**Artificial Intelligence Course**

**Assignment 01**



**Dr.Hashim Yasin**

Student Name: Abdul Salam & Taha Raja

Roll No: 18F-0326 & 18F-0137 Section: CS(6B)

Date performed: 1/04/2021

# Question No 01:

# Question No 02:

# 

# Question No 03:

# Question No 04:

Code:

Graph = {  
 "A": [["B",3], ["C",4]],  
 "B": [["A",3], ["E",7], ["H",10], ["I",11]],  
 "C": [["A",4], ["D", 7], ["F",9]],  
 "D": [["C",7], ["E",9]],  
 "E": [["D",9], ["B",7], ["F",11], ["H",13]],  
 "F": [["C",9], ["E",11], ["G",13]],  
 "G": [["F",13], ["H",15], ["K",18]],  
 "H": [["B",10], ["E",13], ["G",15], ["K",19], ["L",20], ["J",18]],  
 "I": [["B",11], ["J",19]],  
 "J": [["H",18], ["I",19]],  
 "K": [["H",19], ["G",18]],  
 "L": [["H",20]]  
}  
  
  
print(Graph)  
  
def minlol(lists):  
 minValue = 999  
 index = 0  
 Mindex = -1  
 while index < len(lists):  
 lol = lists[index]  
 if minValue > lol[1] :  
 minValue = lol[1]  
 Mindex = index  
 index +=1  
 return Mindex  
  
def BFS(Graph,startingN,goalN):  
 stack = []  
 visited = []  
 path = []  
 pathcost = 0  
 elementcount = 1  
 stack.append(startingN)  
  
 while stack:  
 node = stack.pop()  
 visited.append(node)  
  
 print(str(node) + "->", end=" ")  
 if node == goalN:  
 print("Goal Achived")  
 for pathv in path:  
 pathcost = pathv[1] + pathcost  
 print("Total path cost : " + str(pathcost))  
 print("Total Element Inseted in Stack : " + str(elementcount))  
 break  
  
 for conectedNode in Graph[node]:  
 if conectedNode[0] not in visited and conectedNode[0] not in stack:  
 stack.append(conectedNode[0])  
 elementcount +=1  
 path.append(conectedNode)  
  
  
def DFS(Graph,startingN,goalN):  
 queue = []  
 visited = []  
 path = []  
 pathcost = 0  
 elementcount = 1  
  
 queue.append(startingN)  
  
 while queue:  
 node = queue.pop(0)  
 visited.append(node)  
  
 print(str(node) + "->", end=" ")  
 if node == goalN:  
 print("Goal Achived")  
 for pathv in path:  
 pathcost = pathv[1] + pathcost  
 print("Total path cost : " + str(pathcost))  
 print("Total element inserted in queue : " + str(elementcount))  
 break  
  
 for conectedNode in Graph[node]:  
 if conectedNode[0] not in visited and conectedNode[0] not in queue:  
 queue.append(conectedNode[0])  
 elementcount +=1  
 path.append(conectedNode)  
  
  
  
  
def uniformsearch(Graph,startingN,goalN):  
 listN = []  
 backtrackStack = []  
 visited = []  
 path = []  
 elementcount = 1  
  
 path.append([startingN,0])  
 pathcost = 0  
  
 backtrackStack.append(startingN)  
 while backtrackStack:  
 listN.clear()  
 Node = backtrackStack[-1]  
 visited.append(Node)  
 print(Node + "->", end=" ")  
 if Node == goalN:  
 print("Goal Achived")  
 for pathv in path:  
 pathcost = pathv[1] + pathcost  
 print("Total path cost : " + str(pathcost))  
 print("Total element in inserted in stack : " + str(elementcount))  
 break  
  
 for conectedNode in Graph[Node]:  
 if conectedNode[0] not in visited and conectedNode[0] not in backtrackStack:  
 listN.append(conectedNode)  
 indexo = minlol(listN)  
 if indexo != -1:  
 adjN = listN[indexo]  
 backtrackStack.append(adjN[0])  
 elementcount +=1  
 path.append((listN[indexo]))  
 else:  
 backtrackStack.pop()  
  
  
  
print("\nUniform cost Search :")  
uniformsearch(Graph,"A","G")  
print("\nDfs Search :")  
DFS(Graph,"A","G")  
print("\nBfs Search :")  
BFS(Graph,"A","G")

Screenshot: 