Indexed Search Tree (Trie)

Fawzi Emad
Chau-Wen Tseng
Department of Computer Science
University of Maryland, College Park

Indexed Search Tree (Trie)

- Special case of tree
- Applicable when
  - Key C can be decomposed into a sequence of subkeys $C_1, C_2, \ldots, C_n$
  - Redundancy exists between subkeys
- Approach
  - Store subkey at each node
  - Path through trie yields full key
- Example
  - Huffman tree
Tries

- Useful for searching strings
  - String decomposes into sequence of letters
  - Example
    - “ART” ⇒ “A” “R” “T”

- Can be very fast
  - Less overhead than hashing

- May reduce memory
  - Exploiting redundancy

- May require more memory
  - Explicitly storing substrings

Types of Tries

- Standard
  - Single character per node

- Compressed
  - Eliminating chains of nodes

- Compact
  - Stores indices into original string(s)

- Suffix
  - Stores all suffixes of string
Standard Tries

- **Approach**
  - Each node (except root) is labeled with a character
  - Children of node are ordered (alphabetically)
  - Paths from root to leaves yield all input strings

```
<table>
<thead>
<tr>
<th>A</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>N</td>
</tr>
<tr>
<td>S</td>
<td>D</td>
</tr>
<tr>
<td>I</td>
<td>M</td>
</tr>
</tbody>
</table>
```

Trie for Morse Code

Standard Trie Example

- **For strings**
  - \{ a, an, and, any, at \}
Standard Trie Example

For strings

{ bear, bell, bid, bull, buy, sell, stock, stop }

Standard Tries

Node structure

- Value between 1…m
- Reference to m children
  - Array or linked list

Example

```java
class Node {
    letter value; // Letter V = { V_1, V_2, ..., V_m }
    node child[m];
}
```
Standard Tries

Efficiency

- Uses $O(n)$ space
- Supports search / insert / delete in $O(d \times m)$ time
- For
  - $n$ total size of strings indexed by trie
  - $d$ length of the parameter string
  - $m$ size of the alphabet

Word Matching Trie

- Insert words into trie
- Each leaf stores occurrences of word in the text
Compressed Trie

- **Observation**
  - Internal node $v$ of $T$ is redundant if $v$ has one child and is not the root

- **Approach**
  - A chain of redundant nodes can be compressed
    - Replace chain with single node
    - Include concatenation of labels from chain

- **Result**
  - Internal nodes have at least 2 children
  - Some nodes have multiple characters

**Example**

![Compressed Trie diagram](image)
Compact Tries

- Compact representation of a compressed trie

Approach
- For an array of strings $S = S[0], \ldots, S[s-1]$
- Store ranges of indices at each node
  - Instead of substring
- Represent as a triplet of integers $(i, j, k)$
  - Such that $X = s[i][j..k]$
- Example: $S[0] = \text{“abcd”}$, $(0,1,2) = \text{“bc”}$

Properties
- Uses $O(s)$ space, where $s = \# \text{ of strings in the array}$
- Serves as an auxiliary index structure

Compact Representation

Example

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0, 1, 1</td>
<td>1, 1, 1</td>
<td>0, 0, 0</td>
<td>7, 0, 3</td>
<td>4, 1, 1</td>
<td>4, 2, 3</td>
<td>5, 2, 2</td>
<td>0, 2, 2</td>
<td>2, 3</td>
<td>3, 3, 4</td>
</tr>
</tbody>
</table>
Suffix Trie

- Compressed trie of all suffixes of text
- Example: “IPDPS”
  - Suffixes
    - IPDPS
    - PDPS
    - DPS
    - PS
    - S
- Useful for finding pattern in any part of text
  - Occurrence ⇒ prefix of some suffix
  - Example: find PDP in IPDPS

Suffix Trie

- Properties
  - For
    - String X with length n
    - Alphabet of size m
    - Pattern P with length d
  - Uses $O(n)$ space
  - Can be constructed in $O(n)$ time
  - Find pattern P in X in $O(d \times m)$ time
    - Proportional to length of pattern, not text
Tries and Web Search Engines

- Search engine index
  - Collection of all searchable words
  - Stored in compressed trie
- Each leaf of trie
  - Associated with a word
  - List of pages (URLs) containing that word
    - Called occurrence list
- Trie is kept in memory (fast)
- Occurrence lists kept in external memory
  - Ranked by relevance
Computational Biology

- **DNA**
  - Sequence of 4 different nucleotides (ATCG)
  - Portions of DNA sequence produce proteins (genes)

- **Genome**
  - Master DNA sequence for organism
  - For Human
    - 46 chromosomes
    - 3 billion nucleotides
Tries and Computational Biology

- **ESTs**
  - Fragments of expressed DNA
  - Indicator for genes (& location)
  - 5.5 million sequences at NIH

- **ESTmapper**
  - Build suffix trie of genome
    - 8 hours, 60 Gbytes
  - Search for ESTs in suffix trie
    - 11 hours w/ 8 processor Sun

- **Search genome w/ BLAST**
  - 5+ years (predicted)