Suffix Arrays

CMSC 423

Suffix Arrays

- Even though Suffix Trees are O(n) space, the constant hidden by the big-Oh notation is somewhat "big": ≈ 20 bytes / character in good implementations.

- If you have a 10Gb genome, 20 bytes / character = 200Gb to store your suffix tree. "Linear" but large.

- Suffix arrays are a more efficient way to store the suffixes that can do most of what suffix trees can do, but just a bit slower.

- Slight space vs. time tradeoff.

Example Suffix Array

s = attcatg$

- Idea: lexicographically sort all the suffixes.
- Store the starting indices of the suffixes in an array.

1	attcatg$
2	ttcatg$
3	tcatg$
4	catg$
5	atg$
6	tg$
7	g$
8	$

index of suffix suffix of s

sort the suffixes alphabetically

the indices just "come along for the ride"

8	$
5	atg$
1	attcatg$
4	catg$
7	g$
3	tcatg$
6	tg$
2	ttcatg$

Example Suffix Array

s = attcatg$

- Idea: lexicographically sort all the suffixes.
- Store the starting indices of the suffixes in an array.

1	attcatg$
2	ttcatg$
3	tcatg$
4	catg$
5	atg$
6	tg$
7	g$
8	$

index of suffix suffix of s

sort the suffixes alphabetically

the indices just "come along for the ride"

8
5
1
4
7
3
6
2

Another Example Suffix Array

s = cattcat$

- Idea: lexicographically sort all the suffixes.
- Store the starting indices of the suffixes in an array.

1	cattcat$
2	attcat$
3	ttcat$
4	tcat$
5	cat$
6	at$
7	t$
8	$

index of suffix suffix of s

sort the suffixes alphabetically

the indices just "come along for the ride"

8	$
6	at$
2	attcat$
5	cat$
1	cattcat$
7	t$
4	tcat$
3	ttcat$

Another Example Suffix Array

s = cattcat$

- Idea: lexicographically sort all the suffixes.
- Store the starting indices of the suffixes in an array.

1	cattcat$
2	attcat$
3	ttcat$
4	tcat$
5	cat$
6	at$
7	t$
8	$

index of suffix suffix of s

sort the suffixes alphabetically

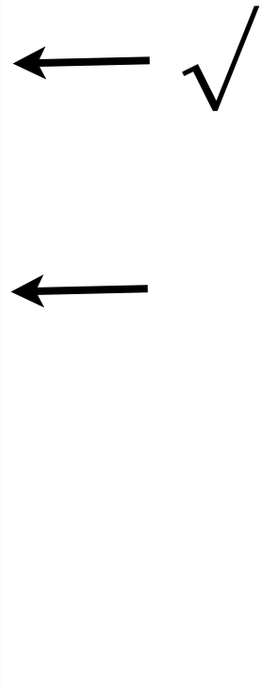
the indices just "come along for the ride"

8
6
2
5
1
7
4
3

Search via Suffix Arrays

s = cattcat$

8	$
6	at$
2	attcat$
5	cat$
1	cattcat$
7	t$
4	tcat$
3	ttcat$



- Does string "at" occur in s?
- Binary search to find "at".

- What about "tt"?

Counting via Suffix Arrays

s = cattcat$

8	$
6	at$
2	attcat$
5	cat$
1	cattcat$
7	t$
4	tcat$
3	ttcat$

- How many times does “at” occur in the string?
- All the suffixes that start with “at” will be next to each other in the array.
- Find one suffix that starts with “at” (using binary search).
- Then count the neighboring sequences that start with at.

K-mer counting

Problem: Given a string s, an integer k, output all pairs (b, i) such that b is a length-k substring of s that occurs exactly i times.

k = 2

		CurrentCount	
8	$	1	
6	at$	1	
2	attcat$	2	
5	cat$	1	(at,2)
1	cattcat$	2	
7	t$	1	(ca,2)
4	tcat$	1	(t$,1)
3	ttcat$	1	(tc,1)
		1	(tt,1)

1. Build a suffix array.

2. Walk down the suffix array, keeping a CurrentCount count

    If the current suffix has length < k, skip it

    If the current suffix starts with the same length-k string as the previous suffix:

        increment CurrentCount

    else

        output CurrentCount and previous length-k suffix

        CurrentCount := 1

Output CurrentCount & length-k suffix.

Constructing Suffix Arrays

- Easy $O(n^2 \log n)$ algorithm:

  sort the n suffixes, which takes O(n log n) comparisons, where each comparison takes O(n).

- There are several direct O(n) algorithms for constructing suffix arrays that use very little space.
- The Skew Algorithm is one that is based on divide-and-conquer.

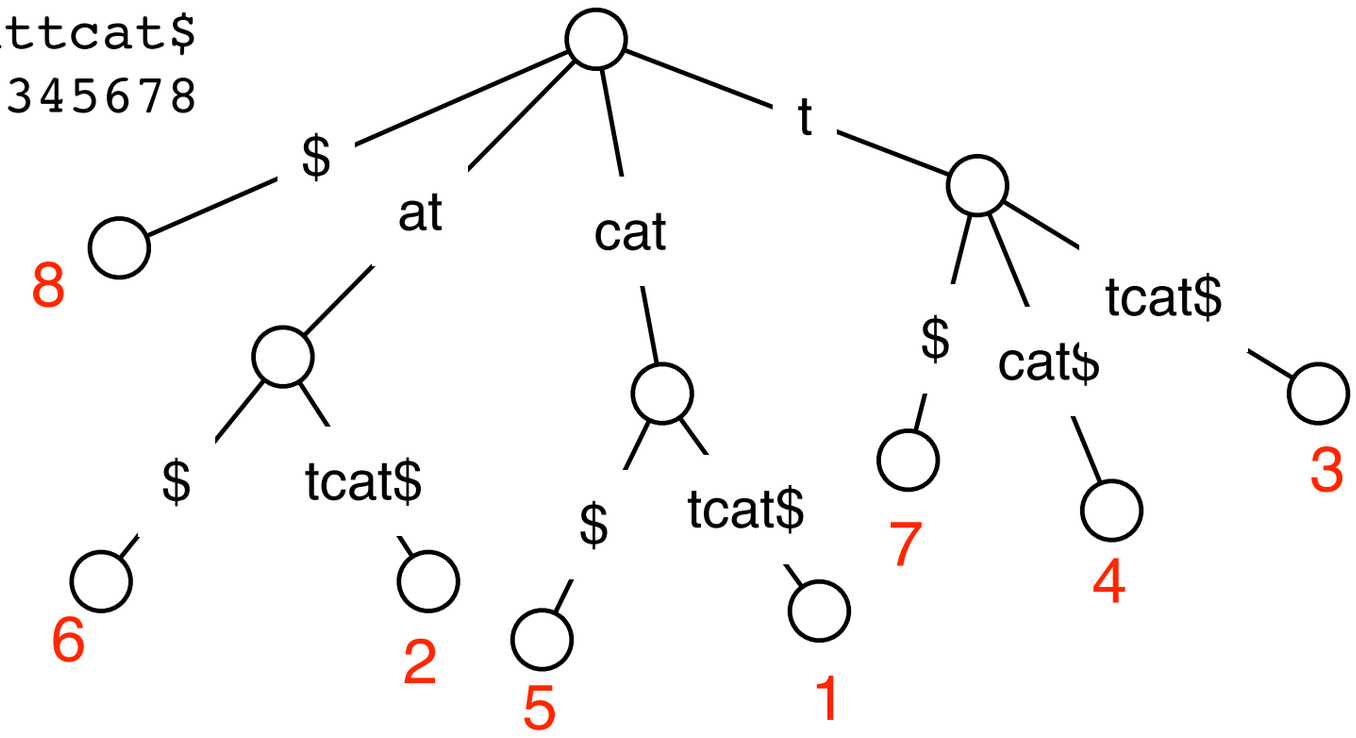
- An simple O(n) algorithm: build the suffix tree, and exploit the relationship between suffix trees and suffix arrays (next slide)

Relationship Between Suffix Trees & Suffix Arrays

$\Sigma = \{\$, a, c, t\}$

s = cattcat$

12345678



Red #s = starting position of the suffix ending at that leaf

Leaf labels left to right: 86251743

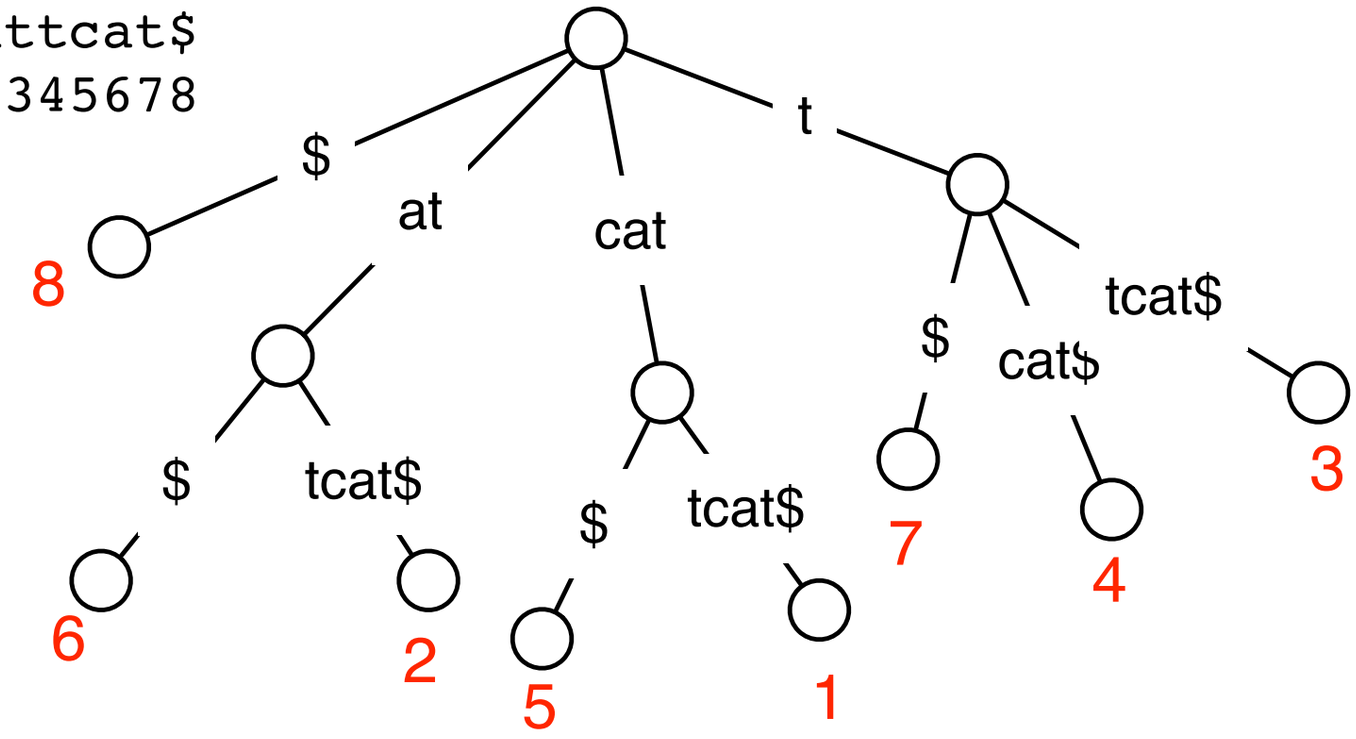
Edges leaving each node are sorted by label (left-to-right).

Relationship Between Suffix Trees & Suffix Arrays

$\Sigma = \{\$,a,c,t\}$

s = cattcat$

12345678



s = cattcat$

8	$
6	at$
2	attcat$
5	cat$
1	cattcat$
7	t$
4	tcat$
3	ttcat$

Red #s = starting position of the suffix ending at that leaf

Edges leaving each node are sorted by label (left-to-right).

Leaf labels left to right: 86251743

Recap

- Suffix arrays can be used to search and count substrings.

- Construction:
  - Easily constructed in $O(n^2 \log n)$
  - Simple algorithms to construct them in O(n) time using possibly $O(n^2)$ space.
  - More complicated algorithms to construct them in O(n) time using at most O(n) space.

- More space efficient than suffix trees: just storing the original string + a list of integers.