Suffix Arrays

While we've seen that suffix trees require $O(m)$ space, the constants in that bound can be very large. A related data structure, suffix arrays, allow a much more compact representation at the cost of a small penalty at search time.

Simple idea: sort the suffixes of string S, store the starting indices of sorted suffixes in an array.

Ex:

S = BANANA$

1. BANANA$
2. ANANA$
3. NANA$
4. ANA$
5. NA$
6. A$
7. $

sort →

7. $
6. A$
4. ANA$
2. ANANA$
1. BANANA$
5. NA$
3. NANA$

store indices →

7
6
4
2
1
5
3

Another example:

S = ca#ca+$

Searching for a query in a suffix array

Use binary search: main idea, suffixes that start with the same prefix will be stored in a contiguous range in the suffix array.

$r = |S|$
$l = 1$
for ~~each~~ ~~c in pattern~~ $i = 1, \ldots, |P|$
  set l to smallest index within $l, \ldots, r$ ~~that~~ matches $P[i]$ in position i of suffix, if any
  set r to largest index within $l, \ldots, r$ that matches $P[i]$ in position i of suffix, if any
if $l \leq r$ return suffixes in range $l, \ldots, r$.

Constructing suffix arrays

(1) Naive algorithm: sort the suffixes, $O(n \log n)$ comparisons, each comparison is $O(n)$. Total: $O(n^2 \log n)$.

(2) Linear algorithm: Construct sorted suffix tree, obtain indices ~~by~~ by traversing leaves in order.

S = BANANA$