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

**Digital Search:**
- Keys are strings over some alphabet \( \Sigma \)
  - Eq. \( \Sigma' = \{a, b, c, \ldots\} \)
  - \( \Sigma' = \{0, 1\} \) Let \( k = |\Sigma'| \)
- Assume chars coded as ints: \( a = 0, b = 1, \ldots z = k - 1 \)

**Example:**
\[ \Sigma' = \{a = 0, b = 1, c = 2\} \]
Keys: \{aab, aba, abc, caa, cab, cbc\}

**Tries and Digital Search Trees I**

**Node:** Multiway of order \( k \)

**Analysis:**
- Space: Smaller by factor \( k \)
- Search Time: Larger by factor of \( k \)

**Example:**
![Example Diagram]

**How to save space?**
- Store 1 char. per node

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

**Same structure/Alt. Drawing**

**Search:**
- First-child/next-sibling

**Next sibling:**
- Advance to next character of search string

**Next character:**
- \( \rightarrow \neq x \rightarrow \text{try next char in } \Sigma' \)
- \( = x \rightarrow \text{advance to next character of search string} \)
Patricia Tries:
- Improves trie by compressing degenerate paths
- \text{PATRICIA} = \text{Practical Alg. to Retrieve Info. Coded in Alpha...}
- Late 1960's: Morrison & Guchenerberger
- Each node has index field, indicates which char to check next (Increase with depth)

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

Example: ID

\text{Example: } S = \text{pamapajaman}$

Def: Substring identifier for $S_i$: is shortest prefix of $S_i$ unique to this string

$S_i$ = amap$ E.g. \text{ID}(S_i) =$ "amap"

ID

Suffix Trees:
- Given single large text $S$
- Substring queries: "How many occurrences of "tree" in CMSC 420 notes"

Notation: $S^t = a_0a_1a_2...a_{n-1}$

- Suffix: $S_i = a_ia_{i+1}...a_{n-1}$
- Special terminal
- Q: What is minimum substring needed to identify suffix $S_i$?
Example: $S = \text{pama pajama}$

Suffix Trees (cont.)

Example: $S = \text{pama pajama}$

PR k-d tree: Can be used for answering same queries as point k-d tree (orth: range, near neigh)

Geometric Applications:

PR kd-Tree: k-d tree based on midpoint subdivision

Assume points lie in unit square

Example:

Example: $S = \text{pama pajama}$

E.g. ID($S_i$) = amap ID($S_i$) = ama$

Substring Queries:

How many occurrences of $t$ in $S$?

- 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 (full of on extern node)
    - compare target with string
      - if matches - found 1 occurrence
      - else - no occurrences

Tries and Digital Search Trees III

Analysis:

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

Claim: This is a trie!

Some cells may be empty
Binary Encoding:
- Assume our points are scaled to lie in unit square $0 \leq x, y < 1$ (can always be done)
- Represent each coordinate as binary fraction:
  $x = 0.a_1 a_2 a_3 \ldots, a_i \in \{0, 1\}$
  $x = \sum a_i \cdot 2^{-i}$

Example:

How do we extend to 2-D?

PR kd-Tree $\equiv$ Trie ??

- Approach: Show how to map any point in $\mathbb{R}^2$ to bit string
- Store bit strings in a trie (alphabet $\Sigma = \{0, 1\}$)
- Prove that this trie has same structure as kd-tree

Further Remarks:
- Techniques for efficiently encoding, building, serializing, compressing...
- Can generalize immediately to PR kd-tree

Tries and Digital Search Trees IV

Bit Interleaving:

Given a point $p = (x, y)$
$0 \leq x, y < 1$

Let:
- $x = 0.a_1 a_2 \ldots$ in binary
- $y = 0.b_1 b_2 \ldots$

Define:

$$\phi(x, y) = a_1 b_1 a_2 b_2 \ldots$$

Called Morton Code of $p$

Lemma: Given a pt set $P \subseteq \mathbb{R}^2$ (in unit square $[0, 1]^2$) let $P = \{p_1, \ldots, p_n\}$ where $p_i = (x_i, y_i)$
Let $\phi(P) = \{\phi(p_1), \phi(p_2), \ldots, \phi(p_n)\}$ (n binary strings)
Then the PR kd-tree for $P$ is equivalent to binary trie for $\phi(P)$.

Proof: By induction on no. of bits
Let $x = 0.a_1 a_2 \ldots$ and consider just $\phi(x, y) = a_1 b_1 a_2 b_2 \ldots$

The PR kd-tree + binary trie assigns pts to same subtrees ($\ldots$ induction)