TOPIC 5.5 : DIJKSTRA'S ALGORITHM USING ADJACENCY MATRIX

Problem Statement  
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] denotes 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).

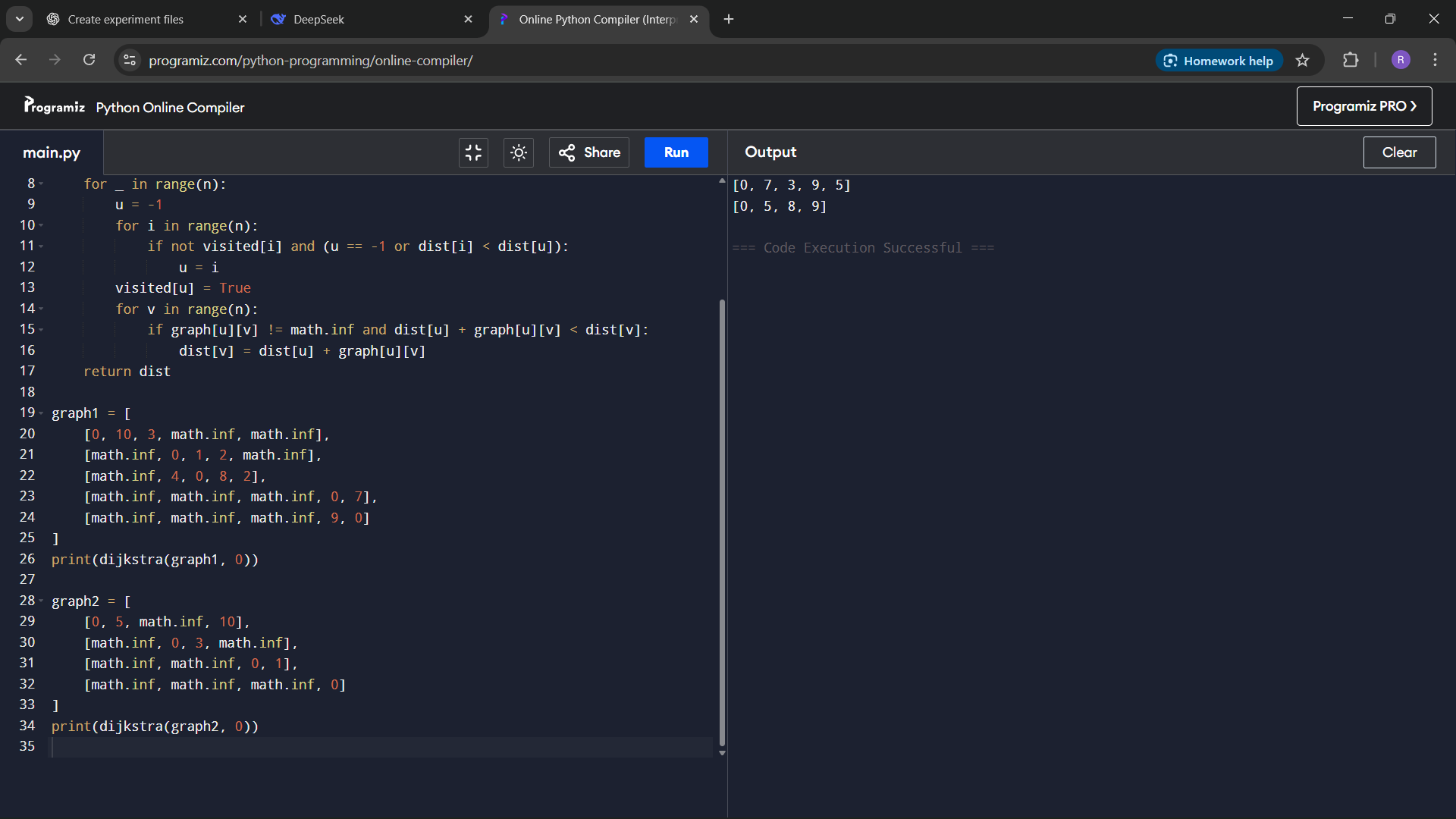
Test Case 1  
Input:  
n = 5  
graph = [[0, 10, 3, Infinity, Infinity],  
[Infinity, 0, 1, 2, Infinity],  
[Infinity, 4, 0, 8, 2],  
[Infinity, Infinity, Infinity, 0, 7],  
[Infinity, Infinity, Infinity, 9, 0]]  
source = 0  
Output: [0, 7, 3, 9, 5]

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]

Aim  
To write a program that finds the shortest distance from a given source vertex to all other vertices using Dijkstra's Algorithm on an adjacency matrix representation of a graph.

Algorithm

1. Start
2. Initialize an array dist[] with Infinity values and set dist[source] = 0
3. Maintain a visited set to track processed vertices
4. Repeat n times:
   * Select the unvisited vertex u with the smallest dist[u]
   * Mark u as visited
   * Update distances of all adjacent vertices v of u:  
     If dist[u] + graph[u][v] < dist[v], update dist[v]
5. After all vertices are processed, dist[] contains the shortest path from source to all vertices
6. Stop

Input and Output  


Result  
The program successfully computes the shortest path distances from the source vertex to all other vertices using Dijkstra's Algorithm.

Performance Analysis  
Time Complexity: O(n^2) for adjacency matrix  
Space Complexity: O(n) for distance and visited arrays