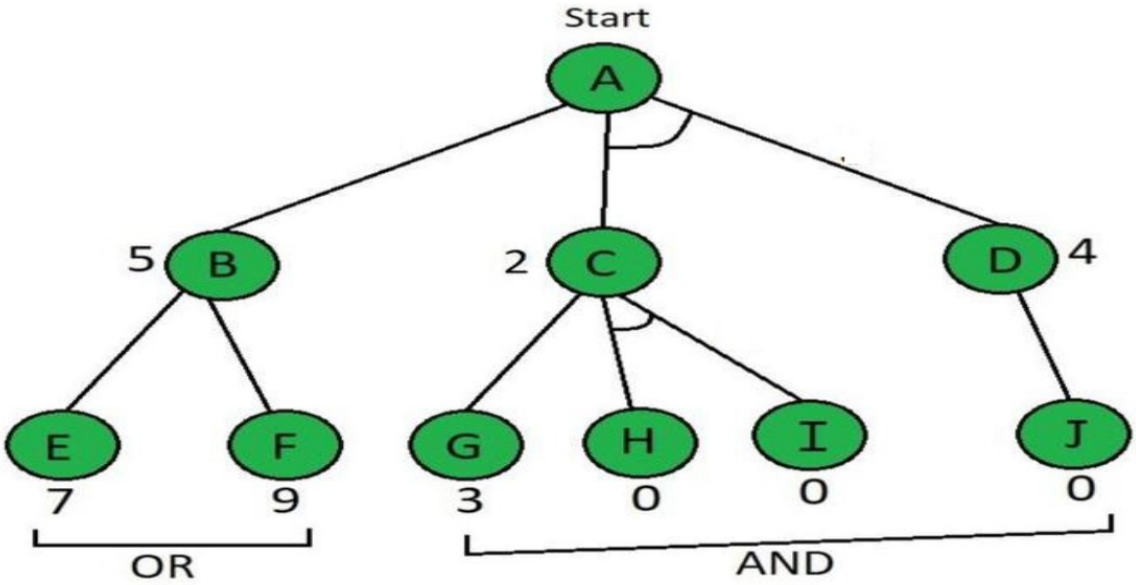
AI LAB | 12 September 2023

Name: Maloth Aditya

Roll No.: 120CS0124

Find the shortest path to node ‘A’ for the following AND-OR graph using the AO* algorithm. Below the Node given is the heuristic value, i.e., h(n). Edge length is considered as 1.



CODE:

```
# 12 September 2023

# Find the shortest path to node 'A' for the given AND-OR graph using the AO* algorithm. Below the Node given is the heuristic value, i.e., h(n).

# Edge length is consideredas 1, i.e., g(n)=1


#minCost returns the minimum cost for the shortest path to node 'A'

minCost = 1000000


def Cost(H, condition, weight = 1):

        cost = {}
```

```
        global minCost
        minCost = 1000000
        if 'AND' in condition:
                AND_nodes = condition['AND']
                Path_A = ' AND '.join(AND_nodes)
                PathA = sum(H[node]+weight for node in AND_nodes)
                cost[Path_A] = PathA
                minCost=min(minCost,PathA)

        if 'OR' in condition:
                OR_nodes = condition['OR']
                Path_B =' OR '.join(OR_nodes)
                PathB = min(H[node]+weight for node in OR_nodes)
                cost[Path_B] = PathB
                minCost=min(minCost,PathB)
        return cost

def updateCost(H, Conditions, weight=1):
        Main_nodes = list(Conditions.keys())
        Main_nodes.reverse()
        least_cost= {}
        for key in Main_nodes:
                condition = Conditions[key]
                print(key,':', Cost(H, condition, weight))
                c = Cost(H, condition, weight)
                H[key] = min(c.values())
                least_cost[key] = Cost(H, condition, weight)
        return least_cost
```

AI LAB | 12 September 2023

```
def shortestPath(Start,Updated_cost, H):
    Path = Start
    if Start in Updated_cost.keys():
        Min_cost = min(Updated_cost[Start].values())
        key = list(Updated_cost[Start].keys())
        values = list(Updated_cost[Start].values())
        Index = values.index(Min_cost)

        # FIND MINIMIMUM PATH KEY
        Next = key[Index].split()
        # ADD TO PATH FOR OR PATH
        if len(Next) == 1:

            Start =Next[0]
            Path += ' <- ' +shortestPath(Start, Updated_cost, H)
        # ADD TO PATH FOR AND PATH
        else:
            Path +=' <- ('+key[Index]+') '

            Start = Next[0]
            Path += '[' +shortestPath(Start, Updated_cost, H) + ' + '

            Start = Next[-1]
            Path += shortestPath(Start, Updated_cost, H) + ']'

    return Path

H = {'A': -1, 'B': 5, 'C': 2, 'D': 4, 'E': 7, 'F': 9, 'G': 3, 'H': 0, 'I':0, 'J':0}
```

```
Conditions = {
'A': {'OR': ['B'], 'AND': ['C', 'D']},
'B': {'OR': ['E', 'F']},
'C': {'OR': ['G'], 'AND': ['H', 'I']},
'D': {'OR': ['J']}
}
# all the edge length are 1
weight = 1
# new cost based on the AO* algorithm
print('New Heuristic:')
UpdatedCost = updateCost(H, Conditions, weight=1)
print()
print()
print('Shortest Path :\n',shortestPath('A', UpdatedCost,H))
print('Cost:', minCost)
```

OUTPUT:

```
[Running] python -u "/home/nit/120CS0124/Sept12/AOStarAlgorithm.py"
New Heuristic:
D : {'J': 1}
C : {'H AND I': 2, 'G': 4}
B : {'E OR F': 8}
A : {'C AND D': 5, 'B': 9}

Shortest Path :
 A <- (C AND D) [C <- (H AND I) [H + I] + D <- J]
Cost: 5

[Done] exited with code=0 in 0.025 seconds
```