Lecture 7

**STRING DS**

**Notation:**
- $\Sigma$: finite alphabet
- $T \in \Sigma^*$: text
- $P \in \Sigma^*$: search pattern
- $T[i:j]$ : sub-string of $T$ from character $i$ to $j$
- $T[0:]$ : suffix starting at $0$

**STRING MATCHING**

Input: text $T$, pattern $P$

Q: Find all occurrences of $P$ in $T$

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

* Best approach takes O(n) time

(Knuth, Morris, Pratt 77, Boyer Moore 77, Rabin, Kar)
How to "store" strings?

**TREE**:
- a tree with fan-out ≤ 2
- edges parent → child are labeled with the letters of Σ

![Tree Diagram]

- common technique: terminate strings with 
  - to distinguish if a path represents a prefix or word
- Root → Leaf path corresponds to strings, stored in the tree

- example: 3 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
  \[ \emptyset 1 2 3 4 5 6 \]

- Length of prefix stored at node \( v \):
  \( L \) depth = length of the prefix

- \( |T| + 1 \) leaves representing the suffixes
  - We store the edge labels \( T[i..j] \) by storing \( i, j \)

  \( O(|T|) \) space [words not links]
Applications - Suffix Trees

- Find all occurrences of $P \preceq T$
  (substring = prefix of a suffix) $O(P + K)$
- Count all occurrences of $P \preceq T$
  (add subtree size in the interior nodes)
- Longest repeated substring $O(N)$
  (branching node with max. letter depth)
- Multiple documents
  (concatenate texts with $S_1, S_2, S_3, \ldots, S_k$)
  - Longest common substring $O(1 + L)$
- Multiple documents with 22 distinct $S_i$
  below
Suffix arrays

- Sort suffixes in \( T \) and store their indices (lexicographic order).

**Example:**

<table>
<thead>
<tr>
<th>Suffix Array</th>
<th>LCP Array</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

- 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.

- Suffix + LCP Array: searchable in \( O(|P| + \log |T|) \) with \( \text{Rmq-Ds} \) (homework).

- LCP Array stores better depth.

- LCP Array can be constructed in \( O(|T|) \) time [Kasai et al.]

- For block wise construction we discuss later for the suffix array.
Suffix Tree $\rightarrow$ 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)

- In-order traversal

- Fill in Suffixes (example)

- LCP's give letter depth
  - length of edge labels
Building Suffix Array in $O(|T|)$ (DC3 Alg.)

Burkhardt

[Kärkkäinen Sanders -03] inspired by

[Farach 97, Farach-Colton, Ferragina, Muthukrishnan 2000]

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

2. Replace every letter in $T$ by its rank in $\Sigma$

3. Transform $T$ into 3 parts

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

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

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

4. Recurse on $T_0$ and $T_1$ (concatenated) (Don't care)

4. We get the suffixes in $T$ sorted

5. Radix-Sort the suffixes in $T_2$, use

   $T_2[i:] = T[3i+2:] = \langle T[3i+2], T[3i+3] \rangle$

   $= T[3i+2] \oplus T_6[3i+2]$
6) Merge the sorted arrays for $T_1$ and $T_2$, case 

Case 4: $T_0[i:] \leq T_2[j:] \Rightarrow T_3[i:] \leq T[3j+2]$

Case 5: $T_1[i:] \geq T_2[j:] \Rightarrow T[3k+1] \geq T[3j+2]$

$\Rightarrow T(c) = T(\frac{2}{3} \cdot u) + O(u) \ (u=|T|)$
Toy Example:

\[ T = \text{bananas} \quad 012345678 \]

\[ T_0 = [\text{ban}] [\text{ana}] [S\;S\;S] \]
\[ T_1 = [\text{ana}] [\text{nes}] \]
\[ T_2 = [\text{nau}] [\text{as}\;S] \]
\[ \%<A<B<C<D<E<\% \]

\[ \overset{\sim}{T} = \text{CAFAD} \quad 012345 \]

Recurse

Suffix Array \( SA(\overset{\sim}{T}) \)

To sort \( T_2 \), write \( T_2 \) in \( \Sigma \times \text{Suffixes}(\overset{\sim}{T}) \)

\[ EB = [\text{nau}] [aS\;S] = (n, [\text{ana}]) [S\;S\;S] = (n, \text{AF}) \]

\[ B = [aS\;S] = (a, [S\;S\;S]) = (a, F) \]
2) Radix sort

SA for T₂ (ignore 8)

\[
\begin{array}{c|c}
B & (u, AF) \\
EB & (α, F) \\
\end{array}
\]

Merging (ignore 8 entries)

\(\bar{\text{SA}}(i)[1] \leftarrow \text{SA}(T₂)[1]\)

\(T₁[0] \leftarrow T₂[1]\) (Case 8)

\[(a, u, \overline{\text{[ana][S$8$S]}}) \leftarrow (u, \overline{\text{[S$8$S]}})\]

AF (Rank 3)

\(\Rightarrow\)

\(\bar{\text{SA}}(i)[2] \leftarrow \text{SA}(T₂)[1]\)

\(T₀[1] \leftarrow T₂[1]\) (Case 4)

\[(a, \overline{\text{[nas]}}) \leftarrow (a, \overline{\text{[S$8$S]}})\]

\(\text{Diff Rank 5} \leq F = \text{Rank} 6\)

3.

\(\bar{\text{SA}}(i)[3] \leftarrow \text{SA}(T₂)[1]\)

\(T₀[0] \leftarrow T₂[1]\)

\[(b, \overline{\text{[ana][nes]}}) \leftarrow (\overline{\text{[S$8$S]}})\]

and so on
Document Retrieval

- Extract k distinct documents in $O(|P| + k)$ time
- Pattems refer to $s_{i...i+k}$ in the Suffix array
  - (suffixes in suffix tree)

---

**Points**: Position of the next $|$ with the same index
- Stored in array $PA$
- We want to find the first occurrences of $s_x$ only (for all $x$)
- We use a Range Minimum DS for the points
• find position of minimum stored at index $k$ in the pointer array

  $\rightarrow$ if $PA[i] < i$ : output $i$,

  recurse on $[i, k-1]$, $[k+1, i]$ (with same $i$ to keep)