Lab Assignment 1: Implementing Dijkstra's Algorithm with a User-Defined Min-Heap.

Objective

This assignment aims to help students understand and implement Dijkstra's shortest path algorithm. Students will also learn how to create and use a custom min-heap data structure to efficiently manage the priority queue operations required by Dijkstra's algorithm.

Instructions

1. Implement a Min-Heap:
   a. Create a user-defined min-heap class that supports the following operations:
      i. `insert(key, value)`: Inserts a new key-value pair into the heap.
      ii. `extract_min()`: Removes and returns the key-value pair with the smallest key.
      iii. `decrease_key(key, new_value)`: Decreases the value associated with a given key.
      iv. `is_empty()`: Checks if the heap is empty.

2. Dijkstra's Algorithm:
   a. Implement Dijkstra's algorithm using your min-heap to find the shortest paths from a source vertex to all other vertices in a given weighted graph. Your algorithm should:
      i. Initialize the distances from the source to all vertices as infinite, except for the source itself (distance 0).
      ii. Use the min-heap to efficiently select the vertex with the smallest known distance.
      iii. Update the distances to the neighboring vertices using the chosen vertex.
3. Input and Output:
   a. Input:
      i. The graph should be represented using an adjacency list.
      ii. The input will consist of the number of vertices, and the number of edges, followed by the edges themselves (each edge specified by a start vertex, an end vertex, and a weight).
   b. Output:
      i. Print the shortest distances from the source vertex to all other vertices.

Example:

Input	Output
5 6 0 1 4 0 2 1 1 3 1 2 1 2 2 3 5 3 4 3	Vertex 0: Distance 0 Vertex 1: Distance 3 Vertex 2: Distance 1 Vertex 3: Distance 4 Vertex 4: Distance 7