AI PRAC 1 BFS

CODE:

graph={

'5':['3','7'],

'3':['2','4'],

'7':['8'],

'2':[],

'4':['8'],

'8':[],

}

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,'5')

AI PRAC 2 DFS

CODE:

graph={

'5':['3','7'],

'3':['2','4'],

'7':['8'],

'2':[],

'4':['8'],

'8':[],

}

start=input("Enter Starting Node: ")

des=input("Enter Destination Node: ")

def dfs(graph,start,li,des):

if start not in li:

li.append(start)

for i in graph[start]:

if li[-1] is des:

break

dfs(graph,i,li,des)

return li

print("Following is The Depth First Seach:",dfs(graph,start,[],des))

AI PRAC 3 DLS

CODE:

graph={

'5':['3','7'],

'3':['2','4'],

'7':['8'],

'2':[],

'4':['8'],

'8':[],

}

def dls(start,goal,path,level,maxD):

print("Current Level is ",level)

print("Goal Testing for Node: ",start)

path.append(start)

if start==goal:

print("Goal Node Found!")

return path

else:

print("Goal Node Test Failed")

if level==maxD:

print("Maximum Level Reached!")

else:

print("Expanding the Current Node")

for child in graph[start]:

if dls(child,goal,path,level+1,maxD):

return path

else:

path.pop()

return False

start='5'

goal=input("Enter Goal Node: ")

path=list()

result=dls(start,goal,path,0,int(input("Enter Maximum Depth: ")))

if (result):

print("Path Founded till the Goal Node as Follows: ")

print("Path: ",path)

else:

print("Goal Node Not Founded Within the Specified Limit!")

AI PRAC 4: IDDFS

CODE:  
graph={

'5':['3','7'],

'3':['2','4'],

'7':['8'],

'2':[],

'4':['8'],

'8':[],

}

def iddfs(graph,n,seen,dst,dep,lim):

if n not in seen:

seen.append(n)

if dep<=lim:

for i in graph[n]:

if seen[-1] is dst:

return seen

iddfs(graph,i,seen,dst,dep+1,lim)

else:

print("Maximum Level Reached")

return None

print(iddfs(graph,'5',[],'8',0,int(input("Enter Maximum Limit:"))))

AI PRAC 5: Recursive Best First Search (RBFS)

NOTE: Give Starting Node as A and Destination Node as D while Running The Code

CODE:

graph={

'A':({'B':1,'C':3,'D':7},12),

'B':({'D':5},1),

'C':({'D':12},3),

'D':({'A':3},0),

}

def greedy\_search\_rec(graph,prev,dst,path,q):

print("Connected Nodes with Current Node",prev,"With H(n) Value:")

for n in graph[prev][0]:

if n not in path:

q[n]=graph[n][1]

print(n,"->",q[n])

while q:

mn=min(q,key=q.get)

print("Taking Minimum H(n) Value: ",mn)

if dst==mn:

return path+[dst]

new\_path=greedy\_search\_rec(graph,mn,dst,path+[mn],q)

if new\_path:

return new\_path

return[]

source=input("Enter Source Node: ")

dst=input("Enter Destination Node: ")

path=greedy\_search\_rec(graph,source,dst,[source],{})

if path:

print(path)

else:

print("Path Not Found")

AI PRAC 6

NOTE: Give Starting Node as B and Destination Node as E and heuristic value as 6 while Running The Code

CODE:

graph={

'A': ({'B':2},{'E':3},11),

'B': ({'C':1},{'G':9},6),

'C': ({'E':2},2),

'E': ({'D':6},0),

'D': ({'G':1},7),

}

def a\_star(graph,prev,dst,path,pcost,q):

print("Connected Nodes with Current Node",prev,"With H(n) Values:")

for n in graph[prev][0]:

if n not in path:

q[n]=(graph[n][1],graph[prev][0][n])

print(n,"->",q[n])

add1=sum(q[n])

path\_cost=add1+pcost

print("A\* Value of ",n,"is: ",path\_cost)

while q:

mn=min(q,key=q.get)

print("Taking Minimum H(n) Value",mn)

if dst==mn:

return path+[dst]

pc=path\_cost+q[mn][1]

new\_path=a\_star(graph,mn,dst,path+[mn],pc,q)

if new\_path:

return new\_path

return []

start=input("Enter Starting Node: ")

dst=input("Enter Destination Node: ")

heuristic=int(input("Enter Heuristic Value For Souce: "))

path=a\_star(graph,start,dst,[],0,{start:(heuristic,0)})

if path:

print(path)

else:

print("Path Not Found")