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

Node: Multiway of order $k$

Example: $\sum = \{ a=0, b=1, c=2 \}$

Keys: \{aab, aba, abc, caa, cab, cbc\}

Digital Search:
- Keys are strings over some alphabet $\Sigma$
- E.g., $\Sigma = \{a, b, c, \ldots\}$
- $\sum = \{0, 1\}$ Let $k=|\Sigma|$.
- Assume chars coded as ints: $a=0$, $b=1$, $c=k-1$

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

Example:
\{aab, aba, abc, caa, cab, cbc\}

Tries and Digital Search Trees I

Search: ~ length of query string [O(1) time per node]

Space:
- No. of nodes ~ total no. of chars in all strings
- Space ~ $k \cdot$ (no. of nodes)

Same structure/Alt. Drawing

How to save space?
de la Briandais trees:
- Store 1 char. per node

This diagram illustrates the structure of tries and digital search trees, showing how keys are stored and how searches are conducted. The nodes are multiway of order $k$, and the keys are strings over some alphabet $\Sigma$. The analysis shows that the space is smaller by a factor of $k$ compared to traditional search trees, but the search time increases by the same factor. The example provided demonstrates the structure and functionality of these data structures.
Patricia Tries:
- Improves trie by compressing degenerate paths
- PATRICIA = Practical Alg. to Retrieve Info. Coded in Alpha...
- Late 1960's: Morrison & Guchinberger
- Each node has index field, indicates which char to check next (Increase with depth)

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

Example:
- ID
  - $S_0: ajam...$
  - $S_1: a$
  - $S_2: apaj...$
  - $S_3: mapaj...$
  - $S_4: amapaj...$
  - $S_5: amapa...$

Tries and Digital Search Trees II

Example: $S = pamapajama$
- Def: Substring identifier for $S_i$ is shortest prefix of $S_i$ unique to this string
- $S_i: amap$. Eq. ID($S_i$) = "amap" ID($S_i$) = "ama$

Suffix Trees:
- Given single large text $S$
- Substring queries: "How many occurrences of "tree" in CMSC 420 notes"
- Notation $S = a_0a_1a_2...a_{n-1}$
- Suffix $S_i = a_{i}a_{i+1}a_{i+2}...a_{n-1}$

- Q: What is minimum substring needed to identify suffix $S_i$?
Example: \( S = \text{pamapajama} \)

Suffix Trees (cont.)
- \( S \) - text string \(|S| = n\)
- \( S_i = i^{th} \) suffix
- Substring ID = min substring needed to identify \( S_i \)
- A suffix tree is a Patricia trie of the \( n+1 \) substring identifiers

Substrings Queries:
- How many occurrences of \( t \) in text?
  - Search for target string \( t \) in trie
  - if we end in \( \) internal node \( \) (or midway on edge) - 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

Example:
- Search("ama") → End at intern node
- Report: 2 occ.
- Search("amapaj") → End at extern node
- Go to \( S_i \), verify

Tries and Digital Search Trees III

Geometric Applications:
- PR k-d tree: Can be used for answering same queries as point kd-tree (orth. range, near. neigh)
- PR k-d Tree: kd-tree based on midpoint subdivision
  - Assume points lie in unit square
  - Example: Geometric Applications:

Analysis:
- Space: \( O(n) \) nodes
  - \( O(n \cdot k) \) total space \( (k = |S| = O(1)) \)
- Search time: \( n \) total length of target string
- Construction time:
  - \( O(n \cdot k) \) [nontrivial]

Claim: This is a trie!
1. PA2 due Thu, 11:59pm

2. PA3 almost ready
   - Combine:
     - Heap $\rightarrow$ QuakeHeap
     - kd-Tree $\rightarrow$ HB-kdTree
     - Hash Map $\rightarrow$ Set
     - add nearestNeighbor
     - Euclidean MST

3. You can drop lowest PA.
   - score: PA $0 + 1a + 1b = 100$ pts
   - Get EC: PA $2 = 100$ pts
   - Exam: PA $3 = 100$ pts

4. Midterm grades