Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

3D5A DATA STRUCUTRES AND ALGORITHM

Assignment 4 Report

Prepared By:

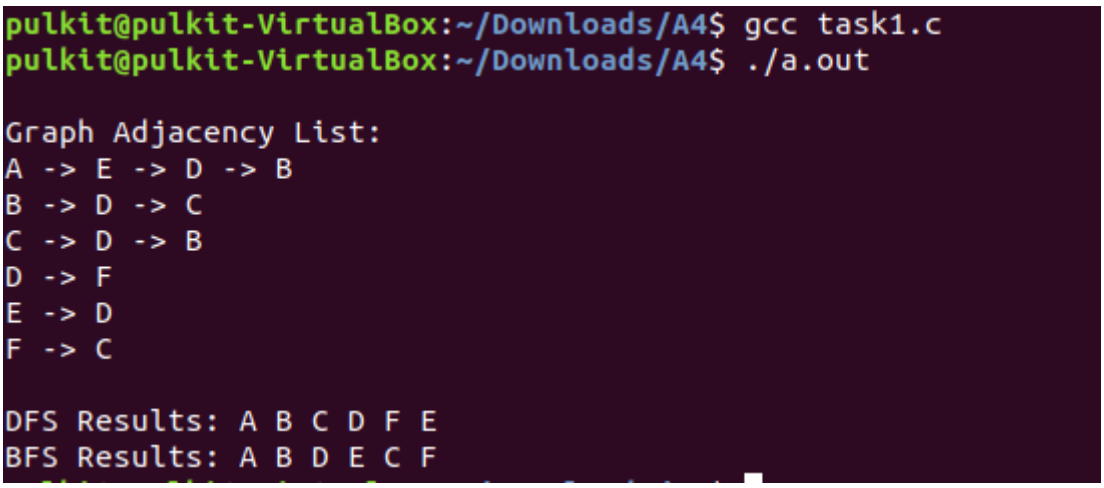
PULKIT SHARMA

19323659

Computer Engineering

Task 1

Choose a suitable graph representation, implement it and represent the graph below. Perform both a Depth First Search and a Breath First search with A as the start vertex, printing the nodes in the order you visit them.



- I chose adjacency list implementation for this task.
- I used this because it is comparatively faster to add/ delete an edge in list than in matrix.
- It is an unweighted graph therefore a lot of extra space would have been wasted building a matrix.
- We implement BFS using a queue data structure. One vertex is selected at a time and it's marked visited. Then its adjacent nodes are visited and stored in a queue and the process is repeated.
- We implement DFS using a stack data structure. It performs in two stages, first the visited vertices are pushed into the stack and second if there are no new vertices the vertices in the stack are popped and the process is repeated.
- BFS is slower than DFS.

Task 2

Choose a suitable graph representation, implement it and represent the graph below (from Tutorial 8). Use Dijkstra's algorithm to calculate the shortest path from A to each of the other nodes. Your algorithm should output the list of the nodes in the order in which they were made permanent, and the shortest distance from A to each of the other nodes.

```
pulkit@pulkit-VirtualBox:~/Downloads/A4/task2$ gcc task2.c
pulkit@pulkit-VirtualBox:~/Downloads/A4/task2$ ./a.out

Order of Vertices being made permanent:
(A)  was made permanent
 (B)  was made permanent
 (C)  was made permanent
 (G)  was made permanent
 (E)  was made permanent
 (D)  was made permanent
 (F)


   Distance of vertex (B) from (A) = 1
Path  = (B) <- (A)


   Distance of vertex (C) from (A) = 2
Path  = (C) <- (B) <- (A)


   Distance of vertex (D) from (A) = 7
Path  = (D) <- (E) <- (C) <- (B) <- (A)


   Distance of vertex (E) from (A) = 5
Path  = (E) <- (C) <- (B) <- (A)


   Distance of vertex (F) from (A) = 7
Path  = (F) <- (E) <- (C) <- (B) <- (A)


   Distance of vertex (G) from (A) = 3
Path  = (G) <- (B) <- (A)
```

- I used adjacency matrix in this task because the graph is a weighted one and we need to traverse into each node.
- Dijkstra algorithm is a greedy algorithm to find the shortest path from source to all the vertices in a given graph.
- We check the minimum distance from a node to adjacent matrix and then calculate the total cost to reach that matrix, if it is lower than the previous cost then it updates the new cost.
- We find the shortest distance from node A to F is coming out to be 7
- We print the order in which the nodes were made permanent. It is coming out to be A-> B-> C-> G-> E-> D-> F.