Topic: Strings

Terminology:
- Text $T$, $|T| = n$
- Pattern $P$, $|P| = m$
- Alphabet $\Sigma$
  - e.g. binary, small (good), large (poly)

Pattern matching

Find occurrence of $P$ in $T$ (as a substring)

There exist solutions in $O(n+m)$:
- Knuth, Morris, Pratt alg.
- Boyer-Moore
  
  etc...

Idea: preprocess pattern
Data structure perspective:

preprocess $T$ in $O(n)$

query $P$ in $O(m)$

Problem 1

Preprocess $T_1, \ldots, T_k$ $\quad n = \sum_i |T_i|$

Query $P$ $\quad$ is $P=Z_i,$

or find pred./succ. of $P$ in $\{T_i\}$

w.r.t. lex. order.

Trie: Rooted tree, where strings correspond to root-leaf paths.

If the tree is in-order $\leadsto$ sorted $T.$

$T = \{ a, cx, cy \}$
<table>
<thead>
<tr>
<th>nodes store children</th>
<th>query</th>
<th>space</th>
</tr>
</thead>
<tbody>
<tr>
<td>array + blank cells store^1</td>
<td>(O(m))</td>
<td>(O(n</td>
</tr>
<tr>
<td>balanced search tree (BST) (set/map)</td>
<td>(O(m \log</td>
<td>\Sigma</td>
</tr>
<tr>
<td>hash table^2</td>
<td>(O(m))</td>
<td>(O(n))</td>
</tr>
<tr>
<td>van Emde-Boas tree</td>
<td>(O(m \log \log</td>
<td>\Sigma</td>
</tr>
<tr>
<td>hash + vEB^3</td>
<td>(O(m + \log</td>
<td>\Sigma</td>
</tr>
<tr>
<td>weight balanced BST</td>
<td>(O(m + \log k))^4</td>
<td>(O(n))</td>
</tr>
<tr>
<td>indirect</td>
<td>(O(m + \log</td>
<td>\Sigma</td>
</tr>
</tbody>
</table>

^1 no PRED/SUCCE
^2 works w.h.p. /vEB only used once
^3 cf. next page
Claim: Going down twice:
- advances \( \mathcal{P} \)
- reduces \# candidates to \( \frac{2}{3} \)

\[ \longrightarrow \text{string sorting } \Omega(n + k \log |\Sigma|) \]
**Compressed trie**

Assume long input (e.g., war and piece) and short queries.

**Suffix tree**: compressed trie of all $T[i]$.

- Banana: $n + 1$ leaves
- $O(n)$ space
- $O(m + \log |\Sigma| + k)$ all occurrences of $P$

$\hat{\#}$ of occurrences
hashing: $O(m + k)$ all occ. of $P$

E.g. Longest Match of $T[i:]$, $[j:] = \text{LCAC}(i)$

→ How do we construct a suffix tree?

In $O(n)$ ... complicated ...

→ Consider instead suffix array.

Claim Construction of SA equiv. to ST.
**ST → SA**: in order traversal
to outer tour gives LCP

**SA → ST**: Cartesian tree $O(n)$
on LCP (mostly)

**SA construction in $O(n + \text{sort}(\Sigma))$**:  
1) Sort $\Sigma$,  
2) replace $\Sigma$ with $[1, \ldots, |\Sigma|]$  
3) $T_0 = [ (T[3i], T[3i+1], T[3i+2])$ for $i = 0, \ldots ]$  
   $T_1 = [ \begin{array}{ccc} \text{3i+1} & \text{3i+2} & \text{3i+3} \end{array} ]$  
   $T_2 = [ \begin{array}{ccc} \text{3i+2} & \text{3i+3} & \text{3i+4} \end{array} ]$  
4) recurse on $T_0 \circ T_1 = \text{relative order of } T[3i], T[3i+1]$
5) radix sort of $T_2$
\[ T_2[i:] \approx T[3i+2:] \approx (T[3i+2], T[3i+3]) \]
\[ \approx (T[3i+2], T_0[i+1:]) \]

6) merge $T_0 \cdot T_3$ with $T_2$

\[ T_0[i:] \text{ vs } T_2[j:] \]
\[ \approx (T[3i:], T_3[i:]) \text{ vs } (T[3j+2], T_0[j+1]) \]

similarly $T_2 \text{ vs } T_1$

\[ \text{e.g. } \text{banana} \]
\[ T_0 \text{ ban ana } \]
\[ T_3 \text{ ana na} \]
\[ T_2 \text{ nan a} \]