# California State University, Fresno

# DEPARTMENT OF COMPUTER SCIENCE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Class: | **Algorithms & Data Structures** | | | Semester: | **Fall 2021** |
|  | | | | | |
| Points |  | Document author: | **Marlon Branham** | | |
|  | Author’s email: | **zekebranham@mail.fresnostate.edu** | | |
| Laboratory number: | **Laboratory 2** | | |
|  | | | | | |

**1. Statement of Objectives**

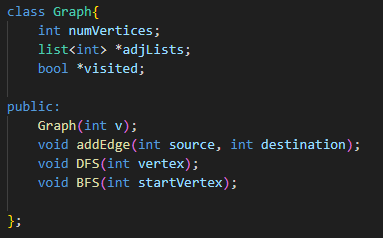
In this lab, were to implement Breadth-First Search and Depth-First Search algorithms. We are to implement both algorithms and output a list of visited elements using both. The purpose of this lab is to gain experience in working with BFS and DFS algorithms while implementing them successfully. In this report, I will be discussing my experimental procedure, an analysis of results, encountered problems, and a conclusion of my findings.

**2. Experimental Procedure**

The experimental procedure for this lab included implementing both BFS and DFS algorithms of a graph with a maximum of 10 different nodes numbered 1-10. The input will be the number of nodes and adjacency list which are held by each node.

I created a class called “Graph” that declares my number of vertices/nodes, a pointer to a list of adjacent vertices, and a pointer to a Boolean array of visited vertices. I have a constructor that takes in the size of the graph, dynamically allocating the memory, and also my three functions: “addEdge”, “DFS”, and “BFS”.

My “addEdge” function takes in the values of the source and the destination. The function pushes the destination value to the front of the source’s adjacency list, effectively creating an edge between the two. My “Graph” constructor takes in the size of the graph (number of vertices) and initiates numVerticies to this value, sets the adjacent lists to a list of vertices, and initializes the Boolean “visited” array while setting the entire array to false. The “DFS” function takes in the vertex to start on, then recursively goes through the adjacency list of the vertex, moving from vertex to vertex as the recursive call unwinds. Finally, my “BFS” function takes in the starting vertex again and uses a queue to hold the values of the nodes at each level.

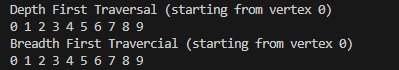
****

**3. Analysis**

My results came out as expected. The Depth-First Search started from vertex 1 and traveled through the adjacency list to output the unvisited nodes. The Breadth- First Search started from vertex 1 and traveled from level to level displaying all the nodes at each level.

A screen shot of a computer program

Description automatically generated



**4. Encountered Problems**

There were no encountered problems with this lab. Following the pseudocode, reviewing notes and online resources allowed me to successfully implement both DFS and BFS algorithms with no issues.

**5. Conclusions**

This lab was meant to provide experience in implementing and using both depth-first searching and breadth-first searching. After completing this lab, I have successfully learned the processes of implementing these algorithms myself and will continue to work on building my knowledge of graphs and different traversal methods.

**6. References**

<https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/#:~:text=Depth%2Dfirst%20search%20is%20an,along%20each%20branch%20before%20backtracking.>

<https://www.programiz.com/dsa/graph-dfs>

<https://m.youtube.com/watch?v=Urx87-NMm6c&pp=ygUMI2xlb2ZkZnNzaG93>

<https://www.hackerearth.com/practice/algorithms/graphs/breadth-first-search/tutorial/>