Lecture 9 - Trays, ~~Mis~~matches and wild cards, level ancestor

Last Time: Text T of length n, Pattern P of length m, Both over alphabet Σ

Suffix Trees (ST)

$O(n|\Sigma|)$ space
$O(m)$ query

or

$O(n)$ space
$O(m \log \Sigma)$ query

Suffix Arrays (SA)

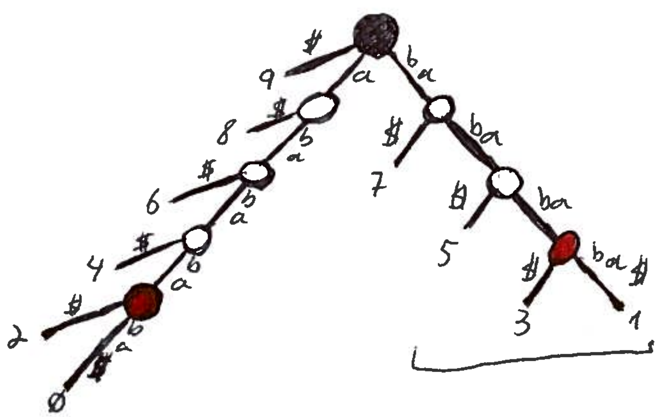
$O(n)$ space
$O(m + \log n)$ query

Suffix Tray [Cole, Kopelowitz, Lewenstein] ICALP 2006

$O(n)$ space
$O(m + \log|\Sigma|)$ query

$T = a\,b\,a\,b\,a\,b\,a\,b\,a\,\#$
$\quad\ \ 0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9$

Idea: tree decomposition; we already saw heavy-path, preferred paths decompositions. we will also see separator decomposition (L18) and ladder decomposition today.



SA: 9 8 6 4 2 0 7 5 3 1 = leaves of ST

- Internal nodes of ST correspond to an interval of SA
- Time to search on interval I is $O(m + \log|I|)$.

Definition: Σ-node is: (1) has at least $|\Sigma|$ leaves in its subtree.
(2) all its children do not.

\# of leaves in subtree of a Σ-node $\leq |\Sigma|^2$ so time to search in SA interval of a Σ-node is $O(m + \log|\Sigma|)$

- This takes care of Σ-nodes

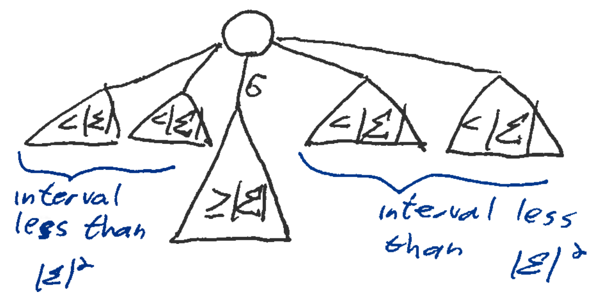
Remaining internal nodes partitioned into:

(1) Branching-Σ-node: has at least two children with subtree $\geq |\Sigma|$

(2) Others

Branching-Σ-nodes is $O(n/|\Sigma|)$ so we can store an array of size $|\Sigma|$ in everyone.

The remaining nodes:



Navigation:
1. Σ-node: jump to SA.
2. Branching-Σ-node: look at Σ-array.
3. Others: look at single character, walk to this child or jump to SA.

Search time: $O(m) + O(m + \log|\Sigma|) = O(m + \log|\Sigma|)$

$O(m)$: to walk Branching-Σ-nodes and single characters

$O(m + \log|\Sigma|)$: search SA interval

Space: $O(n)$

Approximate String Matching

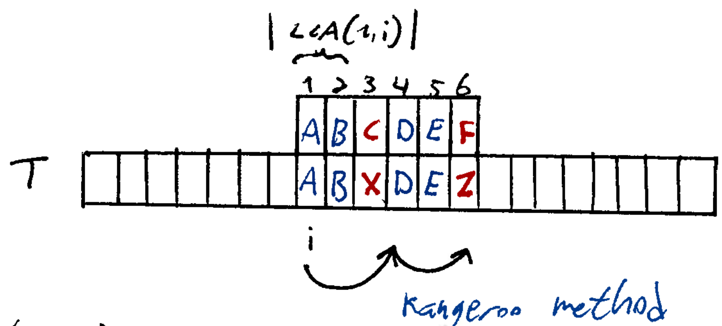
Given a text T, find occurrences of a query pattern p in T with error k

- Hamming Distance: # of character mismatches - (Today)
- Edit distance: # of edits (insertions, deletions, mismatches) needed for an exact match - (Pset)

"Online"

P & T are given together.

- construct suffix tree + LCA on P$T => space $O(|T|)$
- For every location in T
  - use k LCA queries to check if p occurs in this location.



Time: $O(|T| \cdot k)$

"Offline"

T is given in advance, p is given online.

- Best known Bounds: [Cole, Gottlieb, Lewenstein] STOC 2004

Space and preprocessing: $O(|T|(c\lg|T|)^k / k!)$

query time: $O(|p| + (c\lg|T|)^k / k! \lg\lg|T| + \underbrace{3^k}_{\text{only for edit distance}} \cdot \#\text{occurences})$

We focus on subproblem of <u>searching with wildcards</u>

- P contains at most k "don't care" characters (? wildcards)
- Find "exact matches" (? matches anything)

Solution: $O(|T| \lg^k |T|)$ space and preprocessing

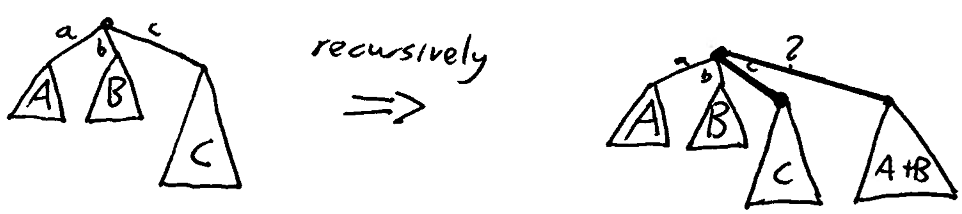
$O(2^k \lg\lg |T| + |P| + \#\text{occurences})$ for query

<u>Simple solution</u> $|\Sigma|^k \cdot |P|$ query

- walk down suffix tree of T as usual
- upon ? in P branch $|\Sigma|$ ways

<u>Better solution</u> $2^k |P|$ query

- Heavy-Light decomposition on <u>suffix tree of T</u>
  primary tree



Search branches 2 ways upon ? in P:
1) heavy edge
2) ? edge

- Each node in primary tree stores secondary tree on the union of light subtrees hanging off node. excluding first character

- recurs k times → k+1 levels of trees
- light depth $\leq \lg |T|$

⇒ each leaf appears in $\leq \lg^k |T|$ trees

⇒ $O(|T| \lg^k |T|)$ space & preprocessing

$2^k \lg\lg |T| + |P|$ solution uses <u>"level ancestor"</u> data structure.
as well as suffix tree + LCA

Determine in $\lg\lg |T|$ time on which of the two edges we branch

Level Ancestor Problem:

- Preprocess static tree
- level ancestor query: find l^{th} ancestor of node v.

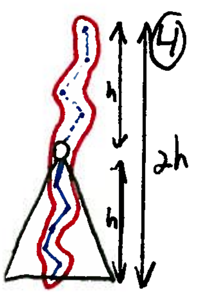
[Berkman & Vishkin - JCSS 94, Dietz - WADS 91, Alstrup & Holm - ICALP 00
Bender & Farach-Colton - TCS 04]
↖ dynamic tree

Solutions:

① lookup table: on all possible queries
- $O(n^2)$ space
- $O(1)$ query

② Jump pointers: (skip list) each node stores $1^{st}, 2^{nd}, 4^{th}, 8^{th}$ ..... ancestor
- $O(n \lg n)$ space
- $O(\lg n)$ query

③ long path decomposition: decompose tree by longest paths. store each path as an array + parent pointer
not the same as heavy-path decomp.
- $O(n)$ space
- query might still visit $\Theta(\sqrt{n})$ paths...
- Notice: node of height h is on path of length $\geq h$. So:



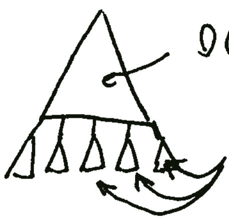
④ ladder decomposition: extend path of length l up by l levels.
- still $O(n)$ space
- $height(v) = h \Rightarrow height(\text{top node in } v\text{'s ladder}) \geq 2h$
- query: as above, but using ladders
  - each step at least doubles height $\Rightarrow O(\lg n)$ query

⑤ Combine jump pointers & ladder decomp.
- query: $O(1)$. 1 jump pointer $\Rightarrow height(v') > l/2$
  1 ladder step $\Rightarrow$ ladder(v') extends $> l/2$ above
  $\Rightarrow$ contains l^{th} ancestor
- Space - $O(n \lg n)$

⑥ jump-pointer tuning: jump-pointers only at leaves

- start queries at leaves
- depth-d ancestor of v = depth-d ancestor of leaf(v)
- $O(\underset{\text{ladder decomp}}{n} + \underbrace{L \lg n}_{\text{jump pointers}})$ space, where L = # leaves

⑦ ART decomposition:



$O(n/\lg n)$ leaves $\Rightarrow$ using ⑥ we get $O(n)$ space

less than $\frac{1}{4}\lg n$ nodes

$\Rightarrow$ # distinct trees on $\frac{1}{4}\lg n$ nodes

$= \underbrace{C_{\frac{1}{4}\lg n}}_{\text{Catalan \#}} \le 4^{\frac{1}{4}\lg n} = \sqrt{n}$

- lookup table for every possible bottom tree $\Rightarrow O(\sqrt{n} \cdot \lg^2 n)$ space
- query: look in bottom tree, else get parent(root), query top
  $\Rightarrow O(1)$ query

* In our "don't cares" data structure we need ancestors in a compressed trie $\Rightarrow$ weights on the edges
  - total weight along any path $\le n$