Digital Trees

Digital Trees

Digital Trees branch based on the letters of the used alphabet

Can become unbalanced

Variants

- Tries (from information re**trie**val)
- Patricia-Trees
- Prefix-Trees

Has its origin in information retrieval

- become popular for database implementation in case of XML-DBMS
  - elements are the alphabet

Trie

Store strings in a tree with an edge for each of its characters.

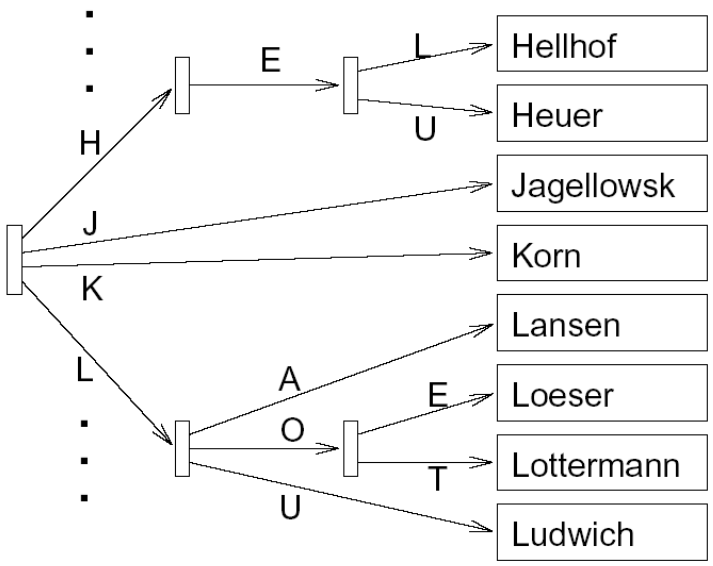
Strings with same prefix share edges in higher tree levels.

The path from the root to a node n describes the word stored in n.

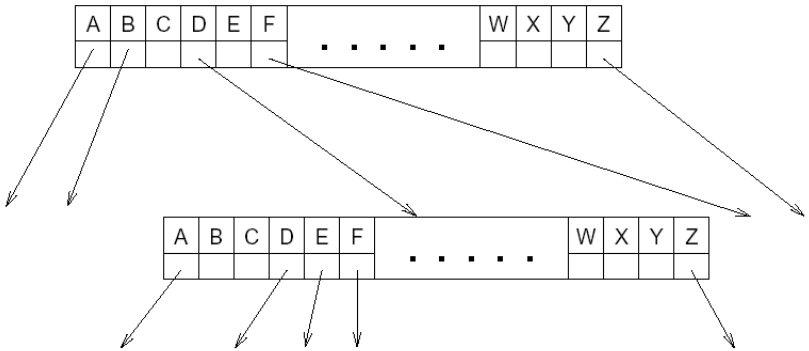
Each node is linked to the results for its corresponding word.

Note, that we have to do up to l steps, where l is the length of the search key.

Example Trie



Nodes of a Trie



Patricia Tree

Problems of a Trie

- Long common sub-words
- Letters that do not occur
- Very unbalanced

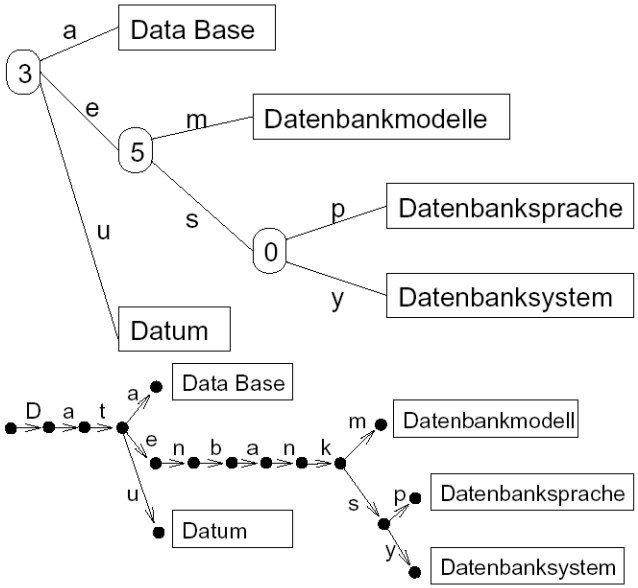
Solution to the first problem: Patricia Trees

- Practical Algorithm To Retrieve Information Coded In Alphanumeric
- Principle: Sub-words that are not needed can be skipped
  - we store the number of letters to be skipped
- We can only determine whether a key exists, when we reach a leaf

Advantages

- Patricia trees are very space and CPU efficient.
- With their inherent “compression”, patricia trees grow slowly, even when inserting large strings

Patricia Tree vs. Trie



Prefix Tree

Similar to Patricia Tree, but we also store the skipped sub-words in the nodes