Tries: History
- de la Briandais (1959)
- Fredkin- “trie from “retrieval”
- Pronounced like “try”

Digital Search:
- Keys are strings over some alphabet $\Sigma$
- $\Sigma = \{a, c, g, t\}$
- $\Sigma' = \{0, 1\}$
- Let $k = |\Sigma'|$
- Assume keys coded as $\Sigma$
- $E_i$ = Saab, aba, abe, can, cab, cbc

Analysis:
- Space: Smaller by factor $k$
- Search Time: Larger by factor of $k$

Example:
- $\Sigma = \{a = 0, b = 1, c = 2\}$
- Keys: $\{aab, aba, abc, caa, cab, cbc\}$

Analysis:
- Search: $\sim$ length of query string [O(1) time per node]
- Space:
  - No. of nodes $\sim$ total no. of chars in all strings
  - Space $\sim k \cdot$ (no. of nodes)

How to save space?
- de la Briandais trees:
- Store 1 char per node
- First child/next-sibling

Node: Multiway of order $k$
- Assume: Strings prefix tree
  (no string is prefix of another)

Example:
- $\sum = \{a = 0, b = 1, c = 2\}$
- Keys: $\{aab, aba, abc, caa, cab, cbc\}$

Space:
- No. of nodes $\sim$ total no. of chars in all strings
Patricia Tries:
- Improves trie by compressing degenerate paths
- PATRICIA = Practical Alg. to Retrieve Info. Coded in Alpha...
- Late 1960’s: Morrison & Gurevich
- Each node has index field, indicates which char to check next (Increase with depth)

Dealing with long Paths:
- To get both good spaces & query time efficiency, need to avoid long degenerate paths.
- Path compression!

Example: $S_0: \text{ajam...aj}$ $S_1: \text{pajam...paj}$ $S_2: \text{apaja...ap}$ $S_3: \text{mapaja...map}$ $S_4: \text{mapaj...map}$ $S_5: \text{amapaj...amap}$ $S_6: \text{jama...j}$ $S_7: \text{pamapa...pam}$

Tries and Digital Search Trees II

Analysis:
- Query time: (Same as std trie) $\times$ search string length (may be less)
- Space:
  - No. nodes $\approx$ No. of strings (irresp. of length)
  - Total space: $K \cdot$ (No. of nodes) (Storage for strings)
- Estimate, substring queries: "How many occurrences of "tree" in CMSC 420 notes"
  - Notation: $S = a_0 a_1 ... a_{n-1}$
  - Suffix: $S_i = a_i a_{i+1} ... a_{n-1}$
  - Def: Substring identifier for $S_i$ is shortest prefix of $S_i$ unique to this string
    - Eg. ID($S_i$) = "ama" $\Rightarrow$ ID($S_i$) = "ama$
    - Q: What is minimum substring needed to identify suffix $S_i$?
Example: $S = p u m a p a j a m a$

Suffix Trees (cont.)

$S$ - text string, $|S| = n$

$S_i$ - $i$th suffix

Substring ID = min substr. needed to identify $S_i$

A suffix tree is a Patricia trie of the $n+1$ substring identifiers

Substring Queries

How many occurrences of $t$ in text?

- Search for target string $t$ in trie
  - if we end in internal node
    - return no. of extern. nodes in this subtree
  - else (fall out at extern. node)
    - compare target with string
      - if matches - found 1 occurrence
      - else - no occurrences

How many substrings of length $l$?

- Search for $S_t$ in trie
  - return no. of substrings of length $l$

How many total substrings?

- Total of all substrings of length $l$

Tries and Digital Search Trees III

Analysis:

- Space: $O(n)$ nodes
- Search time: $O(n.k)$ total space
  
  $(k = |S| = O(1))$

- Construction time: $O(n.k)$ [nontrivial]

Final tree

Example:

Search("ama") → End at intern node → $O$ (ama)

Search("amapaj") → End at extern node

Report: 0 occ.
Announcements - 5/2
- HW 3 - Grades published
- HW 4 - Almost done (later today)
  - New policy - Can drop lowest HW score
- Final Exam
  Mon, May 15 4-6 pm

Room ?? IRB 0324

Programming Assignment 3:
- When to adjust contenders

kd-tree

cell

$r_j > R_{min}$
$c_j$ not contender