HO CHI MINH UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY
SOFTWARE ENGINEERING DEPARTMENT
ADVANCED PROGRAM IN COMPUTER SCIENCE
COURSE: **DATA STRUCTURES**
LECTURER: Dr. ĐINH BÁ TIẾN

Final project

MINI SEARCH ENGINE

- TRƯƠNG PHƯỚC LỘC
- HỒ TUẤN THANH

HCMC, 2015

Introduction

In this project, you will be designing and implementing a mini search engine. You are probably familiar with Google, Bing, which are some of the most popular search engines that you can find on the Web. The task performed by a search engine is, as the name says, to search through a collection of documents. Given a set of texts and a query, the search engine locates all documents that contain the keywords in the query. The problem may be therefore reduced to a search problem, which can be efficiently solved with the data structures we have studied in this class.

Your task

Your task is to design and implement an algorithm that searches a collection of documents. You will have a lot of documents and a set of sample queries. You have the freedom to select the data structures and algorithms that you consider to be more efficient for this task. Of course, you will have to justify your decisions.

First, you will process the documents and store their content (i.e. words / tokens) in the data structures that you selected (in information retrieval, this phase is called indexing). Next, for every input query, you will process the query and search its keywords in the documents; using the previously implemented data structures and an algorithm of your choice (the phase is called retrieval). For each such query, you will have to display the documents that satisfy the query.

English stopwords should be removed if they appears in the queries. Link: http://www.ranks.nl/stopwords

The search engine should support several operator such as:

1. AND
2. OR
3. Manchester –united
4. intitle:hammer nails
5. Peanut Butter +and Jam
6. filetype:txt
7. Search for a price. Put $ in front of a number. For example: camera $400.
8. Search hashtags. Put # in front of a word. For example: #throwbackthursday
9. Search for an exact match. Put a word or phrase inside quotes. For example, "tallest building".

10. Search for wildcards or unknown words. Put a * in your word or phrase where you want to leave a placeholder. For example, "largest * in the world"
11. Search within a range of numbers. Put .. between two numbers. For example, camera $50..$100.
12. Entering "~set" will bring back results that include words like "configure", "collection" and "change" which are all synonyms of "set

History: list of queries that a user entered before. When the user enters one of these queries again, the search engine should suggest it. See auto suggestion feature in Google for inspiration.

Hints

As data structures, you must use trees – such as 2-3 trees, AVL trees, balanced binary search trees, trie

What to turn in

There are two main parts in this project, all of them contributing to the final project grade.

1. You must create 50 data files, at least 500 words in a file.
2. You will have to write a project report (about 7-10 typed pages – single space – 12pt font) that includes:
   - Design issues, what are the problems and how you solve them
   - Data structures you use, the decision behind selecting them
   - Algorithms you employ, again with a justification of your decision
   - Particular emphasis should be placed on the running time of your algorithm
   - Optimization issues: what could you do to further optimize your algorithm
   - You need to specifically address the problem of scalability: would you implementation is efficient in the case of very large text collections?
   - The report will also include results you obtained for the given sample queries, and for at least five additional queries of your choice
   - Any other remarks about your design and implementation
3. You will have to send in a program. It has
   - Input
     - query.txt
       - n

        - query 1
        - query 2
        - …
    - d0001.txt
    - d0002.txt
    - d0003.txt
    - …
- Output
    - StudentID_result.txt
        - Preprocessing time (ms)
        - Result of query1, time (ms). Ex: 500 d0001 d0002
        - …

Grading

- Design issues, data structures efficiency, algorithms, other issues addressed in the written report.
  => 30 points
- Program (+ 3/12 types of queries)
  => 55 points
- Extra
  => Up to 35 points

Notes

- *No late submissions are accepted!*
- Up to 10 points extra credit may be given if you handle some of these issues:
  - Document scoring with respect to the query (i.e. keep track of the number of occurrences of words in documents, and calculate a score based on that). If you plan to implement this part, please see me for additional details.
  - Incomplete matches. That is, allow for a search for number*, which will match all words starting with number, i.e. number, numbers, numbering, etc.
  - Stop words elimination. Given a list of very frequent words, like the, a, of, etc., you may consider avoiding the storage of these elements.