**Complejidad computacional:**

La complejidad computacional es O(a.log(a)), ya que el algoritmo utilizado para la resolución del ejercicio es Kruskal, implementado con lista de aristas ordenada por pesos.

**private** **void** **kruskal**() {

**int** **aristas** = 0;

**int** **nodo1** = 0;

**int** **nodo2** = 0;

**int** **peso** = 0;

**int** **k** = 0;

**int** **noUnir** = 0;

**this**.padres = **new** **int**[**this**.cantCiudades];

**for** (**int** **i** = 0; i < **this**.cantCiudades; i++) {

**this**.padres[i] = i;

}

**for** (**int** **i** = 0; i < **this**.aristas.size(); i++) {

nodo1 = **this**.aristas.get(i).getNodo1();

nodo2 = **this**.aristas.get(i).getNodo2();

peso = **this**.aristas.get(i).getCosto();

**System**.***out***.println(nodo1 + " " + nodo2 + " " + peso);

**if** (!(**this**.centrales.contains(nodo1) && **this**.centrales.contains(nodo2))) {

union(**this**.aristas.get(i).getNodo1(), **this**.aristas.get(i).getNodo2());

**this**.recorrido.add(**new** Arista(nodo1, nodo2));

noUnir = i;

**break**;

}

}

**this**.minimoCosto += peso;

aristas++;

**while** (aristas < (**this**.cantCiudades - **this**.cantCentrales) && k < **this**.aristas.size()) {

**if**(k != noUnir) {

nodo1 = **this**.aristas.get(k).getNodo1();

nodo2 = **this**.aristas.get(k).getNodo2();

peso = **this**.aristas.get(k).getCosto();

**if**(!(**this**.centrales.contains(nodo1) && **this**.centrales.contains(nodo2))) {

**if** (find(nodo1) != find(nodo2)) {

union(nodo1, nodo2);

aristas++;

**this**.recorrido.add(**new** Arista(nodo1, nodo2));

**this**.minimoCosto += peso;

}

}

}

k++;

}

}