Indian Institute of Technology Ropar

Data Structures and Algorithms

CS506 and CS205

Lab Assignment 04

[Due Date: 14th October 2024]

Motivation:

This assignment has been designed to implement advanced data structure problems in hand.

Algorithms Need to Implement:

1. Topological Sort
2. Cycle finding Algorithm using DFS
3. Minimum Spanning Tree Algorithms for undirected graphs
   a. Prim's Algorithm
   b. Kruskal's Algorithm

Problem Statement and Formulation:

For each problem, you will be given an adjacency matrix A of n x n (n will be input). Where A[i][j] will represent the status of the connection between the i^{th} and j^{th} nodes.

In the case of Topological Sort and DFS, the adjacency matrix will be given for a **DIRECTED GRAPH.** The links between the nodes will be **unweighted** hence the matrix will contain values 0/1.

In the case of algorithms related to MST, the adjacency matrix will be given for an **UNDIRECTED GRAPH.** And all the links between the nodes will have some **weightage (cost)**. Hence, the adjacency matrix will contain any real value (even fractional and negative).

For the Topological sort, you need to produce the topological sorted sequence of the nodes only.

For Cycle detection problems using DFS, you need to answer the following:

1. Does there exist a cycle or not? => Yes or No
2. Number of all the back edges, forward edges, cross edges, and tree edges.
3. Show the start (entry) and end (exit) time of each node.

For MST algorithms, you need to produce an adjacency matrix representing the MST of the given undirected graph.

General Note:

1. **You may change the Adjacency Matrix to any other format in between the program (ex: Adjacency list). But input must be an adjacency matrix only.**
2. **For all the problems, the answer will be unique (hence we will not give any input that will have more than one answer).**
3. **Caution:** Please maintain the correct order of each value. It's your responsibility to correctly print each value.
4. **For DFS, if there are multiple choices to visit next, the lowest index numbered node will have higher preference.**

Input Format:

First Line will be the algorithm you want to execute:
(1. Topological Sort 2. DFS 3. Prim’s 4. Kruskal) **[Ordering must be matched]**

The Second Line will be the number of Test Cases

The third line will be the number of vertices (N)

After that, there will be N lines representing each row of a matrix. For each line, there will be N values (with single space separated) representing the columns of that particular row.

Sample Input:

- **Topological Sort**

```
1                       #Selecting Topological Sort
3                       #Number of Test Cases
4
0 1 0 0
0 0 0 1
0 1 0 1
0 0 0 0
3
0 1 0
0 0 1
0 0 0
6
0 1 1 0 0 0
0 0 0 1 0 0
0 0 0 1 1 0
0 0 0 0 0 1
0 0 0 0 0 1
0 0 0 0 0 0
```

- **DFS**

```
2                 #Selecting Cycle Detection using DFS
2                 #Number of Test Cases
4
0 1 0 0
0 0 1 0
0 0 0 1
0 0 0 0
4
0 1 0 0
0 0 1 0
0 0 0 1
1 0 0 0
```

- **Prim's Algorithm**

```
3                         #Selecting Prim's Algorithm
3                         #Number of Test Case
4
0  10  6  0
10  0  5  15
6  5  0  4
0  15  4  0
5
0  8  0  0  7
8  0  5  3  0
0  5  0  4  0
0  3  4  0  2
7  0  0  2  0
9
0  4  0  0  7  0  0  0  0
4  0  8  0  0  0  0  0  0
0  8  0  1  0  5  0  0  0
0  0  1  0  9  0  2  0  0
7  0  0  9  0  0  0  4  0
```

```
0 0 5 0 0 0 6 0 0
0 0 0 2 0 6 0 1 8
0 0 0 0 4 0 1 0 3
0 0 0 0 0 0 8 3 0
```

➢ **Kruskal's Algorithm**

```
4                               #Selecting Kruskal's Algorithm
3                               #Number of Test Cases
4
0 10 6 0
10 0 5 15
6 5 0 4
0 15 4 0
5
0 8 0 0 7
8 0 5 3 0
0 5 0 4 0
0 3 4 0 2
7 0 0 2 0
9
0 4 0 0 7 0 0 0 0
4 0 8 0 0 0 0 0 0
0 8 0 1 0 5 0 0 0
0 0 1 0 9 0 2 0 0
7 0 0 9 0 0 0 4 0
0 0 5 0 0 0 6 0 0
0 0 0 2 0 6 0 1 8
0 0 0 0 4 0 1 0 3
0 0 0 0 0 0 8 3 0
```

Sample Output:

- **Topological Sort**

```
2 0 1 3
0 1 2
0 2 4 1 3 5
```

- **DFS**

```
No
0 0 0 3
1 2 3 4
8 7 6 5
Yes
1 0 0 3
1 2 3 4
8 7 6 5
```

#Output Format
#Yes/No
#No_Back_Edges No_Forward_Edges No_Cross_Edges No_Tree_Edges
#Entry_Time_of_Node_1 Entry_Time_of_Node_2 … Entry_Time_of_Node_N
#Exit_Time_of_Node_1 Exit_Time_of_Node_2 … Exit_Time_of_Node_N

- **Prim's Algorithm**

```
0  0  6  0
0  0  5  0
6  5  0  4
0  0  4  0
0  0  0  0  7
0  0  0  3  0
0  0  0  4  0
0  3  4  0  2
```

```
7 0 0 2 0
0 4 0 0 7 0 0 0 0
4 0 0 0 0 0 0 0 0
0 0 0 1 0 5 0 0 0
0 0 1 0 0 0 2 0 0
7 0 0 0 0 0 0 4 0
0 0 5 0 0 0 0 0 0
0 0 0 2 0 0 0 1 0
0 0 0 0 4 0 1 0 3
0 0 0 0 0 0 0 3 0
```

➢ Kruskal's Algorithm

```
0 0 6 0
0 0 5 0
6 5 0 4
0 0 4 0
0 0 0 0 7
0 0 0 3 0
0 0 0 4 0
0 3 4 0 2
7 0 0 2 0
0 4 0 0 7 0 0 0 0
4 0 0 0 0 0 0 0 0
0 0 0 1 0 5 0 0 0
0 0 1 0 0 0 2 0 0
7 0 0 0 0 0 0 4 0
0 0 5 0 0 0 0 0 0
0 0 0 2 0 0 0 1 0
0 0 0 0 4 0 1 0 3
0 0 0 0 0 0 0 3 0
```

Programming Languages Allowed:
C/C++
(You may use STL libraries for stack, queues, list, vector etc.)
(But using any library/function which directly solves the problem itself, is not allowed)

Expected File(s):
One single file of .c / .cpp

Naming Convention of File

FirstName_EntryNo_CS506_Lab04_Prog.c (or .cpp)
**Students of CS205 will write CS205 instead of CS506*

Example:-
Rejoy_2023CSM1011_CS506_Lab04_Prog.c (or .cpp)

Instructions for Programming:

- Don't put unnecessary pieces of information or codes.
- Good documentation in the codes is highly appreciated. (You may use comments)
- Please remember to make modular codes.

Plagiarism Checking

Do not copy from each other. Plagiarism will be checked. After a certain threshold, a high penalty or even an F grade can be imposed, depending on the plagiarism rate.

Feel Free To Contact The TAs:

Rejoy Chakraborty
Madhav Mishra
Shradha Sharma