

**Artificial intelligence (CSC403)**

**Student name: Razan Abdulrahman Alrashed ID: 443500087**

**ASSIGNMENT 5**

from queue import PriorityQueue

class Graph:

    def \_\_init\_\_(self):

        self.graph = {}

        self.heuristics = {}

    def add\_node(self, node):

        if node not in self.graph:

            self.graph[node] = []

    def add\_edge(self, node1, node2, value):

        if node1 in self.graph and node2 in self.graph:

            self.graph[node1].append((node2, value))

            self.graph[node2].append((node1, value))

    def add\_heuristic(self, node, heuristic):

        self.heuristics[node] = heuristic

def A\_Star(graph, start, target):

    visited = set()

    queue = PriorityQueue()

    queue.put((0, start, [start]))  # (cost, node, path)

    while not queue.empty():

        cost, node, path = queue.get()  # get will remove and return from queue

        if node not in visited:

            visited.add(node)

            if node == target:

                return cost, path

            for neighbor, weight in graph.graph[node]:

                if neighbor not in visited:

                    total\_cost = cost + weight + graph.heuristics.get(neighbor, 0)

                    queue.put((total\_cost, neighbor, path + [neighbor]))

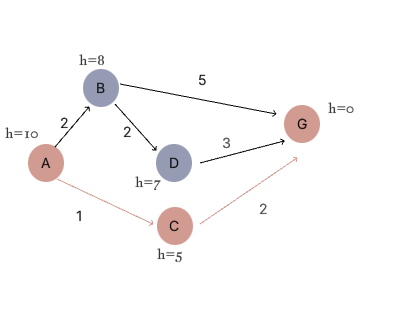
g = Graph()

g.add\_node('A')

g.add\_node('B')

g.add\_node('C')

g.add\_node('D')

g.add\_node('G')

g.add\_edge('A', 'B', 2)

g.add\_edge('A', 'C', 1)

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

g.add\_edge('B', 'G', 5)

g.add\_edge('C', 'G', 2)

g.add\_edge('D', 'G', 3)

g.add\_heuristic('A', 10)

g.add\_heuristic('B', 8)

g.add\_heuristic('C', 5)

g.add\_heuristic('D', 7)

g.add\_heuristic('G', 0)

print(f"PATH: {' -> '.join(path)}")

print(f"Cost from A to G : {cost}")



output