

**Artificial intelligence (CSC403)**

**Student name: Razan Abdulrahman Alrashed**

**ID: 443500087**

**ASSIGNMENT 3**

**Assignment 2**

from queue import PriorityQueue

class Graph:

    def \_\_init\_\_(self):

        self.graph = {}

    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 UCS(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[node]:

                if neighbor not in visited:

                    total\_cost = cost + weight

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

g = Graph()

g.add\_node('A')

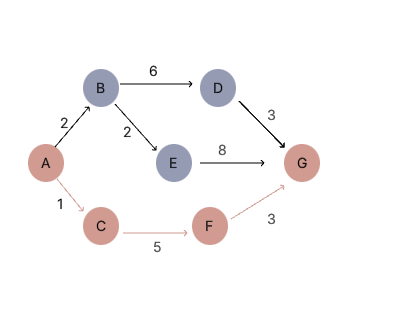
g.add\_node('B')

g.add\_node('C')

g.add\_node('D')

g.add\_node('E')

g.add\_node('F')

g.add\_node('G')

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

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

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

g.add\_edge('B', 'D', 6)

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

g.add\_edge('E', 'G', 8)

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

cost, path = UCS(g.graph, 'A', 'G')

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

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



output

*Output*