TOPIC 5.6 : DIJKSTRA'S ALGORITHM USING EDGE LIST

Problem Statement  
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.

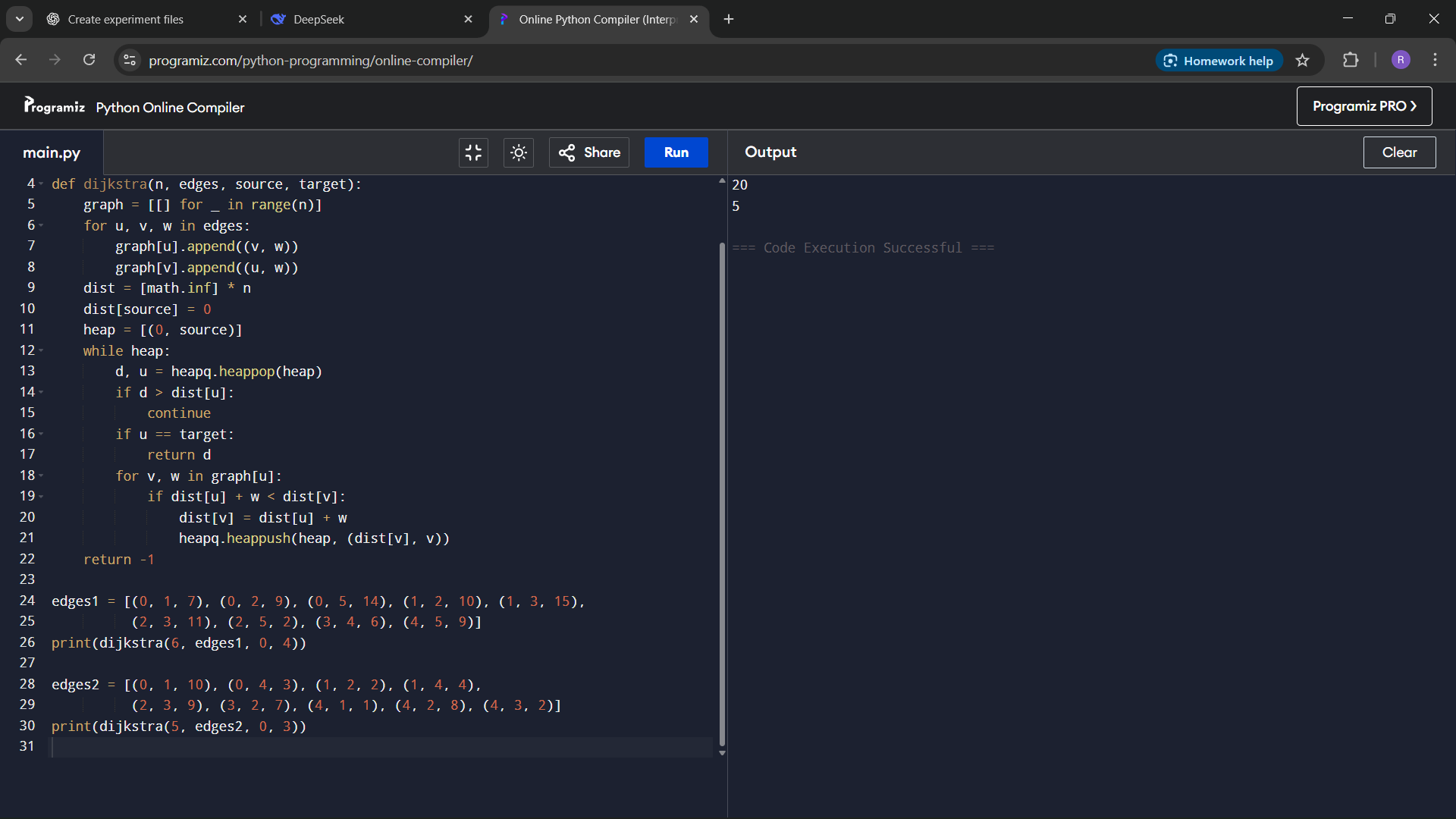
Test Case 1  
Input:  
n = 6  
edges = [(0, 1, 7), (0, 2, 9), (0, 5, 14), (1, 2, 10), (1, 3, 15), (2, 3, 11), (2, 5, 2), (3, 4, 6), (4, 5, 9)]  
source = 0  
target = 4  
Output: 20

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

Aim  
To write a program that finds the shortest path distance from a given source vertex to a target vertex using Dijkstra's Algorithm with an edge list representation of a graph.

Algorithm

1. Start
2. Build an adjacency list from the edge list
3. Initialize distance array with Infinity, set distance[source] = 0
4. Use a priority queue (min-heap) to pick the vertex with the smallest distance
5. For each neighbor v of u with weight w:  
   If dist[u] + w < dist[v], update dist[v] and push it into the heap
6. Repeat until all vertices are processed or target is reached
7. Return dist[target]
8. Stop

Input and Output  


Result  
The program successfully computes the shortest path distance between source and target using Dijkstra's Algorithm on an edge list.

Performance Analysis  
Time Complexity: O((V + E) log V) using a priority queue  
Space Complexity: O(V + E) for adjacency list and heap