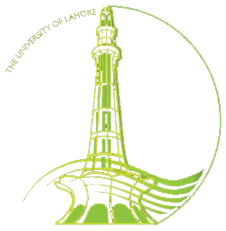

THE UNIVERSITY OF LAHORE
ISLAMABAD CAMPUS

Artificial Intelligence
(CS13217)

Lab Report

Name:	Sameea Naeem
Registration #:	CSU-XS16-139
Lab Report #:	02
Submitted To:	Mr. Usman Ahmed

The University of Lahore, Islamabad Campus
Department of Computer Science & Information Technology

Experiment # 2
Implementing Depth First Search algorithm

Objective
To understand and implement the DFS algorithm.

Software Tool
1. Sub lime text
2. python
3. miktex

1 Theory

The DFS algorithm is a recursive algorithm that uses the idea of backtracking. It involves exhaustive searches of all the nodes by going ahead, if possible, else by backtracking.

Here, the word backtrack means that when you are moving forward and there are no more nodes along the current path, you move backwards on the same path to find nodes to traverse. All the nodes will be visited on the current path till all the unvisited nodes have been traversed after which the next path will be selected.

This recursive nature of DFS can be implemented using stacks. The basic idea is as follows: Pick a starting node and push all its adjacent nodes into a stack. Pop a node from stack to select the next node to visit and push all its adjacent nodes into a stack. Repeat this process until the stack is empty. However, ensure that the nodes that are visited are marked. This will prevent you from visiting the same node more than once. If you do not mark the nodes that are visited and you visit the same node more than once, you may end up in an infinite loop.

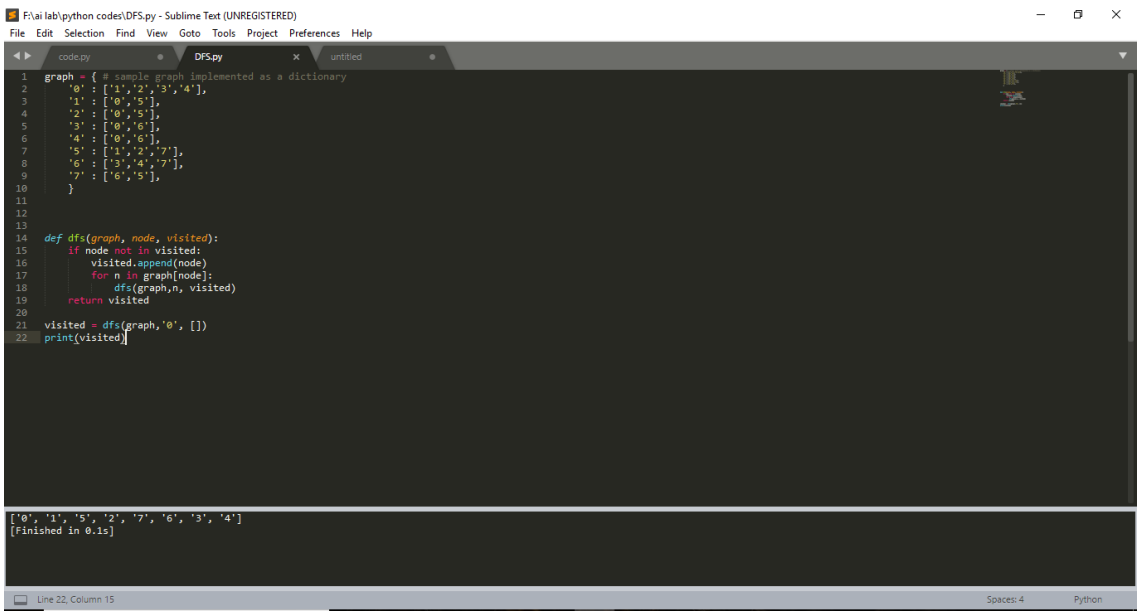

Figure 1: Impplementation of DFS using python

2 Task

Implement DFS on graph shown in figure 2

2.1 Code

```
graph = {
# sample graph implemented as a dictionary
    '0' : ['1','2','3','4'],
    '1' : ['0','5'],
    '2' : ['0','5'],
    '3' : ['0','6'],
    '4' : ['0','6'],
    '5' : ['1','2','7'],
    '6' : ['3','4','7'],
    '7' : ['6','5'],
```

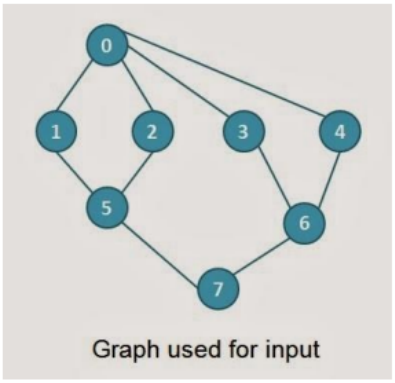

Figure 2: Graph

```
    }
```

```
def dfs(graph, node, visited):
    if node not in visited:
        visited.append(node)
        for n in graph[node]:
            dfs(graph,n, visited)
    return visited

visited = dfs(graph,'0', [])
print(visited)
```

3 Conclusion

['0', '1', '5', '2', '7', '6', '3', '4']