SEARCH ENGINE

Andrew Nady

900184042

Algorithm:

a. In order to make a search engine, you need to think about many things, like which page will appear first to the user, how will the input be parsed in order to do the best search according to a given input.

   SO let's begin with the question: "which page will appear first to the user"

Page Rank:

there is an algorithm that computes the page rank for every page
Reference: https://www.youtube.com/watch?v=P8Kt6Abq_rM
This page rank will be used in an equation to compute the score for every page.

$$score(page) = 0.4 \times PR_{norm} + \left(\left(1 - \frac{0.1 \times impressions}{1 + 0.1 \times impressions}\right) \times PR_{norm} + \frac{0.1 \times impressions}{1 + 0.1 \times impressions} \times CTR\right) \times 0.6$$

There are also two variables in the Equations:
1) CTR: number of clicks/ impressions
2)impressions: this is updated when an input is found in its keywords.

b. Second it is the side if the User, I took a string from the user and parsed it and checked if there is and, or, white spaces between the words.
   1. If it finds and between two word, the program search for the links that contain both keyword
   2. If it finds or, it will search for the URLS which contains any of them
   3. If it finds white spaces it will be treated as OR
   4. It the word between Quotation marks, then the program have to find between the quotation exactly

Pseudo Code:

1) Page rank, I made a function that compute the number of children for a given node.
    a. Complexity: O(N*E), where N is the number of nodes (Vertices) the graph has, and E is the number of edges

```
//----------------------------Page Rank----------------------------//
void PageRank(Graph &g,int childerC[]){
    int size=g.nodes.size();
  vector<double>  Prank(size);
   vector<double> PrankUtil(size);
    for(int i=0;i<size;i++){
        Prank[i]=1.0/double(size);
        PrankUtil[i]=0.0;
    }
    //i represents all the edges
    for (int i=0; i<size; i++)//represent all the sources
        for (int j=0; j<g.edges.size(); j++)
        {
            // cout<<i<<" "<<g.edges[j].dest<<endl;
        int index=0;
            if(g.nodes[i]==g.edges[j].dest)
                {

                PrankUtil[i]=PrankUtil[i]+(Prank[g.edges[j].srcId]/childerC[g.edges[j].srcId]);
                // Prank[i]=PrankUtil[i];
                // cout<<g.edges[j].dest<<" "<<PrankUtil[i]<<" "<<childerC[g.edges[j].srcId]<<en

                }

        }
        for (int i=0; i<size; i++){
            Prank[i]=PrankUtil[i];
             g.urls[i].pr=PrankUtil[i];
            PrankUtil[i]=0;
            // cout<<Prank[i]<<endl;
        }
```

```
void children_count(Graph g,int childerC[]){

    //   int childerC[g.adjList.size()]={0};
        for (int i=0; i<4; i++)
        {
            int counter=0;
            for (string v: g.adjList[i])
            counter++;
            childerC[i]=counter;
            // cout<<counter<<endl;

        }
}
//----------------------------children cou
```

2) Ctr,
    a. I made a new file to store the number of clicks
    b. So, when the user click on a specific Url, it will call this function and modify the number of clicks and then store it in the file

```
void Ctr(Graph &g,string input){

    ofstream Ctr_file;
    Ctr_file.open("Ctr.txt");
    for(int i=0;i<g.urls.size();i++){
        if(g.urls[i].name==input)
        {
            int clks= g.urls[i].clks+1;
            g.urls[i].ctr=(clks/g.urls[i].impression)*100;
            g.urls[i].clks=clks;
            // cout<<g.urls[i].ctr;

        }
    Ctr_file << g.urls[i].name << ","<<g.urls[i].clks<<","<<g.urls[i].ctr<<endl;
    }

    Ctr_file.close();
}
```

3) Impressions,
    a. I did almost the same logic as CTR, so I modify the impressions when it appears to the user, if the input is find in the keywords of the URL
    b. Complexity, O(N), where the N is the number of nodes in the grpah(URLS)

```
//-------------------------------Impressions-------------------------------//
void impressions(Graph &g,string input){

    ofstream impressions_file;
    impressions_file.open("impression.txt");
    for(int i=0;i<g.urls.size();i++){
        if(g.urls[i].name==input)
        {
           g.urls[i].impression++;

        }
    impressions_file << g.urls[i].name << ","<<g.urls[i].impression<<endl;
    }


   impressions_file.close();
}
```

4) Finally, the score of every page,
    a. I just used the computed impressions, ctr to compute the Score
    b. Complexity, O(N), where the N is the number of nodes in the grpah(URLS)

```
//-------------------------------Score-------------------------------//
void score(Graph &g){
    for(int i=0;i<g.nodes.size();i++){
 double temp=(0.1*g.urls[i].impression)/(1+0.1*g.urls[i].impression);
 double temp1=(1-temp)*g.urls[i].pr;
 double temp2=temp*g.urls[i].ctr;
        g.urls[i].score= (0.4*g.urls[i].pr+(temp1+temp2)*0.6);
        // cout<< g.urls[i].score<<endl;
    }

}


//-------------------------------Score-------------------------------//
```

5) Filling the Graph, I am filling the graph and establishing the edges

```
//-------------------------------reading URls to the graph -------------
void readURLs( Graph &g){
    ifstream f;
    f.open("Web_Graph.txt");
   if(f.is_open())
   {
       string line;
       int i=0;
       while(getline(f, line))
       {
           stringstream ss(line);
           string src, dest;
           getline(ss,src,',');
           // cout<<src<<"-> ";
           getline(ss,dest,'\n');
           // cout<<dest<<endl;
           g.addEdge(src,dest);

           if(g.nodes.size()==0)
            g.nodes.push_back(src);
            else{
           int flags=0;
           int flagd=0;
           for(int j=0;j<g.nodes.size();j++)
                {
                    if(g.nodes[j]==src)
                   flags=1;
                   if(g.nodes[j]==dest)
                   flagd=1;
                }
           if(flags==0)
               g.nodes.push_back(src);
            if(flagd==0)
               g.nodes.push_back(dest);

            }

       }
   }
       f.close();
}

//-----------------------------------------reading URls to the graph------
```

6) Then I make another function to read every thing about every UrL and putt it into struct

```
//
struct URL{
string name;
double impression;
double ctr;
double clks;
double pr;
double score;
};
```

7) Finally, I need to make the function of the Searching about keyword, This happened it many steps:
    a. First I need to parse the user input, so that I can know if it is between 2 quotation marks or there is or , and and so
    b. So I make a split function to split the words In the string if there is not qutatin marks
    c. Then I pass the vector of the words to another functoin check if they exist or not in the graph

```
//--------------------parsing input-----------------------------
vector<string> split(string str)
{

    vector<string> name;
    istringstream ss(str);
    string word;
    if(str[0]=='"'&&str[str.size()-1]=='"')
      {

       //   str.erase(remove(str.begin(), str.end(), '"'), str.end())

          name.push_back(str);

      }
    else {
        while (ss >> word)
            {
                name.push_back(word);
            }
    }
        cout<<endl;
    return name;
}
//--------------------parsing input-----------------------------
```

    d. I implemented a function that is responsible for searchin the keyWords from the file, so it takes the vector of the strings and it decide to make and or or according to the words inside of the vector, if the second word is and or it is or it is white space, also it has other handling such as making And, or in the lower case. This function return vector of struct, the struct contains the URL and the score

```
string make_lowercase( string in )
{
  std::string out;

  std::transform( in.begin(), in.end(), std::back_inserter( out ), ::tolower );
  return out;
}
```

8) In the main function I called all these functions in an appropriate way to make my application works, but also I added a quicksort function that sorts that Output Urls according to the scores.

Testing:

1) Testing the and, or , “ ”

```
Welcome!
 1)New Search
 2)Exit
1
search bar .... :data

Search Results:
1)www.test1.com

Would you like to
1)Choose a webpage to open
2)New search
3) Exit
1
```

```
Would you like to
1)Choose a webpage to open
2)New search
3) Exit
2
search bar .... :complexity or data

Search Results:
1)www.test1.com
2)www.test3.com
3)www.test2.com
```

```
2
search bar .... :complexity and data

Search Results:
1)www.test1.com

Would you like to
1)Choose a webpage to open
2)New search
3) Exit
2
```

2) Testing that the Ctr and impressions is updated

Before clicking	after