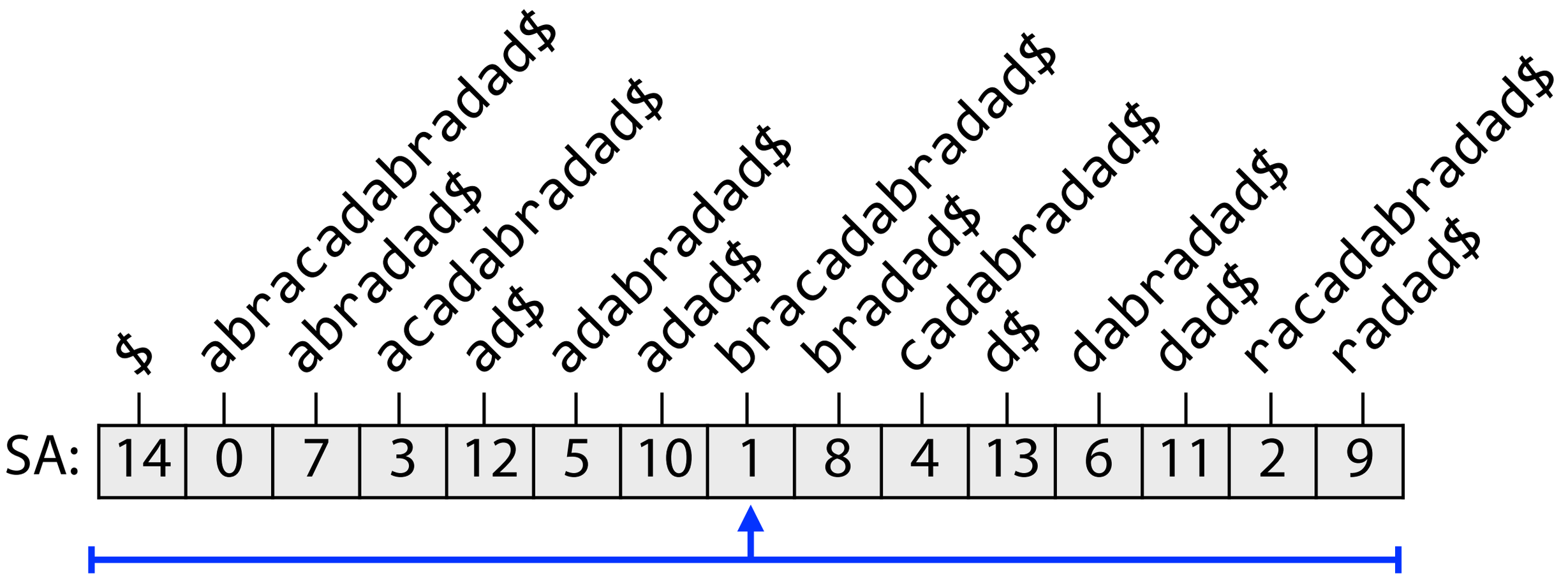
Suffix array: querying



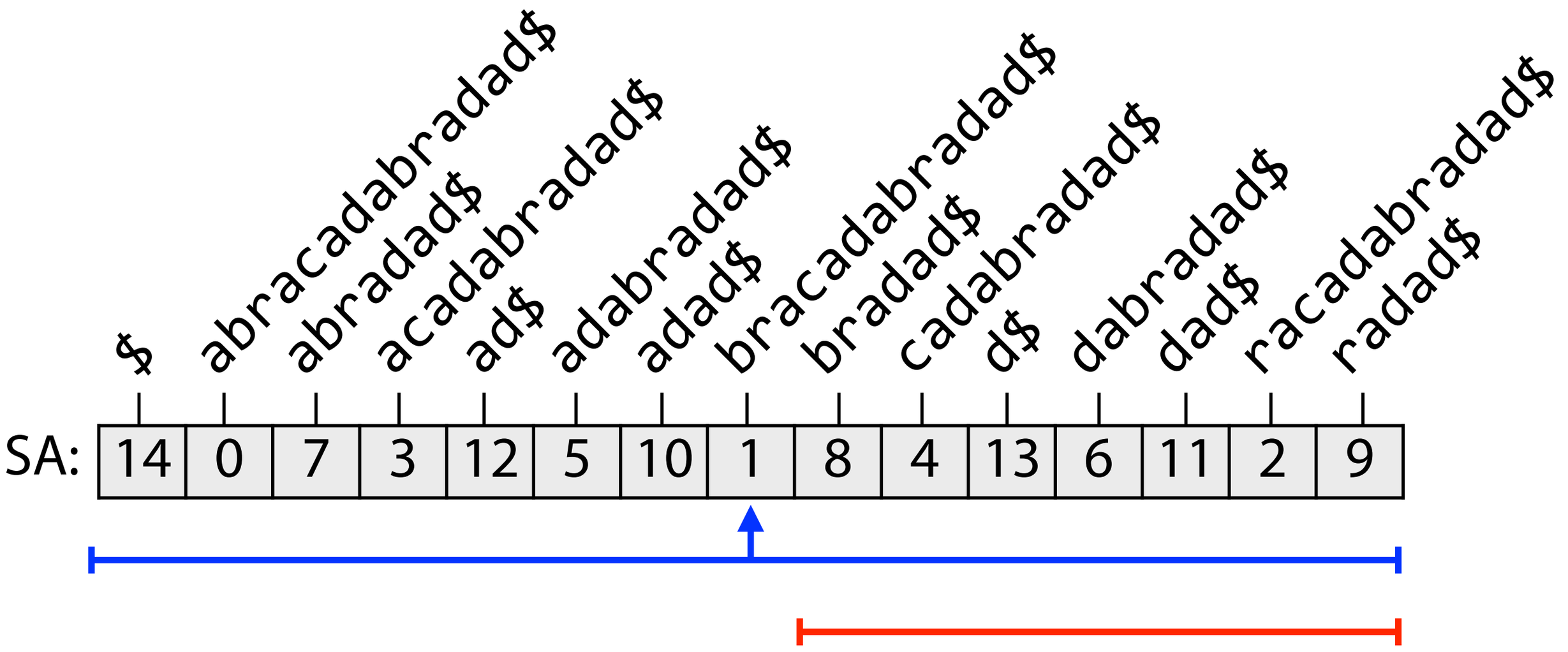
Iteration 1: compare **dad** to pivot **bracadabradad$**

dad is alphabetically after; recurse on right half

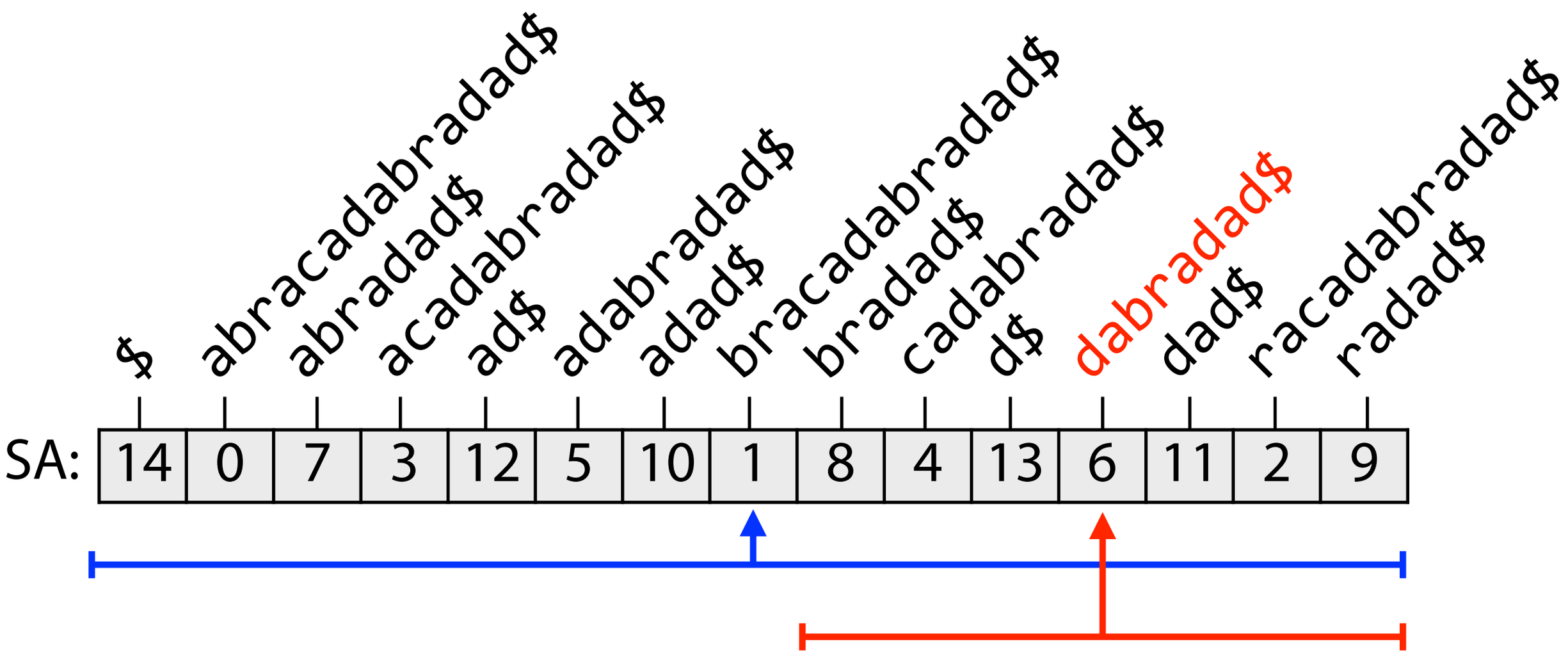
Suffix array: querying



Suffix array: querying



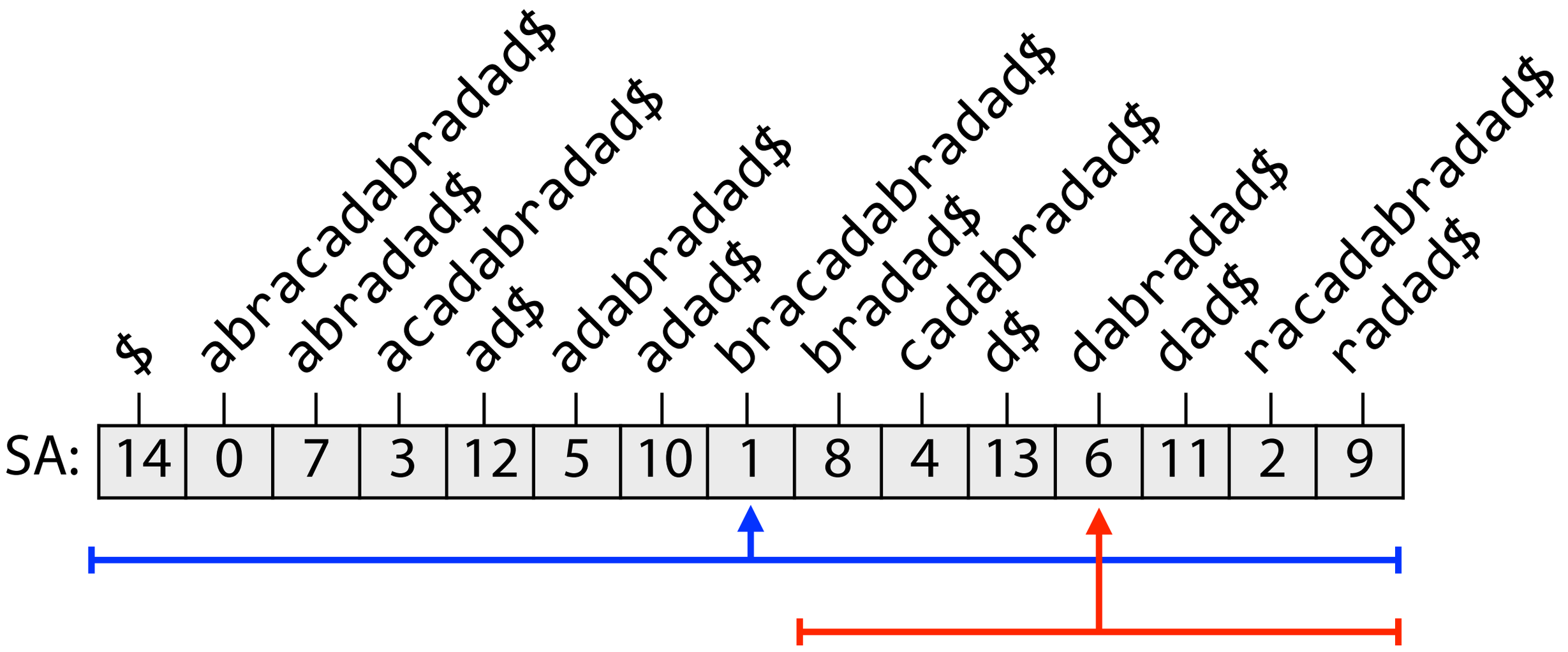
Suffix array: querying



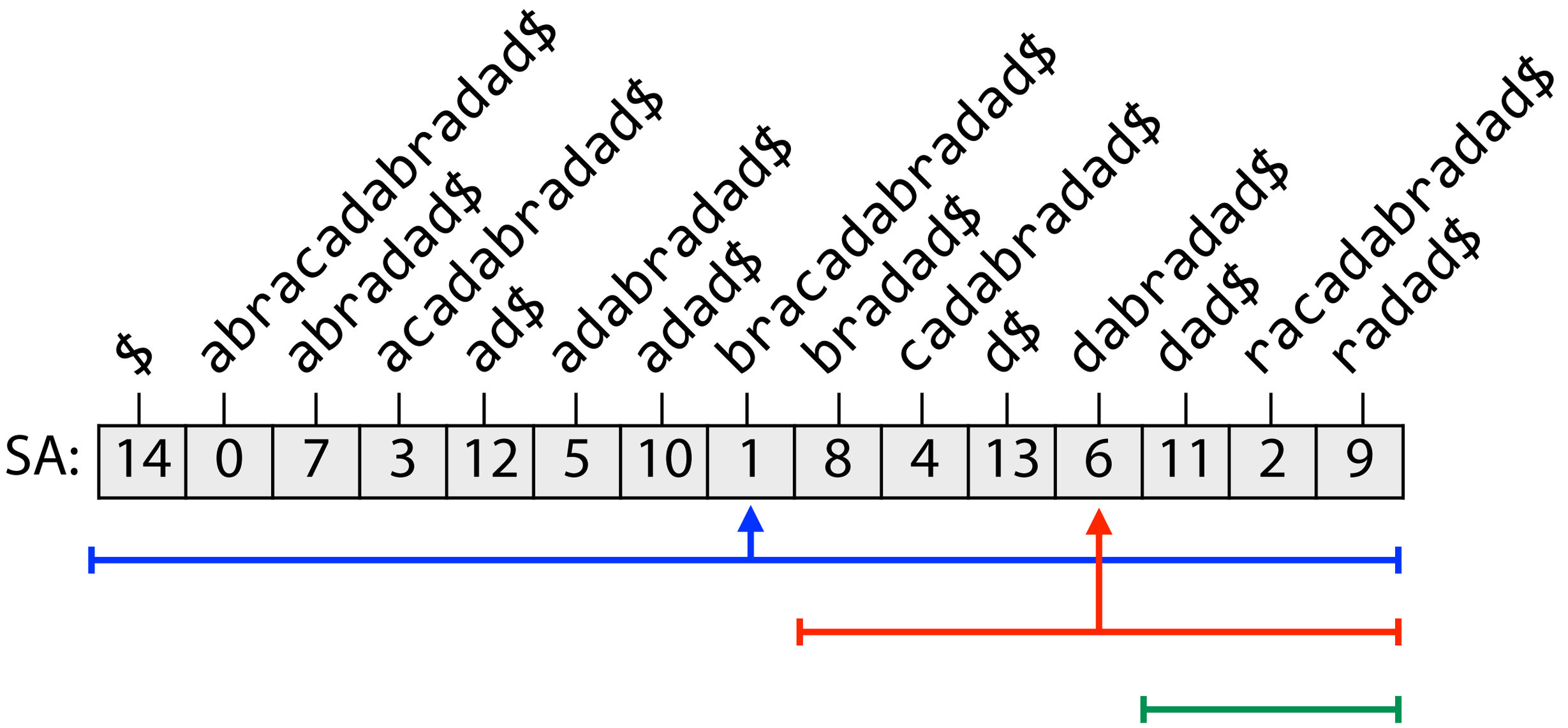
Iteration 2: compare **dad** to pivot **dabradad$**

dad is alphabetically after; recurse on bottom half

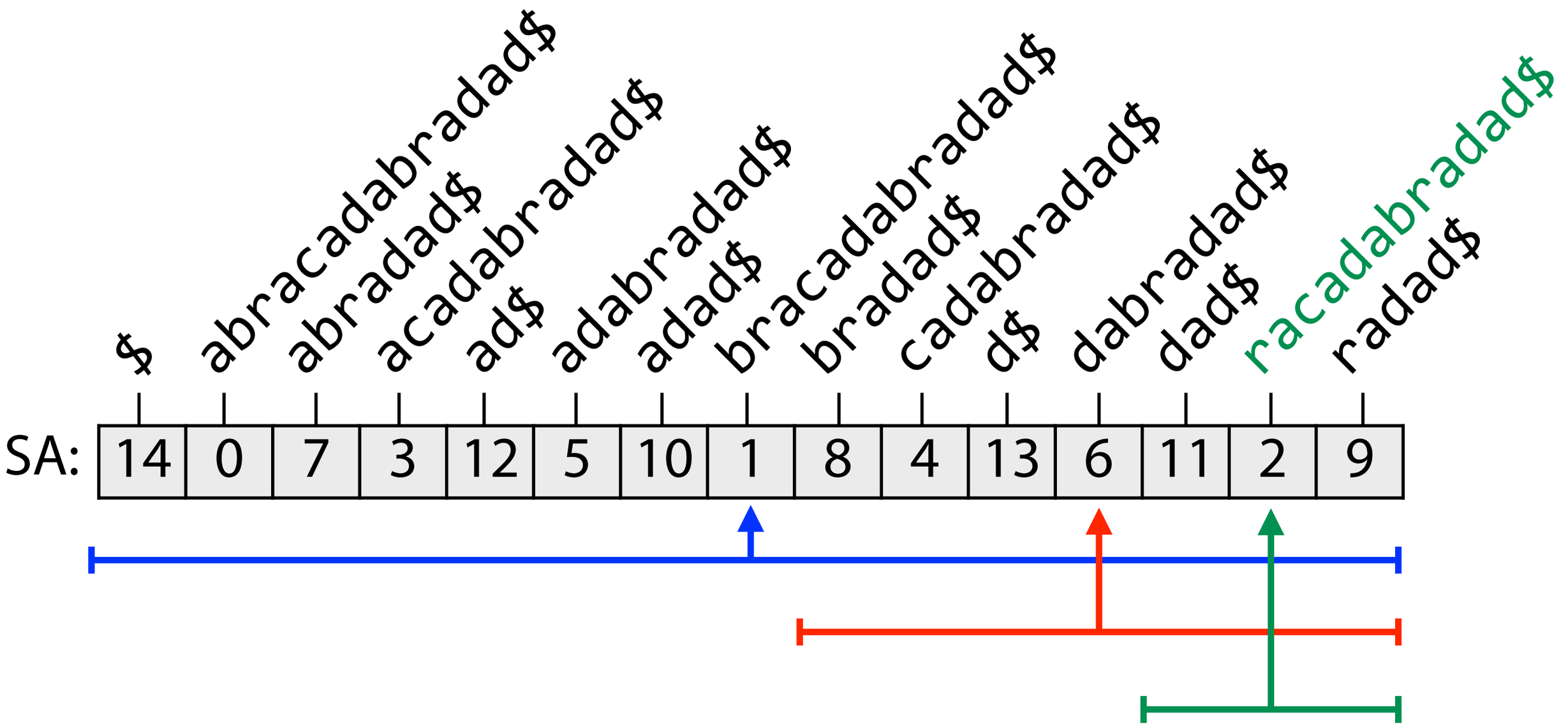
Suffix array: querying



Suffix array: querying



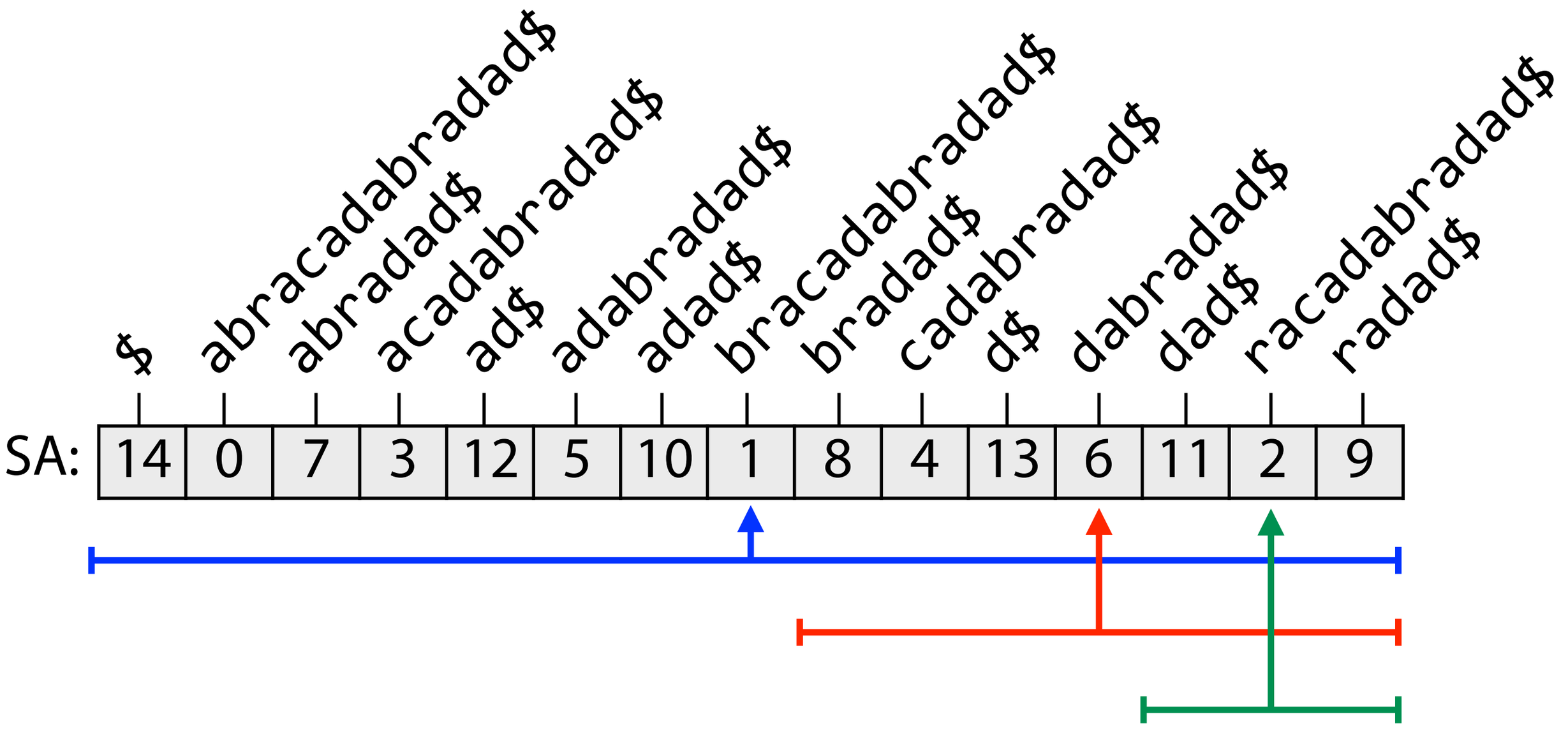
Suffix array: querying



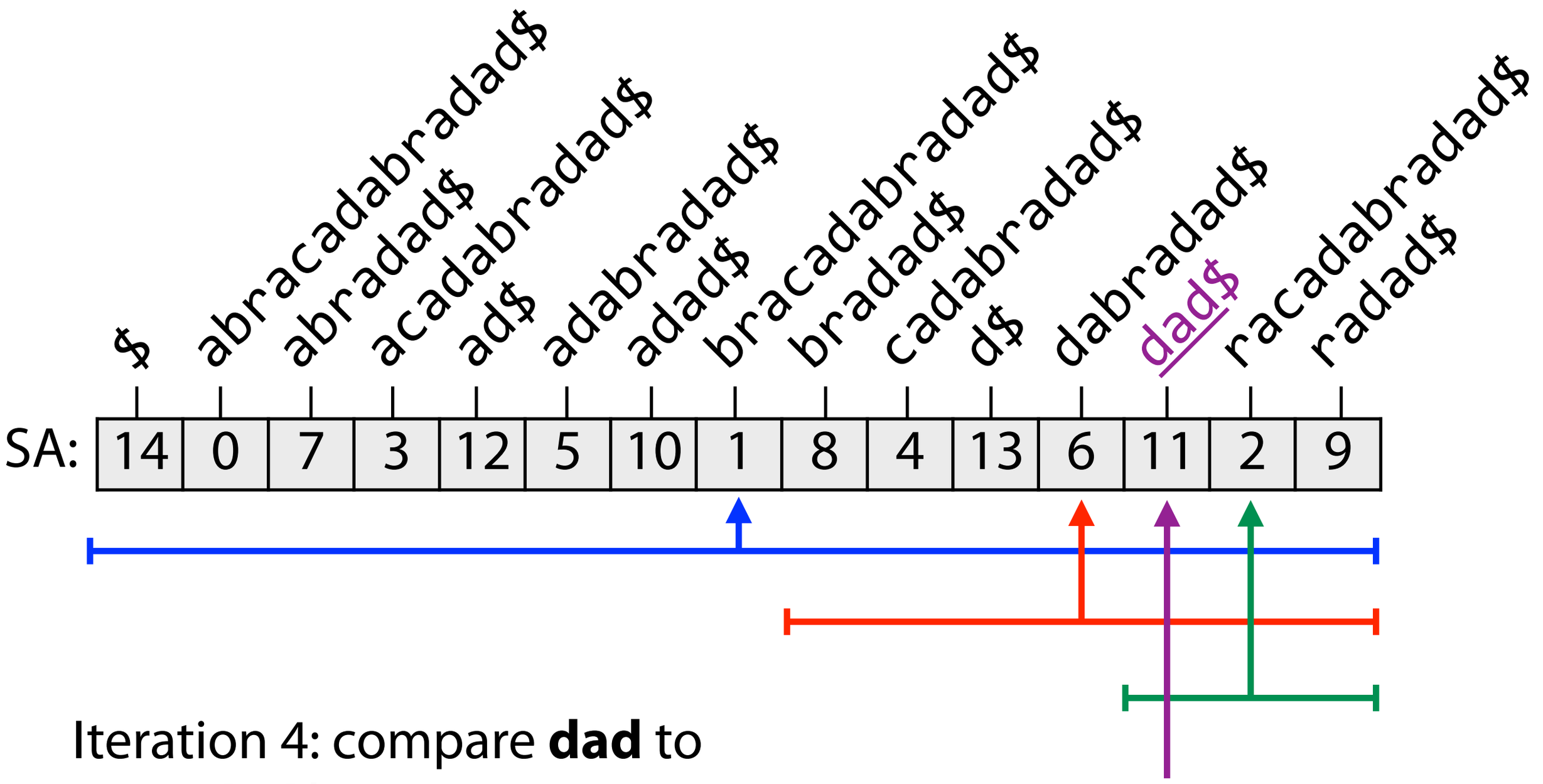
Iteration 3: compare **dad** to pivot **racadabradad$**

dad is before

Suffix array: querying



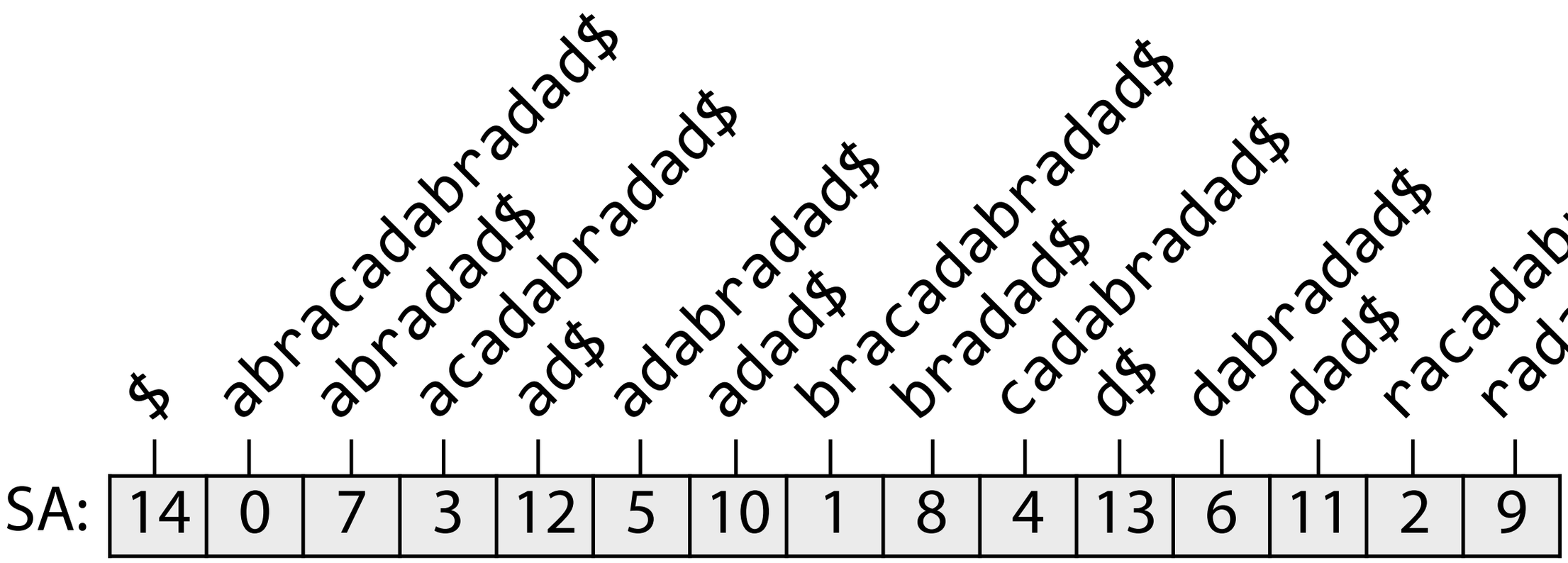
Suffix array: querying



Iteration 4: compare **dad** to pivot **dad$**

Answer: yes, **dad** appears

Suffix array: querying

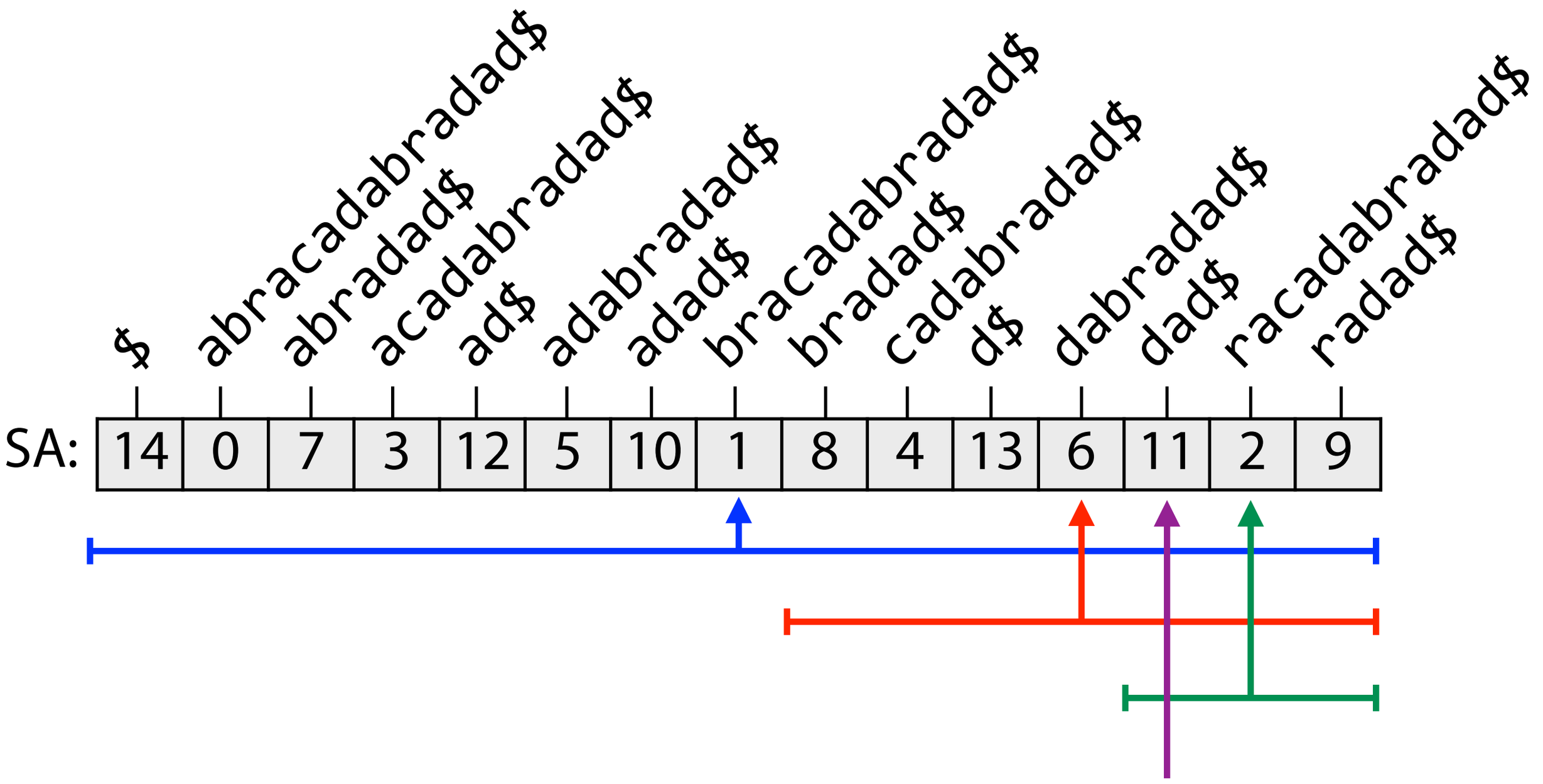


Usually binary search is $O(\log m)$...

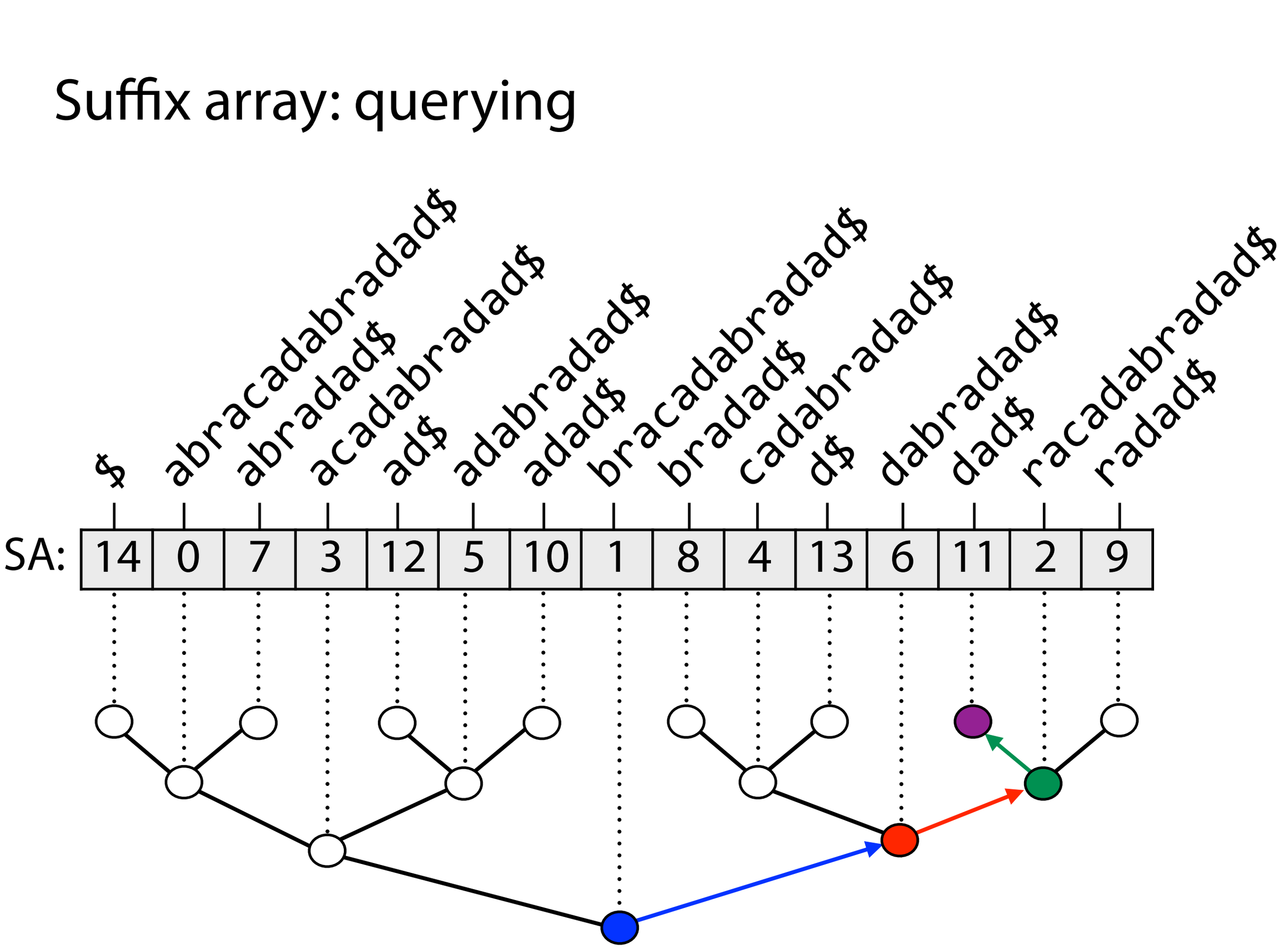
...$O(\log m)$ bisections, $O(1)$ per bisection to compare

But we need $O(n)$ to lexicographically compare the pivot suffix with or length-n query. So $O(n \log m)$ time overall.

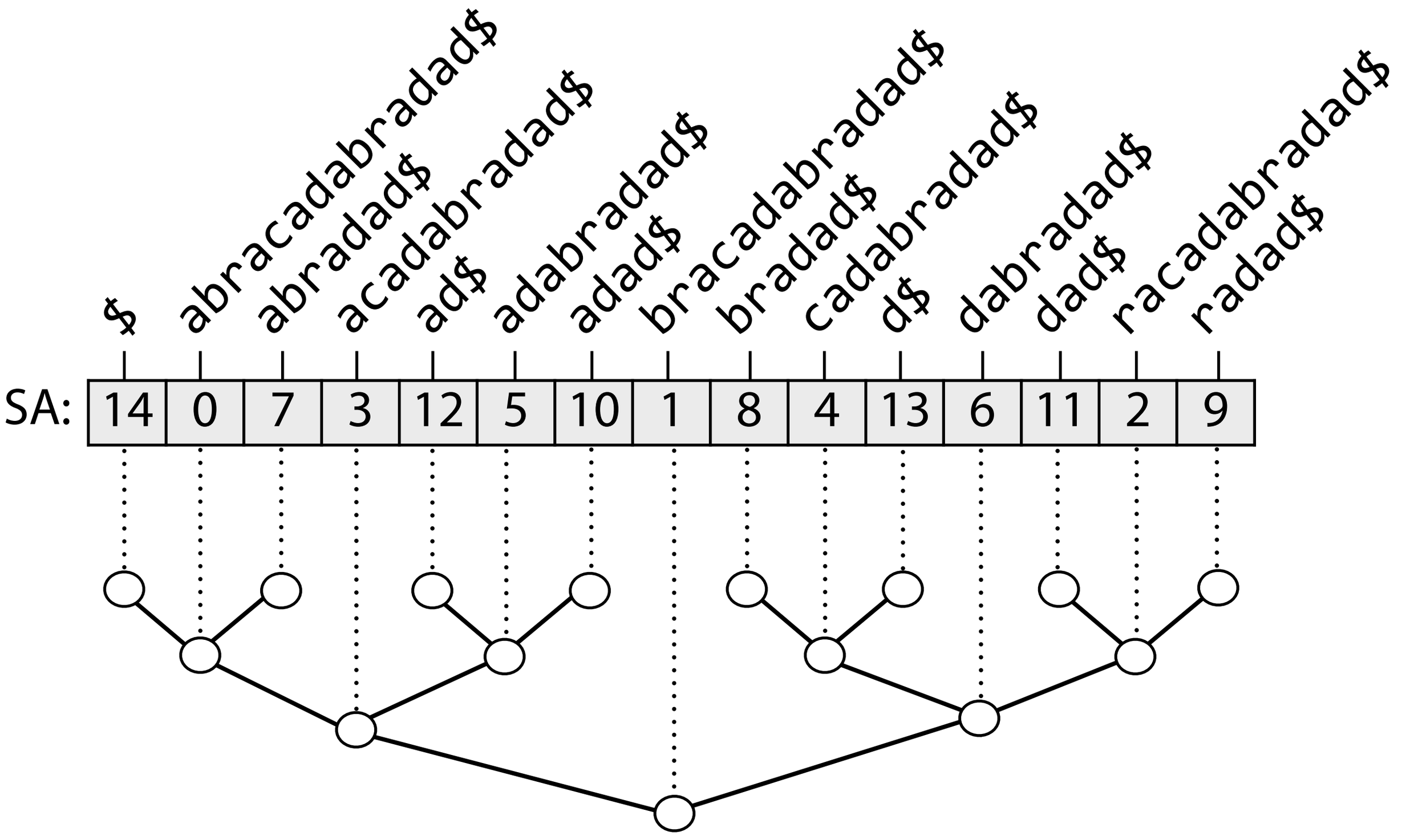
Suffix array: querying



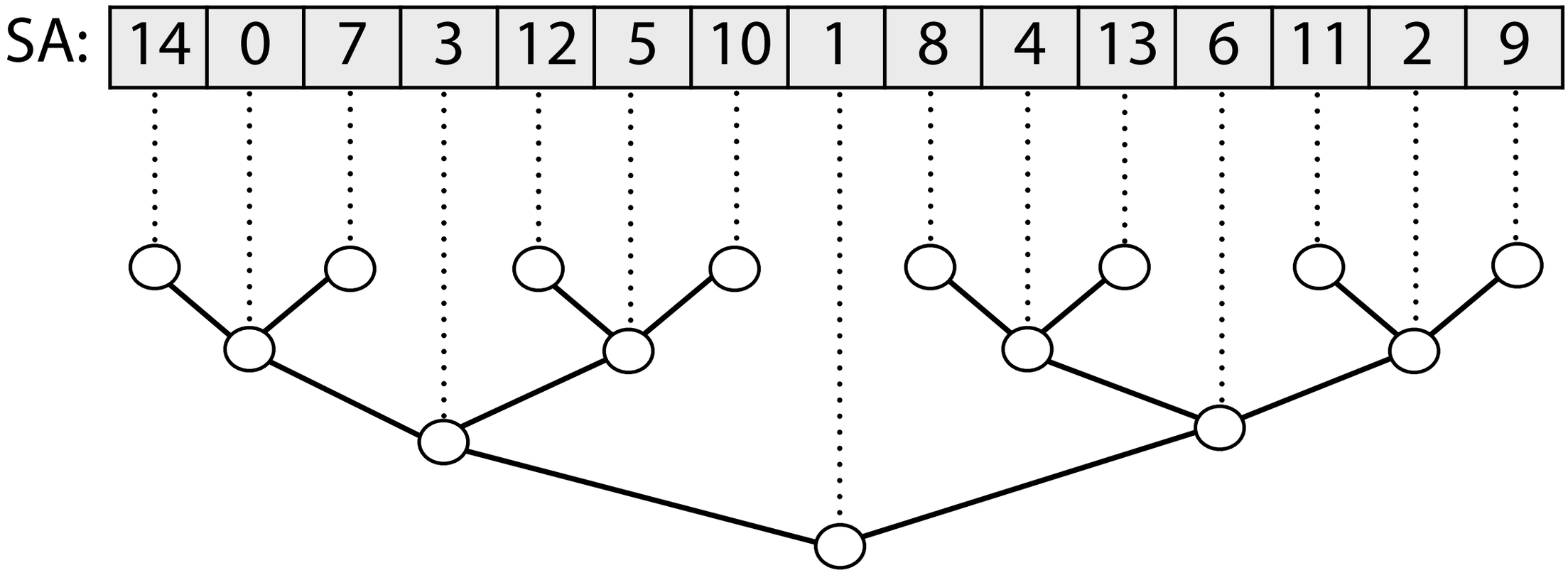
Suffix array: querying



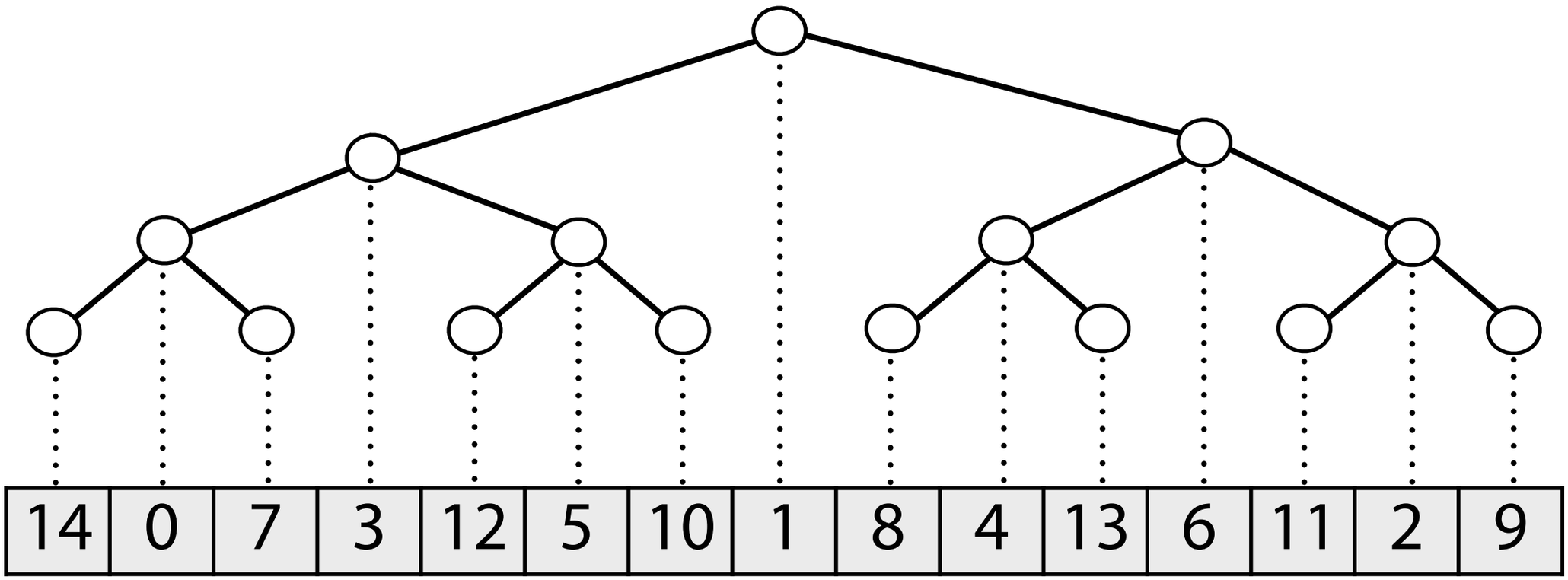
Suffix array: querying



Suffix array

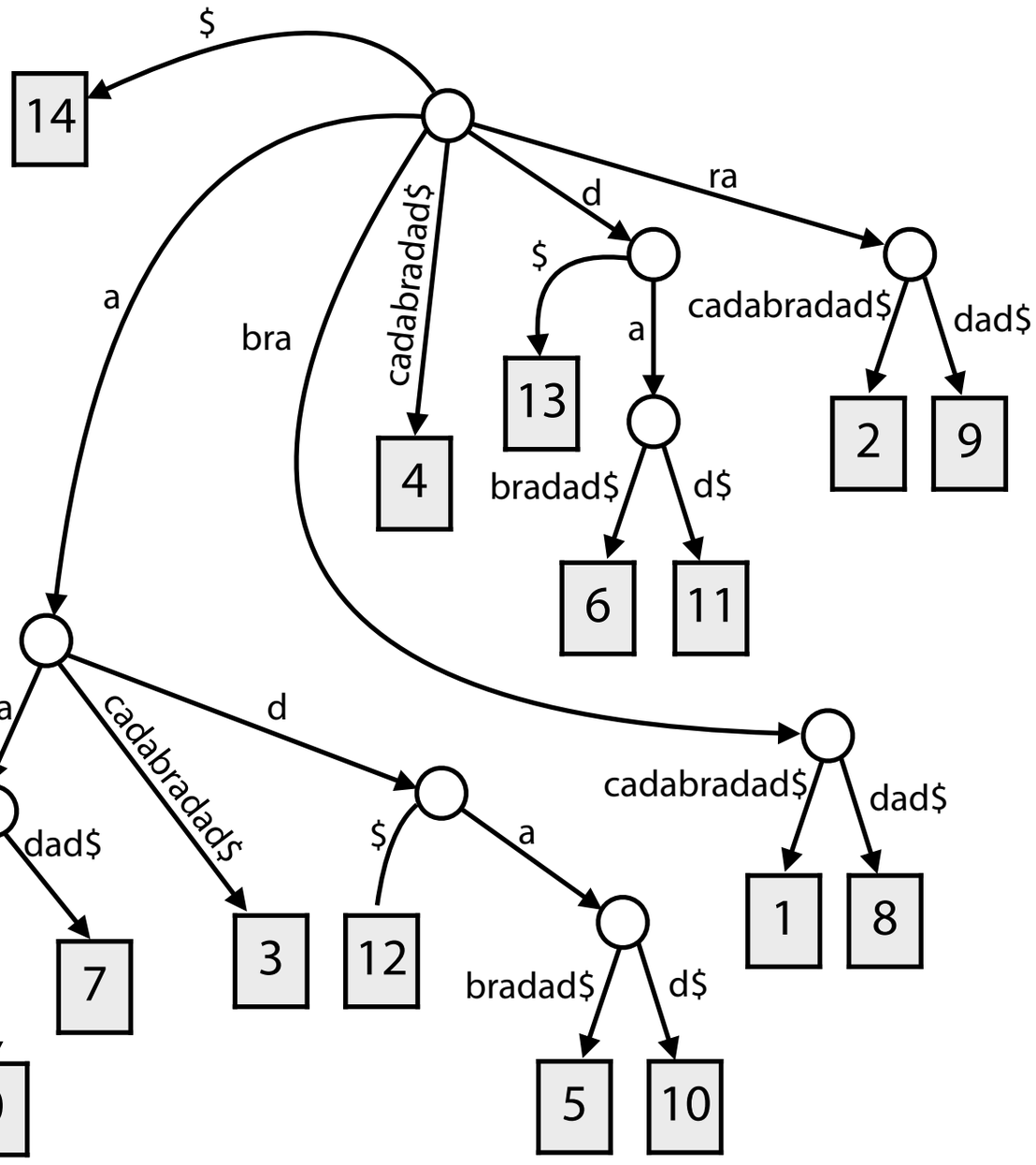


Suffix array

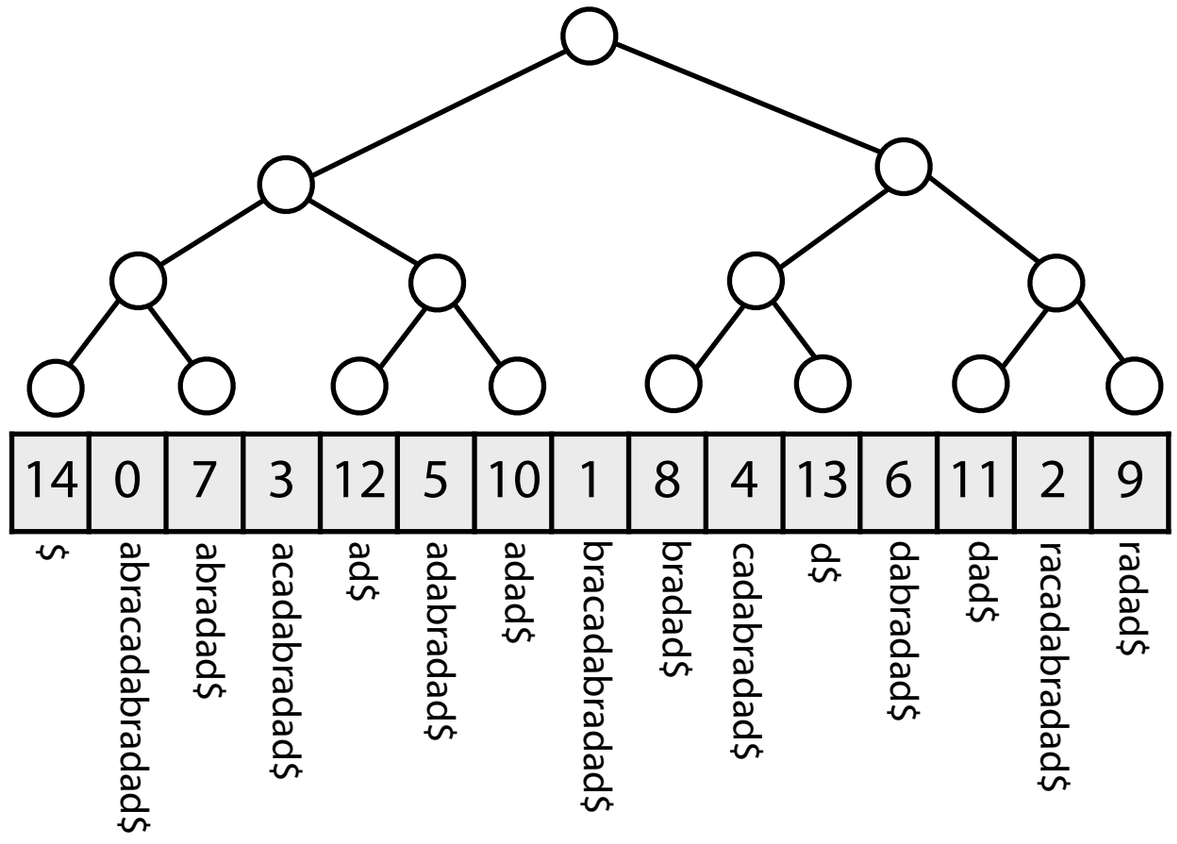


SA:	14	0	7	3	12	5	10	1	8	4	13	6	11	2	9

Suffix array



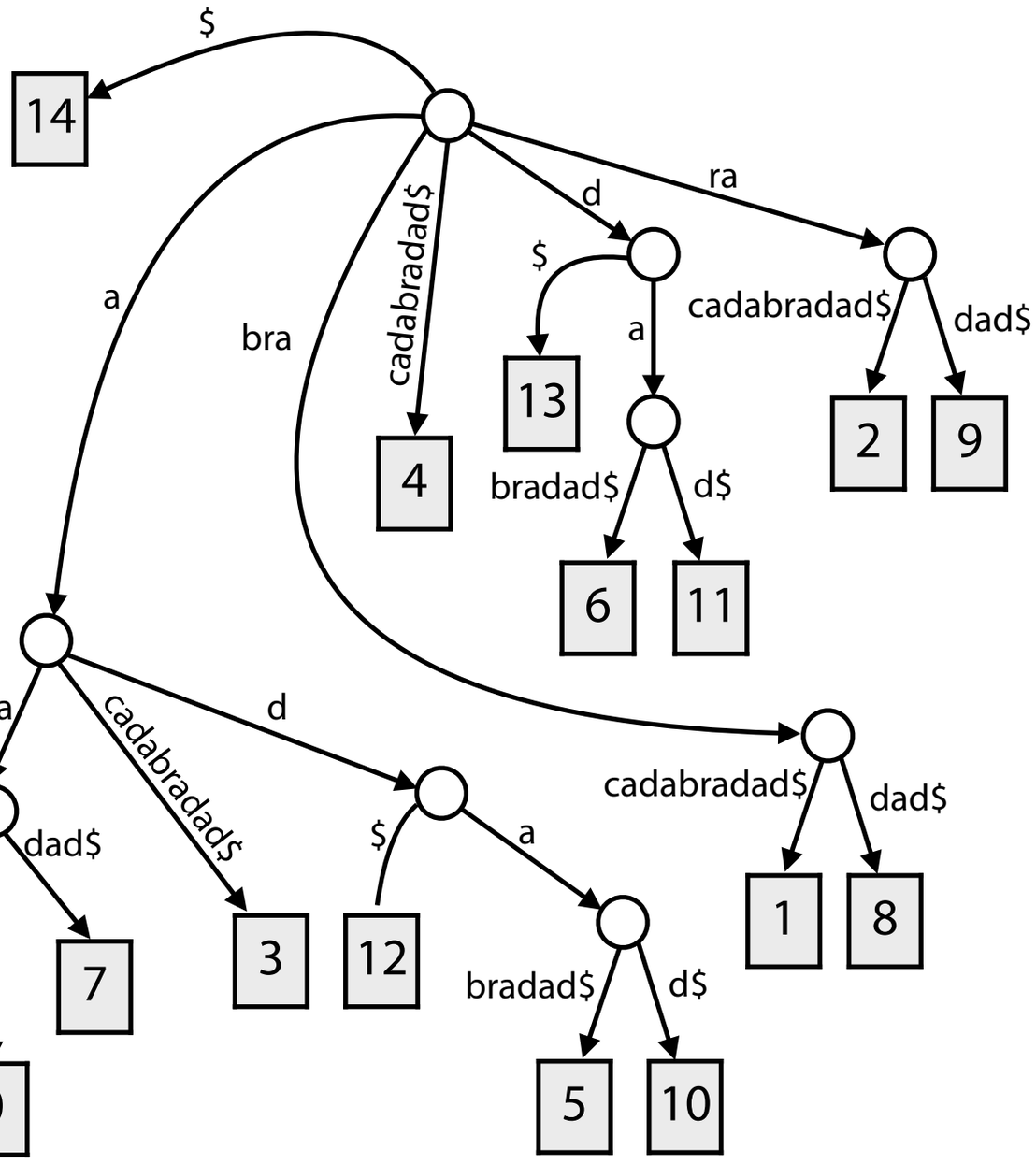
Suffix tree is an **explicit** tree

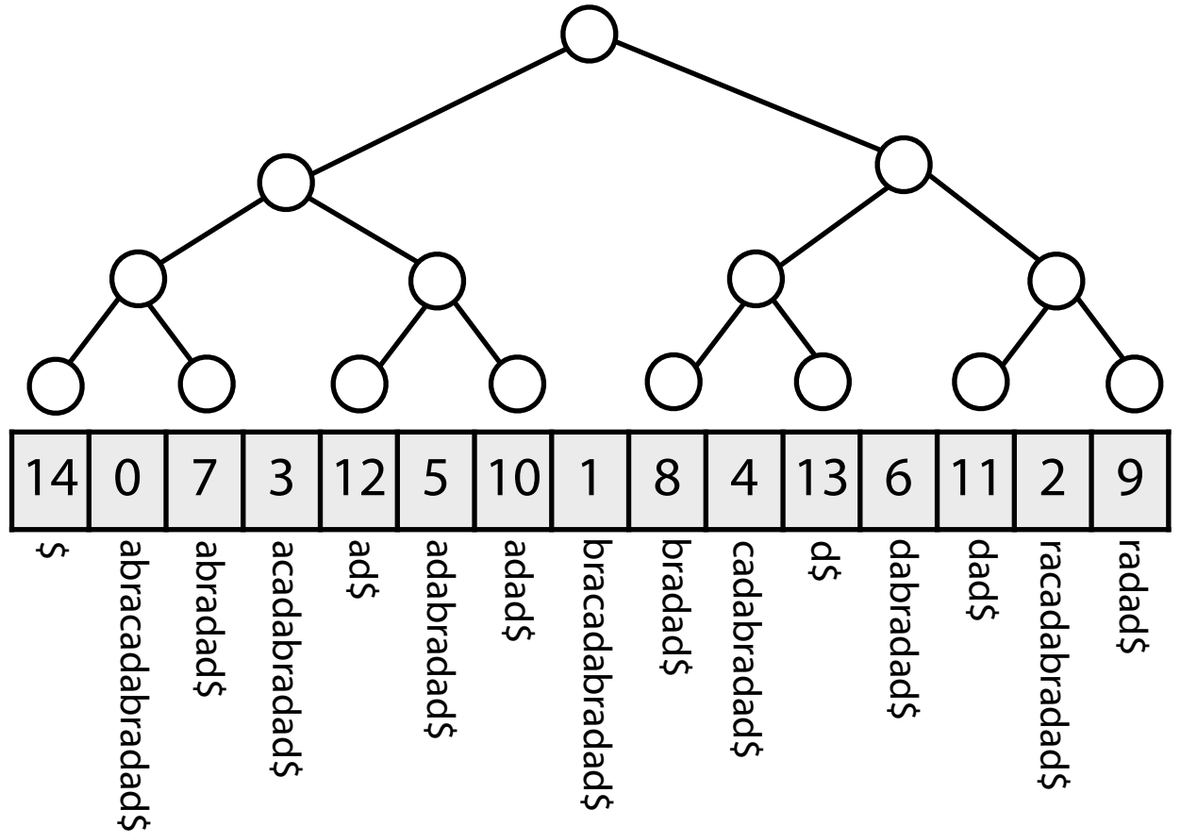


SA binary search tree is **implicit**

...but has same function: narrow the space of suffixes that might have our query as a prefix

Suffix array





Presence/absence is $O(n)$ time for length-n query

🤔🤔🤔

Presence/absence is $O(n \log m)$ time for length-n query