**King Fahd University of Petroleum & Minerals**

**Information & Computer Science Department**

#### ICS-202 Data Structures

#### Lab Project

**Date**

**Duration**

|  |  |  |  |
| --- | --- | --- | --- |
| **Due Date (Tentative)** | **Duration** | **Semester** | **Submission** |
| Monday/Wednesday, 25/27 December 2017 | 6 weeks | 171 | Blackboard + Demo |

Write a program which implements a **basic search engine functionality** using specific data structures. The program should proceed as follows:

(1) A set of text documents will be made available as part of the project files.

(2) You have to build a postings file/inverted index data structure as follows:

1. Before scanning the words in every text file, make sure you do some pre-processing.

This involves removal of punctuation, treating every character as lowercase and removal of unneeded/extra characters.

1. Now assign a numerical index to every text file in the source collection. This may begin from 1.. onwards for every text file.

For our collection, it can be done as follows:

|  |  |
| --- | --- |
| Index | Filename |
| 1 | Source-document0002.txt |
| 2 | Source-document0004.txt |
| 3 | Source-document0006.txt |
| 4 | Source-document0007.txt |

You have to maintain a suitable data structure for this purpose.

1. For the indexing phase: Read every word in the each document and create a 'postings' list. A 'postings' list is an inverted index data structure. It should be a list of words (possibly sorted in alphabetical order) with the list of document indices and the relative frequency in a linked list. Do not include commonly occuring **stopwords** like: the, and, or, of, for, to, be.

An example of a inverted index data structure is shown here:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| data | → | (0.035, 1) | → | (0.15, 2) |  |
|  |  |  |  |  |  |
| text | → | (0.042, 1) | → | (0.269, 5) | → | (0.2, 9) |
|  |  |  |  |  |  |
| information | → | (0.35, 3) |  |  |  |

The explanation of each entry is as follows:

For the term ‘data’ we have the entry (0.035, 1). This means that the term `data’ occurs in document 1 with a relative frequency of 0.035. This relative frequency can be calculated by counting the number of occurrences of `data’ divided by the total number of words in the document.

For example, if you have the following list of documents:

Document 1: *Data and information are interrelated*.

(Number of times of `data’ = 1, Number of words = 5, therefore relative frequency = 1/5 = 0.2).

Document 2: *Data is the source of all information*.

Document 3: *My data was stolen*.

Then the entry of ‘data’ will look as follows:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| data | → | (0.2, 1) | → | (0.142, 2) | → | (0.25, 3) |

(3) Once done, your postings list is the data structure for the search operations.

(4) Your search engine should support the following search operations:

1. single term query: It should return a list of documents sorted by the relative frequency.
2. AND queries: this should search for the presence of two terms in common documents and return the list of common documents sorted by the product of relative frequencies.
3. OR queries: this should search for the presence of two terms in common documents and return the list of common documents ranked by the sum of frequencies.
4. NOT queries: return a list of documents that do not contain the suggested term.
5. The results should be in a format with title followed by some text.
6. BONUS: Any combination of the above queries.

SUBMISSION DETAILS: Will be provided later.