```java
import java.util.Arrays;
import java.util.Scanner;

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

        where E is the number of edges in the graph and
        V is the number of vertices in the graph
 */
class Edge implements Comparable<Edge> {
        int source;
        int dest;
        int weight;

        void printEdge() {
                System.out.println(Math.min(source, dest) + " " + Math.max(source, dest) + " "+weight);
        }

        public int compareTo(Edge e) {
                return this.weight - e.weight;
        }
}

public class Solution {

        static int findParent(int v, int[] parent) {
                if (parent[v] == v) {
                        return v;
                }

                return findParent(parent[v], parent);
        }

        private static void kruskal(Edge[] input, int v, int e) {
                Arrays.sort(input);
                Edge[] output = new Edge[v - 1];
                int[] parent = new int[v];
                for (int i = 0; i < v; i++) {
                        parent[i] = i;
                }

                int count = 0;
                int i = 0;
                while (count != v - 1) {
                        Edge currentEdge = input[i];
                        // Check if we can add the currentEdge in MST or not
                        int sourceParent = findParent(currentEdge.source, parent);
```

```
				int destParent = findParent(currentEdge.dest, parent);

				if (sourceParent != destParent) {
					output[count] = currentEdge;
					count++;
					parent[sourceParent] = destParent;
				}

				i++;
			}

			for (int j = 0; j < v - 1; j++) {
				output[j].printEdge();
			}
		}

		public static void main(String[] args) {

			Scanner sc = new Scanner(System.in);
			int v = sc.nextInt();
			int e = sc.nextInt();
			Edge[] input = new Edge[e];
			for (int i = 0; i < e; i++) {
				input[i] = new Edge();
				input[i].source = sc.nextInt();
				input[i].dest = sc.nextInt();
				input[i].weight = sc.nextInt();
			}
			sc.close();
			kruskal(input, v, e);
		}

	}
```