4. Implement Greedy search algorithm for Dijkstra's Minimal Spanning Tree Algorithm

Java Code :

```
import java.io.*;
import java.lang.*;
import java.util.*;

public class Dijkstras {

        static int minDistance(int[] dist, boolean[] visited) {
               int min = Integer.MAX_VALUE;
               int min_index = -1;

               for (int i = 0; i < dist.length; i++) {
                   if (visited[i] == false && dist[i] <= min) {
                       min = dist[i];
                       min_index = i;
                   }
               }
               return min_index;
          }

       static void dijkstra(int[][] graph, int src, int dest) {
         int n = graph.length;
         int[] dist = new int[n];
         boolean[] visited = new boolean[n];
         HashMap<Integer, ArrayList<Integer>> parent = new HashMap<>();
         List<Integer> path = new ArrayList<>();
         path.add(dest);

         for (int i = 0; i < n; i++) {
             dist[i] = Integer.MAX_VALUE;
             visited[i] = false;
         }

         dist[src] = 0;
         parent.put(src, new ArrayList<>());

         for (int i = 0; i < n - 1; i++) {
             int u = minDistance(dist, visited);
             visited[u] = true;

             for (int v = 0; v < n; v++) {
                 if (
                     !visited[v] && graph[u][v] != 0 &&
                     dist[u] != Integer.MAX_VALUE &&
                     dist[u] + graph[u][v] < dist[v]
                 ) {
                     dist[v] = dist[u] + graph[u][v];

                     if (!parent.containsKey(v))
                         parent.put(v, new ArrayList<Integer>());
                     parent.get(v).add(u);
                 }
             }
         }

         int key = dest;
         while (parent.get(key).size() > 0) {
             int elem = parent.get(key).get(parent.get(key).size()-1);;
             path.add(elem);
```

```
            key = elem;
        }
        Collections.reverse(path);

        System.out.printf("\n\n\nDijkstra Single-Source Shortest
Path::\nPath: %s\nMinimum Cost: %d", path.toString(), dist[dest]);
    }


      public static void main(String ar[])
      {



            int graph[][] = new int[][] {
                  { 0, 4, 0, 0, 0, 0, 0, 8, 0},
                  { 4, 0, 8, 0, 0, 0, 0,11, 0},
                  { 0, 8, 0, 7, 0, 4, 0, 0, 2},
                  { 0, 0, 7, 0, 9,14, 0, 0, 0},
                  { 0, 0, 0, 9, 0,10, 0, 0, 0},
                  { 0, 0, 4,14,10, 0, 2, 0, 0},
                  { 0, 0, 0, 0, 0, 2, 0, 1, 6},
                  { 8,11, 0, 0, 0, 0, 1, 0, 7},
                  { 0, 0, 2, 0, 0, 0, 6, 7, 0}
              };
            Dijkstras obj=new Dijkstras();


            obj.dijkstra(graph, 0, 4);

      }
}
```