```
import java.util.Scanner;

/*
        Time complexity: O(E * log(V))
        Space complexity: O(V^2)

        where E is the number of edges in the graph and
        V is the number of vertices in the graph
 */
public class Solution {

        private static void prims(int[][] adjacencyMatrix) {
                int v = adjacencyMatrix.length;
                boolean visited[] = new boolean[v];
                int weight[] = new int[v];
                int parent[] = new int[v];
                weight[0] = 0;
                parent[0] = -1;
                for (int i = 1; i < v; i++) {
                        weight[i] = Integer.MAX_VALUE;
                }
                for (int i = 0; i < v; i++) {
                        // Pick vertex with min weight
                        int minVertex = findMinVertex(weight, visited);
                        visited[minVertex] = true;
                        // Explore its unvisited neighbors
                        for (int j = 0; j < v; j++) {
                                if (adjacencyMatrix[minVertex][j] != 0 && !visited[j]) {
                                        if (adjacencyMatrix[minVertex][j] < weight[j]) {
                                                weight[j] = adjacencyMatrix[minVertex][j];
                                                parent[j] = minVertex;
                                        }
                                }
                        }
                }

                // Print edges of MST
                for (int i = 1; i < v; i++) {
                        if (parent[i] < i) {
                                System.out.println(parent[i] + " " + i + " " + weight[i]);
                        } else {
                                System.out.println(i + " " + parent[i] + " " + weight[i]);
                        }
                }

        }

        private static int findMinVertex(int[] weight, boolean visited[]) {
```

```
                int minVertex = -1;
                for (int i = 0; i < weight.length; i++) {
                        if (!visited[i] && (minVertex == -1 || weight[i] < weight[minVertex])) {
                                minVertex = i;
                        }
                }
                return minVertex;
        }

        public static void main(String[] args) {

                Scanner s = new Scanner(System.in);
                int v = s.nextInt();
                int e = s.nextInt();
                int adjacencyMatrix[][] = new int[v][v];
                for (int i = 0; i < e; i++) {
                        int v1 = s.nextInt();
                        int v2 = s.nextInt();
                        int weight = s.nextInt();
                        adjacencyMatrix[v1][v2] = weight;
                        adjacencyMatrix[v2][v1] = weight;
                }

                prims(adjacencyMatrix);
        }

}
```