Analysis and Design of Algorithms

Search Engine Project Report

<u>Considerations:</u>

The Engine requires three files to exists on the same folder as the .exe file. They are the impressions file, keywords file, and web graph file. For the click through file, it's automatically generated if it's not found

<u>Data Structures:</u>

1- Graphs to store the nodes and edges of the web graph.
2- Maps to store the adjacency list (web_site string, vector of websites) for each website. The vector of websites here is the websites pointing towards web_site not the other way around. This helps in calculating the Page Rank by going through each list in the adjacency list. Thus, this helps in the ranking algorithm as well
3- Maps to store the other values like:
   a- Number of impressions
   b- Number of click through
   c- Values of CTR
   d- Scores for each page
   e- Number of out edges for each node
   f- Page Rank for each page

   This is done because websites are not numbered

<u>Ranking Algorithm:</u>

```
For number of iterations:
    //new_PR: vector initialized with zeros
    For node in adjacency list:
        Value = 0
        For node in adjacentOf[node]:
            Value = value + PageRank[node]/noOfoutEdges[node]
    PR = new_PR
```

Complexity = number of iterations * (number of nodes + number of edges)*complexity of accessing values

N -> number of iterations

V -> number of pages/nodes

E -> number of edges

$$T(n) \; = N * (|V| + |E|) * log|V|$$

For the space complexity, I have used the map data structure to store the Page Rank along with the noOfEdges map. Each one of those has a complexity of O(|V|)

Indexing Algorithms:

For indexing the pages when inserted into the graph, I use maps to index them based on their text, so there is no algorithm for this. The map data structure resolves the issue

For indexing the pages when printed out on the screen, I used merge sort to sort them from highest to lowest based on the score value. This function is implemented as a function in the graph. It outputs a sorted vector of those websites. The complexity for this is O(nlog(n)) which is the same as the merge sort complexity. For the space complexity of this algorithm, it's linear in the number of websites as I am storing the score in a map also.

Tradeoffs:

1- I used maps instead of indexing pages to numbers because this will make searching for pages in complexity of O(n). However, maps provide this with complexity of (log(n)) according to the documentation