EDA (ETS d'Enginyeria Informàtica). Curs 2020-2021
Pràctica 6: Una aplicació de l'algorisme de Kruskal a la vida real:
disseny de la línia elèctrica entre ciutats
Sesió 1: ¿Com connectar N ciutats instal·lant (només) N-1 cables elèctrics?

Departament de Sistemes Informàtics i Computació. Universitat Politècnica de València

1. Objectius

Aquesta pràctica persegueix un doble objectiu. D'una banda, en concloure les dues sessions que la componen, l'alumne haurà de ser capaç de reconéixer i aplicar els conceptes i resultats més rellevants de la teoria de Grafs a la resolució d'una mena de problemes que, baix múltiples instàncies, es plantegen en la vida real: connectar entre sí un conjunt d'objectes de la forma més "econòmica" possible, o problema de l'Arbre de Recobriment Mínim (*Minimum Spanning Tree* en anglés).

D'altra banda, a més, l'alumne haurà de ser capaç de representar i obtindre solucions factibles i òptimes a aquesta mena de problemes reutilitzant la jerarquia Java `Grafo` que se li proporciona. Específicament, en aquesta pràctica implementarà una versió eficient de l'algorisme de Kruskal que permet optimitzar el disseny de la línia elèctrica entre un conjunt de ciutats donat, el mateix problema pràctic que, en 1926, va portar al matemàtic txec Otakar Boruvka a formular per primera vegada el problema de l'Arbre de Recobriment Mínim i a resoldre'l algorítmicament.

2. Descripció del problema

Una companyia elèctrica necessita renovar la línia elèctrica existent entre un grup de `N` ciutats amb el menor cost possible. Com és molt probable que en l'estesa "sobren" bastants cables (veure Figura 1), la companyia decideix analitzar si és possible redefinir la topologia d'aquesta xarxa elèctrica perquè només es mantinguen actius i renoven `N-1` dels seus cables, que és el número mínim que ha de tindre per a donar servei a `N` ciutats. Note's que aquesta és només una **solució factible** al problema, perquè minimitza el nombre de cables a renovar però no el cost de la seua renovació; com es veurà en la segona sessió d'aquesta pràctica, per a decidir quins dels cables de la xarxa són els `N-1` a renovar amb cost mínim cal fer uns quants càlculs més.

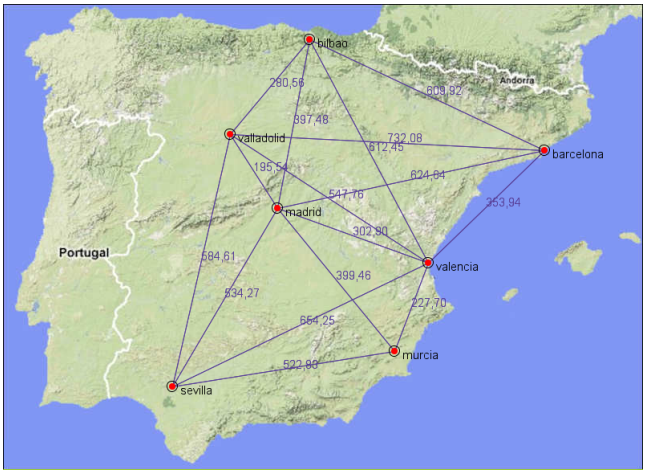


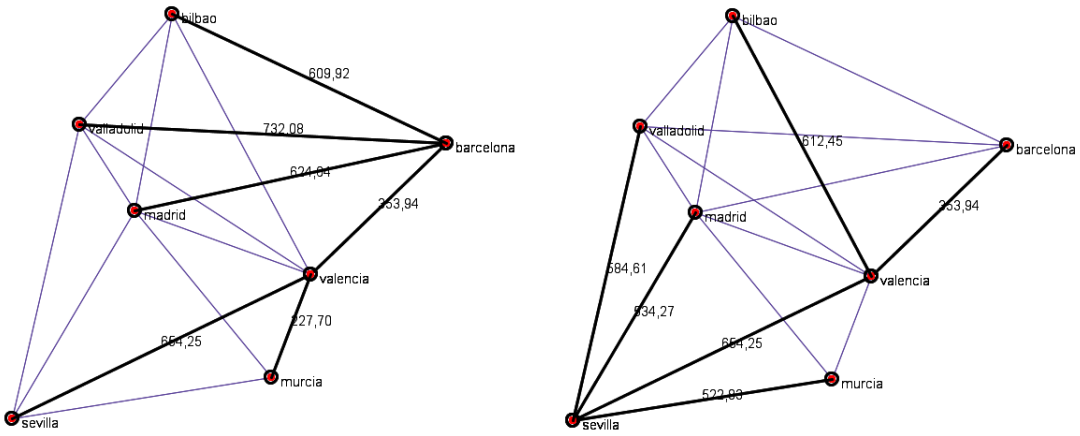
Figura 1: Instal·lació elèctrica de la xarxa que dóna servei a set ciutats espanyoles

Per a dur a terme aquest estudi de viabilitat, l'equip informàtic de la companyia representa la xarxa elèctrica a renovar mitjançant un graf *No Dirigit* en el qual els vèrtexs són les ciutats de la xarxa i les arestes els cables de la línia elèctrica, etiquetades amb els milions d'euros que costaria la seua renovació. Fet això, planteja en termes de Teoria de Grafs si el problema té solució per a qualsevol conjunt de `N` ciutats. El resultat conegut que aplica és que SII el graf que representa al problema és Connex, existeix almenys un conjunt de `N-1` de les seues arestes que permeten connectar entre si qualsevol parell de vèrtexs del graf. Note's que, en ser `N-1` el nombre

mínim d'arestes que té un graf No Dirigit i Connex de `N` vèrtexs, aquest conjunt d'arestes no conté cap aresta Cap endarrere (que forme cicle) i, a més, garanteix que qualsevol parell de vèrtexs del graf estan connectats per un únic camí simple; precisament per això, defineix el que es denomina un Arbre de Recobriment del graf, i.e. un subgrafo Acíclic (Arbre), Connex i que els seus `N` vèrtexs són els vèrtexs del graf.

Sobre la base d'aquest resultat, l'equip informàtic ja sap a quin problema s'enfronta i, a més, quin algorisme emprar per a resoldre-ho eficientment: trobar un Arbre de Recobriment d'un graf No Dirigit de `N` vèrtexs, retornant com a resultat el conjunt de `N-1` arestes que el componen; si aquest conjunt no existeix, perquè el graf no és Connex, el resultat a retornar és un conjunt inexistent d'arestes (o `null`).

Per a obtindre la solució d'aquest problema, n'hi ha prou amb realitzar un Recorregut en Amplària (BFS) d'un sol vèrtex `v` del graf, qualsevol d'ells, i guardar com a resultat les arestes emprades per a visitar tots els vèrtexs assolibles des de `v`. Fet això, si el BFS de `v` retorna un conjunt d'exactament `N-1` arestes, el graf és Connex i, per tant, la solució del problema és la del BFS de `v`; sinó, el graf no és Connex i la solució del problema és `null`. Per a il·lustrar aquest resultat, en la següent figura es mostren amb línies gruixudes negres els Arbres de Recobriment resultat dels BFS dels vèrtexs Barcelona (Figura 2(a)) i Sevilla (Figura 2(b)) del graf exemple (Connex) de la Figura 1.



(a) (b)

Figura 2: Dos Arbres de Recobriment BFS de el graf de la Figura 1

Ja per a concloure, és important recordar que qualsevol Arbre de Recobriment que s'obtinga en realitzar el BFS d'un vèrtex d'un graf és només una solució factible al problema plantejat inicialment, i.e. no és necessàriament un Arbre de Recobriment Mínim; de fet, només ho seria si totes les arestes del graf tingueren el mateix pes, la qual cosa és altament improbable en la vida real... Justament per això, cap dels dos Arbres de Recobriment de la Figura 2 és un Arbre de Recobriment Mínim del graf de la Figura 1. Com es detallarà en la segona sessió d'aquesta pràctica, per a obtindre un Arbre de Recobriment Mínim d'un graf (veure Figura 3) caldrà modificar l'estratègia BFS empleada fins al moment per a optimitzar la selecció de les arestes de l'Arbre de Recobriment "a la Kruskal", és a dir, sobre la base dels seus pesos.

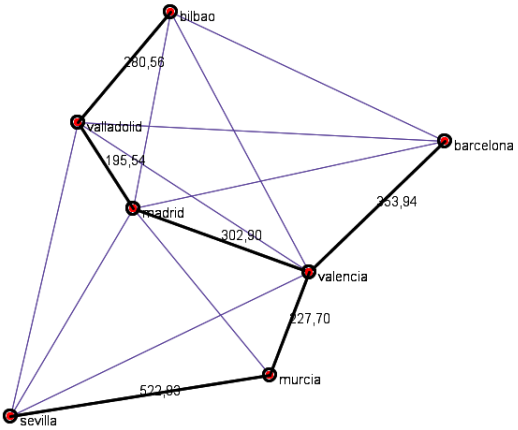


Figura 3: Arbre de Recobriment Mínim de el graf de la Figura1

3. Activitats

Abans de realitzar les activitats que es proposen en aquesta sessió, l'alumne haurà d'actualitzar l'estructura de paquets i classes del seu projecte *BlueJ eda* tal com s'indica a continuació:

- Obrir el projecte *BlueJ eda* i, dins del seu paquete *librerias.estructurasDeDatos*, crear un nou paquet de nom *grafos*. Una volta fet, ***Eixir*** del *BlueJ*.
- Descarregar en la carpeta *grafos* tots els fitxers disponibles en *PoliformaT* per a aquesta primera sessió de la pràctica.
- Invocar de nou al *BlueJ* i accedir al paquet *librerias.estructurasDeDatos.grafos*. Tal com mostra la següent figura (Figura 4), aquest paquet ha de contindre ara les classes de la Jerarquia Java `Grafo` i dues classes més: `Arista` i `TestGrafo`. Una vegada completada, la primera s'incorporarà a la jerarquia `Grafo`, perquè la usen els nous mètodes que l'alumne ha d'implementar en aquesta pràctica; la segona servirà, com el seu nom indica, per a comprovar la correcció del codi desenvolupat durant aquesta sessió

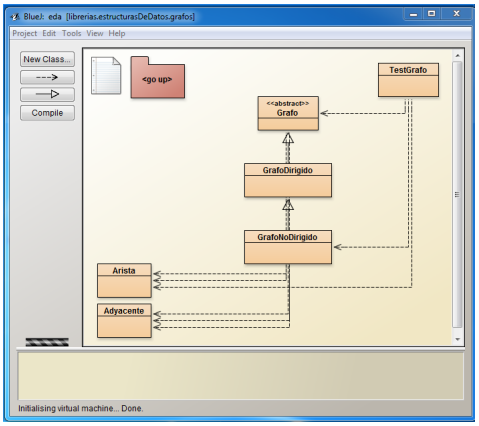


Figura 4: Clasess del paquet *grafos*

3.1. Implementar el mètode `arbolRecubrimientoBFS` i completar la classe `Arista`

En aquesta activitat l'alumne ha d'implementar en la classe `Grafo` un nou mètode de Cerca BFS que retorne una solució factible al problema de l'Arbre de Recobriment Mínim. La seua especificació i capçalera ja figuren en la classe `Grafo` i són com segueix:

```
/** PRECONDICIÓ: !this.esDirigido()
 * Retorna un subconjunt d'arestes que connecten tots els vertices
 * d'un graf No Dirigit i Connex, o null si el graf no és Connex.
 *  @return Arista[], array amb les numV - 1 arestes que connecten els numV
 *                    vèrtex del graf, o null si el graf no és connex
 */
public Arista[] arbolRecubrimientoBFS()
```

A partir d'elles, resulta obvi que abans d'implementar `arbolRecubrimientoBFS` és necessari completar el codi de la classe `Arista`; per a això, basta seguir els comentaris que apareixen en el seu codi.

Quant a la implementació de `arbolRecubrimientoBFS`, es recomana no començar-la des de zero, sinó com segueix:

- Copiar el cos del mètode `toArrayBFS()` que hi ha en la classe `Grafo` en el de `arbolRecubrimientoBFS`.
- Modificar i/o eliminar aquelles línies del cos actual de `arbolRecubrimientoBFS` sobre la base de la descripció del problema realitzada en l'apartat 2 d'aquest butlletí.
- Copiar el mètode `protected` que implementa el Recorregut BFS d'un vèrtex donat d'un Graf i modificar tant la seua capçalera com el seu cos perquè pot ser invocat des del cos de `arbolRecubrimientoBFS`.

3.2. Validar el codi desenvolupat en la pràctica

Per a comprovar la correcció del codi implementat durant la sessió l'alumne ha d'executar el programa `TestGrafo`.