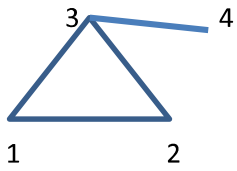
HW-7 CSL2020 (Feb 2021)

Assigned: 03 Feb 2021 Submission deadline: 14 Feb 2021 (Sunday) 9 AM Marks: 9

Submit three C source files with names: hw7_q1_roll_number.c , hw7_q2_roll_number.c, and hw7_q3_roll_number.c as mentioned below.

Helper code: (being written here since some of you may not know file handling in C)

Create a file input.txt with the following data: (It shows the adjacency matrix of an undirected graph G).
The graph can be represented as



Note: If the matrix represents weights then the entries can be any **integer** values. The (i,j)_th entry in the matrix represents if i_th vertex is connected to the j_th vertex (or its weight).

If the matrix has (i,j)_th entry as 2 but (j,i)_th entry as 0 then it is denoting a directed graph where there is an edge (i,j) with weight 2 but the edge (j,i) is absent. An undirected graph's matrix M will always be symmetric.

```
4
0 1 1 0
1 0 1 0
1 1 0 1
0 0 1 0
```

Create a C file with the following helper code:

```
#include <stdio.h>

int main(){
        int i,j, n, **M;
        FILE *fp;
        fp = fopen("r", "input.txt");

        fscanf(fp, "%d", &n);   /* assuming that n is a positive integer. */

        M= (int **) malloc(sizeof(int *)*n);
        for(i=0; i<n; i++)
                M[i] = (int*) malloc(sizeof(int)*n);
        for(i=0; i<n; i++)
          for(j=0; j<n; j++)
              fscanf(fp, "%d", &M[i][j]);

        /*  the above code reads the data from the file into the square matrix M    */

        .....    /* Your code will come here */

        // free the memory allocated by malloc yourself
        fclose(fp);
        return 0;
}
```

The above code will read the adjacency matrix given in the input file. Size and input may change.

1. For a given **directed** unweighted graph (which is described by an adjacency matrix given in input.txt), perform DFS. Whenever there is a choice of vertices, pick the smaller numbered vertex. If the input matrix is of size n x n then assume that the vertices are numbered 1, 2, … n.

Input: Input matrix as defined earlier.

Output: First print the vertex numbers followed by their discovery time and finish time. Assume that the time when the first vertex is discovered is 1, and the time is incremented by 1 at every step as discussed in the class. Then print an n x n matrix, which should mimic the adjacency matrix. Wherever there was no edge between vertex i and vertex j (i.e. when (i,j)_th entry of M is zero), print 0. For the rest of the edges, print their type. Print T for tree edge, F for forward edge, B for back edge and C for cross edge. The output should look like what is given below. (Note: the matrix shown below may not be meaningful. In fact, it is invalid for the given example (there can only be 3 tree edges for 4 vertices). It is being shown only to present the format of the output).

1 $discovery_1$ $finish_1$
2 $discovery_2$ $finish_2$
…
N $discovery_n$ $finish_n$
0 T 0 F
0 0 T C
0 T 0 B
0 T C 0 [3 Marks]

2. Given the **weight matrix** of an **undirected** graph, implement Kruskal's algorithm to find a Minimum Spanning Tree.

Note that this will require you to implement a function to test if a specific edge is "safe" or not. That is, you will need to check if the specific edge is creating a cycle if added to the set A (as discussed in the class).

Input: A weight matrix of size n x n as explained on page 1.

Output:

The weight of the MST you found. [3 Marks]

3. For a given **weighted undirected graph** specified by it weight matrix, and a given source vertex i find the shortest distance of all the vertices from the source vertex using Dijkstra's algorithm.

Input: The first line of the input file will have two entries n and S with space between them, where n will denote the number of vertices (labelled 1 to n) and S will denote the source vertex. S will be an integer from among these n numbers. Rest of the input file will contain the weight matrix in the same way as explained for input.txt in question 1.

Output: You should print n distances corresponding to the shortest path length from the source S, in a sequence, separated by a space character.

That is, you should use a loop to print the output as follows:

```
for(i=1; i<=n; i++)
        printf("%d ", distance[i]);
```

where the array distance[.] is storing distances of vertices from the source vertex.

One of these entries will certainly be 0. And some entries may be UNDEFINED, if the vertex is unreachable from the source vertex. To handle such UNDEFINED entries, we assume that the distances will all be positive integers only, and hence you can take UNDEFINED to be an invalid value -1.

[3 Marks]

NOTE:

The checking of this assignment will be fully automatic. Hence, no consideration of any kind will be permissible later. The deadline will also be strict. We will not entertain any request for delayed submissions for any reason. We will need time to complete the grading hence the above restrictions.