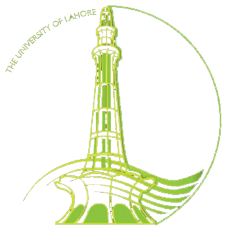


THE UNIVERSITY OF LAHORE
ISLAMABAD CAMPUS

Artificial Intelligence
(CS13217)

Lab Report 3

Name: Kanza Afzal
Registration #: CSU-XS18-132
Lab Report #: 03
Dated: 6-04-2018
Submitted To: Mr. Usman Ahmed

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

Experiment # 3
Implementing Depth First Search Problem

Objective
To understand and implement the Depth First Search

Software Tool
1.
pythaon

1 Theory

Depth-first search (DFS) is an algorithm for traversing or searching tree or graph data structures.It 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.

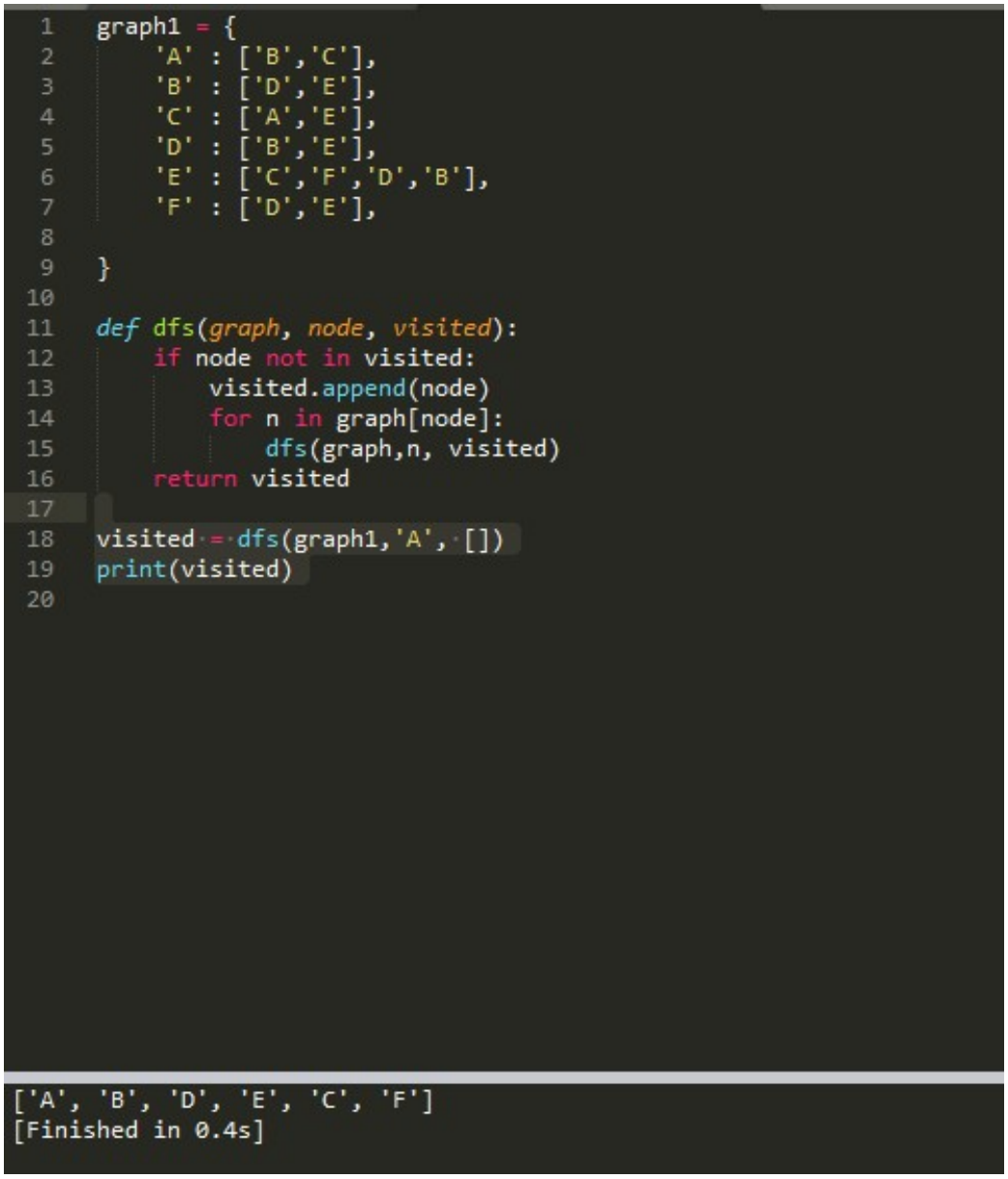
Here, the word backtrack means that when you are moving forward and there are no more nodes along the current path, you move backwards on the same path to find nodes to traverse. All the nodes will be visited on the current path till all the unvisited nodes have been traversed after which the next path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

2 Task

graph1 = 'A' : ['B','C'], 'B' : ['D','E'], 'C' : ['A','E'], 'D' : ['B','E'], 'E' : ['C','F','D','B'], 'F' : ['D','E'],

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(graph1,'A', []) print(visited)



```
graph1 = {
    'A' : ['B','C'],
    'B' : ['D','E'],
    'C' : ['A','E'],
    'D' : ['B','E'],
    'E' : ['C','F','D','B'],
    'F' : ['D','E'],

}

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(graph1,'A', [])
print(visited)
```

```
['A', 'B', 'D', 'E', 'C', 'F']
[Finished in 0.4s]
```

Figure 1: Iplementation of Depth First Search

2.1 Procedure: Task 1

To traverse all the nodes in the graph using dfs technique.

2.2 Procedure: Task 2

3 Conclusion

In depth-first Search the memory requirement is only linear with respect to the search graph. This is in contrast with breadth-first search which requires more space. The reason is that the algorithm only needs to store a stack of nodes on the path from the root to the current node.