AI

Practical 1:Graph Traversal

Q1)write a program to implement Depth First search algorithm

#dfs

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

'B':['C','F'],

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

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

'E':['B','F'],

'F':['A'],

'D':['F'],

'H':['A'],'''

graph={

'M':['N','Q','R'],

'N':['O','Q','M'],

'R':['M'],

'O':['P','N'],

'Q':['M','N'],

'P':['O','Q']

}

def dfs(g,n,seen,d):

if n not in seen:

seen.append(n)

for i in g[n]:

if seen [-1] is d:

break

dfs(g,i,seen,d)

return seen

print(dfs(graph,input("Enter destination:"),[],'R'))

o/p:

Enter destination:M

['M', 'N', 'O', 'P', 'Q', 'R']

Q2)WAP to implement Breadth first search algorithm

#bfs

from queue import Queue

adj\_list={

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

'B':['C','F'],

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

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

'E':['B','F'],

'F':['A'],

'D':['F'],

'H':['A'],

}

visited={}

level={}

parent={}

bfs\_traversal=[]

queue=Queue()

for node in adj\_list.keys():

visited[node]=False

parent[node]=None

level[node]=-1

print(visited)

print(level)

print(parent)

source='A'

visited[source]=True

level[source]=0

queue.put(source)

while not queue.empty():

u=queue.get()

bfs\_traversal.append(u)

for v in adj\_list[u]:

if not visited[v]:

visited[v]=True

parent[v]=u

print(parent[v])

level[v]=level[u]+1

queue.put(v)

print("BFS traversal: ",bfs\_traversal)

#print(level["N"])

#print(level["O"])

node="E"

path=[]

while node is not None:

path.append(node)

node=parent[node]

path.reverse()

print("Shortest path is: ",path)

o/p:

BFS traversal: ['A', 'B', 'D', 'C', 'F', 'E', 'G', 'H']

Shortest path is: ['A', 'B', 'C', 'E']