ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA
Universitat Rovira i Virgili

Estructures de Dades 2022

Pràctica 2: *Grafs*

En aquesta segona pràctica tenim com a objectiu treballar amb un graf. Concretament, generarem un graf dels punts de recàrrega de Catalunya que proporciona l'ICAEN (Institut Català d'Energia): https://xarxarecarrega.icaen.gencat.cat/ICAEN/

Es tracta d'un dataset que conté tots els punts de recàrrega instal·lats a Catalunya amb les seves coordenades per tal de poder-los situar en el mapa.

La informació que ens proporcionen com a input és una llista de connectors en format JSON. Cal notar que un punt de recàrrega pot tenir múltiples connectors (endolls), i a la seva vegada un punt de recàrrega es troba en una localització concreta on pot haver-hi més punts de recàrrega, per tant, és normal que trobeu més d'un connector amb les mateixes coordenades. Per a simplificar el problema, no agruparem els endolls en punts de recàrrega i zones de recàrrega, sinó que agruparem els endolls directament en zones de recàrrega, és a dir, tots els endolls amb les mateixes coordenades formaran part de la mateixa zona de recàrrega, indiferentment de com estiguin organitzats físicament en punts de recàrrega.

Aquesta informació s'haurà d'emmagatzemar en un graf on els nodes del graf representaran les zones de recàrrega que hi ha. **Cada node estarà identificat per un identificador corresponent a la zona de càrrega (podeu utilitzar el vostre propi sistema d'identificació basat en la ciutat, carrer, coordenades o altre propietats del data set que us permetin identificar una zona...), i tindrà una posició del mapa i informació de la potència dels endolls que hi ha a la zona.** A més tindrem que cada node contindrà una llista de les zones properes i la distància que els separa en forma d'aresta.

Per a guardar el graf primer haurem de programar una estructura de dades genèrica que ens permetrà emmagatzemar un graf, i posteriorment haurem de crear una instància d'aquest graf genèric per guardar les dades dels punts de càrrega.

L'estructura de dades genèrica per a guardar dades en forma de graf haurà de respectar la següent interfície:

```
void CrearGraf();
```
- Constructor per inicialitzar la taula.

```
void afegirAresta(V v1, V v2, E e);
```
- Funció per a afegir una aresta.
- L'operació llença una excepció en cas que no es pugui afegir.

```
boolean existeixAresta(V v1, V v2);
```
- Funció que ens diu si una aresta existeix.

`E valorAresta(V v1, V v2);`

- Funció que retorna el valor d'una aresta.
- L'operació llença una excepció en cas que no existeixi.

`LlistaGenerica<V> adjacents(V v);`

- Funció que retorna una llista que conté tots els nodes adjacents al node passat per paràmetre.
- L'operació llença una excepció en cas que no es pugui crear aquesta llista.

Implementació del graf

Per a construir el graf, simplificarem el problema de nou i, en lloc de representar la xarxa de carreteres de Catalunya per a unir els punts de recàrrega, simularem les carreteres a partir d'aquestes dues assumpcions:

1. Totes les zones de recàrrega a menys de 40km de distància estaran connectades entre elles en línia recta i les arestes no seran dirigides. Per tal de calcular la distància entre dos punts utilitzarem la distància entre dues coordenades en un pla 2D (distància euclidià entre dos punts)

2. Si una zona de recàrrega no està connectada amb ningú a partir de la condició anterior, es connectarà amb la zona més propera sigui quina sigui la distància que les separa, de manera que garantirem que hi hagi només hi haurà una component connexa en el graf.

A més, considereu els següents aspectes:

- Implementeu les classes necessàries per tal de definir tots els elements mencionats en la introducció del problema i guardar els atributs corresponents. Una d'aquestes classes serà la classe de "graf genèrica", la qual ha de seguir l'especificació que hem esmentat. La instància d'aquesta ens servirà per a guardar les dades de punts de càrrega.

- Per a guardar la informació dels nodes podeu fer servir l'estructura que vulgueu, tot i que es recomana fer servir la taula de hashing, la que vau fer a la primera pràctica o en el seu defecte, el HashMap de Java.

- Per a guardar les arestes us recomanem una multillista d'adjacència de manera que el cost en memòria sigui òptim, és a dir, el mínim. Valorarem que la multillista d'adjacència guardi el mínim d'informació possible.

- Per a totes aquelles classes que no formin part de l'estructura graf i que requereixin guardar les dades en una estructura seqüencial (ex. llista de resultats, llista d'endolls) es podrà utilitzar la classe ArrayList de java.util.

Tenint en compte que l'objectiu de la pràctica no és el manegament d'inputs i el seu parseig, es permetrà l'ús de qualsevol llibreria per a llegir el fitxer d'input en format JSON i convertir-lo en objectes. Algunes suggerències: Gson, Json-simple, Org.json... us recomanem la utilització d'un gestor de dependències com Maven per afegir la dependència de la llibreria (el vostre IDE us pot ajudar amb això), o bé, podeu baixar el .jar i linkar-lo manualment en el pitjor dels casos.

Implementació d'algoritmes

Un cop construït el graf, es demana que a partir la posició inicial de l'usuari, l'autonomia del seu vehicle en Km i la posició de destí, es busqui la ruta més òptima indicant en quina zona de recàrrega ha d'aturar-se a carregar per arribar abans d'arribar al seu destí. Si hi ha dos punts a la mateixa distància, prioritzarem les parades en aquells endolls amb una potència de recàrrega més gran per tal de que estigui el mínim temps possible carregant.

L'especificació a seguir per aquest algoritme és la següent:

```
Llista<String> camiOptim(String identificador_origen, String
indentificador_desti, int autonomia);
```

- Funció que retorna una llista que conté tots els noms dels punts de càrrega pels quals ha de passar per arribar al seu destí. A partir del resultat d'aquesta funció s'hauria de poder pintar per pantalla el recorregut a fer i la distància total que es recorrerà.
- L'operació llença una excepció en cas que no es pugui crear aquesta llista.

Aprofitant que se'ns ha proporcionat el *dataset*, també avaluarem com evoluciona l'electrificació del territori en base a un dels objectius de la Generalitat, per mirar si es pot garantir que amb la reserva de bateria que proporciona una autonomia determinada (per exemple 30km) des d'una zona de recàrrega en podrem trobar una altra per a poder carregar. Així doncs, donat un punt de partida, indica aquelles zones de recàrrega a les que no podrem arribar si donada una autonomia determinada (i suposant que carreguem a cada punt per a tenir l'autonomia indicada), no compleixen amb aquesta condició i d'aquests per tal de poder traslladar aquesta informació als òrgans competents i puguin instal·lar nous punts de recàrrega en el camí.

L'especificació a seguir per aquest algoritme és la següent:

```
Llista<String> zonesDistMaxNoGarantida(String indentificador_origen,
int autonomia);
```

- Funció que retorna una llista que conté aquelles zones de recàrrega que no compleixen la condició d'estar enllaçades amb, almenys, amb una altra zona de recàrrega a una distància màxima determinada per l'autonomia.
- L'operació llença una excepció en cas que no es pugui crear aquesta llista i retorna una llista buida si es pot arribar a totes les zones de recàrrega.

Per tal de donar solució als dos requeriments anteriors, utilitza els algoritmes vistos a classe que consideris més adients per a obtenir el resultat. Tingueu present que el joc de proves a realitzar ha de ser exhaustiu. Això significa que haureu de provar un nombre de casos elevat dels dos algoritmes implementats per tal de demostrar el seu funcionament, intentant també els casos més extrems, com per exemple la ruta més llarga del mapa. Aquestes proves s'han de poder visualitzar a l'executar el vostre programa.

Lliurament

- Les pràctiques són individuals.
- Es fa el lliurament d'un únic arxiu `PR2-NOM_COGNOMS.zip` que conté:
    - Els arxius de codi font.
    - Informe (en PDF) incloent explicacions dels aspectes més rellevants de les vostres implementacions, jocs de proves realitzats... al final de tot de l'informe també haureu d'incloure tot el vostre codi enganxat.

Puntuació

- Construcció del graf – 4pt
- Buscar ruta òptima – 3pt
- Buscar les zones de recàrrega que no tenen una altra zona de recàrrega a menys de certa distància – 2pt
- Joc de proves – 1pt