**Name: Om Chandrakant Mahajan Date:21/03/24**

**Practical Name: AO\* Search Batch: B3**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

class Graph:

def \_\_init\_\_(self,graph,heuristicNodeList,startNode):

self.graph = graph

self.H = heuristicNodeList

self.start = startNode

self.parent={}

self.status={}

self.solutionGraph={}

def applyAOStar(self):

self.aoStar(self.start,False)

def getNeighbors(self,v):

return self.graph.get(v,'')

def getStatus(self,v):

return self.status.get(v,0)

def setStatus(self,v,val):

self.status[v]=val

def getHeuristicNodeValue(self,n):

return self.H.get(n,0)

def setHeuristicNodeValue(self,n,value):

self.H[n] = value

def printSolution(self):

print("For graph solution traverse, tee graph form the start node :",self.start)

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

print(self.solutionGraph)

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

def computeMinimumCostChildNodes(self,v):

minimumCost = 0

costToChildNodeListDict = {}

costToChildNodeListDict[minimumCost] = {}

flag = True

for nodeInfoTuplelist in self.getNeighbors(v):

cost = 0

nodeList = []

for c,weight in nodeInfoTuplelist :

cost = cost + self.getHeuristicNodeValue(c)+weight

nodeList.append(c)

if flag == True:

minimumCost = cost

costToChildNodeListDict[minimumCost] = nodeList

flag = False

else:

if minimumCost > cost:

minimumCost = cost

costToChildNodeListDict[minimumCost] = nodeList

return minimumCost,costToChildNodeListDict[minimumCost]

def aoStar(self,v,backTracking):

print("Heuristic values :",self.H)

print("Solution graph :",self.solutionGraph)

print("processing node :",v)

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

if self.getStatus(v) >= 0:

minimumCost,childNodeList = self.computeMinimumCostChildNodes(v)

print(minimumCost,childNodeList)

self.setHeuristicNodeValue(v,minimumCost)

self.setStatus(v,len(childNodeList))

solved = True

for ChildNode in childNodeList:

self.parent[ChildNode] = v

if self.getStatus(ChildNode) != -1:

Solved = solved & False

if solved == True:

self.setStatus(v,-1)

self.solutionGraph[v] = childNodeList

if v != self.start:

self.aoStar(self.parent[v],True)

if backTracking == False:

for childNode in childNodeList:

self.setStatus(childNode,0)

self.aoStar(childNode,False)

print("Graph -1")

h1 = {'A':1,'B':5,'C':2,'D':16,'E':4,'F':3}

graph1 = {

'A':[[('B',1),('C',1)],[('D',1)]],

'B':[[('E',1)],[('F',1)]]

}

G1 = Graph(graph1,h1,'A')

G1.applyAOStar()

G1.printSolution()

**OUTPUT:**

**C:\Users\comp\PycharmProject\AI\_89\_B3\venv\Scripts\python.exe C:/Users/comp/PycharmProject/AI\_89\_B3/AO\_89.py**

**Graph -1**

**Heuristic values : {'A': 1, 'B': 5, 'C': 2, 'D': 16, 'E': 4, 'F': 3}**

**Solution graph : {}**

**processing node : A**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**9 ['B', 'C']**

**Heuristic values : {'A': 9, 'B': 5, 'C': 2, 'D': 16, 'E': 4, 'F': 3}**

**Solution graph : {'A': ['B', 'C']}**

**processing node : B**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**4 ['F']**

**Heuristic values : {'A': 9, 'B': 4, 'C': 2, 'D': 16, 'E': 4, 'F': 3}**

**Solution graph : {'A': ['B', 'C'], 'B': ['F']}**

**processing node : A**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Heuristic values : {'A': 9, 'B': 4, 'C': 2, 'D': 16, 'E': 4, 'F': 3}**

**Solution graph : {'A': ['B', 'C'], 'B': ['F']}**

**processing node : F**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**0 {}**

**Heuristic values : {'A': 9, 'B': 4, 'C': 2, 'D': 16, 'E': 4, 'F': 0}**

**Solution graph : {'A': ['B', 'C'], 'B': ['F'], 'F': {}}**

**processing node : B**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Heuristic values : {'A': 9, 'B': 4, 'C': 2, 'D': 16, 'E': 4, 'F': 0}**

**Solution graph : {'A': ['B', 'C'], 'B': ['F'], 'F': {}}**

**processing node : C**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**0 {}**

**Heuristic values : {'A': 9, 'B': 4, 'C': 0, 'D': 16, 'E': 4, 'F': 0}**

**Solution graph : {'A': ['B', 'C'], 'B': ['F'], 'F': {}, 'C': {}}**

**processing node : A**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**For graph solution traverse, tee graph form the start node : A**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**{'A': ['B', 'C'], 'B': ['F'], 'F': {}, 'C': {}}**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Process finished with exit code 0**