Pràctica 1: Navegació

Part 1: Cerca No Informada

Intel·ligència Artificial

2019-2020

Departament de Ciències de la Computació
Universitat Autònoma de Barcelona

1 Introducció

En aquesta pràctica resoldrem un problema simple de navegació sobre un mapa, on donades unes coordenades d'inici i de final, trobarem la millor ruta entre els dos punts. Per motius de simplicitat, només tindrem en compte les possibles rutes entre els dos punts de la ciutat utilitzant el metro com a mitjà de transport, per tant, intentarem trobar el nombre òptim de parades que hem de fer. Per a fer-ho utilitzarem quatre mètodes de cerca explicats a teoria:

1. Cerca en profunditat `(Depth First Search)`
2. Cerca en amplada `(Breadth First Search)`
3. Cerca de cost uniforme `(Uniform Cost Search)`
4. Cerca A* (`A-Star`)

En aquesta primera part de la pràctica ens centrarem en els mètodes de cerca no informada i no òptima.

ATENCIÓ! En aquesta pràctica guardarem els camins en ordre invers a com ho hem fet a la teoria. Els guardarem en una llista on el primer element és l'arrel (estat inicial) i l'últim element és el node fulla (estata actual del camí).

2 Fitxers necessaris

Per a realitzar la pràctica haureu de descarregar les següents carpetes:

1. `cityInformation`: Carpeta que conté els diversos arxius que representaran el mapa de la ciutat per a tota la ciutat (Lyon_bigCity) o per a una simplificació del mapa (Lyon_smallCity). Dins d'aquesta carpeta trobareu els fitxers:

   (a) `InfoVelocity.txt`: Conté la informació sobre la velocitat a la que viatja cada metro.

   (b) `Stations.txt`: Conté els IDs de cada estació, nom, numero de línia i les coordinades on es troba.

   (c) `Time.txt`: Taula on podem veure el temps que es triga per anar d'una estació a una altra. No hi ha connexió entre dues estacions si el valor de la taula es zero.

   (d) `Lyon_city.jpg`: Imatge del mapa de la ciutat.

2. `Code`: Carpeta que conté els fitxers de python amb funcions útils per a la pràctica, i els fitxers on haureu de programar. Dins d'aquesta carpeta trobareu els fitxers:

   (a) `utils.py`: Conté una series de funcions que us poden ser útils per entendre que fa el que programeu.

   (b) `SubwayMap.py`: Conté les dos classes principals amb les que treballarem: Map (Conté tota la informació sobre la ciutat) i Path (Classe que guarda la informació sobre una ruta o successió de parades).

   (c) `TestCases.py`: Arxiu amb el qual podreu comprovar si les funcions que programeu donen el resultat esperat.

   (d) `SearchAlgorithm.py`: Arxiu on haureu de programar tota la pràctica.

3 Preparació

Abans de començar a programar es molt recomanable entendre les dues classes amb les que treballarem: `Map` i `Path`.

Per a poder realitzar la pràctica, heu d'entendre molt bé quines variables podeu obtenir de cada classe i com cridar-les, per aquest motiu us recomanem que abans de començar a programar res obriu l'arxiu `SubwayMap.py` i identifiqueu quines variables i funcions haureu de cridar durant la pràctica.

Per tant, per confirmar que heu entès aquestes classes i abans de seguir, hauríeu de ser capaços de:

1. Accedir a tota la informació d'una parada en concret (Numero de línia, coordenades i velocitat)

2. Accedir a les connexions d'una parada de metro i al seu valor.

3. Entendre i poder crear un `Path`.

4 Què s'ha de programar?

En la primera part d'aquesta pràctica programareu els algorismes de cerca en amplada i cerca en profunditat que tenen una estructura molt similar. Només varia l'ordre en el qual s'exploren els nodes. Per a programar-los necessitareu crear dues funcions prèvies.

4.1 Funcions prèvies: `Expand` i `Remove_cycles`

`Expand:` Funció que pren com a input un `Path pare` i el `Map`, i retorna una llista de `Paths`, on cadascun d'aquests Paths es igual que el Path pare, però s'ha afegit al final de la llista un node que tenia connexió amb el Path pare.

EXEMPLE:

Crida de la funció: `expand([14,13,8,12],MAP)`

Retorna: `[[14,13,8,12,8], [14,13,8,12,11], [14,13,8,12,13]]`

`Remove_cycles:` Funció que pren com a input una `llista de Paths` i cerca i elimina els Paths en que ja s'ha visitat prèviament l'ultima estació que s'ha afegit a cada Path. D'aquesta manera ens assegurem de no caure en bucles infinits on ens movem sempre per les mateixes estacions. Ha de retornar la llista d'entrada sense els Paths amb repeticions.

EXEMPLE:

Crida de la funció: `remove_cycles([[14,8,12,8], [14,8,12,11], [14,8,12,14]])`

Output:`[[14,8,12,11]]`

4.2 Algorisme de Cerca en Profunditat

Heu d'implementar les següents funcions:

`Insert_depth_first_search:` Funció que pren cam a input la `llista de Paths fills que hem calculat` i la Cua de la `llista de Paths que estem explorant` i retorna la unió d'aquestes dues llistes d'acord amb el principi de cerca en profunditat.

`Depth_first_search:` Funció que donada una estació origen, una estació final, i el mapa de la ciutat es capaç de trobar una ruta entre les dues estacions utilitzant l'algorisme que es veu a la figura 1 però vigilant l'ordre en que es guarden els camins.

```
Funció CERCA_profunditat (NodeArrel, NodeObjectiu)
1. Llista=[ [ NodeArrel ] ];
2. Fins que (Cap(Cap(Llista))=NodeObjectiu O bé (Llista=NIL) fer
   a) C=Cap(Llista);
   b) E=Expandir( C );
   c) E=EliminarCicles(E);
   d) Llista=Inserir_davant(E,Cua(Llista));
3. Ffinsque;
4. Si (Llista<>NIL) Retornar(Cap(Llista));
5. Sinó Retornar("No existeix Solucio");
Ffuncio
```

Figure 1: Pseudo-codi de la Cerca en Profunditat vista a Teoria.

4.3 Algorisme de Cerca en Amplada

Heu d'implementar les següents funcions:

`Insert_breadth_first_search:` Funció que pren cam a input la `llista de Paths fills que hem calculat` i la Cua de la `llista de Paths que estem explorant` i retorna la unió d'aquestes dues llistes d'acord amb el principi de cerca en amplada.

`Breadth_first_search:` Funció que donada una estació origen, una estació final, i el mapa de la ciutat es capaç de trobar una ruta entre les dues estacions utilitzant l'algorisme que que es veu a la figura 2 però vigilant l'ordre en que es guarden els camins.

```
Funció CERCA_amplada (NodeArrel, NodeObjectiu)
1. Llista=[ [ NodeArrel ] ];
2. Fins que (Cap(Cap(Llista))=NodeObjectiu O bé (Llista=NIL) fer
   a) C=Cap(Llista);
   b) E=Expandir( C );
   c) E=EliminarCicles(E);
   d) Llista=Inserir_darrera(E,Cua(Llista));
3. Ffinsque;
4. Si (Llista<>NIL) Retornar(Cap(Llista));
5. Sinó Retornar("No existeix Solucio");
Ffuncio
```

Figure 2: Pseudo-codi de la Cerca en Amplada vista a Teoria.

4.4 Una funció final: Coord2station

Com trobaríem la ruta òptima en cas de que l'usuari no es troba exactament al costat d'una parada de metro?

Aquesta darrera funció ens servirà per trobar l'estació o estacions més properes donades unes coordenades qualsevols del mapa.

`Coord2station:` Funció que pren com entrada una llista amb dos valors [X,Y] amb les coordenadeson es troba l'usuari i el Mapa, i retornarà una llista amb els IDs de les estacions més properes (en cas de que una parada tingui varies línies ha de tornar el ID de cada línia, i si l'usuari es troba entre varies parades a la mateixa distància també).

EXEMPLE:

Crida de la funció: `coord2station([100,200],MAP)`

Output:`[8,12,13]`

5 Entrega de la Part 1

Per a l'avaluació d'aquesta primera part de la pràctica haureu de pujar al Moodle el vostre fitxer `SearchAlgorithm.py` que ha de contenir el vostre NIA a la variable authors i el vostre grup al la variable group (a l'inici de l'arxiu). L'entrega s'ha de fer abans del dia **01/03/2020** a les **23:55**.

ATENCIÓ! és important que tingueu en compte els següents punts:

1. La correcció del codi es fa de manera automàtica, per tant assegureu-vos de penjar els arxius amb la nomenclatura i format correctes.
2. El codi esta sotmès a detecció automàtica de plagis durant la correcció.
3. Qualsevol part del codi que no estigui dins de les funcions de l'arxiu `SearchAlgorithm.py` no podrà ser avaluada, per tant no modifiqueu res fora d'aquest arxiu.
4. Per evitar que el codi entri en bucles hi ha un límit de temps per a cada exercici, per tant si les vostres funcions triguen massa les comptarà com a error.