REPORT

IT-427

PROG ASSIGNMENT – 3

ULID – SPARAMK

SECRET DIRECTORY- “marvel”

Question- For this assignment, you are asked to use two approaches, Breath First Search and Depth First Search, to find all connected components of undirected graphs in a given file.

The commands to run the code for the given file

udGraphs.txt

Compile – javac graphcc.java

Run – java graphcc udGraphs.txt

Code Explanation-

I have created a class named graphcc which contains the main method. I am taking file name/ file path from command line arguments. I am using

File Reader and BufferedReader to parse my file.

I am using an **adjacency matrix** to store my graph data. Once when I have the graph data ready, I am using arraylist which contains all the vertices and using them to call the BFS and DFS staring with vertex 0.

Methods used-

solveBFS(int[][] adjmatrix,int check,ArrayList<Integer> checker)

I am taking adjacencymatrix and an integer(check) which is the vertex for which we have to find BFS , arraylist checker this helps me to run BFS on unconnected components of the graph.

I am maintain a Queue and adding check value to queue and running a while loop until queue is empty.

I am making the check vertex as visited and then checking the neighbors of the check vertex and if the vertex is not visited then I am adding it

into the result set and removing it from checker list. Once the neighbors are done the remaining vertices will be executed. Once when the while loop is done I am checking the checker array list to see if there are any vertices left, If yes, then I am calling the SolveBFS() method with first element of list as the source vertex to search. In this manner all the unconnected components will be dealt.

solveDFS(int[][] adjmatrix,int check,ArrayList<Integer> checker,ArrayList<Integer> visited,String res,HashSet<Integer> hs)

In this method I am passing adjacencymatrix , check interger i.e the vertex to check DFS, checker arraylist for unconnected components. Then visited list , result String, and a HashSet which adds the results in it. I am using a queue similar to BFS and running the similar while loop

Now if I find a unvisited node as a neighbor then immediately I am calling solveDFS() method for the new vertex which does recursion and calculates the DFS.

Data Structures Used-

- List
- Set
- Queue

Algorithm- I used algorithms which were taught in class but enhanced it using some own logics or methods.

Time complexity of my code-

Big O (V^2)

Space complexity of my code-

Big O (V^2)

V= number of vertices.

This is because I am using adjacency matrix to store the graph data. In worst case if every vertex has all the other vertices connected to it then the for loops run V^2 times.