Assignment 5 ─Graph

TA: Kile (a256673@hotmail.com)

Deadline: Dec. 26, 11:59pm

1. Introduction

In this assignment, you need to implement three graph algorithms.

- Find the cost of the shortest path between two specified vertices by using **Dijkstra's algorithm**.
- Find the minimal cost spanning tree by using **Kruskal's algorithm**.
- Implement a graph traversal algorithm: **Breadth First Search**

2. Descriptions

You should use an adjacency matrix using an array to implement the program.

(1) Dijkstra's algorithm (30%)

✧ Function details:

Your program should read an input file to build the graph. Next, ask the user input two variables to specify two vertices. Then, show the cost of the shortest path between two specified vertices in the weighted directed graph.

✧ Input file format:

```
5
0 2 5 0 0
0 0 6 4 0
0 0 0 0 3
0 0 9 0 0
0 0 0 0 0
```

Figure 1. Example of the input file.

The example of the input file is shown in Figure 1:

- The input file includes the information of the weighted directed graph.
- The first line of the input file is *n* representing the number of vertices. The vertex number starts from 1 and ends at *n*, that is, V={1, 2, …, n}.
- The second line to the last line of the input file represent an adjacency matrix.
-

Next, read two variables from the user as the starting and the ending nodes from the command linked. The weighted directed graph corresponding to the adjacency matrix in Figure 1 is shown in Figure 2.

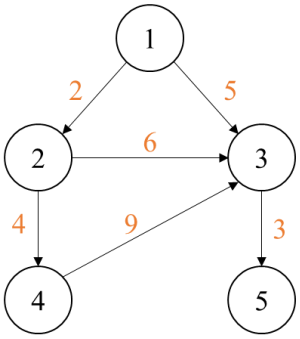

Figure 2. Example of the weighted directed graph.

✧ Output format:

```
Enter two vertices(start end):1 5
The cost from 1 to 5 is : 8

Enter two vertices(start end):3 4
The cost from 3 to 4 is : -1
```

Figure 3. Example of the output.

The examples of the outputs are shown in Figure 3. Print out the cost of the shortest path between two specified vertices. If the two verities are not reachable, print -1.

(2) Kruskal's algorithm (30%)

✧ Function details:

Your program should read an input file and show each edge of the minimum cost spanning tree according the processing order.

✧ Input file format:

```
5
0 2 5 0 0
2 0 6 4 0
5 6 0 9 3
0 4 9 0 0
0 0 3 0 0
```

Figure 4. Example of the input file.

The example of the input file is shown in Figure 4:

- The input file includes the information of the weighted undirected graph.
- The first line of the input file is *n* representing the number of vertices. The vertex number starts from 1 and ends at *n*, that is, V={1, 2, …, n}.
- The second line to the last line of the input file represent an

adjacency matrix.

The weighted undirected graph corresponding to the adjacency matrix in Figure 4 is shown in Figure 5.

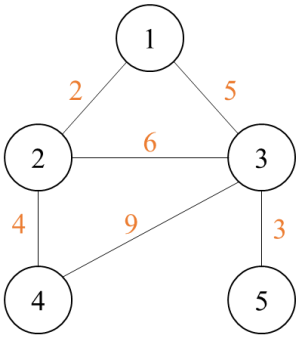

Figure 5. Example of the weighted undirected graph.

✧ Output format:

```
The edges of Minimum Cost Spanning Tree are

Edge 1:(1 2) cost:2
Edge 2:(3 5) cost:3
Edge 3:(2 4) cost:4
Edge 4:(1 3) cost:5
Minimum cost = 14
```

Figure 6. Example of the output.

The example of the output is shown in Figure 6: The output consists of the starting node and the ending node of an edge with its corresponding cost.

(3) Breadth First Search (30%)

✧ Function details:

Your program should use the output of the previous part of this assignment and traverse the undirected graph by Breadth First Search.

✧ Input format:

(A) You need to use the output from Kruskal's algorithm. We call this graph Graph A. After connecting each node of Graph A together, we get a new fully-connected graph named Graph B. We get a new graph named Graph C that is the difference sets of Graph A and Graph B. Graph C is the input to be searched.

(B) Select the edge with highest cost in the Graph A; then, from the edge, select the vertex with the largest number as the starting node for the Breath First Search.

(C) Traverse Graph C by Breadth First Search, starting from the starting node.

For example, Figure 7 is the abovementioned Graph A. After connecting each node of Graph A together, we get a new graph named Graph B (see

Figure 8). Take difference sets of Graph A and Graph B, we get a new graph named Graph C (see Figure 9). Edge $\{V_1,V_3\}$ with a cost of 5 is the longest edges in the Graph A. Node 3 has been chosen as the starting node, because 3 is larger than 1. Finally, we traverse Graph C by Breadth First Search started from the #3 node.

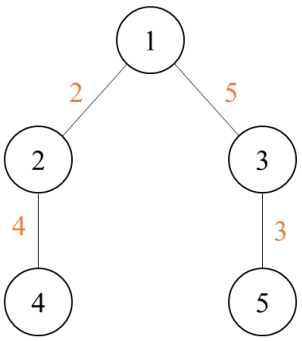

Figure 7. The output after performing the Kruskal's algorithm (i.e. Graph A).

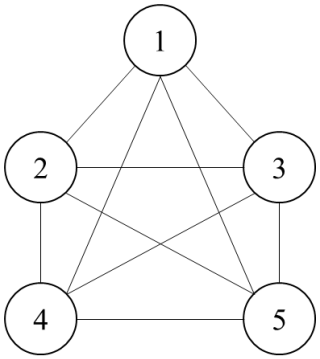

Figure 8. Example of Graph B.

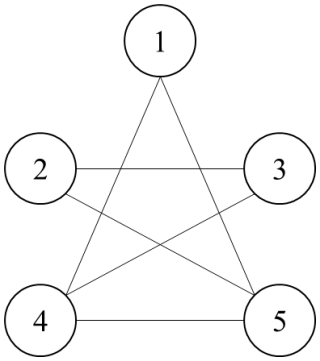

Figure 9. Example of Graph C.

✧ Output format:

```
Level Name
0 3
1 2
1 4
2 5
2 1
```

Figure 10. Example of the output.

The example of the output is shown in Figure 10: The output consists of the traversal sequence of the graph.

3. Readme, comments and style (10%)

- Readme file should include your name, class ID, a brief description of how to compile your program, and other issues you think that will be helpful for the TAs to understand their homework.
- You should add some comments to your source code to keep source code maintainable and readable.

4. How to submit

To submit your file electronically, enter the following command from the csie workstation:

turnin ds.hw5 [your files…]

To check the file you turnin, enter the following command from the csie workstation:

turnin –ls ds.hw5

You can see other description about turnin from following link:
https://www.cs.ccu.edu.tw/~lab401/doku.php?id=turninhowto

5. Grade

The TA(s) will mark and give points according to the following grading polices:

- 10% Source code can be compiled without any error
- 10% Readme file, code style, and comments in source code
- 80% Three required sorting algorithms and the result correctness (result1.txt, result2.txt, and result3.txt)

A readme file should include your name, class ID, a brief description of the code, and other issues students think that will be helpful for the TAs to understand their homework.