```java
/**-------------------------------------------
 * -- Estructuras de Datos.  2018/19
 * -- 2º Curso del Grado en Ingeniería [Informática | del Software | de Computadores].
 * -- Escuela Técnica Superior de Ingeniería en Informática. UMA
 * --
 * -- Examen 4 de febrero de 2019
 * --
 * -- ALUMNO/NAME:
 * -- GRADO/STUDIES:
 * -- NÚM. MÁQUINA/MACHINE NUMBER:
 * --
 * -------------------------------------------
 */

import dataStructures.graph.WeightedGraph;
import dataStructures.graph.WeightedGraph.WeightedEdge;

import dataStructures.dictionary.Dictionary;
import dataStructures.dictionary.HashDictionary;
import dataStructures.priorityQueue.PriorityQueue;
import dataStructures.priorityQueue.LinkedPriorityQueue;
import dataStructures.set.Set;
import dataStructures.set.HashSet;

public class Kruskal {
    public static <V,W> Set<WeightedEdge<V,W>> kruskal(WeightedGraph<V,W> g) {
        Set<WeightedEdge<V,W>> sol = new HashSet<>();
        LinkedPriorityQueue<WeightedEdge<V,W>> edges = new LinkedPriorityQueue<>();

        Set<WeightedEdge<V,W>> ed = g.edges();

        for (WeightedEdge<V,W> aux: ed) {
            edges.enqueue(aux);
        }

```

```
36          Dictionary<V,W> dic = new HashDictionary<>();
37
38          for (V v: g.vertices()) {
39              dic.insert(v, (W) v);
40          }
41
42          while (!edges.isEmpty()){
43              WeightedEdge<V, W> arista = edges.first
    ();
44              V src = arista.source();
45              V dst = arista.destination();
46
47              W representante_src = dic.valueOf(src);
48              W representante_dst = dic.valueOf(dst);
49
50              if(!representante_src.equals(
    representante_dst)){
51                  for (V origen: g.vertices()) {
52                      if (dic.valueOf(origen).equals(
    representante_dst)) {
53                          dic.insert(origen,
    representante_src);
54                      }
55                  }
56                  sol.insert(arista);
57              }
58
59              edges.dequeue();
60          }
61
62          return sol;
63      }
64
65      // Sólo para evaluación continua / only for part
    time students
66      public static <V,W> Set<Set<WeightedEdge<V,W>>>
    kruskals(WeightedGraph<V,W> g) {
67
68          // COMPLETAR
69
70          return null;
```

```
71      }
72  }
73
```