Vyhľadávanie v texte

Broňa Brejová

- Kontakt: brejova@fmph.uniba.sk, M163
- Webstránka: http://compbio.fmph.uniba.sk/vyuka/vvt/
- Zapíšte sa na predmet v Moodli heslom “hladaj slovo”

  kým sa prvý raz neprihlásite, nemôžem vám evidovať body

Full-text keyword search

Plnotextové vyhľadávanie kľúčových slov

Problem statement

Document: Sequence of words

Goal: Create an index for a static set of documents to answer the following queries efficiently.

Query: Given a word w, find all documents containing w.

Example:

Document 0: `Ema ma mamu.`

Document 1: `Mama ma Emu.`

Document 2: `Mama sa ma. Ema sa ma.`

Query `Mama` returns documents 1,2.

Preprocessing: divide into words, convert to lowercase, remove duplicates within document,. . .

Document 0: ema, ma, mamu

Document 1: ma, mama, emu

Document 2: ma, mama, sa, ema

Inverted index: for each word a list of occurrences (document IDs)

ema: 0,2

emu: 1

ma: 0,1,2

mama: 1,2

mamu: 0

sa: 2

Queries with multiple keywords

Find intersection of two sorted arrays of occurrences of lengths $\mathfrak{m}$ and $\mathfrak{n}$

- Linear-time merge $O(\mathfrak{m} + \mathfrak{n})$
- $\mathfrak{m}$-times binary search $O(\mathfrak{m} \log \mathfrak{n})$
- Doubling search $O(\mathfrak{m} \log \frac{\mathfrak{n}}{\mathfrak{m}})$

More than two arrays: add one by one, or use a different algorithm

Also possibly preprocess sets for faster answers [Cohen, Porat 2010]

Full-text keyword search

Plnotextové vyhľadávanie kľúčových slov

Example:

Document 0: `Ema ma mamu.`

Document 1: `Mama ma Emu.`

Document 2: `Mama sa ma. Ema sa ma.`

Query `Mama` returns documents 1,2.

Inverted index: for each word a list of occurrences (document IDs)

ema: 0,2

emu: 1

ma: 0,1,2

mama: 1,2

mamu: 0

sa: 2

Comparison of inverted index representations

Optional homework: fill in table

	Query	Preprocessing
Sorted array		
Binary search tree (balanced)		
Hashing - expected/average case		
Hashing - worst case		
Trie		

m – max. length of a word

n – the number of distinct words

N – total number of words

σ – alphabet size

p – the number of documents found

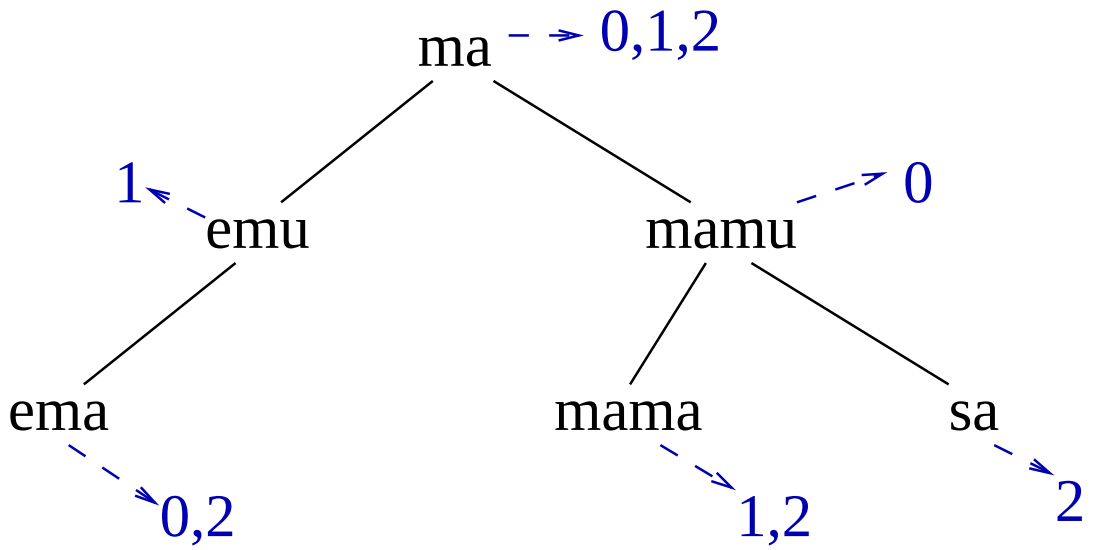
Implementing inverted index with balanced search trees

Balanced binary search tree, (e.g. red-black tree):

search, insert, delete using $O(\log n)$ comparisons

We use binary search tree, in each node list of occurrences

Running time: query $O(m \log n + p)$, preproccessing $O(mN \log n)$



Trie (lexikografický strom)

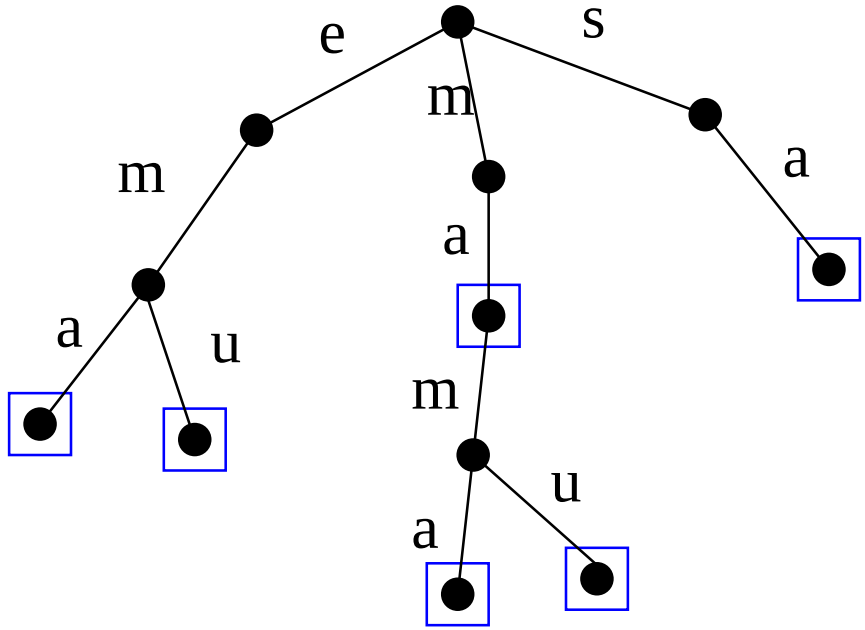
Represents a set of words

Edges labeled with characters

A node represents string read along the path from the root

(Root represents empty string)

In each node store flag if node in the set, plus other data



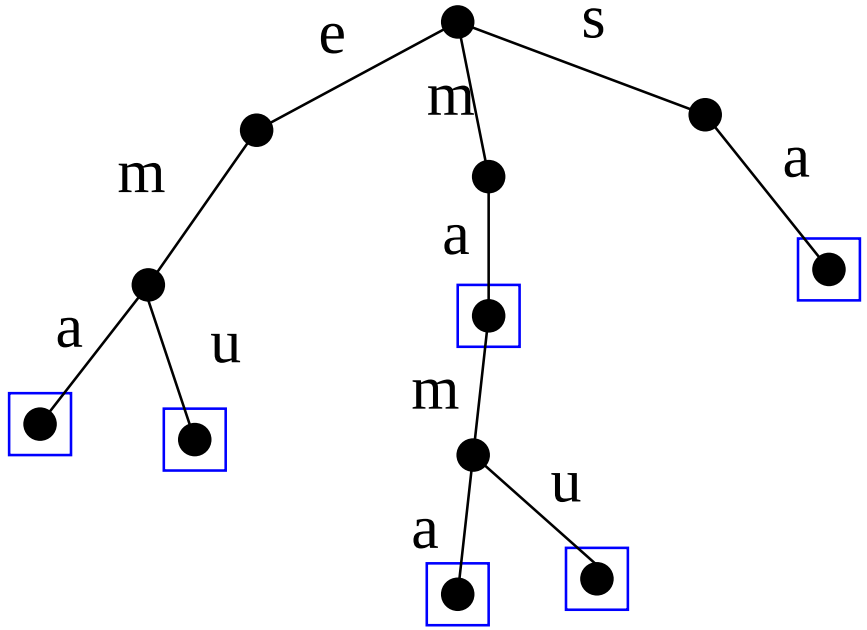
Trie (lexikografický strom)

Assume word of length m, small alphabet

Insert, search, delete in $O(m)$ time if alphabet is small

How to store each node if alphabet large?

Optional homework: details



Trie (lexikografický strom)

In each node: map from alphabet to pointers to children nodes

Implementation of this map for an alphabet of size σ:

	Search	Insert	Memory
Array of size σ	$O(m)$	$O(m\sigma)$	$O(D\sigma)$
Bin. search tree	$O(m \log \sigma)$	$O(m \log \sigma)$	$O(D)$
Linked list	$O(m\sigma)$	$O(m\sigma)$	$O(D)$

D – total length of all words

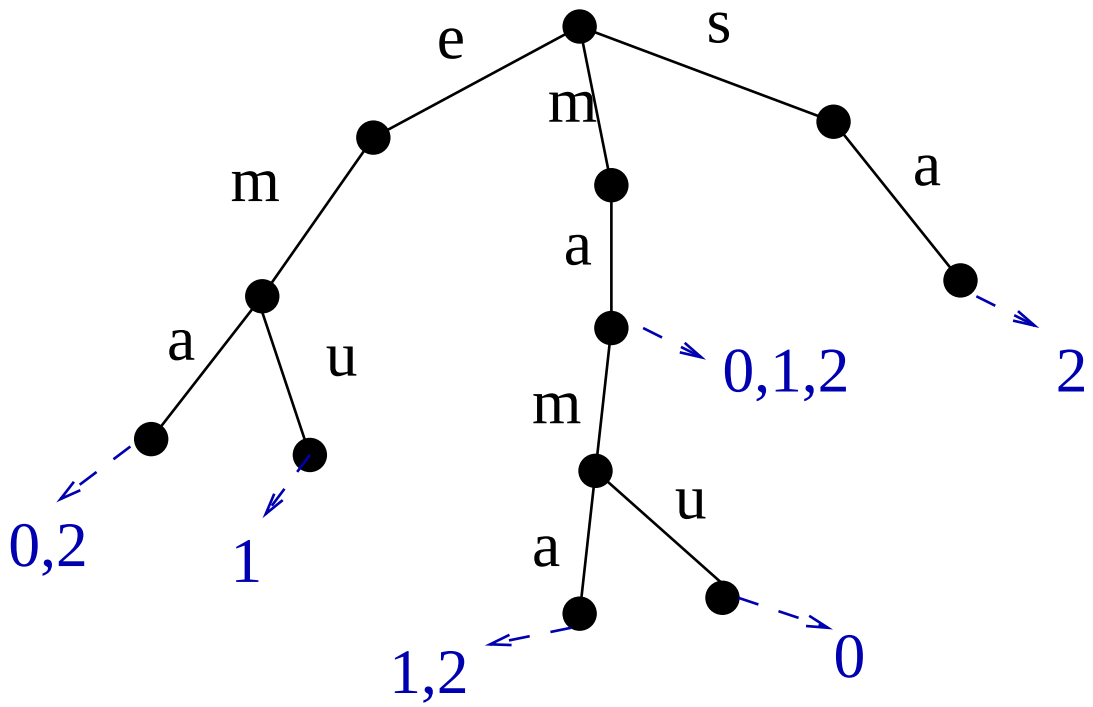
m – length of the word to be searched/inserted

σ – alphabet size

Tries used for implementing inverted index

List of occurrences in each node corresponding to a word

Running time: query $O(m \log \sigma + p)$, preprocessing $O(mN \log \sigma)$



Simple implementation of inverted index with arrays

ema: 0,2

emu: 1

ma: 0,1,2

mama: 1,2

mamu: 0

sa: 2

Concatenated occurrences in array A:

0, 2, 1, 0, 1, 2, 1, 2, 0, 2

Sorted words in array B:

ema, emu, ma, mama, mamu, sa, $

First index in occurrences array for a word in array C:

0, 2, 3, 6, 8, 9, 10

Simple implementation of inverted index with arrays

Concatenated occurrences in array A:

0, 2, 1, 0, 1, 2, 1, 2, 0, 2

Sorted words in array B:

ema, emu, ma, mama, mamu, sa, $

First index in occurrences array for a word in array C:

0, 2, 3, 6, 8, 9, 10

Example: find occurrences of “ma”:

binary search for “ma” in B, find at $B[2]$

return document IDs $A[C[2] \dots C[3]-1]$, i.e. $A[3..5] = 0, 1, 2$

Running time $O(m \log n + p)$ per query, $O(mN \log N)$ preprocessing

Simple to implement, little memory overhead

Implementations of inverted index

	Query	Preprocessing
Sorted array	$O(m \log n + p)$	$O(mN \log N)$
Binary search tree (balanced)	$O(m \log n + p)$	$O(mN \log n)$
Hashing - expected/average case	$O(m + p)$	$O(mN)$
Hashing - worst case	$O(mn + p)$	$O(mNn)$
Trie	$O(m \log \sigma + p)$	$O(mN \log \sigma)$

m – max. length of a word

n – the number of distinct words

N – total number of words

σ – alphabet size

p – the number of documents found

Notation

Finite alphabet Σ, its size σ

String (slovo) $S = s_0 s_1 \ldots s_{n-1}$, length $n = |S|$

Empty string ε

$S[i..j]$ is the substring $s_i s_{i+1} \ldots s_j$, kde $0 \leq i \leq j < n$

For $i > j$, $S[i..j] = \varepsilon$

$S[i]$ denotes ith character of S

Prefix: substring $S[0..j]$ for some j

Suffix: substring $S[i..n-1]$ for some i

Proper substring (vlastné podslovo) S is a substring of S, which is not equal to S

Similarly proper prefix and suffix

Concatenation of strings S and S' denoted SS'

String matching (vyhľadávanie vzorky v texte)

Given: pattern (vzorka) P of length m, text T of length n.

Goal: Find all positions $\{i_1, i_2, \ldots\}$ such that $T[i_j..i_j + m - 1] = P$.

Example:

Input: P =ma, T =Ema ma mamu

Output: 1, 4, 7

Input: P ="a ma", T =Ema ma mamu

Output: 2, 5