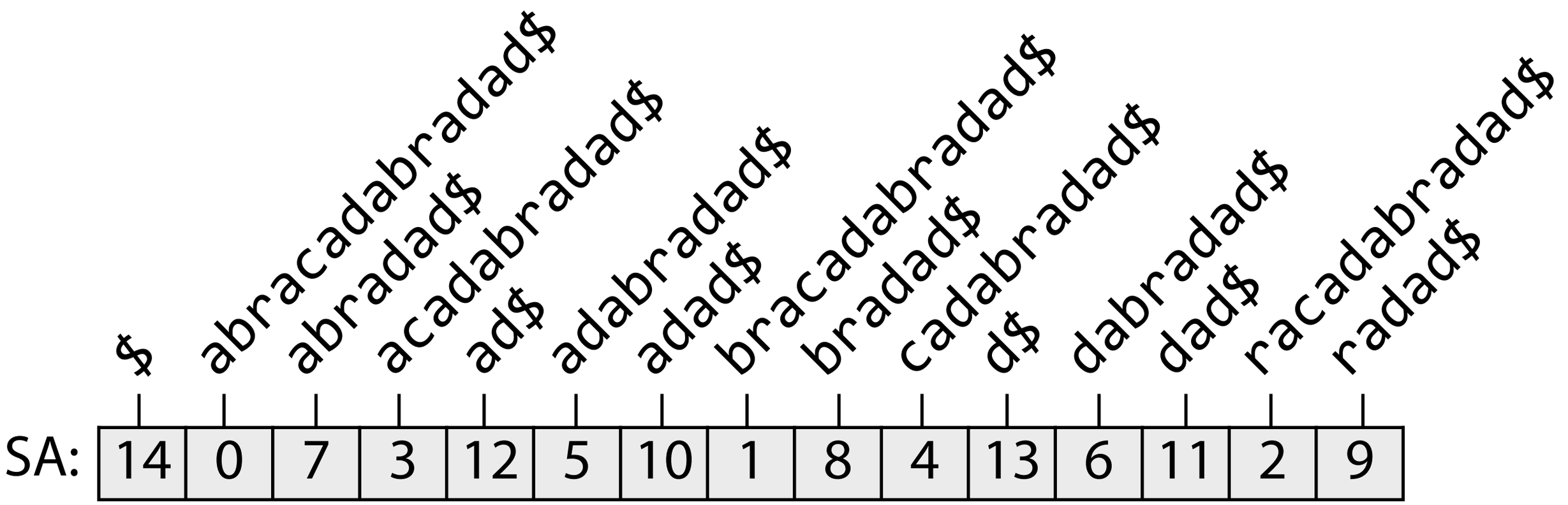
Suffix Arrays: building

Ben Langmead

Suffix array



Building the suffix array is fundamentally about **sorting** the suffixes

Suffix array: sorting suffixes

Should we use our favorite sort, e.g., **quicksort**?

0	$
1	a$
2	aaba$
3	aba$
4	abaaba$
5	ba$
6	baaba$

```
def quicksort(q):
    lt, gt = [], []
    if len(q) <= 1:
        return q
    for x in q[1:]:
        if x < q[0]:
            lt.append(x)
        else:
            gt.append(x)
    return quicksort(lt) + q[0:1] + quicksort(gt)
```

We learn this is
$O(m \log m)$ expected time

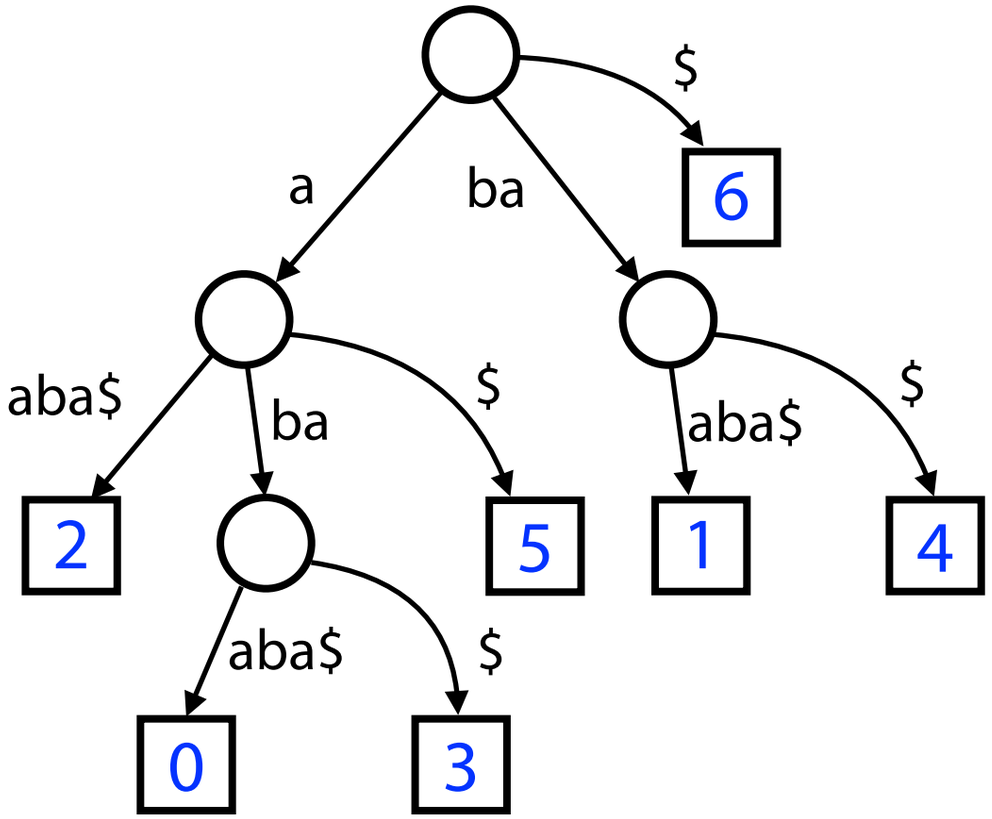
Quicksort is $O(m \log m)$ (expected) when sorting items that can be compared **in constant time**

But our **lexicographic** comparisons are $O(m)$ time!

So $O(m^2 \log m)$ overall 😢

Suffix array: sorting suffixes

Can a suffix **tree** help us build a suffix **array**?

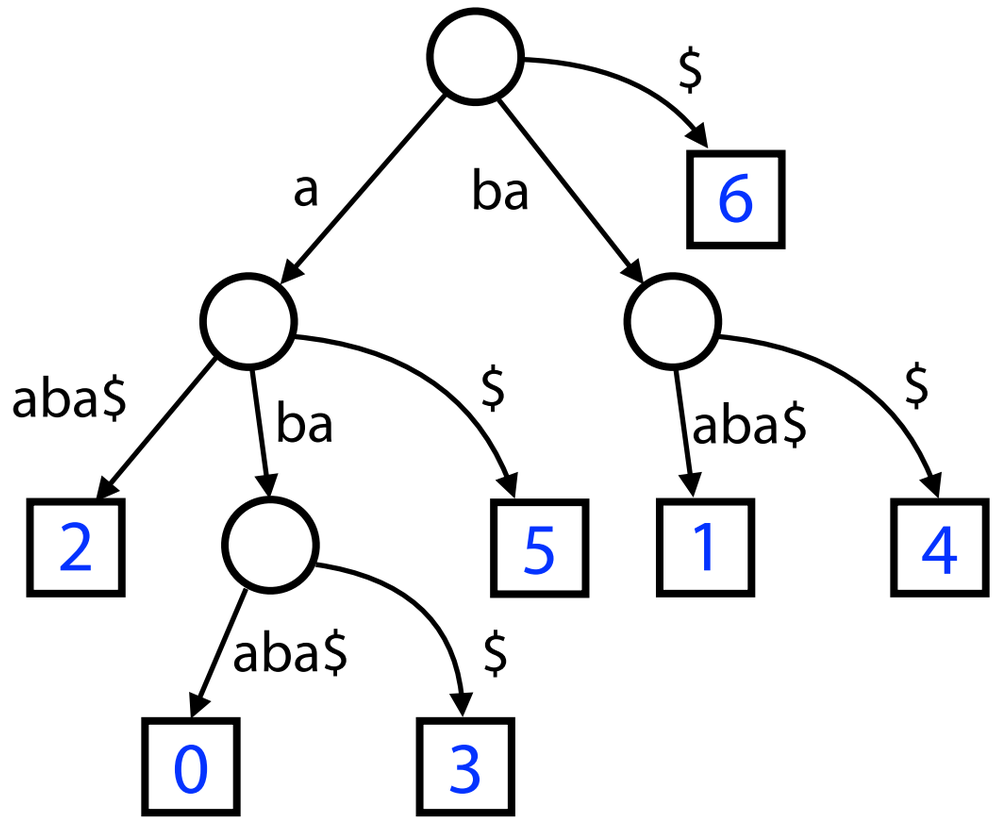


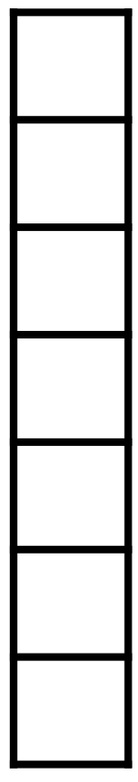
6	$
5	a$
3	aaba$
1	aba$
0	abaaba$
4	ba$
2	baaba$

Recall the leaves of the tree are also the elements of the array. It's a matter of visiting in the right order.

Suffix array: sorting suffixes

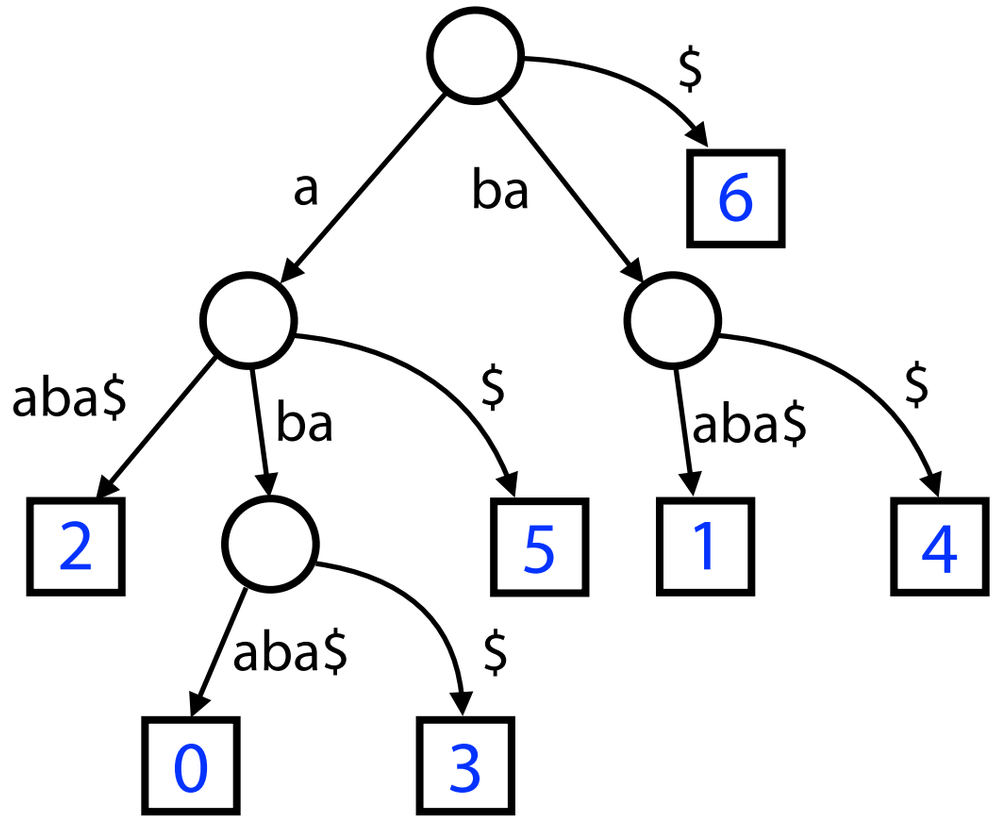
Try: depth-first traversal, always visiting children in lexicographical order





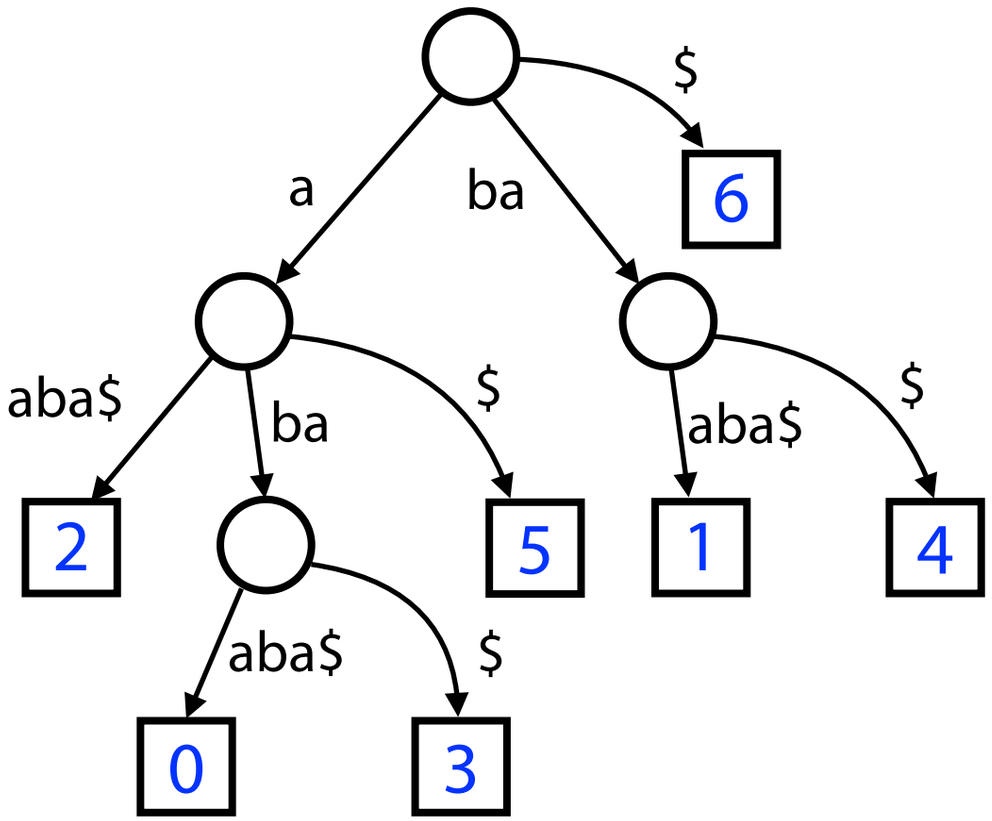
Suffix array: sorting suffixes

Try: depth-first traversal, always visiting children in lexicographical order



6	$
5	a$
3	aaba$
1	aba$
0	abaaba$
4	ba$
2	baaba$

Suffix array: sorting suffixes



6	$
5	a$
3	aaba$
1	aba$
0	abaaba$
4	ba$
2	baaba$

Suffix tree & array are both $O(m)$ space

Suffix tree construction with Ukkonen is $O(m)$ time

Traversal is $O(m)$ time

So this is $O(m)$ time & space ✅

Suffix array: sorting suffixes

To avoid overhead of building a tree first, we can choose a "direct" algorithm

Fast-in-practice $O(m \log m)$ algorithms

Manber U, Myers G. "Suffix arrays: a new method for on-line string searches." SIAM Journal on Computing 22.5 (1993): 935-948.

Larsson NJ, Sadakane K. Faster suffix sorting. Technical Report LU-CS-TR:99-214, LUNDFD6/(NFCS-3140)/1-43/(1999), Department of Computer Science, Lund University, Sweden, 1999.

Newer $O(m)$ algorithms as well!

Kärkkäinen J, Sanders P. "Simple linear work suffix array construction." Automata, Languages and Programming (2003): 187-187.

Ko P, Aluru S. "Space efficient linear time construction of suffix arrays." *Combinatorial Pattern Matching*. Springer Berlin Heidelberg (2003): 200--210.

Nong G, Zhang S, and Chan WH. "Two efficient algorithms for linear time suffix array construction." IEEE transactions on computers 60.10 (2010): 1471-1484. ⭐