1. **Graph Class**
   * **Data Members:**
     + unordered\_map<char, const unordered\_map<char, int>> vertices: Stores the graph as an adjacency list where each vertex points to its neighboring vertices and the respective edge weights.
   * **Methods:**
     + void add\_vertex(char name, const unordered\_map<char, int> &edges): Adds a vertex to the graph with its edges.
     + vector<char> shortest\_path(char start, char finish): Implements Dijkstra's algorithm to find the shortest path from the start vertex to the finish vertex.

**Detailed Explanation**

1. **add\_vertex Method**
   * Adds a new vertex to the vertices map.
   * Each vertex is associated with its neighboring vertices and the corresponding edge weights.
2. **shortest\_path Method**
   * **Initialization:**
     + unordered\_map<char, int> distances: Stores the shortest distance from the start vertex to each vertex.
     + unordered\_map<char, char> previous: Stores the previous vertex in the shortest path for each vertex.
     + vector<char> nodes: Contains all vertices to be processed.
     + vector<char> path: Stores the shortest path from start to finish.
     + A custom comparator lambda function is defined for managing the priority queue.
   * **Setup:**
     + Initializes the distance to the start vertex as 0 and all other distances as infinity.
     + Pushes all vertices into the priority queue (nodes).
   * **Processing:**
     + Extracts the vertex with the smallest distance.
     + Updates the distances to its neighboring vertices if a shorter path is found.
     + Reorganizes the priority queue after each update.
   * **Path Construction:**
     + Once the finish vertex is reached, the path is constructed by backtracking using the previous map.
   * **Return Value:**
     + Returns the shortest path as a vector of characters.
3. **Main Function**
   * Creates a Graph object g.
   * Adds vertices 'A' through 'H' with their respective edges and weights using add\_vertex.
   * Finds and prints the shortest path from vertex 'A' to vertex 'H'.