Mohamed Basuony
900182339
Analysis and Design of Algorithms
Final project report

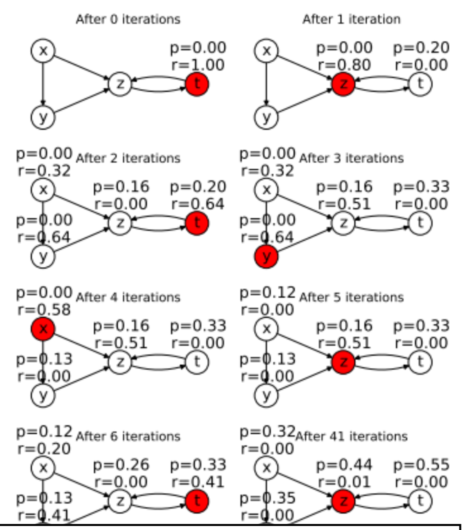
Table of Contents

The pseudocode for my indexing algorithm:

```
indexing(string Usearch, map tempmap)//storing the sites and the
                        //values of the verticies in the grap
   i;
   string s1, s2;
   if (Usearch[0] == "")
   {
      Usearch.erase Usearch.begin
      Usearch.erase(Usearch.end() - 1);
      stringstream yo(Usearch);
      Usearch = s1;
      stringstream g(Usearch);
      Usearch = s2;
     searcho(s1,s2, tempmap);// passing to a function searching both
                                   words



   }
   if (Usearch[0] == "")
   {
      stringstream yo(Usearch);
      searcho(Usearch);//searching one word only

   }
}
```

The pseudocode of my pagerank algorithm

```
Initialize (sparse) estimate-vector ps = ~0 and (sparse) residual-vector rs = es
(i.e. rs[v] = 1 if v = s; else 0)
2: while 9u 2 V s:t: rs[u]
du > rmax do
for v 2 N[u] do
rs[v] += (1 - α)rs[u]=du
end for
ps[u] += αrs[u]
rs[u] = 0
 end while
 return (ps; rs)
```




Retrieved from stanford.edu

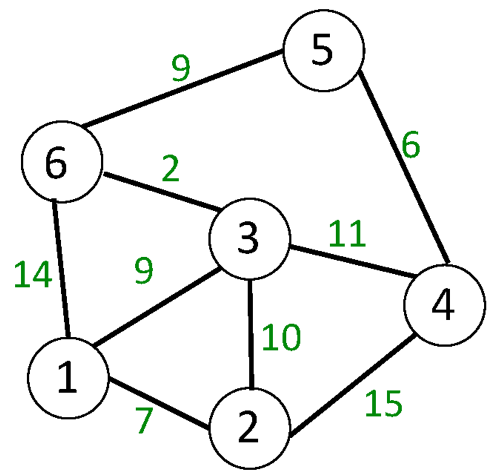
The complexity of PageRank algorithm:

Page rank algorithm determines the importance of a page when showing the results after performing a search. The higher the rank of a certain page the higher priority it gets while outputting the results to the user. The highest priority websites are showed first and the lowest priority sites are showed at the end. Since we are working on a graph data structure, the algorithm typically needs to visit every node in the graph to determine the top ranking pages (links) to show in the result. This operation will have need a maximum n+m times to be done, where m is the number of edges, and n is the number of nodes. This is the worst case scenario because in this case, all the vertices need to be visited and checked. The result of this will be the last node the algorithm will visit. This means that the page rank algorithm when operating on a graph of n nodes and m edges will have a complexity of O(n+m).

The complexity of indexing algorithm:

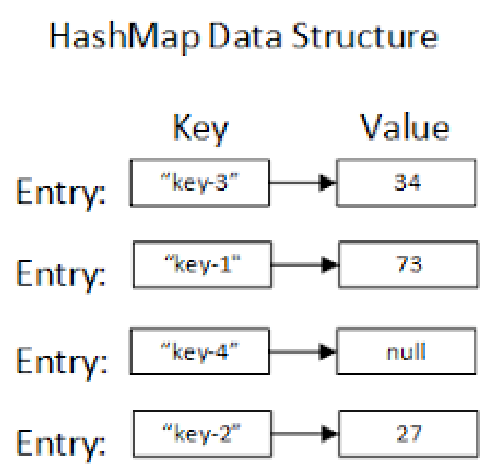
Indexing algorithm is used to retrieve the search query from the user and determine whether it is using AND operator or OR operator. If the User place quotation marks around the search query, then the indexing algorithm will search for both keywords in the websites. If the user entered the words without placing quotation marks around them, the indexing algorithm will display results for the first keyword or the second keyword. Since splitting the user’s input and placing it in the search algorithm is the main function of the indexing algorithm. Its complexity is O(n) where n is the number of keywords inputted by the user.

The main data structure used in the project (Graph):




Graphs are widely used complex data structures. There are 2 types of graphs be directed or undirected. There are other 2 subtypes of graphs which are weighted or unweighted. The graph structure consists of finite set of vertices and edges which connects these two vertices. The directed graph is different from the undirected graph in one thing. The order of pairs matter in a directed graph while in a directed graph it does not. In my project I used graphs because they can easily store information and the information can be organized in multiple ways. Furthermore, multiple operations can be performed on information while it is stored in a graph. In the project, an undirected, unweighted graph of int was used. The numbers stored in the graph were linked to the websites that were retrieved from the user's file. Each website has a unique number that is stored in a vertex in the graph. In this way, each site can be assigned a unique PageRank and unique number of impression. The cost of traversing the graph is O(N) where N is the number of nodes which gives it a disadvantage over the tree data structure for example (O(logn)). However, if you know the index of the node you're searching for it takes O (1) to retrieve the data which give it the upper hand in this situation.

Another type of data structure used in the project (Maps):



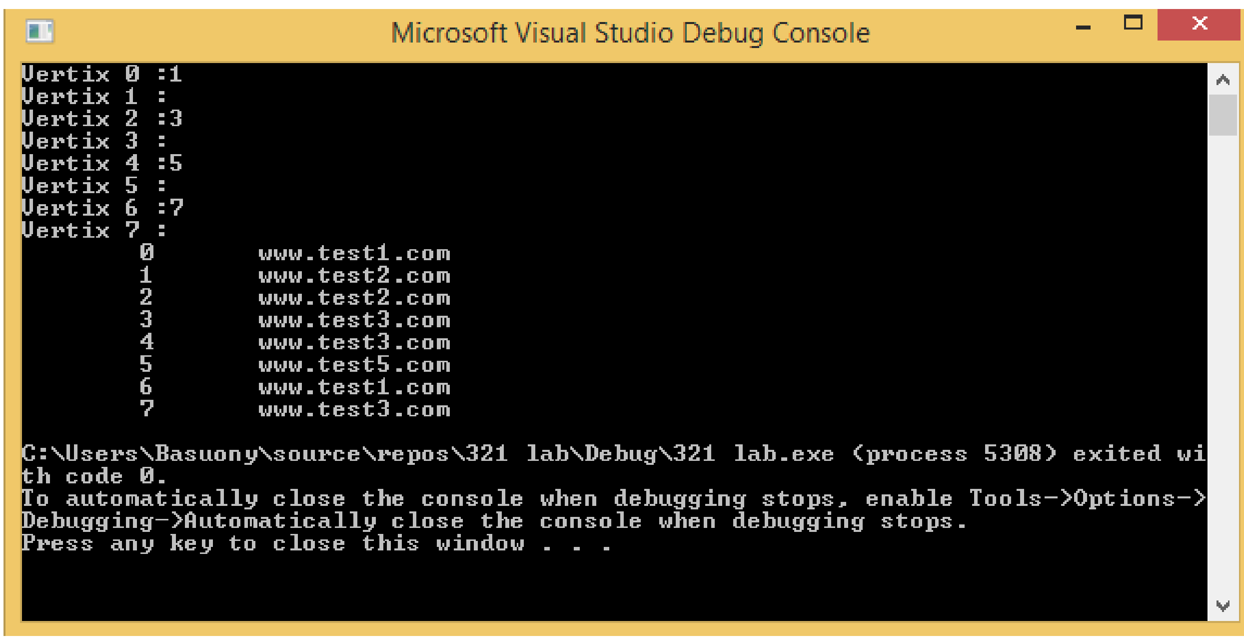

Maps are an abstract data type which consists of a key value pair. It's similar to looking up word in the dictionary. The values are stored in an array. Maps were used in this project because it is manageable to deal with multiple types while using maps. Assigning unique IDs the websites was made possible by using and int to string map. Other multiple maps were used in the operations of searching, printing, and organizing the graph. The maps were the second step in implementing the graph. The first being importing the data and then storing them in map and finally using the maps to build up the graph

Tradeoffs made during implementing the project:

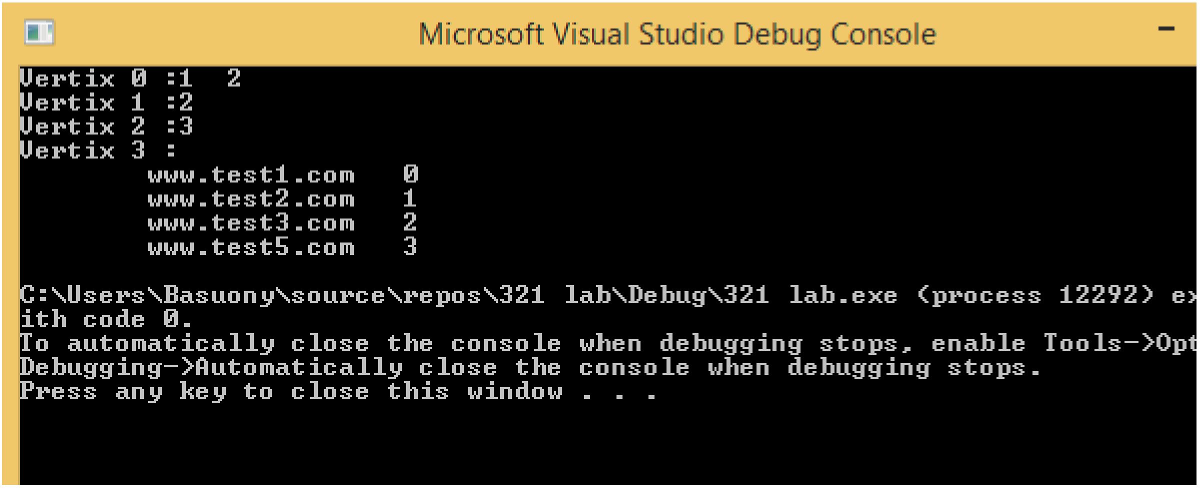
Using maps instead of Vectors:

The first plan that was in mind was to use multiple vectors to store the information retrieved from parsing the file and giving each site a unique ID to be stored inside the graph. Vectors were chosen at first because the use less memory than maps and they are quicker than them. However, in order to manage the files and graph, so many vectors needed to be created. In order to deal with many vectors, the code will start to get messy and confusions will happen more often. Therefore, I decided to use maps instead of vectors although they more memory than vector data structure. The reason that they use more data structure is that they have pointer manipulation between nodes.

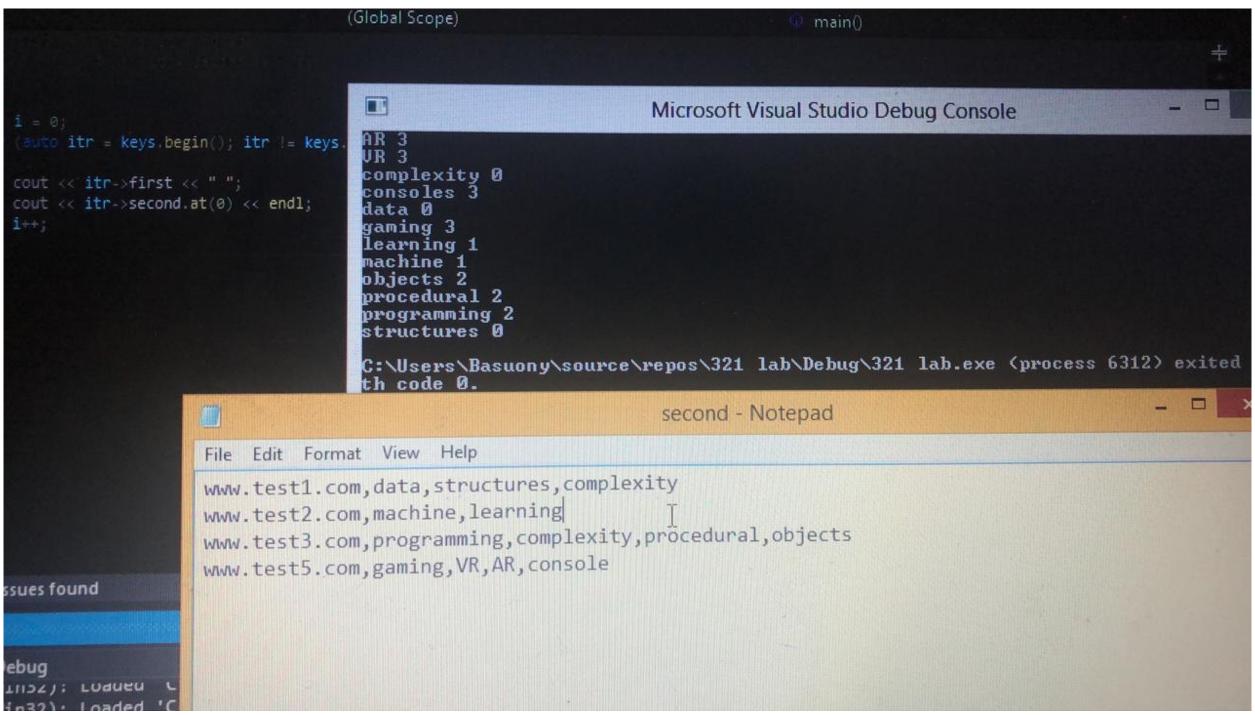
The results of running the project:



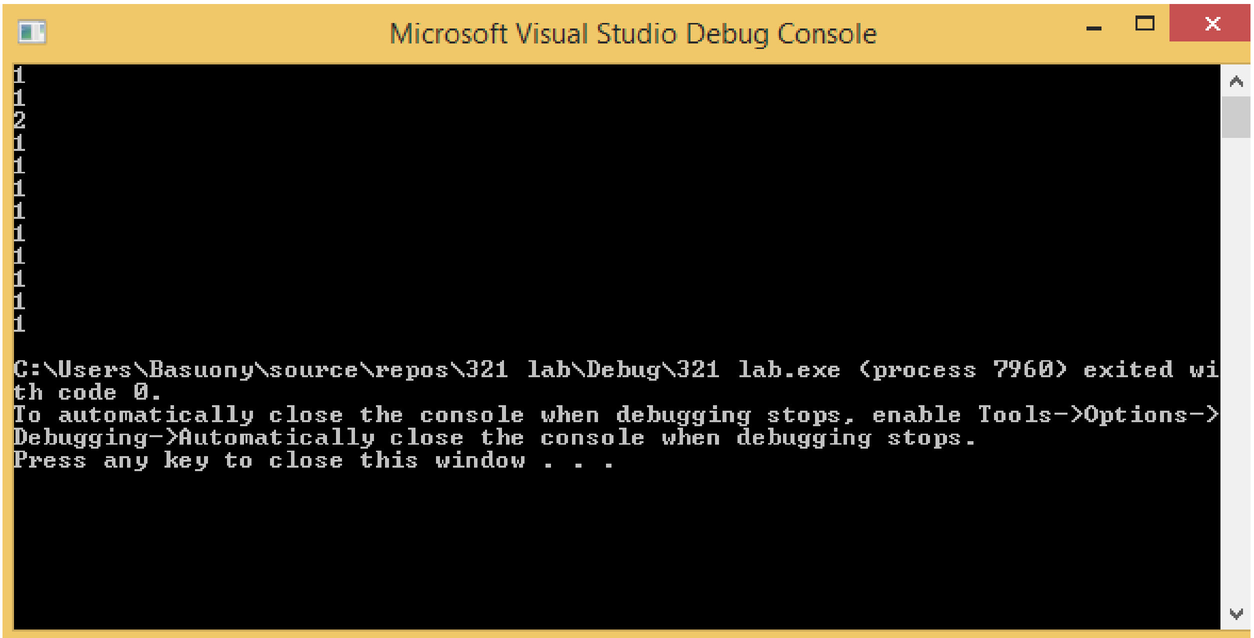

The following screenshot is the first screenshot after retrieving the info from the file. Each site in every line was given a unique ID and stored in the graph. However, sites were stored twice and vertices were duplicated.



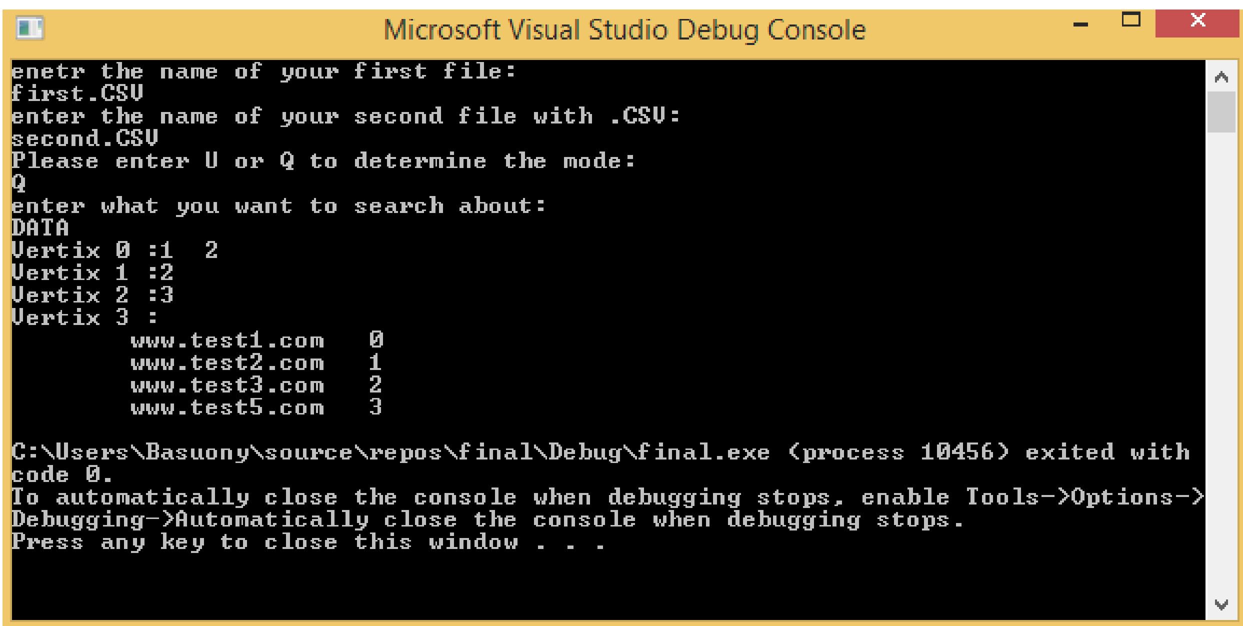

The second screenshot was after storing each site in a map and removing duplicates. After that, the vertices were connected after reading the first file line by line.



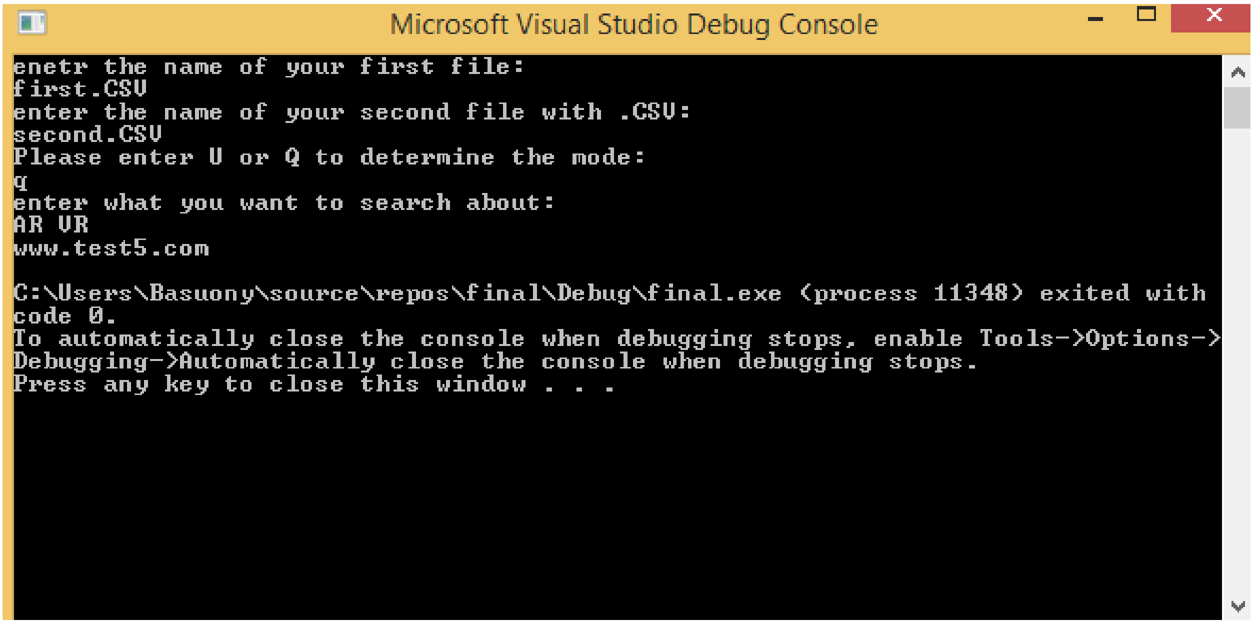

The following photo was after reading all the keywords from the second file and storing it in a string to int map. After that another map was made to connect the keywords to the node of the graph. Every keyword has the corresponding ID in front of it.




The following numbers are the number of the vertices each keyword represent. The number 2 corresponds to the only keyword represented twice “complexity” as it is a keyword to 2 websites




The following screenshot was after coding the 2 modes available for the user Q and U. However, unfortunately, the search didn’t output any result.




The final screenshot is of the working program. The following result was retrieved without the PageRank algorithm. The search doesn’t always work, but at least it outputs some results.