THE AMERICAN UNIVERSITY IN CAIRO
الجامعة الأمريكية بالقاهرة

CSCE220303 – Analysis &Design of Algorithms Lab

Spring 21

Instructor:

Dr. Mohamed Alhalaby

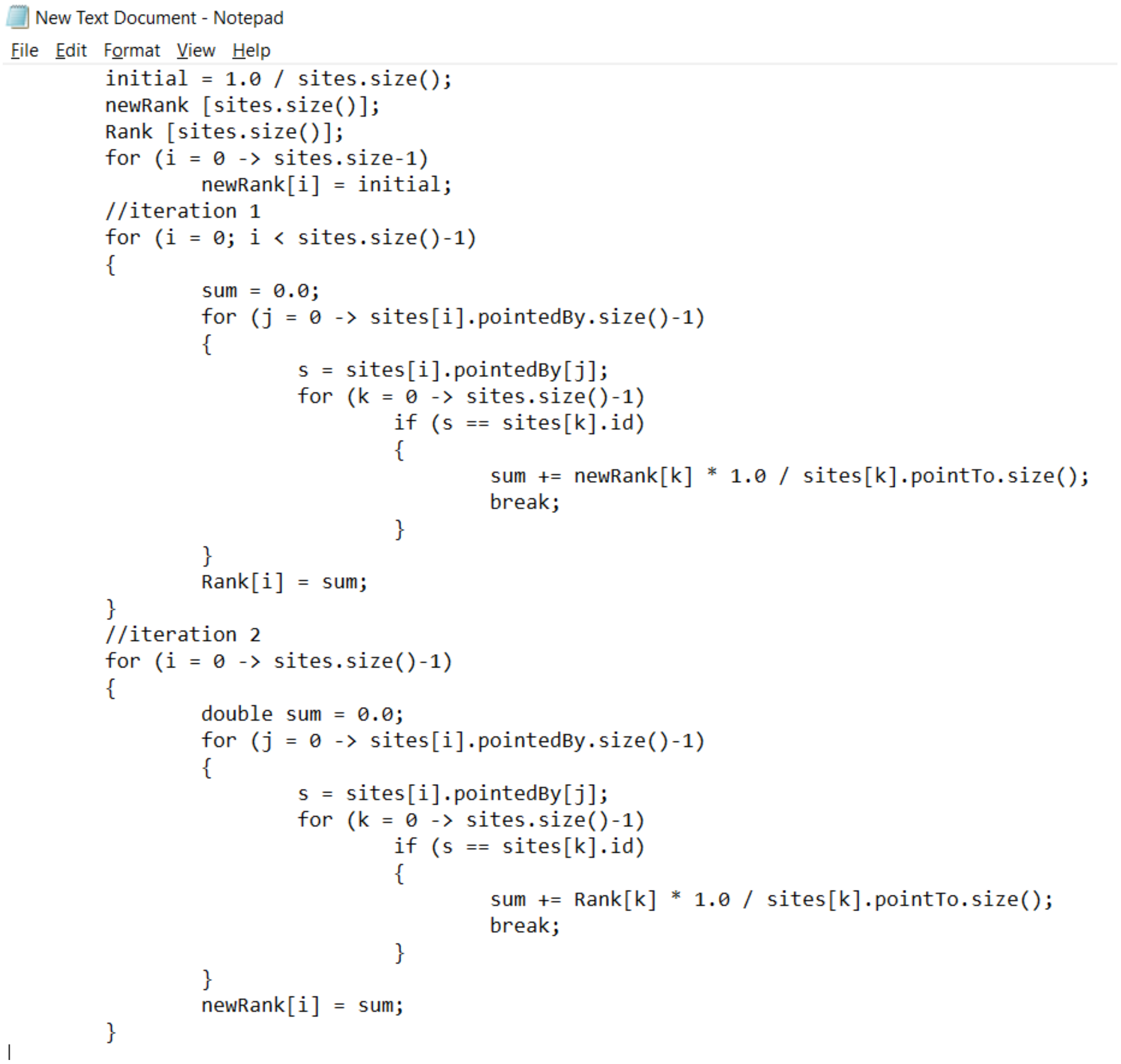
Students:

Sherif Sakran

Search Engine

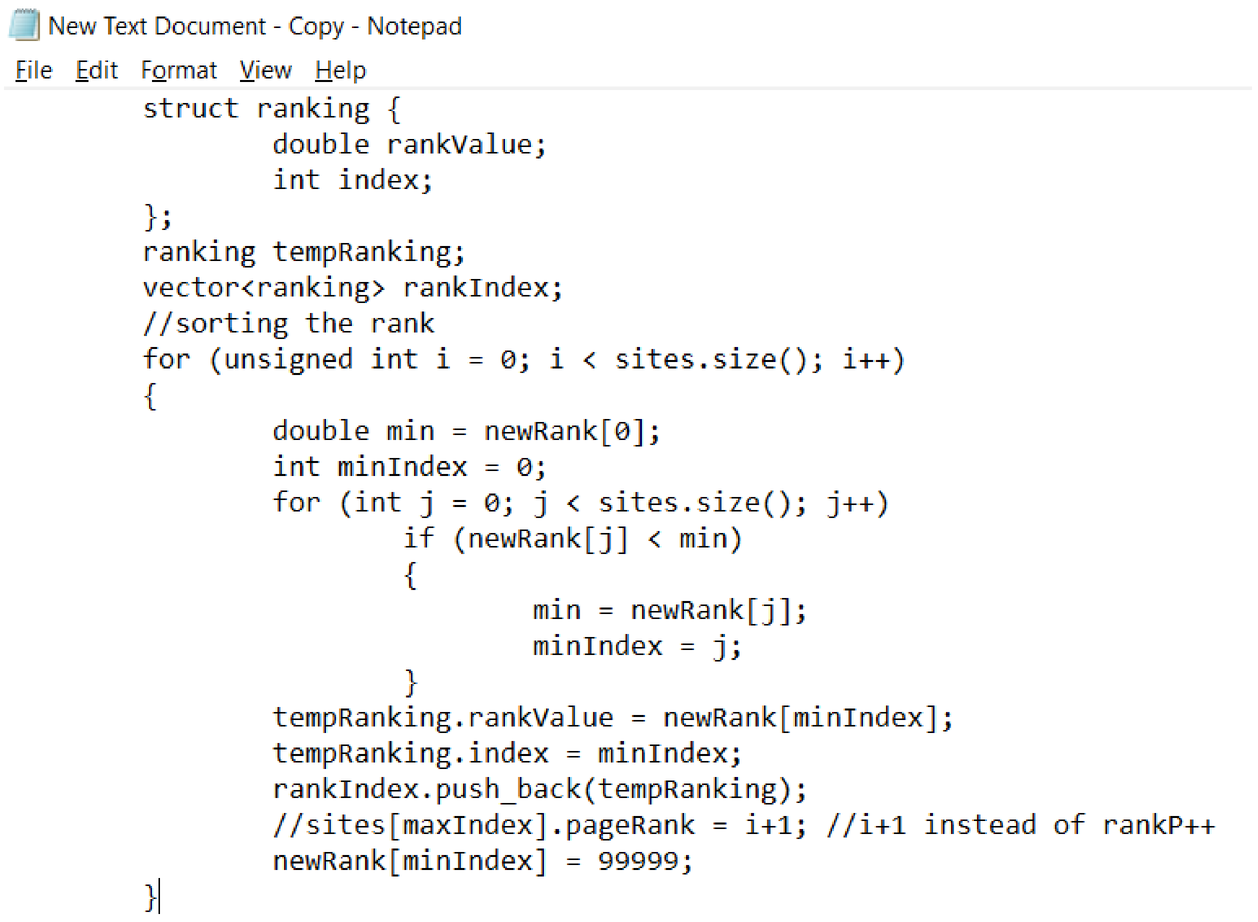
Rank and indexing calculation pseudocode:

The main idea for calculating the rank was to read the connections between the sites correctly which was verified by making a testing graph represented by an adjacency matrix (left in the code as a comment). Followingly, the rank was calculated as following:



```
        initial = 1.0 / sites.size();
        newRank [sites.size()];
        Rank [sites.size()];
        for (i = 0 -> sites.size-1)
                newRank[i] = initial;
        //iteration 1
        for (i = 0; i < sites.size()-1)
        {
                sum = 0.0;
                for (j = 0 -> sites[i].pointedBy.size()-1)
                {
                        s = sites[i].pointedBy[j];
                        for (k = 0 -> sites.size()-1)
                                if (s == sites[k].id)
                                {
                                        sum += newRank[k] * 1.0 / sites[k].pointTo.size();
                                        break;
                                }
                }
                Rank[i] = sum;
        }
        //iteration 2
        for (i = 0 -> sites.size()-1)
        {
                double sum = 0.0;
                for (j = 0 -> sites[i].pointedBy.size()-1)
                {
                        s = sites[i].pointedBy[j];
                        for (k = 0 -> sites.size()-1)
                                if (s == sites[k].id)
                                {
                                        sum += Rank[k] * 1.0 / sites[k].pointTo.size();
                                        break;
                                }
                }
                newRank[i] = sum;
        }
```

However, there was a problem that I face for equal ranks at the beginning. It is not a usual case, and it might seem not that important, but I dealt with it any way by making sure that no two sites have the same rank. I did so by sorting then checking if sequent sites have the same rank as following:


New Text Document - Copy - Notepad
File Edit Format View Help

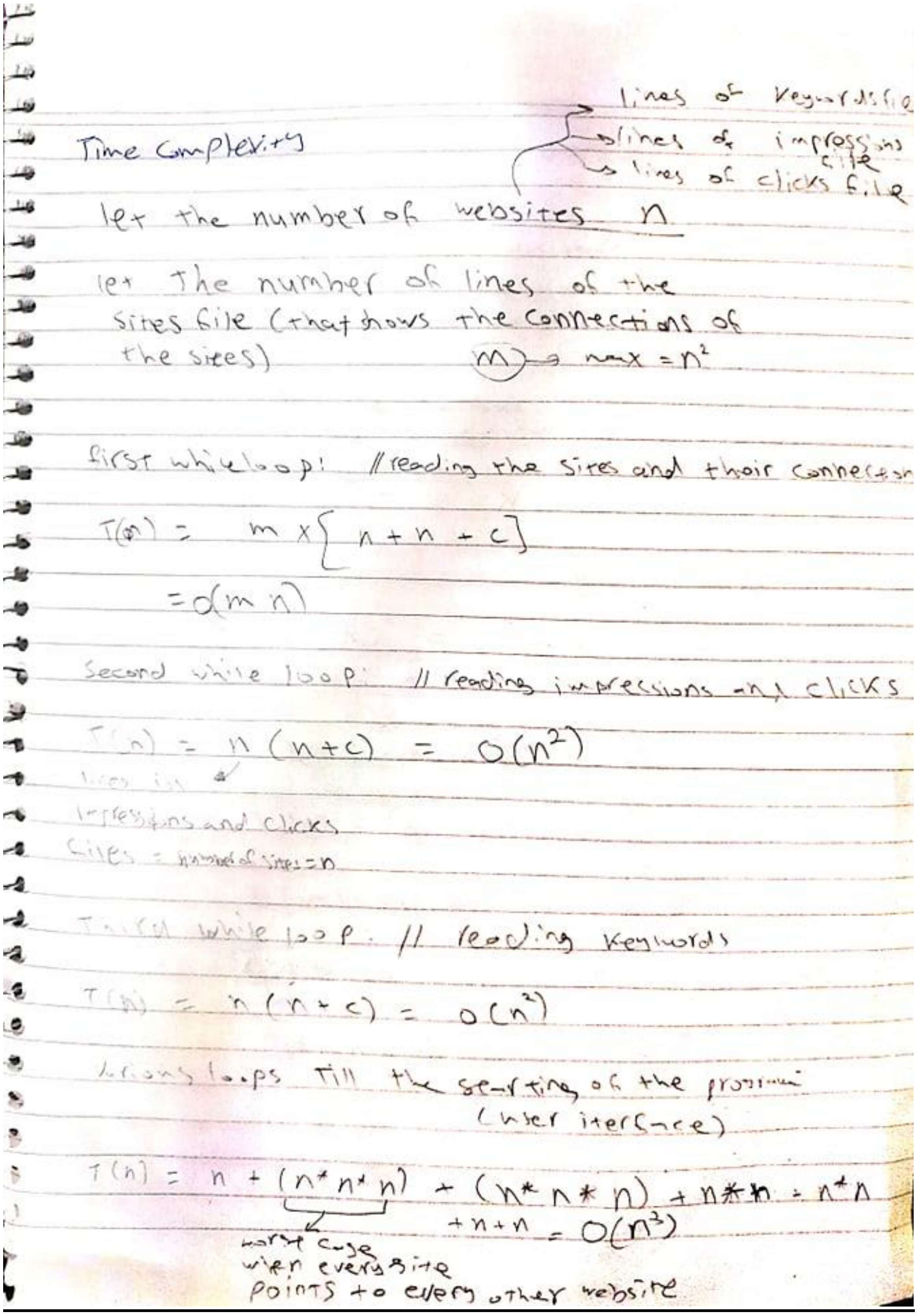
```
struct ranking {
        double rankValue;
        int index;
};
ranking tempRanking;
vector<ranking> rankIndex;
//sorting the rank
for (unsigned int i = 0; i < sites.size(); i++)
{
        double min = newRank[0];
        int minIndex = 0;
        for (int j = 0; j < sites.size(); j++)
                if (newRank[j] < min)
                {
                        min = newRank[j];
                        minIndex = j;
                }
        tempRanking.rankValue = newRank[minIndex];
        tempRanking.index = minIndex;
        rankIndex.push_back(tempRanking);
        //sites[maxIndex].pageRank = i+1; //i+1 instead of rankP++
        newRank[minIndex] = 99999;
}
```

Space and time complexity

Space:

- Let n be the size of the websites:
  - 2 vectors of websites of size n
    - One that stores the sites
    - One that shows the result (size n in the worst case when the keyword exist in all sites)
  - 2 1-D arrays for ranking calculations of size n
  - 1 vector of ranking structure of size n
  - 1 vector of words that splits the query into keywords. (Input dependent size)
  - Different variables of the pre-define data types were used in the program, and some of them are objects form the websites structure that I created. (used as a temp to push in the vector for example).

Time:



Time Complexity

lines of keywords file
lines of impressions file
lines of clicks file

let the number of websites n

let The number of lines of the sites file (that shows the connections of the sites) $m \rightarrow max = n^2$

first while loop: // reading the sites and their connection

$T(n) = m \times [n + n + c]$

$= O(m\,n)$

Second while loop: // reading impressions and clicks

$T(n) = n\,(n+c) = O(n^2)$

lines in impressions and clicks

sites = number of sites = n

Third while loop: // reading keywords

$T(n) = n\,(n+c) = O(n^2)$

various loops till the starting of the program (user interface)

$T(n) = n + \underbrace{(n * n * n)}_{\text{worst case when every site points to every other website}} + (n * n * n) + n * n + n * n + n + n = O(n^3)$

Starting of the program until until Sorting the results

let the number of key words be K → could be in any range

let the number of words of the query q

Case 1: Quotations

$T(n) = n \times q \times K = O(nqK)$

Case 2: "AND"

$T(n) = c(n \times q \times K) = O(nqK)$
c ← number of "AND" in the query

Case 3, 4: OR, no operations

$T(n) = c\,(n \times q \times K) = O(nqK)$
c ← number of "OR" in the query

Sorting the results until the end

$T(n) = n^2 + n + 2n + n + n + n$

$= O(n^2)$

Total complexity   constants were not significantly considered

$T(n) = O(mn) + O(n^2) + O(n^2 + O(n^3)$

$+ O(nqk) + O(nqk) + O(nqk)$

$+ O(n^2)$

n is the number of websites

where m is the number of lines of sites file

max $\underline{m} = n^2$

q is the number of words in the query
" must be less than the number of keywords"
" most probably less than the number of websites"
assume it to be n

k is the number of keywords per each site
(could be much larger than the number of websites)

$\Rightarrow T(n) = O(mn) + O(n^2) + O(n^3) + O(nqk)$

$= \boxed{O(nqk) \geq O(n^3)}$

n : number of websites
q : no of words in the query
k : " " keywords in each website

Data Structures used

- Pre-defined data types: strings, int, double.
    - Vectors
- Created data types:
    - website structure to represent each website which included
        - string name;
        - vector<int> pointTo;
        - vector<int> pointedBy;
        - int id;
        - int clicks = 0;
        - int impressions = 1;
        - double CTR = 0;
        - int pageRank = 0;
        - double score = 0;
        - vector<string> keywords;
    - ranking structure that was used to solve the problem of equal ranks in which I stored the rank and index of each site
        - int index;
        - double rankValue;

Tradeoffs

In the keywords, due to the nature of the .csv files that add commas at the end of shorter lines to make all of them have the same number of cells filled. In order to handle this issue easily, I supposed that the words "ENDOFWEBSITES" is added after the keywords of each site. That helped me to extract the keywords easily without a need to loop over the words twice.

I could have done something similar in during reading the websites, but the issue that prevented me from doing so is that the mechanism by which the sites are read is overlapping with the connections between the sites. Therefore, what could be done is to add the sites at first and gives them ids identical to their indexes then add the connections between sites. I did not do so since it would have been a change in the prompt of the project itself.

Summary

The search engine was tested on various number of websites with different keywords, and it produced results as expected. The engine considers special cases and validates inputs. It calculates the score as expected, and in turn, it shows the results in the required order.

Have fun with my Search Engine!