Pràctica 1: Navegació

Part 1: Cerca No Informada

Intel·ligència Artificial

2023-2024

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 (estat 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 qual viatja cada metro.

(b) `Stations.txt`: Conté els IDs de cada estació, nom, número 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 és 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 sèrie de funcions que us poden ser útils per entendre que fa el que programeu.

(b) `SubwayMap.py`: Conté les dues classes principals amb les quals 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 és molt recomanable entendre les dues classes amb les quals 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 (Número 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.

A l'inici del fitxer cal que poseu el vostre NIU, per exemple:
__author__ = '1124601'

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 és 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 que ja s'ha visitat prèviament l'última 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 com 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 és capaç de trobar una ruta entre les dues estacions utilitzant l'algorisme que es veu a la figura 1 però vigilant l'ordre que es guarden els camins.

4.3 Algorisme de Cerca en Amplada

Heu d'implementar les següents funcions:

`Insert_breadth_first_search`: Funció que pren com 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 és capaç de trobar una ruta entre les dues estacions utilitzant l'algorisme que es veu a la figura 2 però vigilant l'ordre 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 vist a Teoria.

```
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 vist a Teoria.

4.4 Una funció final: distance_to_stations

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

Aquesta darrera funció ens servirà per calcular la distància entre qualsevol punt del mapa i totes les estacions de metro d'aquest.

`distance_to_stations:` Funció que pren com entrada les coordenades on es troba l'usuari, que és una llista amb dos valors [X,Y], i un Mapa. Retorna un diccionari que conté, els IDs de les estacions de metro com a claus, i la distància entre el punt on es troba l'usuari i les estacions de metro com a valors. Aquest diccionari que retorna està ordenat, primer per distància (ascendent), i segon per id de l'estació (ascendent).

EXEMPLE:

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

Output: `{8:10.0, 12:10.0, 13:10.0, 9:24.76, 7:58.73, 14:60.03, 11:66.48, 6:93.94, 1:125.42, 2:149.45, 5:149.45, 10:149.45, 3:151.61, 4:177.56}`.
***Per motius de claredat, les distàncies han estat arrodonides a dos decimals.*

5 Entrega de la Part 1

Per a l'avaluació d'aquesta primera part de la pràctica haureu de pujar al Campus Virtual el vostre fitxer `SearchAlgorithm.py` que ha de contenir el vostre NIU a la variable authors i el vostre grup a la variable group (a l'inici de l'arxiu). L'entrega s'ha de fer abans del dia **03/March/2024** 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 està 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. Tant la correctesa com l'eficiència del codi es valoraran, per tant si les vostres funcions triguen massa, el codi les comptarà com a errònies.