Estructuras de Datos y Algoritmos - EDA (ETS de Ingeniería Informática).
Curso 2023-2024

Práctica 6 - Parte 1. Una aplicación del algoritmo de Kruskal: diseño del tendido eléctrico entre ciudades

¿Cómo conectar N ciudades tendiendo (solo) N-1 cables?

Departamento de Sistemas Informáticos y Computación. Universitat Politècnica de València

1. Objetivos

Esta práctica tiene un doble objetivo. Por un lado, al concluirla, el alumno deberá ser capaz de reconocer y aplicar los conceptos y resultados más relevantes de la teoría de Grafos a la resolución de un tipo de problemas que, bajo múltiples instancias, se plantea en la vida real: conectar entre sí un conjunto de objetos de la forma más "económica" posible, o problema del **Árbol de Recubrimiento Mínimo**, **ARM** (o *Minimum Spanning Tree*, *MST*).

Por otro lado, además, el alumno deberá ser capaz de representar y obtener soluciones factibles y óptimas a este tipo de problemas reutilizando la jerarquía Java `Grafo` proporcionada. Específicamente, implementará una versión eficiente del algoritmo de Kruskal que permite optimizar el diseño del tendido eléctrico entre un conjunto de ciudades dado (el mismo problema práctico que, en 1926, llevó al matemático checo Otakar Boruvka a formular por primera vez el problema del Árbol de Recubrimiento Mínimo y a resolverlo algorítmicamente).

2. Descripción del problema

Una compañía eléctrica necesita renovar el tendido eléctrico existente entre un conjunto de `N` ciudades con el menor coste posible. Es muy probable que en el tendido "sobren" cables (ver Figura 1), y la compañía decide analizar si puede redefinir la topología de la red para mantener activos y renovar solo `N-1` cables, que es el número mínimo necesario para dar servicio a `N` ciudades. Esta es solo una **solución factible** al problema, pues minimiza el número de cables a renovar, pero no el coste de su renovación. (Se verá, en la segunda parte de esta práctica, cómo elegir los `N-1` cables a renovar con coste mínimo).

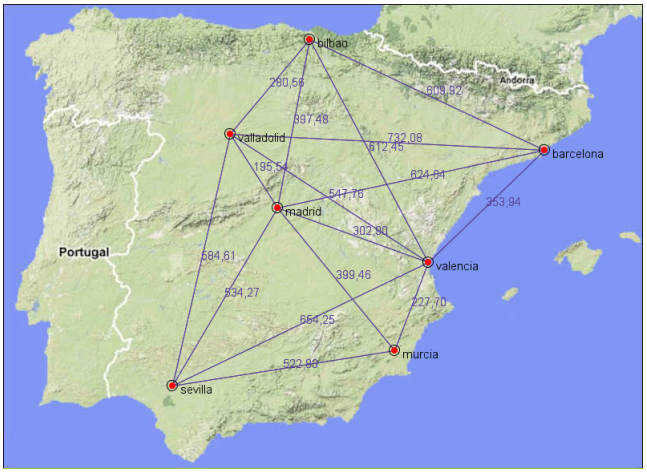


Figura 1: Tendido eléctrico de la red que conecta (da servicio a) siete ciudades

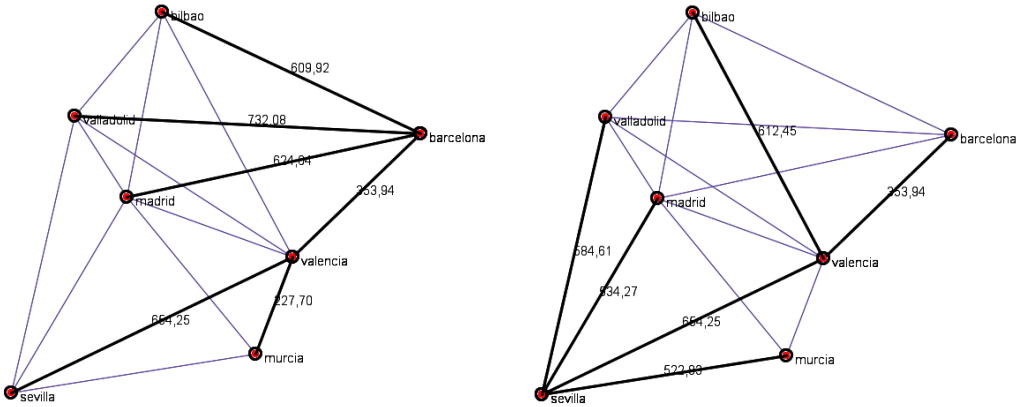
Para llevar a cabo este estudio, se representa la red eléctrica a renovar mediante un grafo No Dirigido cuyos vértices son las ciudades de la red y cuyas aristas son los cables del tendido eléctrico, etiquetadas con el coste económico de su renovación. A continuación, se plantea, en términos de Teoría de Grafos, si el problema tiene solución para cualquier conjunto de `N` ciudades. Hay solución SII el grafo que representa al problema es Conexo, entonces existe al menos un conjunto de `N-1` aristas que permiten conectar entre sí cualquier par de vértices

del grafo. Al ser `N-1` el número mínimo de aristas que tiene un grafo No Dirigido y Conexo de `N` vértices, este conjunto de aristas no contiene ninguna arista Hacia Atrás (que forme ciclo) y, además, garantiza que cualquier par de vértices están conectados por un único camino simple. Por ello, define lo que se denomina un Árbol de Recubrimiento del grafo, i.e. un subgrafo Acíclico (Árbol), Conexo y cuyos `N` vértices son los vértices del grafo.

Teniendo en cuenta todo esto, el algoritmo a emplear para resolver el problema es el de búsqueda de un Árbol de Recubrimiento de un grafo No Dirigido de `N` vértices, obteniendo el conjunto de `N-1` aristas que lo componen. En concreto:

- Se hace un Recorrido en Anchura (BFS) de un vértice cualquiera, `v`, del grafo.
- Como resultado del recorrido BFS de `v`, se guardan las aristas empleadas para visitar todos los vértices alcanzables desde `v`.
- Si dicho resultado es un conjunto de exactamente `N-1` aristas, entonces es la solución del problema.
- Si no así, el grafo no es Conexo, y no hay solución del problema, por lo que se devolverá `null`.

Como ejemplos, se muestran, con líneas gruesas negras, los Árboles de Recubrimiento resultado de los BFS de los vértices Barcelona (Figura 2(a)) y Sevilla (Figura 2(b)) del grafo ejemplo (Conexo) de la Figura 1.



(a) (b)

Figura 2: Dos Árboles de Recubrimiento BFS del grafo de la Figura 1

Cualquier árbol de recubrimiento obtenido al realizar el BFS de un vértice es solo una solución factible al problema, i.e. no es necesariamente un ARM. De hecho, solo lo sería si todas las aristas del grafo tuvieran el mismo peso, lo que es improbable en la vida real... Por ello, ninguno de los dos árboles de recubrimiento de la Figura 2 es un ARM del grafo de la Figura 1. Como se verá en la segunda parte de esta práctica, para obtener un ARM de un grafo (ver Figura 3) habrá que modificar la estrategia BFS para optimizar la selección de las aristas del árbol de recubrimiento teniendo en cuenta sus pesos (algoritmo de Kruskal).

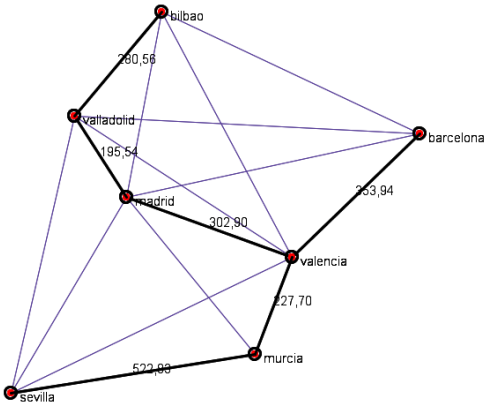


Figura 3: Árbol de Recubrimiento Mínimo (ARM) del grafo de la Figura 1

3. Actividades

Antes de realizar las actividades propuestas, se actualizará la estructura de paquetes y clases de su proyecto *BlueJ eda* del siguiente modo:

- Dentro del paquete `librerias.estructurasDeDatos`, crear un nuevo paquete de nombre `grafos`. Hecho esto, ***Salir*** de *BlueJ*.
- Añadir al paquete `grafos` todos los ficheros disponibles en *PoliformaT* para esta primera sesión de la práctica.
- Invocar de nuevo *BlueJ* y acceder al paquete `grafos`. Tal como muestra la Figura 4, este paquete debe contener las clases de la Jerarquía Java `Grafo` y dos clases más: `Arista` y `TestGrafo`. `Arista` se incorpora a la jerarquía `Grafo`, pues la usan los nuevos métodos que se deben implementar. `TestGrafo` servirá para comprobar la corrección del código desarrollado.

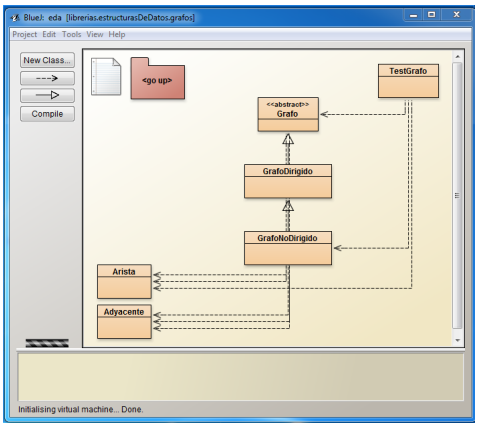


Figura 4: Clases del paquete *grafos*

3.1. Completar la clase `Arista` e implementar el método `arbolRecubrimientoBFS`

En primer lugar, es necesario completar el código de la clase `Arista`. Para ello, basta seguir los comentarios que aparecen en su código.

A continuación, hay que implementar, en la clase `Grafo`, un nuevo método de Búsqueda BFS que devuelva una solución factible al problema del Árbol de Recubrimiento. Su especificación y cabecera ya figuran en la clase `Grafo` y son:

```
/** PRECONDICIÓN: !this.esDirigido()
 *  Devuelve un subconjunto de aristas que conectan todos los vertices
 *  de un grafo No Diridigo y Conexo, o null si el grafo no es Conexo.
 *  @return Arista[], array con las numV - 1 aristas que conectan los numV
 *                    vertices del grafo, o null si el grafo no es Conexo
 */
public Arista[] arbolRecubrimientoBFS()
```

Se recomienda no empezar desde cero la implementación de `arbolRecubrimientoBFS`, sino hacer lo siguiente:

- Copiar el cuerpo del método `toArrayBFS()`, que hay en la clase `Grafo`, en el de `arbolRecubrimientoBFS`.
- Modificar y/o eliminar las líneas del cuerpo actual de `arbolRecubrimientoBFS` según la descripción del problema hecha en el apartado 2.
- Copiar el método `protected` que implementa el Recorrido BFS de un vértice dado de un Grafo y modificar tanto su cabecera como su cuerpo para que puede ser invocado desde el cuerpo de `arbolRecubrimientoBFS`.

3.2. Validar el código desarrollado

Para comprobar la corrección del código implementado durante la sesión, el alumno debe ejecutar el programa `TestGrafo`.