AMRITA
VISHWA VIDYAPEETHAM

Design & Analysis Of Algorithm

Lab Experiment -2

NAME : SURIYAPRAKASH.C

ROLL NO:CH.EN.U4CSE20170

SUBJECT: DESIGN & ANALYSIS OF ALGORITHM

SUBJECT CODE: 19CSE302

Submitted to – Mrs. Ashwini,
Department of CSE,
ASE Chennai campus.

EX NO: 2

1) Breadth First Search
2) Depth First Search

AIM:

To write an algorithm to implement depth first search.

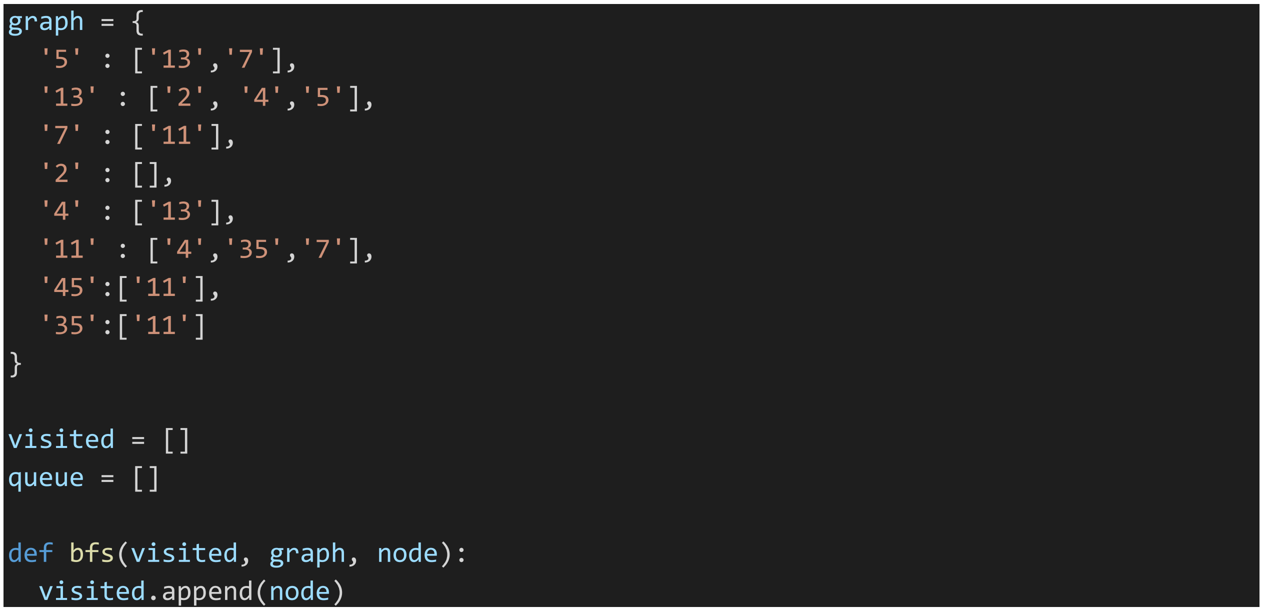
ALGORITHM:

1) SET STATUS = 1 (ready state) for each node in G
2) Push the starting node A on the stack and set its STATUS = 2 (waiting state)
3) Repeat Steps 4 and 5 until STACK is empty
4) Pop the top node N. Process it and set its STATUS = 3 (processed state)
5) Push on the stack all the neighbors of N that are in the ready state (whose STATUS = 1) and set their STATUS = 2 (waiting state)

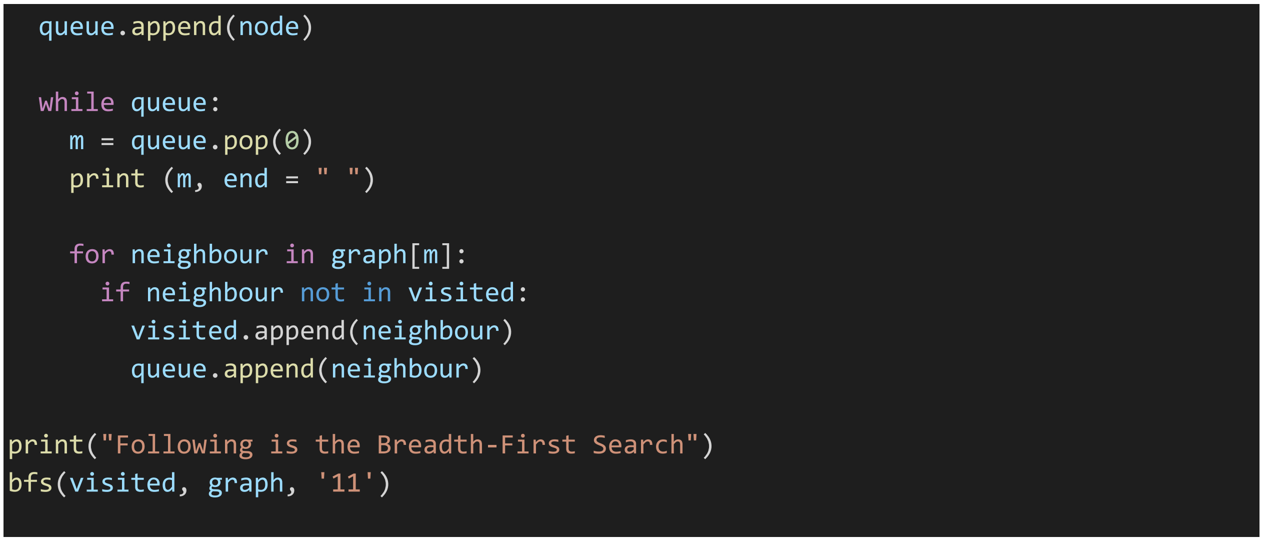
[END OF LOOP]

6) EXIT.

CODE SCREEN:

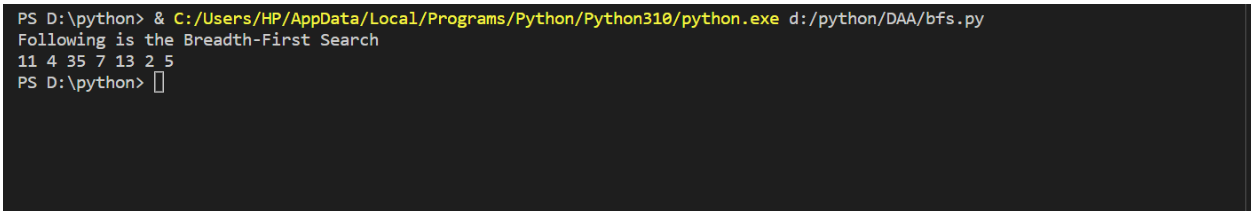


```
graph = {
  '5' : ['13','7'],
  '13' : ['2', '4','5'],
  '7' : ['11'],
  '2' : [],
  '4' : ['13'],
  '11' : ['4','35','7'],
  '45':['11'],
  '35':['11']
}

visited = []
queue = []

def bfs(visited, graph, node):
  visited.append(node)
```



```
  queue.append(node)

  while queue:
    m = queue.pop(0)
    print (m, end = " ")

    for neighbour in graph[m]:
      if neighbour not in visited:
        visited.append(neighbour)
        queue.append(neighbour)

print("Following is the Breadth-First Search")
bfs(visited, graph, '11')
```

OUTPUT SCREEN :



```
PS D:\python> & C:/Users/HP/AppData/Local/Programs/Python/Python310/python.exe d:/python/DAA/bfs.py
Following is the Breadth-First Search
11 4 35 7 13 2 5
PS D:\python> 
```

TIME COMPLEXITY:

When adjacency list is used, time complexity is O(V+E)

When adjacency matrix is used, time complexity is O(V^2)

RESULT:

I have studied and understood the Breadth first search in python language and executed the program successfully.

AIM:

To write an algorithm to implement depth first search algorithm .

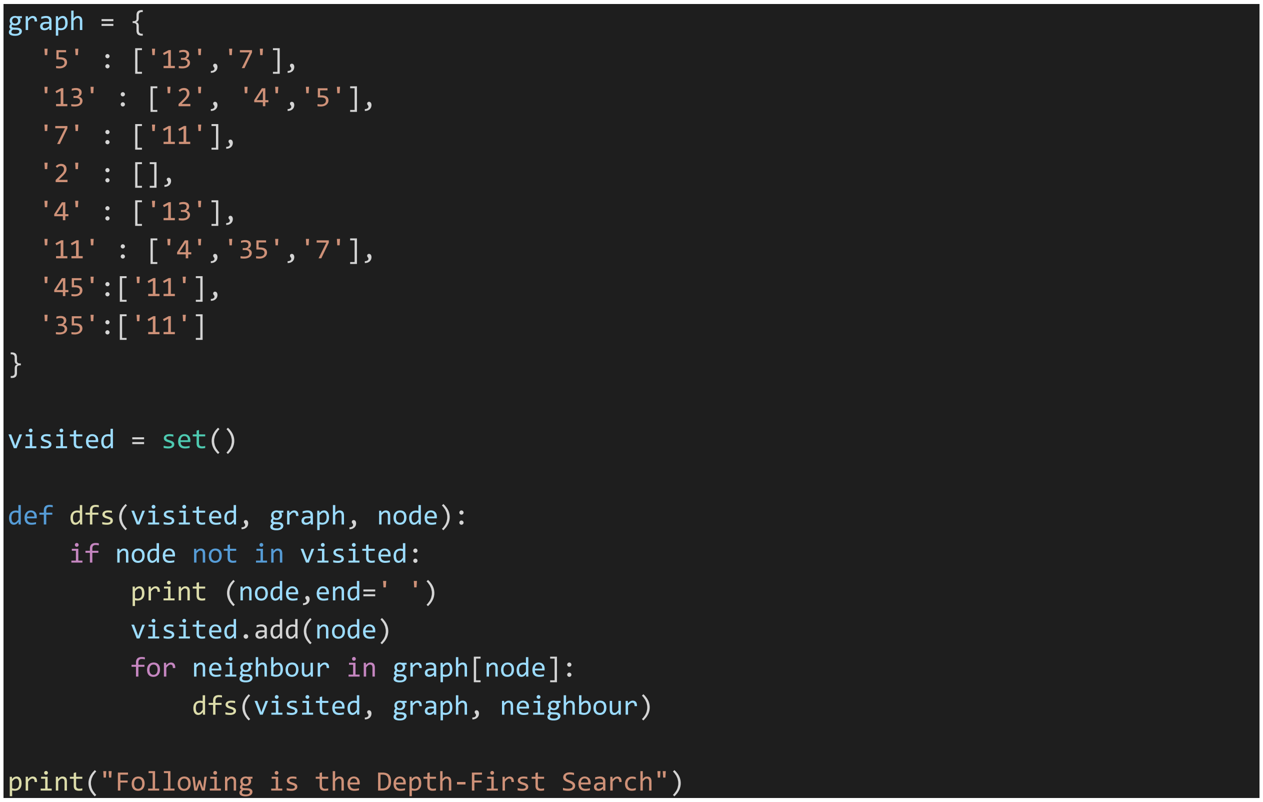
ALGORITHM:

1) SET STATUS = 1 (ready state) for each node in G
2) Push the starting node A on the stack and set its STATUS = 2 (waiting state)
3) Repeat Steps 4 and 5 until STACK is empty
4) Pop the top node N. Process it and set its STATUS = 3 (processed state)
5) Push on the stack all the neighbors of N that are in the ready state (whose STATUS = 1) and set their STATUS = 2 (waiting state)

   [END OF LOOP]

6) EXIT.

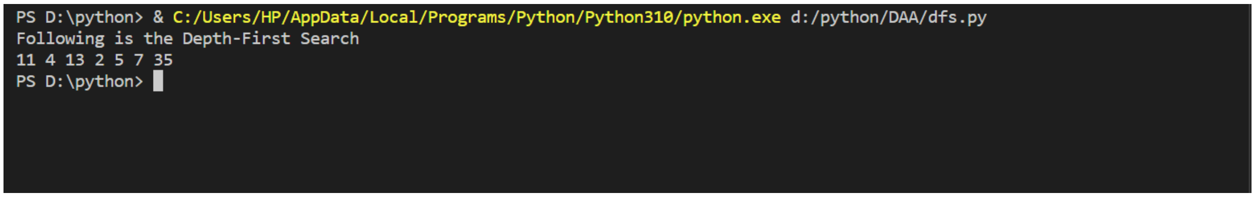
CODE SCREEN:



```python
graph = {
  '5' : ['13','7'],
  '13' : ['2', '4','5'],
  '7' : ['11'],
  '2' : [],
  '4' : ['13'],
  '11' : ['4','35','7'],
  '45':['11'],
  '35':['11']
}

visited = set()

def dfs(visited, graph, node):
    if node not in visited:
        print (node,end=' ')
        visited.add(node)
        for neighbour in graph[node]:
            dfs(visited, graph, neighbour)

print("Following is the Depth-First Search")
```

```
dfs(visited, graph, '11')
```

OUTPUT SCREEN :



```
PS D:\python> & C:/Users/HP/AppData/Local/Programs/Python/Python310/python.exe d:/python/DAA/dfs.py
Following is the Depth-First Search
11 4 13 2 5 7 35
PS D:\python> 
```

TIME COMPLEXITY:

When adjacency list is used, time complexity is O(V+E)

When adjacency matrix is used, time complexity is O(V^2)

RESULT:

I have studied and understood the depth first search in python language and executed the program successfully.

THANK YOU !!