Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)

NAME: **Bhuvi Ghosh**	BATCH: **D22**	SAPID: 60009210191
COURSE CODE: **DJ19DSL502**	YEAR**: 2023-2024**	DATE: **29 / 11 / 2023**
COURSE NAME: **Artificial Intelligence Laboratory**		CLASS: **T.Y.B.Tech (D1)**

Experiment 10

Aim: Implement a plan using AO*.

Theory:

The Depth-first search and Breadth-first search given earlier for OR trees or graphs can be easily adopted by AND-OR graph. The main difference lies in the way termination conditions are determined since all goals following an AND node must be realized; whereas a single goal node following an OR node will do. So for this purpose, we are using AO* algorithm. Like A* algorithm here we will use two arrays and one heuristic function.

OPEN: It contains the nodes that have been traversed but yet not been marked solvable or unsolvable.

CLOSE: It contains the nodes that have already been processed.

AO* Search Algorithm

Step 1: Place the starting node into OPEN.

Step 2: Compute the most promising solution tree say T0.

Step 3: Select a node n that is both on OPEN and a member of T0. Remove it from OPEN and place it in CLOSE

Step 4: If n is the terminal goal node then leveled n as solved and leveled all the ancestors of n as solved. If the starting node is marked as solved then success and exit.

Step 5: If n is not a solvable node, then mark n as unsolvable. If starting node is marked as unsolvable, then return failure and exit.

Step 6: Expand n. Find all its successors and find their h (n) value, push them into OPEN.

Step 7: Return to Step 2.

Step 8: Exit.

Lab Assignment to do:

Consider the use case of a plan to travel from Mumbai to Goa to attend a wedding at Taj Aguada. The plan needs to be decided based on the cost. You can either travel by train or bus or flight and stay in a hotel near or far to the wedding venue. The three options of the venues are Westin, Kennel Worth and Maria Rica hotels. You can choose between a two days package for stay and meal together or separately. Other option for your travel and stay will be a vanity van. There you need to decide if you want to cook or eat outside. Implement AO* to find the most suitable plan in terms of cost.

Output Screenshot:

```
FOR GRAPH SOLUTION, TRAVERSE THE GRAPH FROM THE START NODE: Mumbai
------------------------------------------------------------
{'Bus': [], 'Travel': ['Bus'], 'Stay': [], 'Meal': [], 'Maria Rio': ['Stay', 'Meal']}
------------------------------------------------------------
```

Thus, the most suitable plan would be travelling to Goa by bus and staying in a hotel ‘Maria Rio’ and paying for ‘Stay’ and ‘Meal’ separately.

Code with Output:

In [ ]:

```
class Graph:
    def __init__(self, graph, heuristicNodeList, startNode): #instantiate graph object w
ith graph topology, heuristic values, start node
        self.graph = graph
        self.H=heuristicNodeList
        self.start=startNode
        self.parent={}
        self.status={}
        self.solutionGraph={}

    def applyAOStar(self): # starts a recursive AO* algorithm
        self.aoStar(self.start, False)

    def getNeighbors(self, v): # gets the Neighbors of a given node
        return self.graph.get(v,'')

    def getStatus(self,v): # return the status of a given node
        print("Status= ",self.status)
        return self.status.get(v,0)

    def setStatus(self,v, val): # set the status of a given node
        self.status[v]=val

    def getHeuristicNodeValue(self, n):
        return self.H.get(n,0) # always return the heuristic value of a given node

    def setHeuristicNodeValue(self, n, value):
        self.H[n]=value # set the revised heuristic value of a given node

    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): # Computes the Minimum Cost of child nodes
of a given node v
        minimumCost=0
        costToChildNodeListDict={}
        costToChildNodeListDict[minimumCost]=[]
        flag=True
        for nodeInfoTupleList in self.getNeighbors(v): # iterate over all the set of chi
ld node/s
            cost=0
            nodeList=[]
            for c, weight in nodeInfoTupleList:
                cost=cost+self.getHeuristicNodeValue(c)+weight
                nodeList.append(c)
            if flag==True: # initialize Minimum Cost with the cost of first set of child
node/s
                minimumCost=cost
                costToChildNodeListDict[minimumCost]=nodeList # set the Minimum Cost chi
ld node/s
                flag=False
            else: # checking the Minimum Cost nodes with the current Minimum Cost
                if minimumCost>cost:
                    minimumCost=cost
                    costToChildNodeListDict[minimumCost]=nodeList # set the Minimum Cost
child node/s
        return minimumCost, costToChildNodeListDict[minimumCost] # return Minimum Cost a
nd Minimum Cost child node/s

    def aoStar(self, v, backTracking): # AO* algorithm for a start node and backTracking
status flag
        print("HEURISTIC VALUES :", self.H)
        print("SOLUTION GRAPH :", self.solutionGraph)
        print("PROCESSING NODE :", v)
```

```
        print("------------------------------------------------------------------------
----------------")
        if self.getStatus(v) >= 0: # if status node v >= 0, compute Minimum Cost nodes o
f v
            minimumCost, childNodeList = self.computeMinimumCostChildNodes(v)
            print(minimumCost, childNodeList)
            self.setHeuristicNodeValue(v, minimumCost)
            self.setStatus(v,len(childNodeList))
            solved=True # check the Minimum Cost nodes of v are solved
            for childNode in childNodeList:
                self.parent[childNode]=v
                if self.getStatus(childNode)!=-1:
                    solved=solved & False
            if solved==True: # if the Minimum Cost nodes of v are solved, set the current
node status as solved(-1)
                self.setStatus(v,-1)
                self.solutionGraph[v]=childNodeList # update the solution graph with the
solved nodes which may be a part of solution
            if v!=self.start: # check the current node is the start node for backtrackin
g the current node value
                self.aoStar(self.parent[v], True) # backtracking the current node value
with backtracking status set to true
            if backTracking==False: # check the current call is not for backtracking
                for childNode in childNodeList: # for each Minimum Cost child node
                    self.setStatus(childNode,0) # set the status of child node to 0(need
s exploration)
                    self.aoStar(childNode, False) # Minimum Cost child node is further e
xplored with backtracking status as false
```

In [ ]:

```
# Input to AO Star Search Algorithm
# Implementation of AO Star Search Algorithm
# For simplicity we'll consider heuristic distances given
h1 = {'Mumbai': 0, 'Travel': 0, 'Hotel Book': 0, 'Vanity Van': 0, 'Train': 2200, 'Bus':
2000, 'Air': 7000, 'Maria Rio': 0, 'Western': 0, 'Kennel Worth': 0, 'Stay':0, 'Meal':0,
'Package':0, 'Cook':2000, 'Cost of VV':30000, 'Eat Outside':3000}
graph1 = {
    'Mumbai': [[('Travel', 0), ('Hotel Book', 0)], [('Vanity Van', 100)]],
    'Travel': [[('Train', 4090)], [('Bus', 1180)],[('Air', 4050)]],
    'Hotel Book': [[('Maria Rio', 600)],[('Western', 1340)],[('Kennel Worth', 4030)]],
    'Maria Rio': [[('Stay', 8000), ('Meal', 4000)],[('Package', 40000)]],
    'Western': [[('Stay', 34000), ('Meal', 6000)],[('Package', 14000)]],
    'Kennel Worth': [[('Stay', 20000), ('Meal', 6000)],[('Package', 32000)]],
    'Vanity Van': [[('Cook', 0), ('Cost of VV', 0)],[('Eat Outside', 0), ('Cost of VV',
0)]]
}

G1= Graph(graph1, h1, 'Mumbai')
G1.applyAOStar()
G1.printSolution()
```

```
HEURISTIC VALUES : {'Mumbai': 0, 'Travel': 0, 'Hotel Book': 0, 'Vanity Van': 0, 'Train':
2200, 'Bus': 2000, 'Air': 7000, 'Maria Rio': 0, 'Western': 0, 'Kennel Worth': 0, 'Stay':
0, 'Meal': 0, 'Package': 0, 'Cook': 2000, 'Cost of VV': 30000, 'Eat Outside': 3000}
SOLUTION GRAPH : {}
PROCESSING NODE : Mumbai
------------------------------------------------------------------------------------
--
Status=  {}
0 ['Travel', 'Hotel Book']
Status=  {'Mumbai': 2}
Status=  {'Mumbai': 2}
HEURISTIC VALUES : {'Mumbai': 0, 'Travel': 0, 'Hotel Book': 0, 'Vanity Van': 0, 'Train':
2200, 'Bus': 2000, 'Air': 7000, 'Maria Rio': 0, 'Western': 0, 'Kennel Worth': 0, 'Stay':
0, 'Meal': 0, 'Package': 0, 'Cook': 2000, 'Cost of VV': 30000, 'Eat Outside': 3000}
SOLUTION GRAPH : {}
PROCESSING NODE : Travel
------------------------------------------------------------------------------------
--
Status=  {'Mumbai': 2, 'Travel': 0}
3180 ['Bus']
```

```
Status=  {'Mumbai': 2, 'Travel': 1}
HEURISTIC VALUES : {'Mumbai': 0, 'Travel': 3180, 'Hotel Book': 0, 'Vanity Van': 0, 'Train
': 2200, 'Bus': 2000, 'Air': 7000, 'Maria Rio': 0, 'Western': 0, 'Kennel Worth': 0, 'Stay
': 0, 'Meal': 0, 'Package': 0, 'Cook': 2000, 'Cost of VV': 30000, 'Eat Outside': 3000}
SOLUTION GRAPH : {}
PROCESSING NODE : Mumbai
--------------------------------------------------------------------------------
--
Status=  {'Mumbai': 2, 'Travel': 1}
100 ['Vanity Van']
Status=  {'Mumbai': 1, 'Travel': 1}
HEURISTIC VALUES : {'Mumbai': 100, 'Travel': 3180, 'Hotel Book': 0, 'Vanity Van': 0, 'Tra
in': 2200, 'Bus': 2000, 'Air': 7000, 'Maria Rio': 0, 'Western': 0, 'Kennel Worth': 0, 'St
ay': 0, 'Meal': 0, 'Package': 0, 'Cook': 2000, 'Cost of VV': 30000, 'Eat Outside': 3000}
SOLUTION GRAPH : {}
PROCESSING NODE : Bus
--------------------------------------------------------------------------------
--
Status=  {'Mumbai': 1, 'Travel': 1, 'Bus': 0}
0 []
HEURISTIC VALUES : {'Mumbai': 100, 'Travel': 3180, 'Hotel Book': 0, 'Vanity Van': 0, 'Tra
in': 2200, 'Bus': 0, 'Air': 7000, 'Maria Rio': 0, 'Western': 0, 'Kennel Worth': 0, 'Stay'
: 0, 'Meal': 0, 'Package': 0, 'Cook': 2000, 'Cost of VV': 30000, 'Eat Outside': 3000}
SOLUTION GRAPH : {'Bus': []}
PROCESSING NODE : Travel
--------------------------------------------------------------------------------
--
Status=  {'Mumbai': 1, 'Travel': 1, 'Bus': -1}
1180 ['Bus']
Status=  {'Mumbai': 1, 'Travel': 1, 'Bus': -1}
HEURISTIC VALUES : {'Mumbai': 100, 'Travel': 1180, 'Hotel Book': 0, 'Vanity Van': 0, 'Tra
in': 2200, 'Bus': 0, 'Air': 7000, 'Maria Rio': 0, 'Western': 0, 'Kennel Worth': 0, 'Stay'
: 0, 'Meal': 0, 'Package': 0, 'Cook': 2000, 'Cost of VV': 30000, 'Eat Outside': 3000}
SOLUTION GRAPH : {'Bus': [], 'Travel': ['Bus']}
PROCESSING NODE : Mumbai
--------------------------------------------------------------------------------
--
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1}
100 ['Vanity Van']
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1}
HEURISTIC VALUES : {'Mumbai': 100, 'Travel': 1180, 'Hotel Book': 0, 'Vanity Van': 0, 'Tra
in': 2200, 'Bus': 0, 'Air': 7000, 'Maria Rio': 0, 'Western': 0, 'Kennel Worth': 0, 'Stay'
: 0, 'Meal': 0, 'Package': 0, 'Cook': 2000, 'Cost of VV': 30000, 'Eat Outside': 3000}
SOLUTION GRAPH : {'Bus': [], 'Travel': ['Bus']}
PROCESSING NODE : Hotel Book
--------------------------------------------------------------------------------
--
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 0}
600 ['Maria Rio']
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 1}
HEURISTIC VALUES : {'Mumbai': 100, 'Travel': 1180, 'Hotel Book': 600, 'Vanity Van': 0, 'T
rain': 2200, 'Bus': 0, 'Air': 7000, 'Maria Rio': 0, 'Western': 0, 'Kennel Worth': 0, 'Sta
y': 0, 'Meal': 0, 'Package': 0, 'Cook': 2000, 'Cost of VV': 30000, 'Eat Outside': 3000}
SOLUTION GRAPH : {'Bus': [], 'Travel': ['Bus']}
PROCESSING NODE : Mumbai
--------------------------------------------------------------------------------
--
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 1}
100 ['Vanity Van']
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 1}
HEURISTIC VALUES : {'Mumbai': 100, 'Travel': 1180, 'Hotel Book': 600, 'Vanity Van': 0, 'T
rain': 2200, 'Bus': 0, 'Air': 7000, 'Maria Rio': 0, 'Western': 0, 'Kennel Worth': 0, 'Sta
y': 0, 'Meal': 0, 'Package': 0, 'Cook': 2000, 'Cost of VV': 30000, 'Eat Outside': 3000}
SOLUTION GRAPH : {'Bus': [], 'Travel': ['Bus']}
PROCESSING NODE : Maria Rio
--------------------------------------------------------------------------------
--
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 1, 'Maria Rio': 0}
12000 ['Stay', 'Meal']
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 1, 'Maria Rio': 2}
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 1, 'Maria Rio': 2}
HEURISTIC VALUES : {'Mumbai': 100, 'Travel': 1180, 'Hotel Book': 600, 'Vanity Van': 0, 'T
```

```
rain': 2200, 'Bus': 0, 'Air': 7000, 'Maria Rio': 12000, 'Western': 0, 'Kennel Worth': 0,
'Stay': 0, 'Meal': 0, 'Package': 0, 'Cook': 2000, 'Cost of VV': 30000, 'Eat Outside': 300
0}
SOLUTION GRAPH : {'Bus': [], 'Travel': ['Bus']}
PROCESSING NODE : Hotel Book
--------------------------------------------------------------------------------
--
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 1, 'Maria Rio': 2}
1340 ['Western']
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 1, 'Maria Rio': 2}
HEURISTIC VALUES : {'Mumbai': 100, 'Travel': 1180, 'Hotel Book': 1340, 'Vanity Van': 0, '
Train': 2200, 'Bus': 0, 'Air': 7000, 'Maria Rio': 12000, 'Western': 0, 'Kennel Worth': 0,
'Stay': 0, 'Meal': 0, 'Package': 0, 'Cook': 2000, 'Cost of VV': 30000, 'Eat Outside': 300
0}
SOLUTION GRAPH : {'Bus': [], 'Travel': ['Bus']}
PROCESSING NODE : Mumbai
--------------------------------------------------------------------------------
--
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 1, 'Maria Rio': 2}
100 ['Vanity Van']
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 1, 'Maria Rio': 2}
HEURISTIC VALUES : {'Mumbai': 100, 'Travel': 1180, 'Hotel Book': 1340, 'Vanity Van': 0, '
Train': 2200, 'Bus': 0, 'Air': 7000, 'Maria Rio': 12000, 'Western': 0, 'Kennel Worth': 0,
'Stay': 0, 'Meal': 0, 'Package': 0, 'Cook': 2000, 'Cost of VV': 30000, 'Eat Outside': 300
0}
SOLUTION GRAPH : {'Bus': [], 'Travel': ['Bus']}
PROCESSING NODE : Stay
--------------------------------------------------------------------------------
--
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 1, 'Maria Rio': 2, 'Stay':
0}
0 []
HEURISTIC VALUES : {'Mumbai': 100, 'Travel': 1180, 'Hotel Book': 1340, 'Vanity Van': 0, '
Train': 2200, 'Bus': 0, 'Air': 7000, 'Maria Rio': 12000, 'Western': 0, 'Kennel Worth': 0,
'Stay': 0, 'Meal': 0, 'Package': 0, 'Cook': 2000, 'Cost of VV': 30000, 'Eat Outside': 300
0}
SOLUTION GRAPH : {'Bus': [], 'Travel': ['Bus'], 'Stay': []}
PROCESSING NODE : Maria Rio
--------------------------------------------------------------------------------
--
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 1, 'Maria Rio': 2, 'Stay':
-1}
12000 ['Stay', 'Meal']
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 1, 'Maria Rio': 2, 'Stay':
-1}
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 1, 'Maria Rio': 2, 'Stay':
-1}
HEURISTIC VALUES : {'Mumbai': 100, 'Travel': 1180, 'Hotel Book': 1340, 'Vanity Van': 0, '
Train': 2200, 'Bus': 0, 'Air': 7000, 'Maria Rio': 12000, 'Western': 0, 'Kennel Worth': 0,
'Stay': 0, 'Meal': 0, 'Package': 0, 'Cook': 2000, 'Cost of VV': 30000, 'Eat Outside': 300
0}
SOLUTION GRAPH : {'Bus': [], 'Travel': ['Bus'], 'Stay': []}
PROCESSING NODE : Hotel Book
--------------------------------------------------------------------------------
--
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 1, 'Maria Rio': 2, 'Stay':
-1}
1340 ['Western']
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 1, 'Maria Rio': 2, 'Stay':
-1}
HEURISTIC VALUES : {'Mumbai': 100, 'Travel': 1180, 'Hotel Book': 1340, 'Vanity Van': 0, '
Train': 2200, 'Bus': 0, 'Air': 7000, 'Maria Rio': 12000, 'Western': 0, 'Kennel Worth': 0,
'Stay': 0, 'Meal': 0, 'Package': 0, 'Cook': 2000, 'Cost of VV': 30000, 'Eat Outside': 300
0}
SOLUTION GRAPH : {'Bus': [], 'Travel': ['Bus'], 'Stay': []}
PROCESSING NODE : Mumbai
--------------------------------------------------------------------------------
--
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 1, 'Maria Rio': 2, 'Stay':
-1}
100 ['Vanity Van']
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 1, 'Maria Rio': 2, 'Stay':
```

```
-1}
HEURISTIC VALUES : {'Mumbai': 100, 'Travel': 1180, 'Hotel Book': 1340, 'Vanity Van': 0, '
Train': 2200, 'Bus': 0, 'Air': 7000, 'Maria Rio': 12000, 'Western': 0, 'Kennel Worth': 0,
'Stay': 0, 'Meal': 0, 'Package': 0, 'Cook': 2000, 'Cost of VV': 30000, 'Eat Outside': 300
0}
SOLUTION GRAPH : {'Bus': [], 'Travel': ['Bus'], 'Stay': []}
PROCESSING NODE : Meal
-----------------------------------------------------------------------------------
--
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 1, 'Maria Rio': 2, 'Stay':
-1, 'Meal': 0}
0 []
HEURISTIC VALUES : {'Mumbai': 100, 'Travel': 1180, 'Hotel Book': 1340, 'Vanity Van': 0, '
Train': 2200, 'Bus': 0, 'Air': 7000, 'Maria Rio': 12000, 'Western': 0, 'Kennel Worth': 0,
'Stay': 0, 'Meal': 0, 'Package': 0, 'Cook': 2000, 'Cost of VV': 30000, 'Eat Outside': 300
0}
SOLUTION GRAPH : {'Bus': [], 'Travel': ['Bus'], 'Stay': [], 'Meal': []}
PROCESSING NODE : Maria Rio
-----------------------------------------------------------------------------------
--
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 1, 'Maria Rio': 2, 'Stay':
-1, 'Meal': -1}
12000 ['Stay', 'Meal']
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 1, 'Maria Rio': 2, 'Stay':
-1, 'Meal': -1}
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 1, 'Maria Rio': 2, 'Stay':
-1, 'Meal': -1}
HEURISTIC VALUES : {'Mumbai': 100, 'Travel': 1180, 'Hotel Book': 1340, 'Vanity Van': 0, '
Train': 2200, 'Bus': 0, 'Air': 7000, 'Maria Rio': 12000, 'Western': 0, 'Kennel Worth': 0,
'Stay': 0, 'Meal': 0, 'Package': 0, 'Cook': 2000, 'Cost of VV': 30000, 'Eat Outside': 300
0}
SOLUTION GRAPH : {'Bus': [], 'Travel': ['Bus'], 'Stay': [], 'Meal': [], 'Maria Rio': ['St
ay', 'Meal']}
PROCESSING NODE : Hotel Book
-----------------------------------------------------------------------------------
--
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 1, 'Maria Rio': -1, 'Stay':
-1, 'Meal': -1}
1340 ['Western']
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 1, 'Maria Rio': -1, 'Stay':
-1, 'Meal': -1}
HEURISTIC VALUES : {'Mumbai': 100, 'Travel': 1180, 'Hotel Book': 1340, 'Vanity Van': 0, '
Train': 2200, 'Bus': 0, 'Air': 7000, 'Maria Rio': 12000, 'Western': 0, 'Kennel Worth': 0,
'Stay': 0, 'Meal': 0, 'Package': 0, 'Cook': 2000, 'Cost of VV': 30000, 'Eat Outside': 300
0}
SOLUTION GRAPH : {'Bus': [], 'Travel': ['Bus'], 'Stay': [], 'Meal': [], 'Maria Rio': ['St
ay', 'Meal']}
PROCESSING NODE : Mumbai
-----------------------------------------------------------------------------------
--
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 1, 'Maria Rio': -1, 'Stay':
-1, 'Meal': -1}
100 ['Vanity Van']
Status=  {'Mumbai': 1, 'Travel': -1, 'Bus': -1, 'Hotel Book': 1, 'Maria Rio': -1, 'Stay':
-1, 'Meal': -1}
FOR GRAPH SOLUTION, TRAVERSE THE GRAPH FROM THE START NODE: Mumbai
------------------------------------------------------------
{'Bus': [], 'Travel': ['Bus'], 'Stay': [], 'Meal': [], 'Maria Rio': ['Stay', 'Meal']}
------------------------------------------------------------
```