Vyhľadávanie v texte (String/Pattern Matching Algorithms)

Broňa Brejová

- Kontakt: brejova@fmph.uniba.sk, M163
- Prednášky streda 14:50-16:20 M-XI a štvrtok 16:30-18:00 M-III

  Vyhovuje tento čas?
- Webstránka: http://compbio.fmph.uniba.sk/vyuka/vvt/
- Moodle na odovzdávanie úloh a evidenciu bodov,

  prihláste sa heslom “hladaj slovo”
- Konzultácie: po dohode
- Neváhajte sa klásť otázky na hodine, na konzultáciách, príp. emailom
- Anglické prednášky?

Goals of the course

- Introduce efficient data structures and algorithms for working with text data.
- Used in web search, databases, text editors, compilers, computational biology, antivirus programs, . . .
- Use or improve your knowledge of efficient algorithm design and analysis, formal languages.
- Other skills: work with scientific literature, presentation, computational experiments.

Topics covered

- Keyword search (inverted index, trie)
- Pattern search (Knuth-Morris-Pratt algorithm and others), searching for multiple patters, 2D search, . . .
- Text indexing (suffix trees, suffix arrays), succint data structures
- Edit distance (dynamic programming, improvements), approximate pattern occurrences
- Other topics: regular expressions, applications in biology, FFT, LCA
- Your presentations

Pravidlá

Všetci:

Aktivita na hodine 10% (+ najviac 10% bonus)

Prezentácia článku 15%

Alternatíva A:

Domáca úloha 15%

Skúška 60%

A: >=90, B:80...89, C: 70...79, D: 60...69, E: 50...59, FX: <50

Prácu cez semester berte vážne.

Pravidlá

Všetci:

Aktivita na hodine 10% (+ najviac 10% bonus)

Prezentácia článku 15%

Alternatíva A:

Domáca úloha 15%

Skúška 60%

Alternatíva B:

Domáca úloha 15%

Projekt 60%

Alternatíva C:

Projekt 75%

Alternatíva A je základ

V prípade záujmu o B alebo C sa príďte dohodnúť do **31.3.**

A: >=90, B:80...89, C: 70...79, D: 60...69, E: 50...59, FX: <50

Prácu cez semester berte vážne.

Pravidlá

Domáca úloha

V prvej polovici semestra, implementácia algoritmov a ich experimentálne porovnanie, bonusové otázky

Prezentácie

Vyberte si vedecký článok súvisiaci s témou predmetu (do 15.4.)

Prezentácie cez prednášku na konci semestra

Skúška

Písomná skúška s riešením problémov (tvorba/analýza algoritmov)

Ústna skúška: vysvetlite algoritmus z prednášky, detaily vášho článku

Povolený ťahák 2 listy A4

Pravidlá

Projekt (nepovinný)

Mini-diplomovka (5-15 strán)

Rôzne možnosti: Prehľad/implementácia a experimenty/teória

Výber témy do 31.3., dohodnite sa s vyučujúcou

Aktivita

Odpovedanie otázky cez prednášku, prezentácia nepovinnej DÚ (1-5)

Písanie/zlepšovanie poznámok z prednášok v LaTeXu (1-5)

Hľadanie chýb v poznámkach/na prednáške

Neopisovať

Môžete sa o domácej úlohe rozprávať so spolužiakmi

Ale každý vlastnú implementáciu, experimenty a text

Nekopírovať z internetu, citujte prípadné zdroje

Literature

- Dan Gusfield (1997) Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology. Cambridge University Press. In library I-INF-G-8.
- Alberto Apostolico, Zvi Galil, editors (1997) Pattern Matching Algorithms, Oxford University Press.
- Gonzalo Navarro, Mathieu Raffinot (2002) Flexible Pattern Matching in Strings. Cambridge University Press.
- Research papers
- Class notes, slides

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` return documents 1,2.

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.

Practical issues

Document: webpage/email/book/chapter/abstract/...

Preprocessing: lower/upper case, stemming (úprava na základný tvar), what is a word/word separator?, synonyms, ...

If there are many documents, how to rank them? (Information/text retrieval)

Full-text keyword search

Example:

Document 0: `Ema ma mamu.`

Document 1: `Mama ma Emu.`

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

Query `Mama` return documents 1,2.

- How would you implement this using standard library data structures in C++/Java?
- How would you implement this using well-known data structures (arrays, lists, various trees, hash tables)?
- Running time?

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

Trie (lexikografický strom)

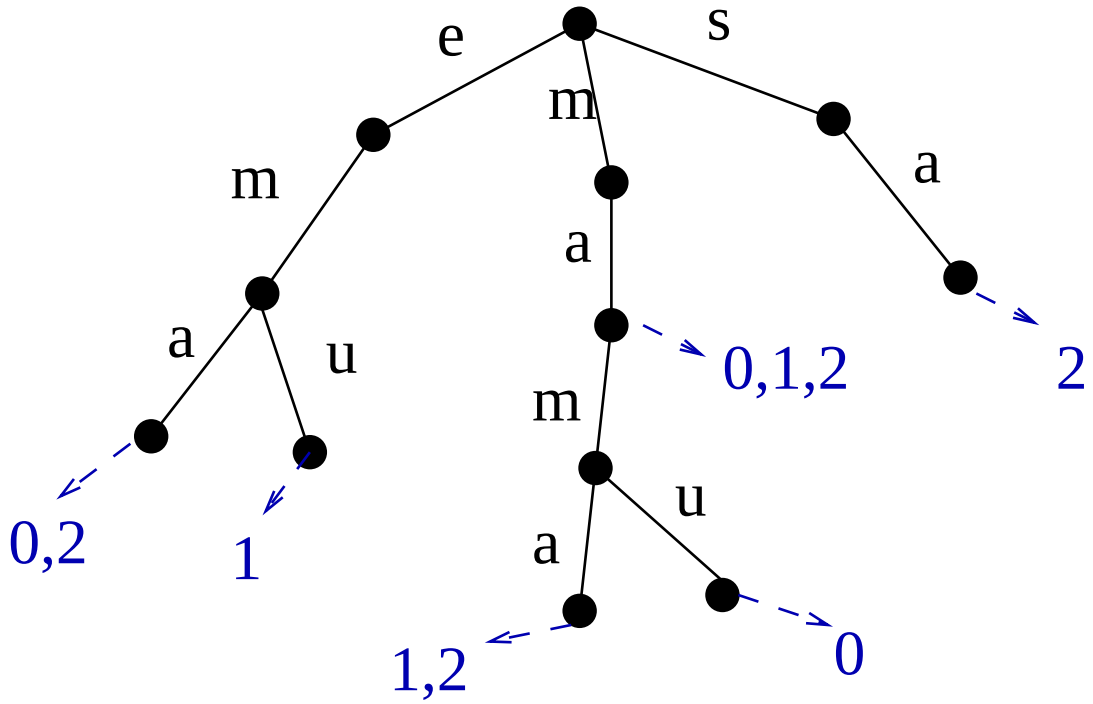
Represents a set of words

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

How to store each node if alphabet large?

Optional homework: details