Name : Mohamed Hesham Riad

ID: 900192833

Final Project

"Search Engine"

-The pseudo-code for indexing and ranking algorithms:

**Indexing:**

I did not use any indexing algorithm, instead I stored the sites in a set and whenever I need to loop over the set I used the range based iterators. Additionally, the attributes of each file is stored in a map and using the range based iterators and the set of sites, I could access whatever information I need.

**Ranking:**

void rank\_calc(){

map<string, double > tempo\_rank=rank;

double lowest = 99;

double highest = -1;

for (auto i : sites) {

double temp\_rank = 0;

for (auto j : links) {

if (j.dest == i)

temp\_rank += (tempo\_rank[j.src]) / numOfLinksOut[j.src];

}

rank[i] = temp\_rank;

if (rank[i] < lowest)

lowest = rank[i];

if (rank[i] > highest)

highest = rank[i];

}

double diff = highest - lowest;

for (auto q : sites)

rank[q] = (rank[q] - lowest) / diff;

}

In my initialization function, while reading data from CSV files, I kept track of the number of the links going out of every website. Then in the ranking algorithm, I calculated the page rank by looping over the set of site and inside this loop, I looped over the vector that contain the links (edges). And I calculated the rank according to the known ranking algorithm shown above. After calculating the rank, I calculated the normalized rank using normalization rule.

-A time and space complexity analysis for ranking algorithm:

Time: To calculate the rank, the algorithm uses two nested loops that have total complexity of O(mn) where m is the number of sites and n is the total number of edges in the web graph.

Space: Throughout the algorithm, the only container that is used to temporarily store the sites and its ranks is a map. We know that the space complexity of a map is O(n), where n is the number of sites used in the database.

-The main data structures used by my algorithm:

The most important data structure I used is maps. It was used to store and update all the attributes of a given file during runtime. Maps allowed me to search in log(n) time. Additionally, it allowed me to access different attributes of the sites using their names. Additionally, I used sets because all of its elements must be unique; this allowed be to eliminate any duplicate possibility. Also, its elements always follow a specific order, so I don’t need to use any sorting algorithms. I also used vectors to store the links (edges).

-Design tradeoffs:

I have chosen Time Complexity over Space Complexity. That’s why I have chosen to use maps instead of vector of vectors or 2D dynamic arrays. Moreover, the maps offered me search complexity of log n compared to n of the vectors. And the noticeable advantage that the maps add is the ability to use the [] operator to access attributes of the sites using their names.