Lecture 7 6.851

André Schulz: Notes

STRING DS

Notation:
- Σ, finite Alphabet
- $T \in \Sigma^*$ text
- $P \in \Sigma^*$ search pattern
- $T[i,j]$ substring of T from character $i \ldots j$
- $T[i:]$ suffix starting at i
- concatenation $\circ$

STRING MATCHING:

Input: text T, pattern P
Q: Find all* occurences of P in T

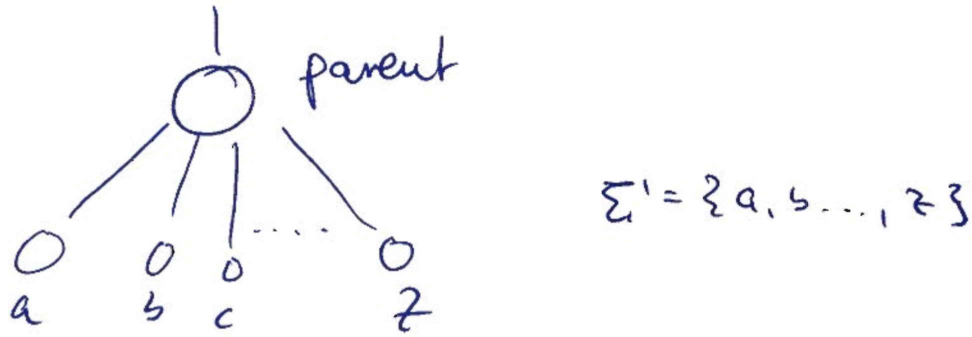
* (also some, first, first k...)

- 1 shot approach takes $O(|T|)$ time
(KNUTH, MORRIS, PRATT 77,
Boyer Moore 77,
Rabin, Karp 87)

How to "store" strings?

<u>TRIE</u> : • tree with fan-out $\leq |\Sigma|$

• egdes parent -> child are labeled with the letters of Σ

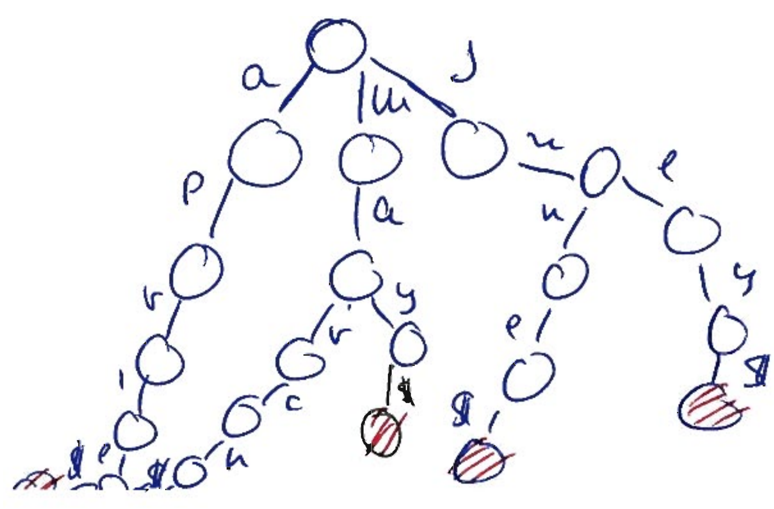


• common technique: terminate strings with "\$" ($\$ \notin \Sigma$)

(to distinguish if a path represents a prefix or word)

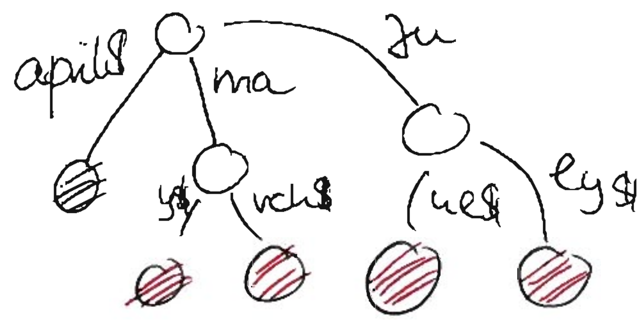
• Root -> leaf paths corresponds to strings, stored in the TRIE

• example : {march, april, may, june, july}



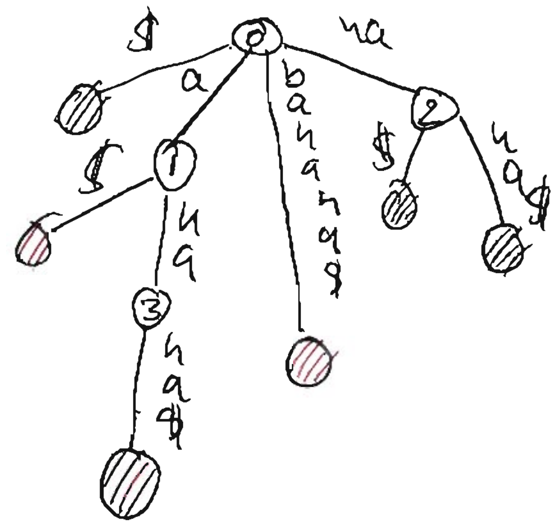
compressed trie:

(contract non-branching paths to single edges)



Suffix tree : compressed trie of all $|T|+1$ suffixes of T

- example : banana$
  ∅ 1 2 3 4 5 6



- ~~length of edges~~ letterdepth
  l.depth(v): length of the prefix stored at node v
- $|T|+1$ leaves representing the suffixes
- we store the edge labels $T[i,j]$ by storing i,j

  ↳ $O(|T|)$ space [words] not bits]

Applications - Suffix Trees

- Find all occurences of P in T
  (Substring = Prefix of a Suffix) $O(|P|+k)$
- Count all occurences of P in T
  (~~add~~ store subtreesize in the interior nodes)
- longest repeated substring in $O(|T|)$
  (branching node with max. letter depth)
- multiple documents
  (concatenate Texts with $\$_1, \$_2, \$_3 \dots \$_k$)
- longest common substring $O(|T|)$
  - max letter depth with ≥ 2 distinct \$'s below

- (Lemma als 3.5)

Suffix arrays

- sort suffixes in T and store their indices (lexicograpic order)

- Example:

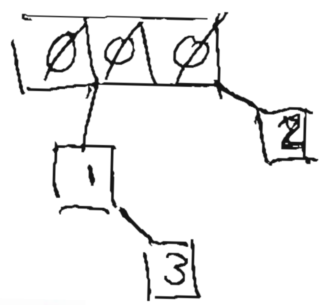
Suffix Array		LCP-Array
6	\$	∅
5	a\$	1
3	ana\$	3
1	anana\$	∅
∅	banana\$	∅
4	na\$	2
2	nana\$	

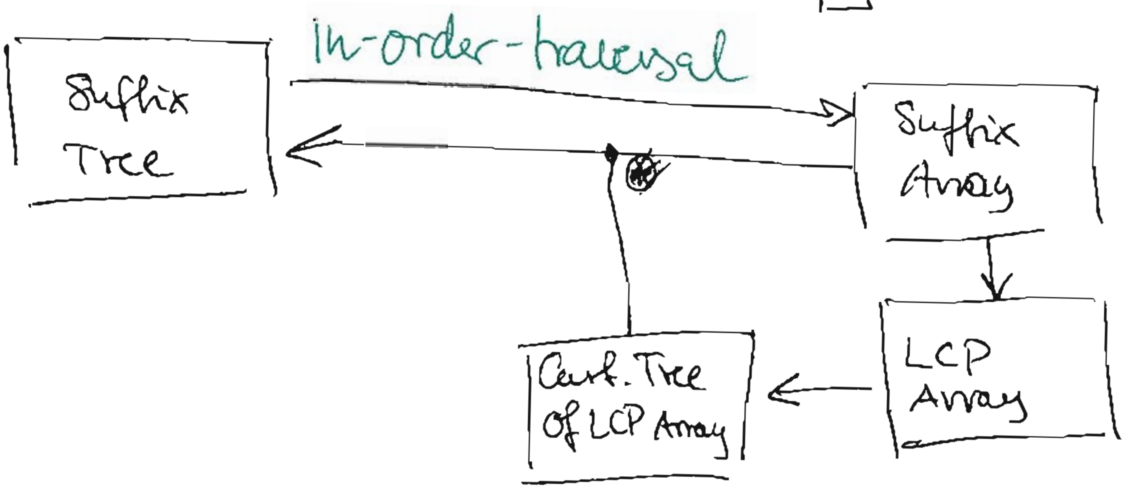
- Searchable in $O(|P| \log |T|)$ via binary search (~~speed up: homework~~)
- LCP Array: we store in cell i the length of the i-th and (i+1)-th entry of the suffix array
  Common prefix
- Suffix + LCP Array: searchable in $O(|P| + \log |T|)$
  with RMQ-DS
  (Homework)
- ~~LCP Array stores letter depth~~
- LCP Array can be constructed in $O(T)$ time [Kasai et al.] or with the construction we discuss later for the suffix-Array

Suffix Tree ⟷ Suffix array

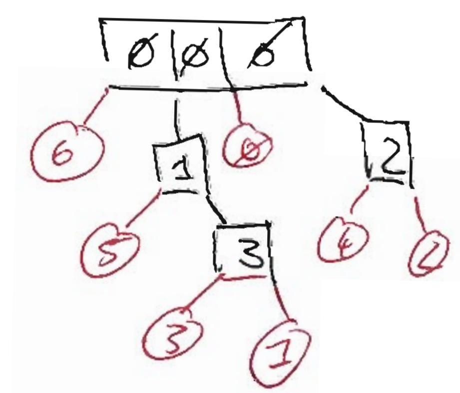
- ST & SA can be transformed into each other in Linear time

- Cartesian-Tree (of LCP Array)
  - Put all mins at the root
  - recurse in remaining array pieces
  - example:
    (banana)





(*) fill in suffixes (example)



- LCP's give letter depth
  = length of edge labels

Building Suffix Array in ~~$O(|T|)$~~ $O(|T| + \text{sort}\,|\Sigma|)$ — DC3 Alg.

[Kärkkäinen Sanders Burkhardt -03] inspired by
[Farach 97, Farach-Colton, Ferragina, Muthukrishnan 2000]

1. Sort Σ in $|\Sigma| \log |\Sigma|$ time
   (remark later sorting uses Radix Sort $(O(|T|))$)

2. Replace every letter in T by its rank in Σ

3. Transform ~~Split~~ T into 3 parts

$$T_0 = \langle (T[3i], T[3i+1], T[3i+2]),\ i = 0,1,\dots \rangle$$

$$T_1 = \langle (T[3i+1], T[3i+2], T[3i+3]),\ i = 0,1,\dots \rangle$$

$$T_2 = \langle (T[3i+2], T[3i+3], T[3i+4]),\ i = 0,1,\dots \rangle$$

4. Recurse on $\underbrace{T_0 \circ T_1}_{\tilde{T}}$ (concatenated) (Difference cover)
   ↳ we get the suffixes in $\tilde{T}$ sorted

5. Radix-Sort the suffixes in T_2, use

$$T_2[i:] = T[3i+2:] = T[3i+2] \circ T[3i+3]$$

$$= T[3i+2] \circ T_0[i+1]$$

⑥ Merge the sorted arrays for $\tilde{T}$ & T_2, use

Case A $T_0[i:] \geq T_2[j:] \iff T_3[i:] \geq T[3j+2]$

$$\iff T[3i] \circ \underbrace{T[3i+1:]}_{T_1[i:]} \geq T[3j+2] \circ \underbrace{T[3j+3}_{T_0[j+:}$$

Case B $T_1[i:] \geq T_2[j:] \iff T[3i+1] \geq T[3j+2]$

$$\iff T[3i+1] \circ T[3i+2] \circ \underbrace{T[3i+3:]}_{T_0[i+1:]} \geq T[3j+2] \circ T[3j+3] \circ T[3j$$

$T_2[j+1:$

$$\Rightarrow T(n) = T\left(\tfrac{2}{3} \cdot n\right) + O(n) \quad (n = |T|)$$

Toy example:

$T =$ bananas \$ \$
0 1 2 3 4 5 6 7 8

$\cancel{\$}\ \cancel{\$}\ a\ \cancel{b}\ \cancel{n}\ \cancel{s}$

$T_0 = \underbrace{[ban]}_{C}\ \underbrace{[ana]}_{A}\ \underbrace{[s\$\$]}_{F}$

$T_1 = \underbrace{[ana]}_{A}\ \underbrace{[nas]}_{D}$

$T_2 = \underbrace{[nan]}_{E}\ \underbrace{[as\$]}_{B}$

$\emptyset < A < B < C < D < E < F$

$\tilde{T} =$ C A F A D
0 1 2 3 4 5

Recurse
Suffix Array $SA(\tilde{T})$

5	\$
1	AD \$
3	AFAD \$
0	CAFAD \$
4	D \$
2	FAD \$

Rank T_0

i	pos in SA
0	4.
1	3.
2	6.

Rank T_1

i	pos in SA
0	2.
1	5.
2	1.

To sort T_2, write T_2 in $\Sigma \times$ Suffixes $(\tilde{T})$

$E B = [nan]\,[as\$] = (n, [ana]\,[s\$\$])$
$= (n, AF)$

$B = [as\$] = (a, [s\$\$]) = (a, F)$

⇒ Radix Sort

SA for T_2 (ignore \$)

SA(T_2)	
B	= (n, AF)
EB	= (a, F)

<u>Merging</u> (ignore \$ entrys)

1. $SA(\tilde{T})[1] \leftrightarrow SA(T_2)[1]$

$T_1[0] \leftrightarrow T_2[1]$ (Case B)

$(a, n, \underbrace{[ana][s\$\$]}_{\substack{AF \\ (\text{Rank } 3)}}) \leftrightarrow$ ~~(n, a, [s\$\$]~~

$(a, s, \$)$

(Rank \$)

≤

2. $SA(\tilde{T})[2] \leftrightarrow SA(T_2)[1]$

$T_0[1] \leftrightarrow T_2[1]$ (Case A)

$(a, \underbrace{[nas]}_{D = \text{Rank } 5}) \leftrightarrow (a, \underbrace{[s\$\$]}_{F = \text{Rank } 6})$

≤

3. $SA(\tilde{T})[3] \leftrightarrow SA(T_2)[1]$

$T_0[0] \leftrightarrow T_2[1]$ case (A)

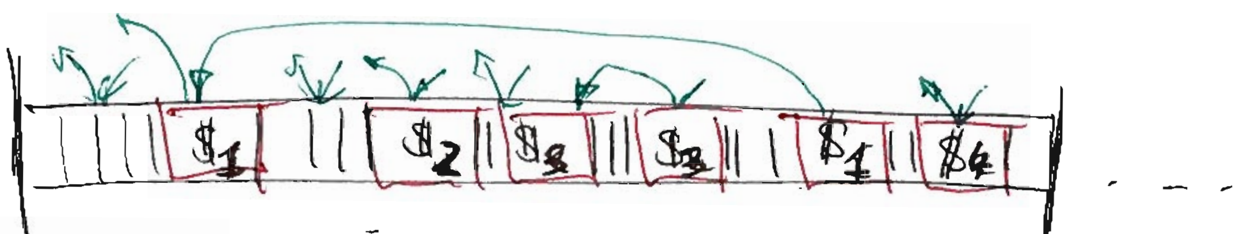
$(b, [ana][nas]) \leftrightarrow (a, [s\$\$])$

\>

and so on

Document Retrieval [Muthukrishnan 2002]

- extract k distinct documents in $O(|P| + k)$ time
- Patterns refer to ~~subtrees~~ interval $[i, j]$ in the suffix ~~tree~~ array
  = (subtrees in suffix-tree)



interval in suffix array

Pointers: Position of the ~~next~~ previous \$ with the same index

↳ Stored in Array PA

- We want to find the first occurrences of $\$_x$ only (for all x)
- We use a Range Minimum DS for the pointers
-

- find position of minimum stored at index k in the pointer array

  → if $PA[k] < i$: output k,
  recurse on $[i, k-1]$, $[k+1, j]$
  (with same i to test