1. Single Source Shortest Paths: Dijkstra's Algorithm

Q1:Given a graph represented by an adjacency matrix, implement Dijkstra's Algorithm to find the shortest path from a given source vertex to all other vertices in the graph.The graph is represented as an adjacency matrix where graph[i][j] denote the weight of the edge from vertex i to vertex j. If there is no edge between vertices i and j, the value is Infinity (or a very large number).

Test Case 1:

Input:

n = 5

graph = [[0, 10, 3, Infinity, Infinity],[Infinity, 0, 1, 2, Infinity],[Infinity, 4, 0, 8, 2],

[Infinity, Infinity, Infinity, 0, 7],[Infinity, Infinity, Infinity, 9, 0]]

source = 0

Output:[0, 7, 3, 9, 5]

Test Case 2:

Input:

n = 4

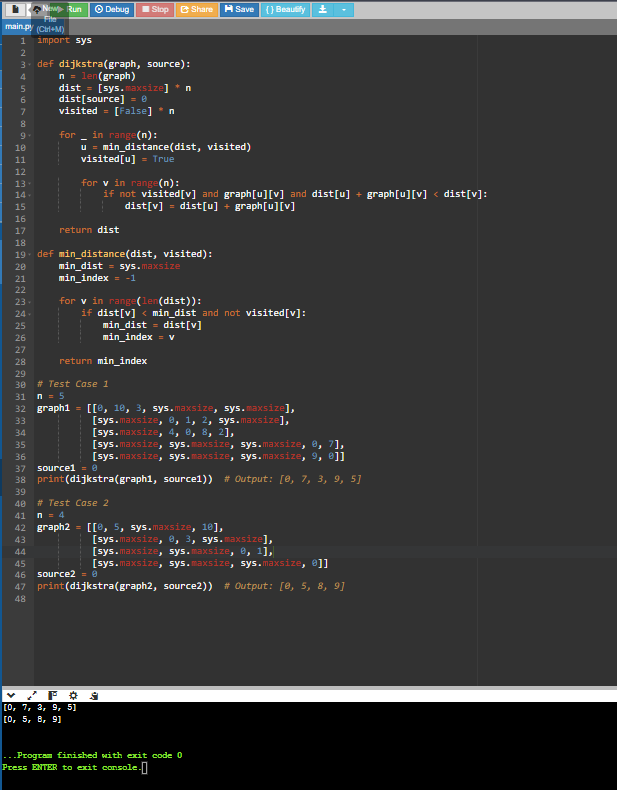
graph = [[0, 5, Infinity, 10],[Infinity, 0, 3, Infinity],[Infinity, Infinity, 0, 1],

[Infinity, Infinity, Infinity, 0]]

source = 0

Output:[0, 5, 8, 9]

OUTPUT:-



Given a graph represented by an edge list, implement Dijkstra's Algorithm to find the shortest path from a given source vertex to a target vertex.The graph is represented as a list of edges where each edge is a tuple (u, v, w) representing an edge from vertex u to vertex v with weight w.

Test Case 1:

Input:n = 6

edges = [(0, 1, 7), (0, 2, 9), (0, 5, 14),(1, 2, 10), (1, 3, 15),

(2, 3, 11), (2, 5, 2),(3, 4, 6),(4, 5, 9)]

source = 0

target = 4

Output:20

Test Case 2:

Input:

n = 5

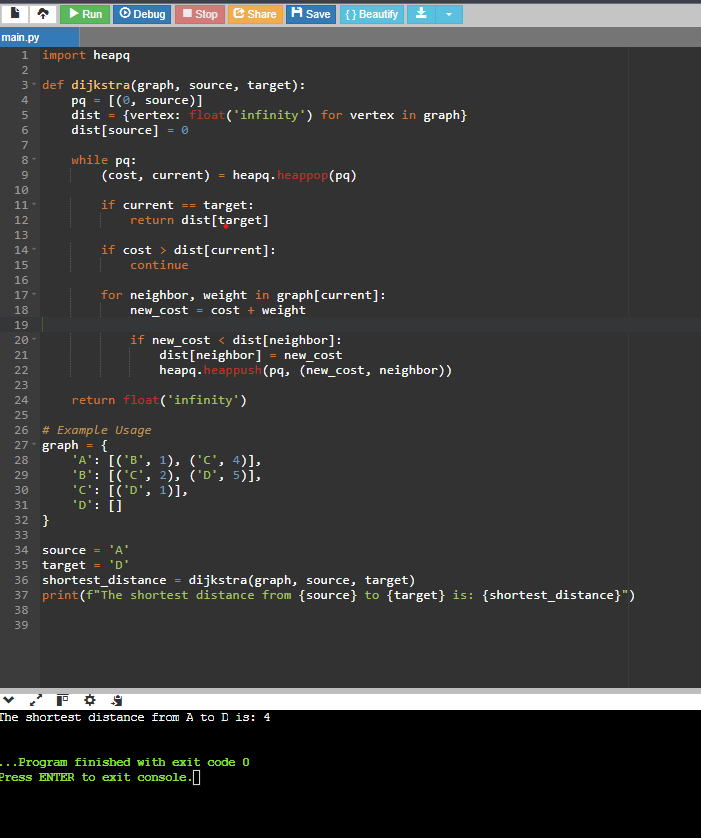
edges = [(0, 1, 10), (0, 4, 3),(1, 2, 2), (1, 4, 4),(2, 3, 9),(3, 2, 7),(4, 1, 1), (4, 2, 8), (4, 3, 2)]

source = 0

target = 3

Output:8

OUTPUT:-



Given a set of characters and their corresponding frequencies, construct the Huffman Tree and generate the Huffman Codes for each character.

Test Case 1:

Input:

n = 4

characters = ['a', 'b', 'c', 'd']

frequencies = [5, 9, 12, 13]

Output:[('a', '110'), ('b', '10'), ('c', '0'), ('d', '111')]

Test Case 2:

Input:

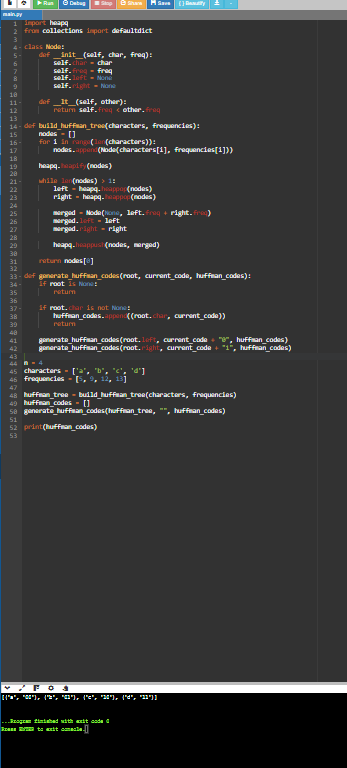
n = 6

characters = ['f', 'e', 'd', 'c', 'b', 'a']

frequencies = [5, 9, 12, 13, 16, 45]

Output:[ ('a', '0'), ('b', '101'), ('c', '100'), ('d', '111'), ('e', '1101'), ('f', '1100')]

OUTPUT:-



Given a Huffman Tree and a Huffman encoded string, decode the string to get the original message.

Test Case 1:

Input:

n = 4

characters = ['a', 'b', 'c', 'd']

frequencies = [5, 9, 12, 13]

encoded\_string = '1101100111110'

Output:"abacd"

Test Case 2:

Input:

n = 6

characters = ['f', 'e', 'd', 'c', 'b', 'a']

frequencies = [5, 9, 12, 13, 16, 45]

encoded\_string = '110011011100101111001011'

Output:"fcbade"

Output:-

