**Name: Usman Ul Haq**

**Roll# BSAIM-035**

**AI LAB TASKs**

**LAB TASK 7:**

**Write code for A\* Algorithm**

class Graph:

def \_\_init\_\_(self):

self.graph = {}

def add\_edge(self, node, neighbor, cost):

if node not in self.graph:

self.graph[node] = []

self.graph[node].append((neighbor, cost))

def a\_star(self, start, goal, heuristic):

open\_list = [(0 + heuristic[start], 0, start, [])]

closed\_set = []

while open\_list:

open\_list.sort() # Sort list manually since heapq is not allowed

\_, g, current, path = open\_list.pop(0)

if current in closed\_set:

continue

path.append(current)

if current == goal:

return path, g

closed\_set.append(current)

for neighbor, cost in self.graph.get(current, []):

if neighbor not in closed\_set:

open\_list.append((g + cost + heuristic[neighbor], g + cost, neighbor, path[:]))

return None, float("inf")

graph = Graph()

graph.add\_edge('A', 'B', 1)

graph.add\_edge('A', 'C', 4)

graph.add\_edge('B', 'D', 2)

graph.add\_edge('C', 'D', 1)

graph.add\_edge('C', 'E', 3)

graph.add\_edge('D', 'F', 5)

graph.add\_edge('E', 'F', 2)

heuristic = {'A': 6, 'B': 4, 'C': 4, 'D': 2, 'E': 3, 'F': 0}

start\_node = 'A'

goal\_node = 'F'

path, cost = graph.a\_star(start\_node, goal\_node, heuristic)

if path:

print("Shortest Path:", " → ".join(path))

print("Total Cost:", cost)

else:

print("No path found.")

**Output**

**Shortest Path: A → B → D → F**

**Total Cost: 8**