Assignment 3


Submission: https://submit.ceng.metu.edu.tr

Important Note: Obey the specifications strictly. Failing to do so, you will lose significantly much points.

You know that both depth-first search and breath-first search can be implemented using a priority-first search. With the appropriate choice of priorities, the priority-first search forest of a graph becomes a breath-first forest or a depth-first forest.

In this third homework, you are required to implement a priority-first search algorithm on graphs. And by assigning appropriate priorities to nodes, print the breath-first and depth-first search trees (or forests) of a given graph. After that you are also going to print the cycles of the graph if there are any. Also note that you are required to implement the priority queue effectively (it means heap :).

Input to the algorithm is an adjacency list representation of a directed graph. Nodes are named with integers. An example graph G(V,E):

```
1 3 4
2 1 3
3 4
4 1 2
```

and V={1, 2, 3, 4} and E={(1,3), (1,4), (2,1), (2,3), (3,4), (4,1), (4,2)}

There are 2 cycles in the graph. You will print the cycles starting from the smallest numbered node. In this case

```
1 3 4 1
1 3 4 2 1
```

One other important thing is that you have to print the trees in Inorder.
Inorder(Leftof(node))
Print(node)
Inorder(Rightof(node))

One other important thing is that, breath-first and depth-first search does not specify which of the adjacent nodes to expand next. You have to expand the node named with the smallest number so that your trees will be the same. You will start your algorithm with node 1 as the start node. And for example in depth-first search when you have to decide which adjacent node to expand, you have to choose node 3 instead of node 4.

Summary:

Input: An adjacency list representation of a graph (you will read from stdin)
Output:
Breath-first search tree printed in inorder form
Depth-first search tree printed in inorder form
Cycles if there are any.