Pattern Matching

Rationale

The advent of the worldwide web, next generation sequencing, and increased use of satellite imaging have all contributed to the current information explosion. One of the most basic tasks common to many applications is the discovery of patterns in the available data. To render the searching of big-data feasible, it is imperative that the underlying algorithms be efficient, both in terms of time and space. Pattern Matching is a branch of theoretical computer science whose ideas are used in practice daily in many different data-driven areas, including (but not limited to) word processors, web search engines, biological sequence alignments, intrusion detection systems, data compression, database retrieval, and music analysis. This course gives a student training in the process of developing and analyzing efficient algorithms through the study of pattern matching algorithms that are used for searching and indexing large textual data.

Description

Pattern Matching is one of the fundamental problems in Computer Science. In its classical form, the problem consists of 1-dimensional string matching. Given a string (or text) T and a shorter string (or pattern) P, find all occurrences of P in T. Over the last four decades, research in Pattern Matching has developed the field into a rich area of algorithmics. This course covers several variants of the pattern matching problem. Emphasis is placed on the algorithmic techniques used to speed up naive solutions, and on the time complexity analysis of the algorithms.

Topic List

Topics may include but are not limited to:

- Exact String Matching
    - Knuth-Morris-Pratt

  - Boyer-Moore
- Suffix Trees and Applications
  - Wiener
  - Ukkonen
  - Practical Implementation Issues
- Multiple Pattern Matching
  - Aho-Corasick
  - Generalized Suffix Tree
- Approximate Pattern Matching
  - Hamming Distance
  - Edit Distance (dynamic programming)
  - Don't Cares (convolutions)
- Lowest Common Ancestor
  - Range Minimum Query
  - Complete binary trees
- Periodicity
  - Squares
  - Repetitions
  - Approximate tandem repeats
- Palindromes
  - Manacher
  - Suffix trees
  - Palindromes with errors
- Naming
  - Karp-Miller-Rosenberg

    - Lyndon Word Naming

- 2-Dimensional Matching
    - Bird-Baker
    - Dueling for Alphabet Independent Matching (Amir-Benson-Farach)
    - Small Space 2d Matching
    - 2d Dictionary Matching

- Suffix Arrays
    - Definition and Construction
    - String Matching with suffix array
    - Burrows-Wheeler Transform

Learning Goals

The student must be able to demonstrate knowledge of how to apply and analyze the following algorithmic tools:

- Finite Automata
- Dynamic Programming
- Suffix Trees
- Naming
- Convolutions
- Dueling

Assessment

Two written exams, one midterm and one final will be used to assess students' knowledge of the subject. The questions on the exams will address problems related to those that were discussed in class, and the student will have to demonstrate the ability to apply techniques learned to new problems. For example, after learning dynamic programming for edit distance with unit cost, a student will have to answer a question related to weighted edit distance.