**2.** **Implement AO\* Search algorithm.**

THEORY: **AO\* Algorithm**

AO\* Algorithm basically based on  problem decompositon (Breakdown problem into

small pieces)

When a problem can be divided into a set of sub problems, where each sub problem

can be solved separately and a combination of these will be a solution, AND-OR

graphs or AND - OR trees are used for representing the solution.

The decomposition of the problem or problem reduction generates AND arcs.

**How AO\* works:**

The algorithm always moves towards a lower cost value.

Basically, We will calculate the cost function here (F(n)= G (n) + H (n))

H:  heuristic/ estimated  value of the nodes. and G: actual cost or edge value (here

unit value).

Here we have taken the edges value 1 , meaning we have to focus solely on

the heuristic value.

1. The Purple color values are edge values (here all are same that is one).

2. The Red color values are Heuristic values for nodes.

3. The Green color values are New Heuristic values for nodes.

**PROCEDURE / PROGRAM :**

**class** Graph:

**def** \_\_init\_\_(self, graph, heuristicNodeList, startNode):  *#instantiate graph object with graph topology, heuristic values, start node*

        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 THE GRAPH FROM 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:

                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)

            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)

h1 = {'A': 1, 'B': 6, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 5, 'H': 7, 'I': 7, 'J': 1, 'T': 3}

graph1 = {

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

    'B': [[('G', 1)], [('H', 1)]],

    'C': [[('J', 1)]],

    'D': [[('E', 1), ('F', 1)]],

    'G': [[('I', 1)]]

}

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

G1.applyAOStar()

G1.printSolution()

h2 = {'A': 1, 'B': 6, 'C': 12, 'D': 10, 'E': 4, 'F': 4, 'G': 5, 'H': 7}

graph2 = {

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

    'B': [[('G', 1)], [('H', 1)]],

    'D': [[('E', 1), ('F', 1)]]

}

G2 = Graph(graph2, h2, 'A')

G2.applyAOStar()

G2.printSolution()

**OUTPUT:**

HEURISTIC VALUES : {'A': 1, 'B': 6, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 5, 'H': 7, 'I': 7, 'J': 1, 'T': 3}

SOLUTION GRAPH : {}

PROCESSING NODE : A

-----------------------------------------------------------------------------------------

HEURISTIC VALUES : {'A': 0, 'B': 6, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 5, 'H': 7, 'I': 7, 'J': 1, 'T': 3}

SOLUTION GRAPH : {}

PROCESSING NODE : B

-----------------------------------------------------------------------------------------

HEURISTIC VALUES : {'A': 0, 'B': 0, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 5, 'H': 7, 'I': 7, 'J': 1, 'T': 3}

SOLUTION GRAPH : {}

PROCESSING NODE : A

-----------------------------------------------------------------------------------------

HEURISTIC VALUES : {'A': 0, 'B': 0, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 5, 'H': 7, 'I': 7, 'J': 1, 'T': 3}

SOLUTION GRAPH : {}

PROCESSING NODE : G

-----------------------------------------------------------------------------------------

HEURISTIC VALUES : {'A': 0, 'B': 0, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 0, 'H': 7, 'I': 7, 'J': 1, 'T': 3}

SOLUTION GRAPH : {}

PROCESSING NODE : B

-----------------------------------------------------------------------------------------

HEURISTIC VALUES : {'A': 0, 'B': 0, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 0, 'H': 7, 'I': 7, 'J': 1, 'T': 3}

SOLUTION GRAPH : {}

PROCESSING NODE : A

-----------------------------------------------------------------------------------------

HEURISTIC VALUES : {'A': 0, 'B': 0, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 0, 'H': 7, 'I': 7, 'J': 1, 'T': 3}

SOLUTION GRAPH : {}

PROCESSING NODE : I

-----------------------------------------------------------------------------------------

HEURISTIC VALUES : {'A': 0, 'B': 0, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 0, 'H': 7, 'I': 0, 'J': 1, 'T': 3}

SOLUTION GRAPH : {'I': []}

PROCESSING NODE : G

-----------------------------------------------------------------------------------------

HEURISTIC VALUES : {'A': 0, 'B': 0, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 0, 'H': 7, 'I': 0, 'J': 1, 'T': 3}

SOLUTION GRAPH : {'I': [], 'G': ['I']}

PROCESSING NODE : B

-----------------------------------------------------------------------------------------

HEURISTIC VALUES : {'A': 0, 'B': 0, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 0, 'H': 7, 'I': 0, 'J': 1, 'T': 3}

SOLUTION GRAPH : {'I': [], 'G': ['I'], 'B': ['G']}

PROCESSING NODE : A

-----------------------------------------------------------------------------------------

HEURISTIC VALUES : {'A': 0, 'B': 0, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 0, 'H': 7, 'I': 0, 'J': 1, 'T': 3}

SOLUTION GRAPH : {'I': [], 'G': ['I'], 'B': ['G']}

PROCESSING NODE : C

-----------------------------------------------------------------------------------------

HEURISTIC VALUES : {'A': 0, 'B': 0, 'C': 0, 'D': 12, 'E': 2, 'F': 1, 'G': 0, 'H': 7, 'I': 0, 'J': 1, 'T': 3}

SOLUTION GRAPH : {'I': [], 'G': ['I'], 'B': ['G']}

PROCESSING NODE : A

-----------------------------------------------------------------------------------------

HEURISTIC VALUES : {'A': 0, 'B': 0, 'C': 0, 'D': 12, 'E': 2, 'F': 1, 'G': 0, 'H': 7, 'I': 0, 'J': 1, 'T': 3}

SOLUTION GRAPH : {'I': [], 'G': ['I'], 'B': ['G']}

PROCESSING NODE : J

-----------------------------------------------------------------------------------------

HEURISTIC VALUES : {'A': 0, 'B': 0, 'C': 0, 'D': 12, 'E': 2, 'F': 1, 'G': 0, 'H': 7, 'I': 0, 'J': 0, 'T': 3}

SOLUTION GRAPH : {'I': [], 'G': ['I'], 'B': ['G'], 'J': []}

PROCESSING NODE : C

-----------------------------------------------------------------------------------------

HEURISTIC VALUES : {'A': 0, 'B': 0, 'C': 0, 'D': 12, 'E': 2, 'F': 1, 'G': 0, 'H': 7, 'I': 0, 'J': 0, 'T': 3}

SOLUTION GRAPH : {'I': [], 'G': ['I'], 'B': ['G'], 'J': [], 'C': ['J']}

PROCESSING NODE : A

-----------------------------------------------------------------------------------------

FOR GRAPH SOLUTION, TRAVERSE THE GRAPH FROM THE START NODE: A

------------------------------------------------------------

{'I': [], 'G': ['I'], 'B': ['G'], 'J': [], 'C': ['J'], 'A': ['B', 'C']}

------------------------------------------------------------

HEURISTIC VALUES : {'A': 1, 'B': 6, 'C': 12, 'D': 10, 'E': 4, 'F': 4, 'G': 5, 'H': 7}

SOLUTION GRAPH : {}

PROCESSING NODE : A

-----------------------------------------------------------------------------------------

HEURISTIC VALUES : {'A': 0, 'B': 6, 'C': 12, 'D': 10, 'E': 4, 'F': 4, 'G': 5, 'H': 7}

SOLUTION GRAPH : {}

PROCESSING NODE : B

-----------------------------------------------------------------------------------------

HEURISTIC VALUES : {'A': 0, 'B': 0, 'C': 12, 'D': 10, 'E': 4, 'F': 4, 'G': 5, 'H': 7}

SOLUTION GRAPH : {}

PROCESSING NODE : A

-----------------------------------------------------------------------------------------

HEURISTIC VALUES : {'A': 0, 'B': 0, 'C': 12, 'D': 10, 'E': 4, 'F': 4, 'G': 5, 'H': 7}

SOLUTION GRAPH : {}

PROCESSING NODE : G

-----------------------------------------------------------------------------------------

HEURISTIC VALUES : {'A': 0, 'B': 0, 'C': 12, 'D': 10, 'E': 4, 'F': 4, 'G': 0, 'H': 7}

SOLUTION GRAPH : {'G': []}

PROCESSING NODE : B

-----------------------------------------------------------------------------------------

HEURISTIC VALUES : {'A': 0, 'B': 0, 'C': 12, 'D': 10, 'E': 4, 'F': 4, 'G': 0, 'H': 7}

SOLUTION GRAPH : {'G': [], 'B': ['G']}

PROCESSING NODE : A

-----------------------------------------------------------------------------------------

HEURISTIC VALUES : {'A': 0, 'B': 0, 'C': 12, 'D': 10, 'E': 4, 'F': 4, 'G': 0, 'H': 7}

SOLUTION GRAPH : {'G': [], 'B': ['G']}

PROCESSING NODE : C

-----------------------------------------------------------------------------------------

HEURISTIC VALUES : {'A': 0, 'B': 0, 'C': 0, 'D': 10, 'E': 4, 'F': 4, 'G': 0, 'H': 7}

SOLUTION GRAPH : {'G': [], 'B': ['G'], 'C': []}

PROCESSING NODE : A

-----------------------------------------------------------------------------------------

FOR GRAPH SOLUTION, TRAVERSE THE GRAPH FROM THE START NODE: A

------------------------------------------------------------

{'G': [], 'B': ['G'], 'C': [], 'A': ['B', 'C']}

------------------------------------------------------------