Suffix arrays

Ben Langmead

JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

Department of Computer Science

Suffix array

```
T = abaaba$
    0123456
```

As with suffix tree,
T is part of index

SA(*T*) =

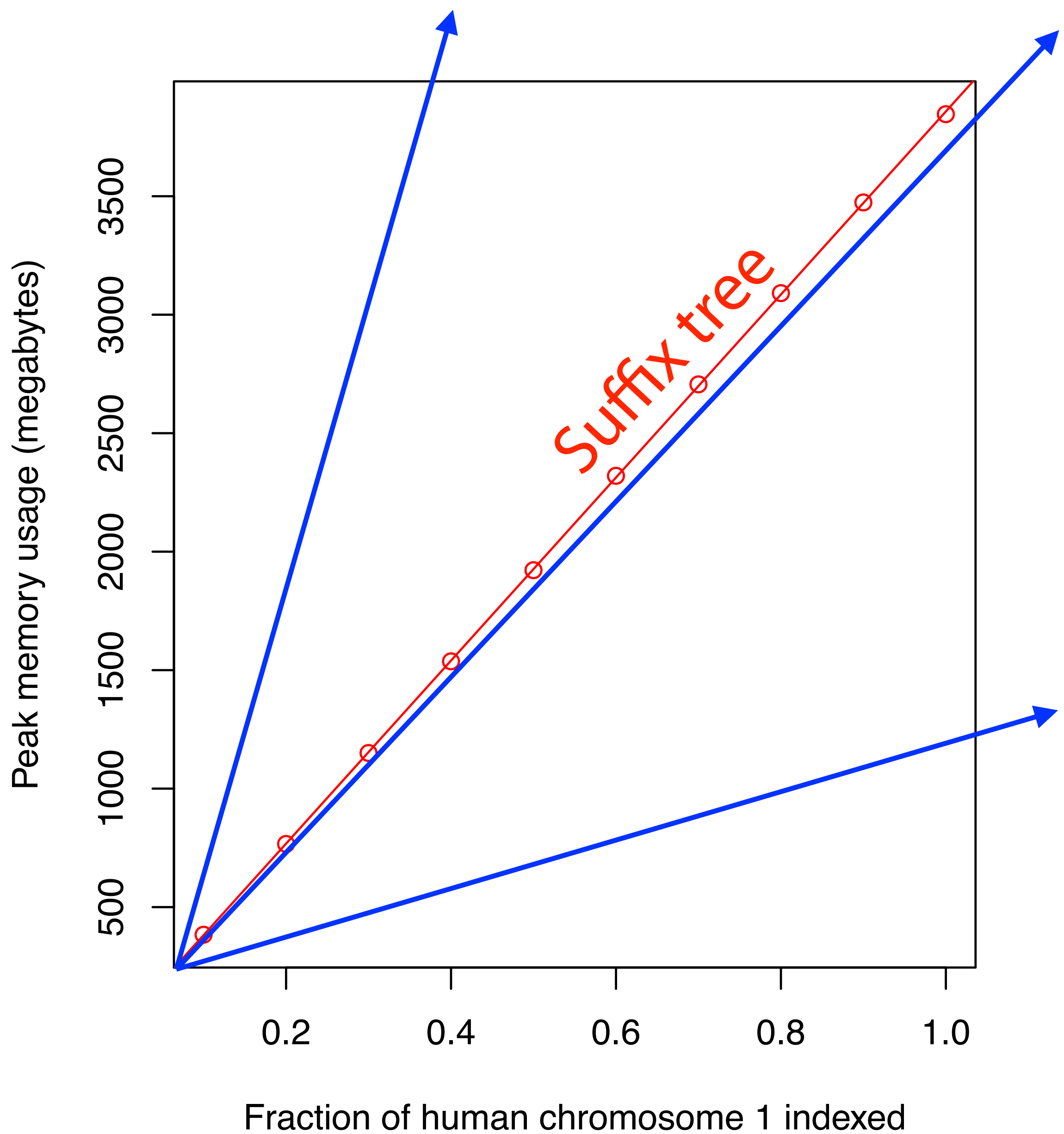
SA(*T*)	Suffix
6	$
5	a $
2	a a b a $
3	a b a $
0	a b a a b a $
4	b a $
1	b a a b a $

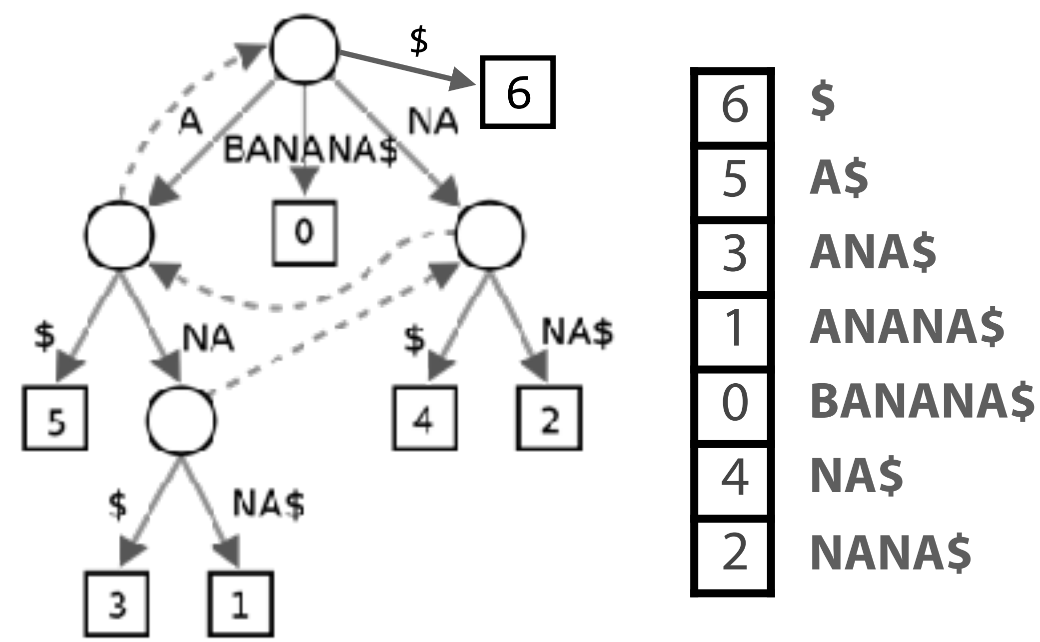
m integers

Suffix array of *T* is an array of integers in [0, *m*) specifying lexicographic (alphabetical) order of *T*'s suffixes

Suffix array

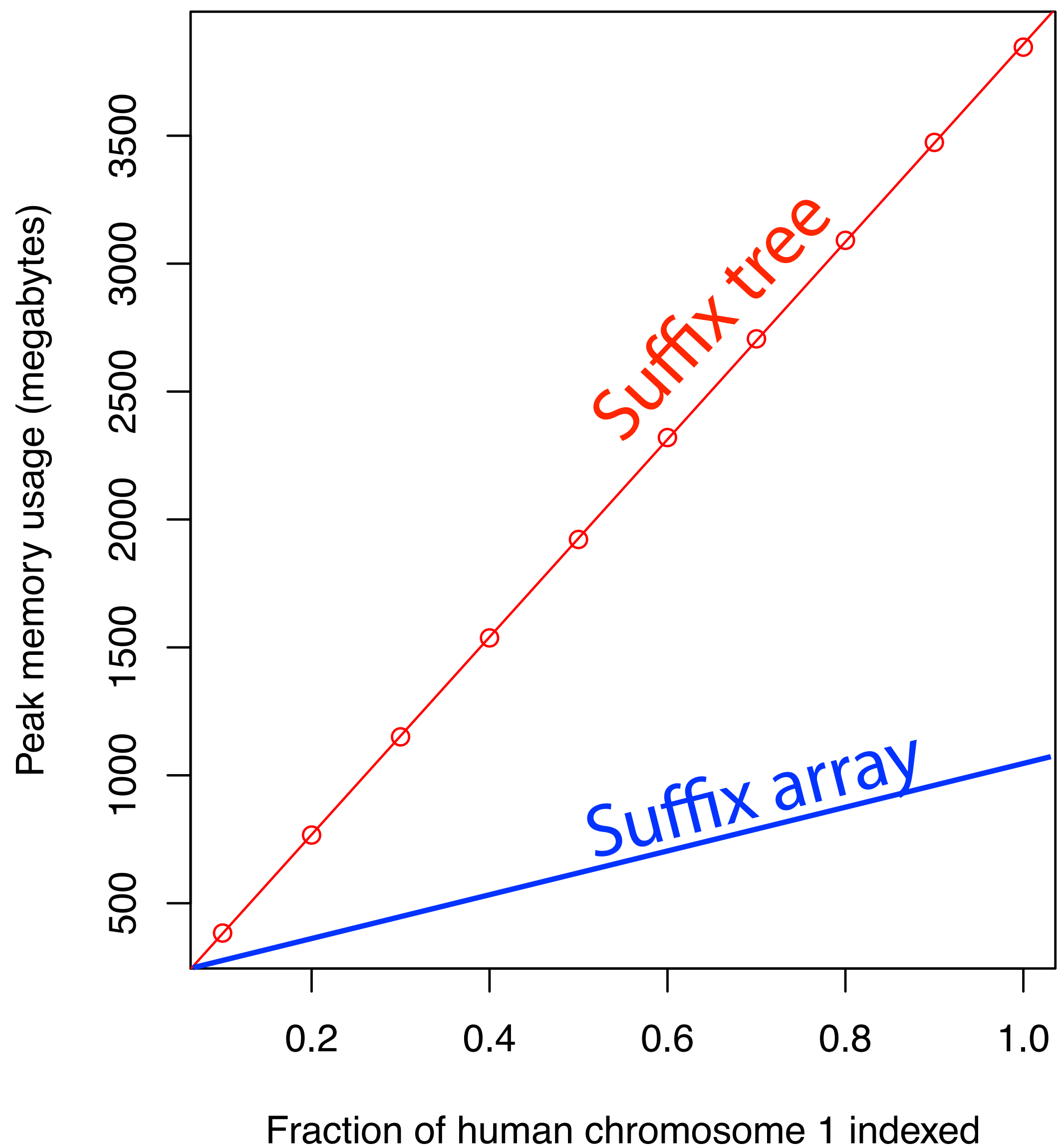
$O(m)$ space, like suffix tree   Is "constant factor" worse, better, same?

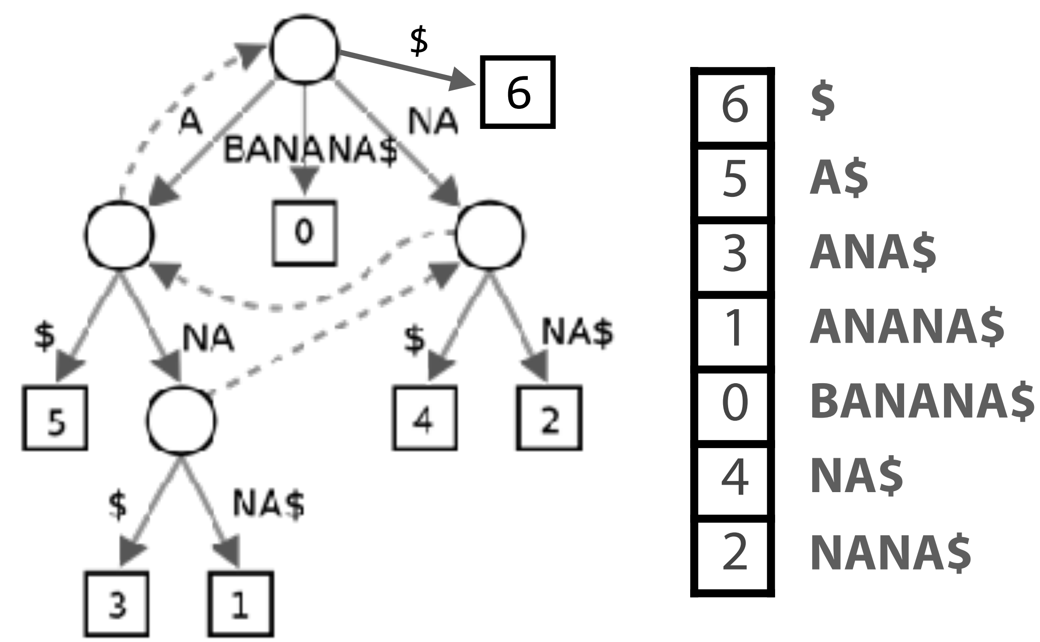




Suffix array

32-bit integers sufficient for human genome, so fits in
~4 bytes/base × 3 billion bases ≈ 12 GB. Suffix tree is >45 GB.





Suffix array: querying

Is P a substring of T?

T = abaaba$

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

Use binary search

6	$
5	a $
2	a a b a $
3	a b a $
0	a b a a b a $
4	b a $
1	b a a b a $

Suffix array: querying

Is P a substring of T?

Do binary search, check whether P is a prefix of the suffix there

Query time is $O(\;?\;)$...

...$O(\log_2 m)$ bisections, $O(n)$ comparisons per bisection, so $O(n \log m)$

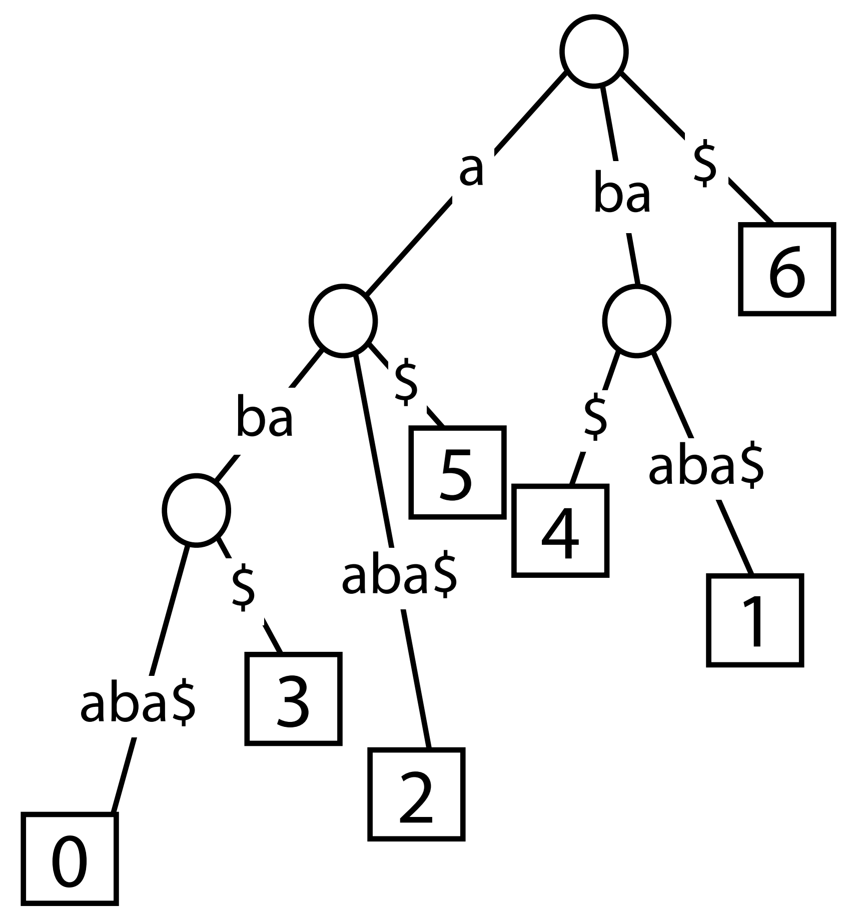
T = abaaba$

6	$
5	a $
2	a a b a $
3	a b a $
0	a b a a b a $
4	b a $
1	b a a b a $

Suffix array: querying

Contrast suffix array query time: O(n log m) with suffix tree: O(n)

6	$
5	a $
2	a a b a $
3	a b a $
0	a b a a b a $
4	b a $
1	b a a b a $



Time can be improved to O(n + log m), but we won't discuss here (See Gusfield 7.17.4). For this class, we'll consider it O(n log m).

Suffix array: sorting suffixes

Use your favorite sort, e.g., quicksort

0	a b a a b a $
1	b a a b a $
2	a a b a $
3	a b a $
4	b a $
5	a $
6	$

```
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)
```

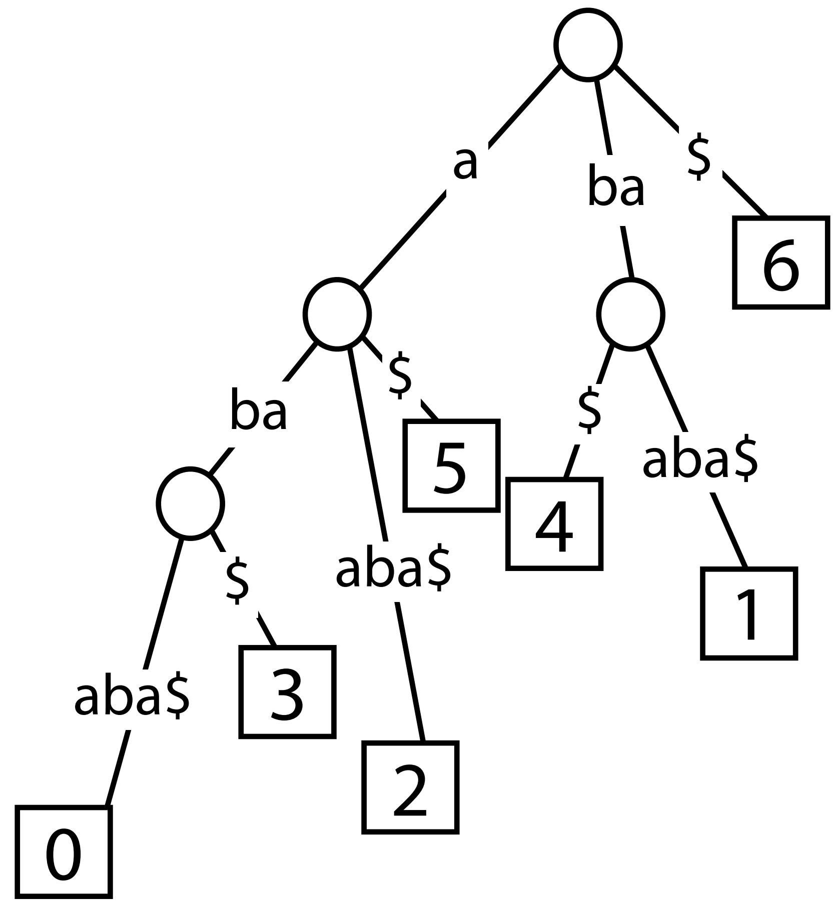
Expected time: O($m^2 \log m$)

Not $O(m \log m)$ because a suffix comparison is O(m) time

Suffix array: building

How to build a suffix array?

6	$
5	a $
2	a a b a $
3	a b a $
0	a b a a b a $
4	b a $
1	b a a b a $



(a) Build suffix tree, (b) traverse in alphabetical order,
(c) upon reaching leaf, append suffix to array

Suffix array: sorting suffixes

Another idea: Use a sort algorithm that's aware that the items being sorted are all suffixes of the same string

Original suffix array paper suggested an $O(m \log m)$ algorithm

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

Other popular $O(m \log m)$ algorithms have been suggested

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.

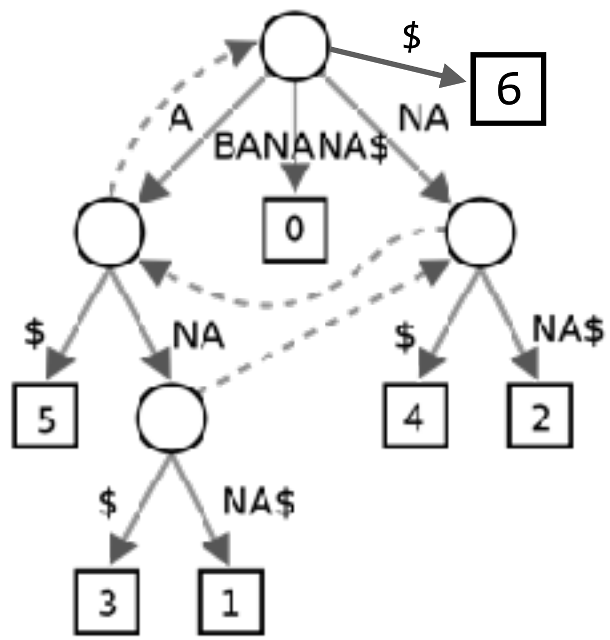
More recently $O(m)$ algorithms have been demonstrated!

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

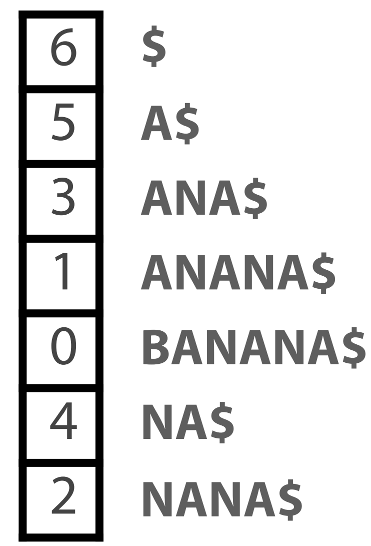
Ko P, Aluru S. "Space efficient linear time construction of suffix arrays." *Combinatorial Pattern Matching*. Springer Berlin Heidelberg, 2003.

Suffix array: summary

Just *m* integers, with O(*n* log *m*) query time



Suffix Tree



Suffix Array

Constant factor greatly reduced compared to suffix tree: human genome index fits in ~12 GB instead of > 45 GB