Pattern matching

Metodický koncept k efektivní podpoře klíčových odborných kompetencí s využitím cizího jazyka ATCZ62 - CLIL jako výuková strategie na vysoké škole

Interreg
EVROPSKÁ UNIE
Rakousko-Česká republika
Evropský fond pro regionální rozvoj

Europäische Union
Evropská unie
Europäischer Fonds für regionale Entwicklung
Evropský fond pro regionální rozvoj

VŠTE

UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

Pattern matching

- Strings
  - A string is a sequence of characters
  - An alphabet is the set of possible characters for a family of strings
    - ASCII
    - Unicode
    - {0, 1}, {A, C, G, T}
  - Let P be a string of size m
  - A substring P[i .. j] of P is the subsequence of P consisting of the characters with ranks between i and j
  - A prefix of P is a substring of the type P[0 .. i]
  - A suffix of P is a substring of the type P[i ..m - 1]

Interreg Rakousko-Česká republika
Evropský fond pro regionální rozvoj
EVROPSKÁ UNIE
Europäische Union
Evropská unie
Europäischer Fonds für regionale Entwicklung
Evropský fond pro regionální rozvoj
VŠTE
UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

Pattern matching

- Given strings $\boldsymbol{T}$ (text) and $\boldsymbol{P}$ (pattern), the pattern matching problem consists of finding a substring of $\boldsymbol{T}$ equal to $\boldsymbol{P}$
- Applications:
  - Text editors
  - Search engines
  - Biological research

Interreg Rakousko-Česká republika
Evropský fond pro regionální rozvoj
Europäische Union
Evropská unie
Europäischer Fonds für regionale Entwicklung
Evropský fond pro regionální rozvoj

Brute-Force algorithm

- Goes through text from left to right
- compares the pattern P with the text T for each possible shift of P relative to T, until either
  - a match is found, or
  - all placements of the pattern have been tried
- Time complexity: *O(nm)*

Algorithm *BruteForceMatch*(*T, P*)
  Input text T of size n and pattern P of size m
  Output starting index of a substring of T equal to P or -1 if no such substring exists
  for $i \leftarrow 0$ **to** $n - m$
    { test shift i of the pattern }
    $j \leftarrow 0$
    while $j < m \wedge T[i + j] = P[j]$
      $j \leftarrow j + 1$
    if $j = m$
      return i { match at i }
    else
      return -1 { no match }

Boyer-Moorův algoritmus

- Searching from right to left
- The Boyer-Moore’s pattern matching algorithm is based on two heuristics
  - Looking-glass heuristic: Compare P with a subsequence of T moving backwards
  - Character-jump heuristic: When a mismatch occurs at T[i] = c
    - If P contains c, shift P to align the last occurrence of c in P with T[i]
    - Else, shift P to align P[0] with T[i + 1]

Interreg Rakousko-Česká republika
Evropský fond pro regionální rozvoj
Europäische Union
Evropská unie
Europäischer Fonds für regionale Entwicklung
Evropský fond pro regionální rozvoj

Boyer-Moorův algoritmus

- For text is faster than brute-force
- Time complexity: *O(mn + A),* where A is size of alphabet
- Fast for large alphabets
- Boyer-Moore’s algorithm preprocesses the pattern P and the alphabet S to build the last-occurrence function L mapping S to integers, where L(c) is defined as
  - the largest index i such that P[i] = c or
  - -1 if no such index exists

Knuth-Morris-Pratt (KMP) algoritmus

- Prohledává text zleva doprava
- Nedělá všechna porovnání
- Pokud narazíme na neshodu, posune se o více než o jedno písmeno.
  - O maximální prefix *P(0 .. j-1)*, který je suffixem *P(1 .. j-1)*
  - Suffix a prefix se nesmí překrývat.
- Najdeme-li kus *P* (od začátku, tedy prefix), znaky tohoto prefixu odpovídají textu, není třeba je kontrolovat znovu
- Konec nalezeného podřetězce může být také obsažen v začátku tohoto podřetězce. Takovou shodou je samozřejmě celá nalezená část P, proto hledáme od P+1. Takže jdeme od konce nalezeného kusu P zleva a zprava, a ve chvíli, kdy nenalezneme shodu, víme, o kolik se můžeme posunout.
- Toto lze předpočítat do tabulky – pak je vše O(1)

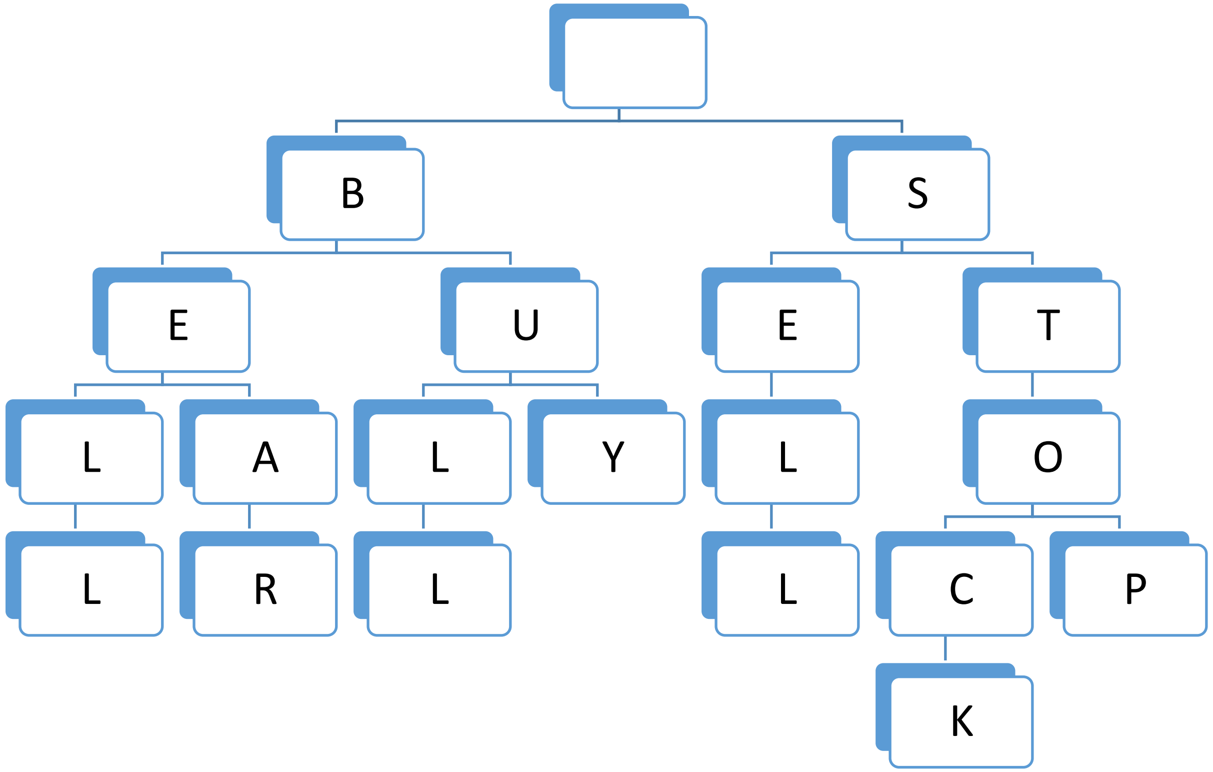
Trie

- The standard trie for a set of strings S is an ordered tree such that:
  - Each node but the root is labeled with a character
  - The children of a node are alphabetically ordered
  - The paths from the external nodes to the root yield the strings of S
- A standard trie uses O(n) space and supports searches, insertions and deletions in time O(dm), where:
  - n total size of the strings in S
  - m size of the string parameter of the operation
  - d size of the alphabet
- In Trie we can store whole text. Each node contains one word.

Interreg Rakousko-Česká republika
Evropský fond pro regionální rozvoj
EVROPSKÁ UNIE
Europäische Union
Evropská unie
Europäischer Fonds für regionale Entwicklung
Evropský fond pro regionální rozvoj

Trie

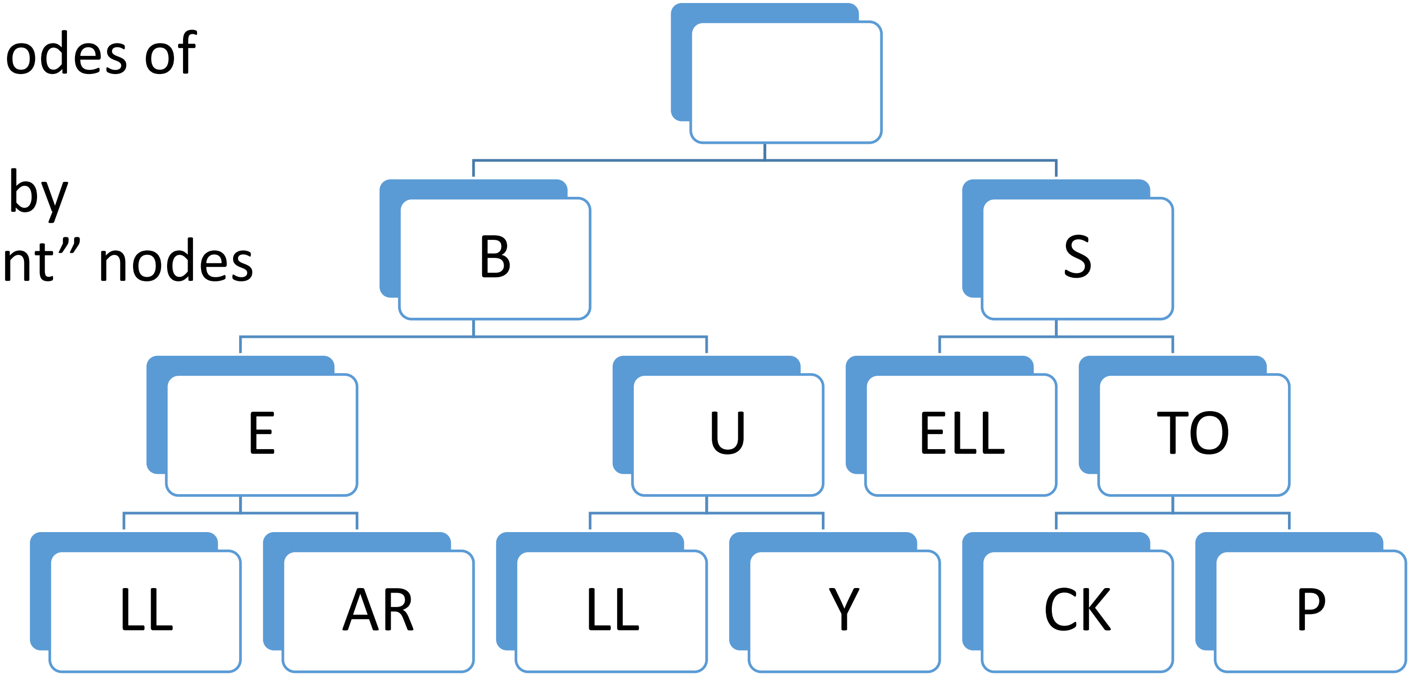
S={BELL, BEAR, BULL, BUY, SELL, STOCK, STOP}



Compressed trie

S={BELL, BEAR, BULL, BUY, SELL, STOCK, STOP}

A compressed trie has internal nodes of degree at least two
It is obtained from standard trie by compressing chains of “redundant” nodes



Interreg
EVROPSKÁ UNIE
Rakousko-Česká republika
Evropský fond pro regionální rozvoj
Europäische Union
Evropská unie
Europäischer Fonds für regionale Entwicklung
Evropský fond pro regionální rozvoj
VŠTE
UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA