CS-430 Course project

Members:

Name	CWID	HAWK ID
Tamilarasee Sethuraj	**A20553416**	tsethuraj@hawk.iit.edu
Rithish Murugan	**A20554949**	rmurugan@hawk.iit.edu
Aravinth Ananth	**A20537468**	aananth7@hawk.iit.edu

Project Description:

The project implements a faithful iterative DFS with stack in linear time such that

- Each node enters the stack exactly once
- The parent of a node is set once and
- The same DFS forest and node order as the recursive DFS are returned.

You can use any language (e.g., C/C++/JAVA) to implement.

Explanation of Algorithm:

1. We initialize the stack with the root node (s) and iterate through each node to perform the DFS using the while loop.
2. When the stack is not empty, we peek at the top of the stack and validate if the node is visited and if not visited, we mark it as visited and then perform DFS on the node.
3. The node is then added to the preorder list to display the order of nodes in which DFS was done.
4. For every node picked for DFS, all the adjacent nodes are checked if they are visited.

5. If all the nodes are visited, then there is no path to perform DFS further from that node and hence it is popped from stack and added to post order list similar to exiting the recursive call of DFS in the recursive algorithm.
6. If any of the adjacent nodes are unvisited, then we append the node to stack, track the parent in a list and then perform DFS for the last inserted node similar to calling the recursive DFS function in the recursive algorithm.
7. While all nodes are visited and DFS is performed for each of them, the stack becomes empty and the loop is terminated thereby displaying the preorder, post order and DFS forest for a given input graph identical to the recursive algorithm.

Attached are the Python codes and their respective outputs for Recursive and Iterative DFS algorithm. Please review

https://drive.google.com/drive/folders/1TnY-KuahZJFrcLWQLTvjgVd-xHq2-aGy?usp=sharing

Time complexity:

For a given graph of V nodes and E edges, the proposed iterative algorithm visits each node (using while loop) and each edge (for loop) only once. Hence, the time complexity of the Iterative algorithm is linear (V+E).

Code Outputs:

Output 1:

```
46
47  G = IterativeDFS(6)
48
49  G.add_edge( u: 0,  v: 1)
50  G.add_edge( u: 1,  v: 2)
51  G.add_edge( u: 2,  v: 3)
52  G.add_edge( u: 3,  v: 4)
53  G.add_edge( u: 4,  v: 5)
54
55  G.dfs()
56
```

```
Run  Iterative
C:\Users\User\PycharmProjects\TamilPracticeCodes\venv\Scripts\python.exe C:\Users\User\PycharmProjects\TamilPracticeCodes\Iterative.py
Forest 1
Preorder: [0, 1, 2, 3, 4, 5]
Postorder: [5, 4, 3, 2, 1, 0]
DFS Forest: [(0, 1), (1, 2), (2, 3), (3, 4), (4, 5)]

Process finished with exit code 0
```

Output 2:

```
G = IterativeDFS(6)

G.add_edge( u: 0,  v: 5)
G.add_edge( u: 1,  v: 2)
G.add_edge( u: 1,  v: 3)
G.add_edge( u: 2,  v: 4)
G.add_edge( u: 3,  v: 4)
G.add_edge( u: 4,  v: 3)
G.add_edge( u: 4,  v: 1)

G.dfs()
```

```
Run    Iterative

C:\Users\User\PycharmProjects\TamilPracticeCodes\venv\Scripts\python.exe C:\Users\User\PycharmProjects\TamilPracticeCodes\Iterative.py
Forest 1
Preorder: [0, 5]
Postorder: [5, 0]
DFS Forest: [(0, 5)]

Forest 2
Preorder: [1, 2, 4, 3]
Postorder: [3, 4, 2, 1]
DFS Forest: [(1, 2), (2, 4), (4, 3)]
```