**Aim:** Write a program to implement the following searching techniques: BFS and DFS.

**Code:**

**BFS(complete traversal)**

import collections

graph = {'A': ['B','C'],

'B': ['A','D','E'],

'C': ['F','G','A'],

'D': ['B'],

'E': ['H','B'],

'F': ['C'],

'G': ['C'],

'H': ['E']

}

def bfs(graph, root):

visited, queue = set([root]), collections.deque([root])

while queue:

vertex = queue.popleft()

visit(vertex)

for node in graph[vertex]:

if node not in visited:

visited.add(node)

queue.append(node)

def visit(n): print(n)

bfs(graph, 'A')

**DFS (search)**

import collections

graph = {‘A’: [‘B’,’C’],

‘B’: [‘A’,’D’,’E’],

‘C’: [‘F’,’G’,’A’],

‘D’: [‘B’],

‘E’: [‘H’,’B’],

‘F’: [‘C’],

‘G’: [‘C’],

‘H’: [‘E’]

}

def dfs(graph,start,end,route,list):

route+=[start]

if start == end:

list.extend(route)

else:

for node in graph[start]:

if node not in route:

dfs(graph,node,end,route,list)

def dfs\_route(graph,start,end):

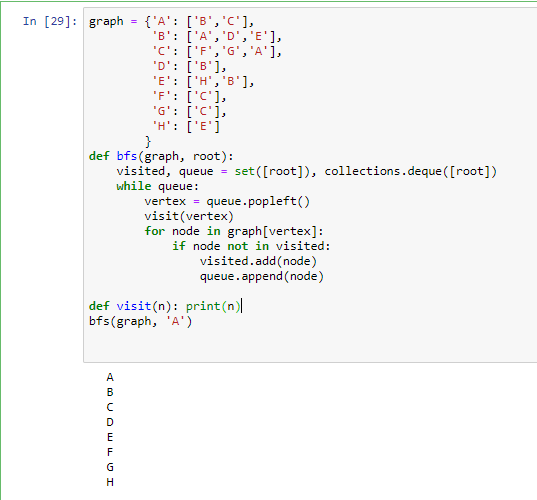
list = []

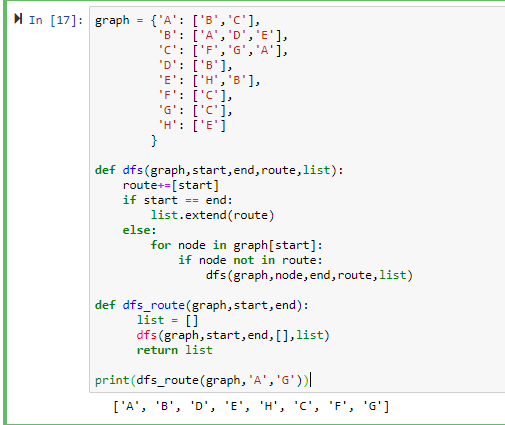
dfs(graph,start,end,[],list)

return list

print(dfs\_route(graph,’A’,’G’))

**Output:**





**DFS(COMPLETE TRAVERSAL)**

graph1 = {

'A' : ['B','S'],

'B' : ['A'],

'C' : ['D','E','F','S'],

'D' : ['C'],

'E' : ['C','H'],

'F' : ['C','G'],

'G' : ['F','S'],

'H' : ['E','G'],

'S' : ['A','C','G']

}

visited = []

def dfs(graph,node):

global visited

if node not in visited:

visited.append(node)

for n in graph[node]:

dfs(graph,n)

dfs(graph1,'A')

print("The traversal order is:" )

print(visited)

**Output**

The traversal order is:

['A', 'B', 'S', 'C', 'D', 'E', 'H', 'G', 'F']

**BFS(Search)**

def bfs(graph,start,search):

explored = []

queue = [start]

found = 1

while found:

node = queue.pop(0)

if(node == search):

found = 0

if node not in explored:

explored.append(node)

neighbours = graph[node]

for neighbour in neighbours:

queue.append(neighbour)

print(explored)

search = int(input("enter the number you want to search-"))

graph = {1: [2, 3, 5],

2: [1,4, 5],

3: [1, 6, 7],

4: [2],

5: [1, 2,4],

6: [3],

7: [3]}

bfs(graph,1,search)

**Output:**



