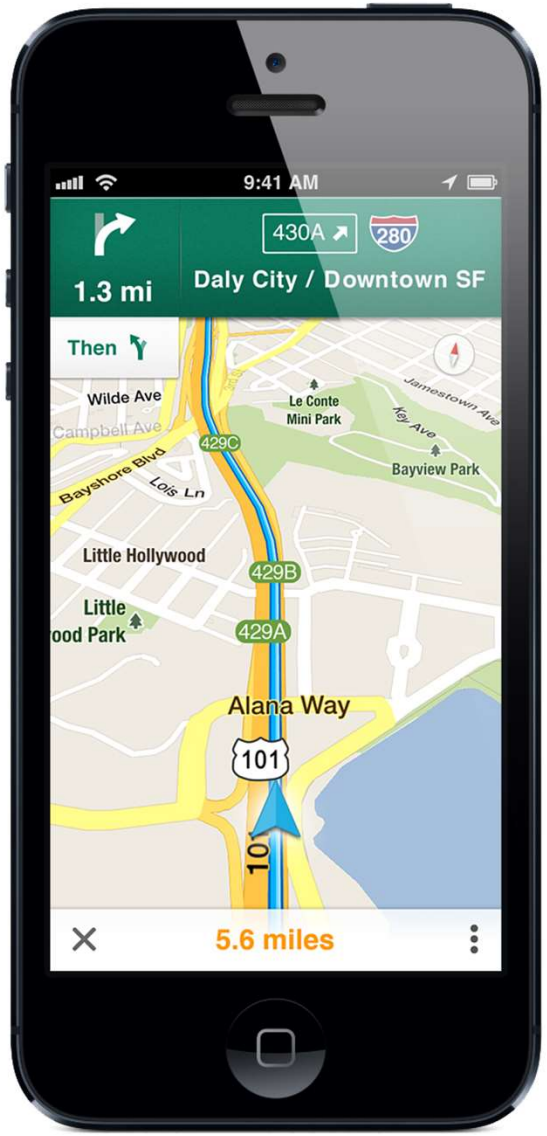


PROJECTE 1
Navegador

SO: Sessió d'Orientació
Projecte 1 - Part 2

Intel·ligència Artificial

2023-2024

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 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 **2a Part de la pràctica** ens centrarem en els mètodes de

Cerca Informada i Cerca Òptima

2. Fitxers necessaris

Exactament igual que a la Part 1

1. 📁 **CityInformation:** Files representing the map of the city.

   A. InfoVelocity.txt: It contains information about the **speed** at which each metro line travels.
   B. Stations.txt: It contains the **IDs** of each station, **name**, **line number** and coordinates where located.
   C. Time.txt: Table where we can see the **time** it takes to get from one station to another. There is no connection between two stations if the value of the table is zero.
   D. Lyon_city.jpg: Image of the city map.

```
Vel. line 1 : 10
Vel. line 2 : 14
Vel. line 3 : 45
Vel. line 4 : 3
```

InfoVelocity.txt

1	MASSENA 1	67	79	
2	CHARPENNES	1	140	56
3	REPUBLIQUE	1	167	64
4	LE TONKIN	2	140	27
5	CHARPENNES	2	140	56
6	COLLEGE BELLECOMBE	2	140	115
7	THIERS-LAFAYETTE	2	140	157
8	PART-DIEU	2	108	206
9	PARTDIEU SERVIENT	2	82	217
10	CHARPENNES	3	140	56
11	BROTTEAUX	3	108	134
12	PART-DIEU	3	108	206
13	PART-DIEU	4	108	206
14	DAUPHINE LACASSAGNE	4	152	230

Stations.txt

```
0.00000 9.05376 0.00000 0.00000 0.00000 0.00000 0.00000
9.05376 0.00000 4.21603 0.00000 20.00000 0.00000 0.00000
0.00000 4.21603 0.00000 0.00000 0.00000 0.00000 0.00000
0.00000 0.00000 0.00000 0.00000 5.42857 0.00000 0.00000
0.00000 20.00000 0.00000 5.42857 0.00000 7.14286 0.00000
0.00000 0.00000 0.00000 0.00000 7.14286 0.00000 4.21429
0.00000 0.00000 0.00000 0.00000 0.00000 4.21429 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 6.03739
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000
0.00000 15.00000 0.00000 0.00000 8.00000 0.00000 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000
```

Time.txt

Lyon_city.jpg

2. Fitxers necessaris

És igual que a la part 1, però hi ha una nova versió disponible.
Heu d'actualitzar els següents fitxers

2. 📁 **Code:** Python files with functions for practice.

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**).
   - Path (Classe que guarda la informació sobre una **ruta** o successió de parades).
C. TestCases.py: Arxiu per comprovar si les funcions que programeu donen el **resultat esperat**.
D. SearchAlgorithm.py: Arxiu on haureu de programar tota la pràctica.

3. Preliminars

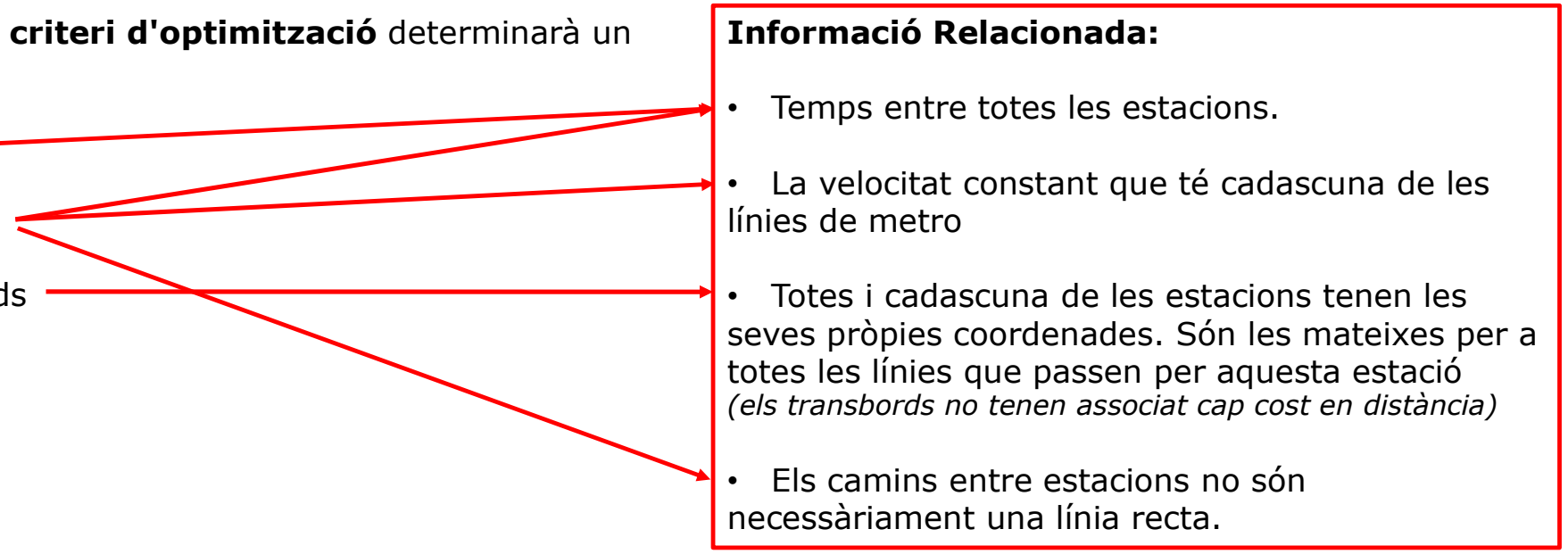
Recomanem que abans de començar a programar tingueu ben clars el següents conceptes:

Cost & Heurística

El **Cost** del camí el representem amb la **funció g**. Heu de veure quina informació del Mapa necessiteu per al seu càlcul:

L'elecció del **criteri d'optimització** determinarà un o altre cost:

- Temps
- Distància
- Transbords
- …



3. Preparació

Recomanem que abans de començar a programar tingueu ben clars el següents conceptes:

Cost & Heurística

El **Cost** del camí el representem amb la **funció g**. Heu de veure quina informació del Mapa necessiteu per al seu càlcul:

Informació Relacionada:

- Temps entre totes les estacions.
- La velocitat constant que té cadascuna de les línies de metro
- Totes i cadascuna de les estacions tenen les seves pròpies coordenades. Són les mateixes per a totes les línies que passen per aquesta estació.
  (*els transbords no tenen associat cap cost en distància)*
- Els camins entre estacions no són necessàriament una línia recta.

Distància real entre les dues estacions (considerant el temps entre una i altra i la velocitat de la línia que les connecta)

class **Map**:

Atributs

self.stations = {}
self.connections = {}
self.velocity = {}

Funcions

add_station(self, id, name, line, x,
add_connection(self, connections)
combine_dicts(self)
add_velocity(self, velocity)

```
0.00000 9.05376 0.00000 0.00000 0.00000 0.00000 0.00000
9.05376 0.00000 4.21603 0.00000 20.00000 0.00000 0.00000
0.00000 4.21603 0.00000 0.00000 0.00000 0.00000 0.00000
0.00000 0.00000 0.00000 0.00000 5.42857 0.00000 0.00000
0.00000 20.00000 0.00000 5.42857 0.00000 7.14286 0.00000
0.00000 0.00000 0.00000 0.00000 7.14286 0.00000 4.21429
0.00000 0.00000 0.00000 0.00000 0.00000 4.21429 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 6.03739
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000
0.00000 15.00000 0.00000 0.00000 8.00000 0.00000 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000
```

Time.txt

```
Vel. line 1 : 10
Vel. line 2 : 14
Vel. line 3 : 45
Vel. line 4 : 3
```

InfoVelocity.txt

1	MASSENA 1	67	79	
2	CHARPENNES	1	140	56
3	REPUBLIQUE	1	167	64
4	LE TONKIN	2	140	27
5	CHARPENNES	2	140	56
6	COLLEGE BELLECOMBE	2	140	115
7	THIERS-LAFAYETTE	2	140	157
8	PART-DIEU	2	108	206
9	PARTDIEU SERVIENT	2	82	217
10	CHARPENNES	3	140	56
11	BROTTEAUX	3	108	134
12	PART-DIEU	3	108	206
13	PART-DIEU	4	108	206
14	DAUPHINE LACASSAGNE	4	152	230

Stations.txt

3. Preparació

Recomanem que abans de començar a programar tingueu ben clars el següents conceptes:

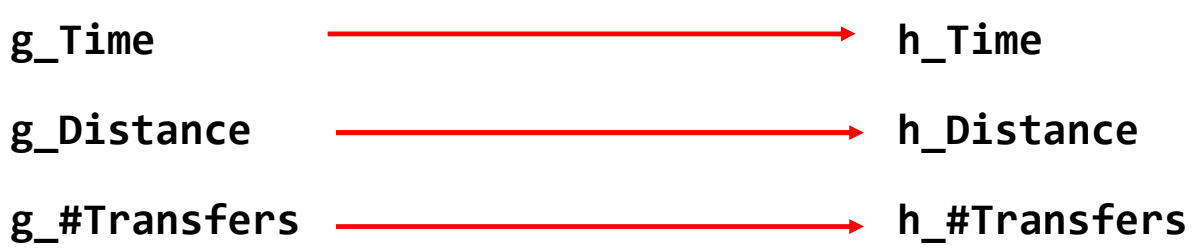
Cost & Heurística

L'Heurística del camí la representem amb la **funció h.** És una representació del cost entre el node actual i el node destí

L'elecció del **criteri d'optimització** determinarà un o altre valor de **h**:

- Temps
- Distància
- Transbords
- ...

Per a cada **Cost** tenim la seva pròpia **Heurística:**



4. Què s'ha de programar?

Funcions que heu de programar per aquesta primera part d'aquesta pràctica:

4.1 Algorisme de Cerca de cost uniforme

- Calculate_cost
- Insert_Cost
- Uniform_cost_search

4.2 Algorisme de cerca A*

- Calculate_heuristics
- Update_f
- Remove_redundant_paths
- Insert_cost_f
- A_star

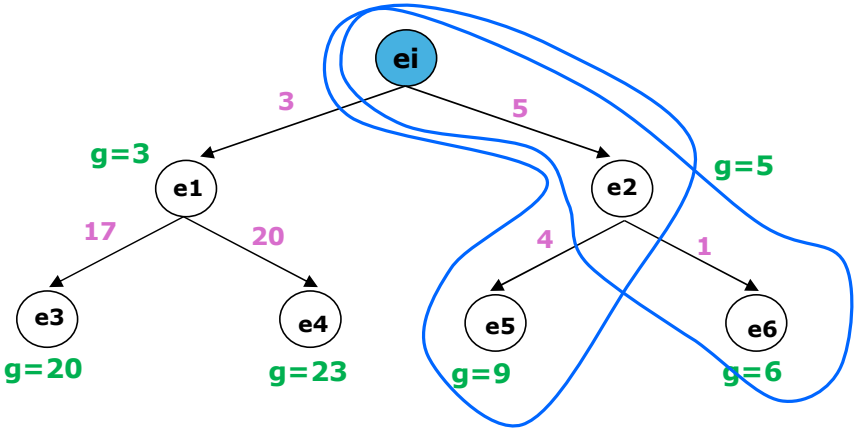
5 Una funció addicional: coordenades *enlloc d'*estacions

- Astar_improved

4.1 Cerca de Cost Uniforme

Calculate_cost

Sessió de Teoria



$$Path\ cost = \sum_{\forall i} C_i$$

Sessió de Pràctica

Per a cadascun dels Path fills, calculeu el cost entre la penúltima estació i el node actual actualitzant el valor del cost total.

Input:
- Llista de *Path* fills
- Mapa
- Elecció de Criteri (Adjacència per defecte)

Retorna:
- Una llista de *Path* resultat de l'expansió amb el cost actualitzat

Exemple de crida:
- `calculate_cost([[14,13,8,12,8],[14,13,8,12,11], [14,13,8,12,13]],Map, 1)`

Retorna:
- `[[14,13,8,12,8,g=12],[14,13,8,12,11,g=2.6],[14, 13,8,12,13,g=15]]`

*** Pista: Feu servir la funció "update_g" de la classe Path.

4.1 Cerca de Cost Uniforme

Insert_Cost

Sessió de Teoria

```
Funció CERCA_cost_uniforme (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=Inserció_ordenada_g(E,Cua(Llista));
3. Ffinsque;
4. Si (Llista<>NIL) Retornar(Cap(Llista));
5. Sinó Retornar(“No existeix Solucio”);
Ffuncio
```

Sessió de Pràctica

Fer la unió de la llista de camins expandits amb la llista de camins explorats.

Input:

- Llista de *Path* expandits
- Llista de *Path* a explorar

Retorna:

- Llista de *Path expandits (*Els camins són inserits en ordre de cost ascendent)

Exemple de crida:

- `insert_cost([[14,13,8,7,g=3],[14,13,8,9,g=3],[14,13,8,12,g=3]],[[14,13,12,g=2]])`

Retorna:

- `[14,13,12],[14,13,8,7],[14,13,8,9],[14,13,8,12]`

4.1 Cerca de Cost Uniforme

Uniform_Cost_Search

Sessió de Teoria

```
Funció CERCA_cost_uniforme (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=Inserció_ordenada_g(E,Cua(Llista));
3. Ffinsque;
4. Si (Llista<>NIL) Retornar(Cap(Llista));
5. Sinó Retornar(“No existeix Solucio”);
Ffuncio
```

Sessió de Pràctica

Trobar la ruta òptima entre dues estacions fent servir l'estratègia de **cerca de cost uniforme**.

Input:
- ID Estació origen
- ID Estació destí
- Mapa
- Elecció de Criteri (Adjacència per defecte)

Retorna:
- El *Path q*ue representa la ruta òptima

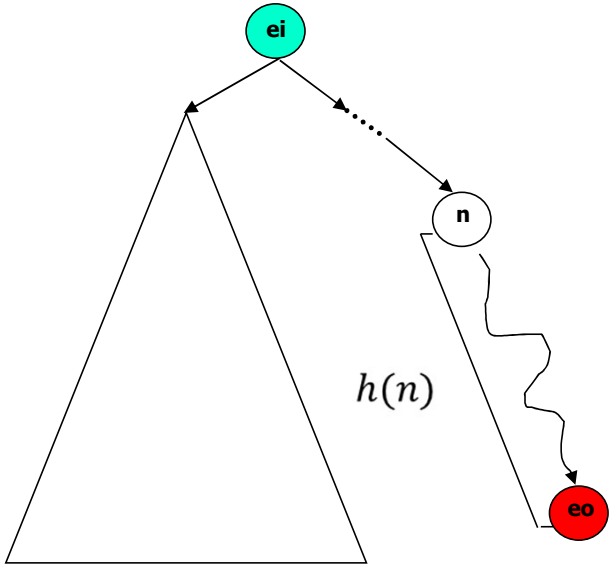
Exemple de crida:
- `uniform_cost_search(14,7,Map,1)`

Retorna:
- `[14,13,8,7]`

4.2 Cerca A*

Calculate_heuristics

Sessió de Teoria



Sessió de Pràctica

Segons l'elecció del criteri, calcular i actualitzar el valor de **l'heurística** d'uns camins donats.

Input:

- ID Estació origen
- ID Estació destí
- Mapa
- Elecció de Criteri

Retorna:

- Llista de Path expandits amb el valor de **h** actualitzat.

Exemple de crida:

- `calculate_heuristics([[14,13,8,7],[14,13,8,9],[14,13,8,12]],Map,9,1)`

Retorna:

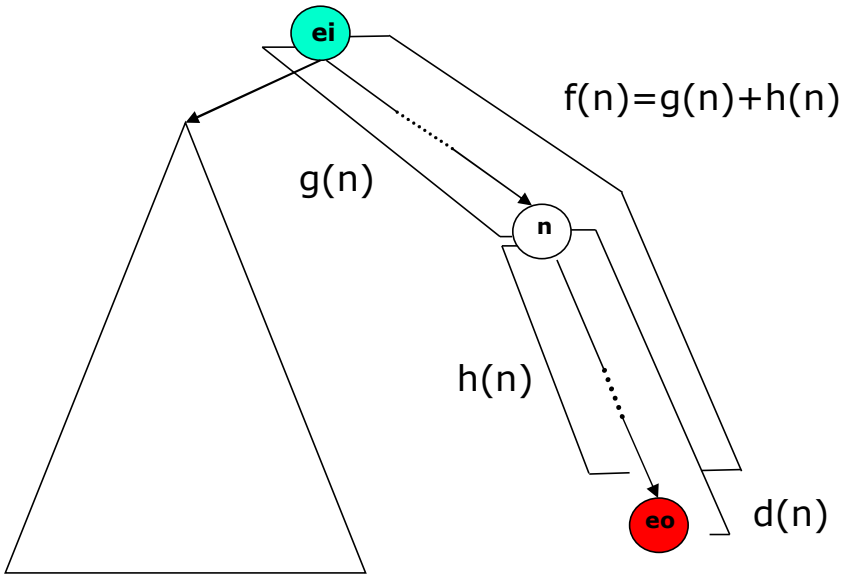
- `[[14,13,8,7,h=1.85],[14,13,8,9,h=0],[14,13,8,12,h=0.63]]`

*** Pista: Feu servir la funció "update_h" de la classe Path.

4.2 Cerca A*

Update_f

Sessió de Teoria



Sessió de Pràctica

Segons l'elecció del criteri, calcular i actualitzar el valor del **cost** d'uns camins donats.

Input:
- Llista de Path expandits

Retorna:
- Llista de Path expandits amb el valor de **f** actualitzat.

Exemple de crida:
- `Update_f([[8,12,11,f=2],[8,12,13,f=2])`

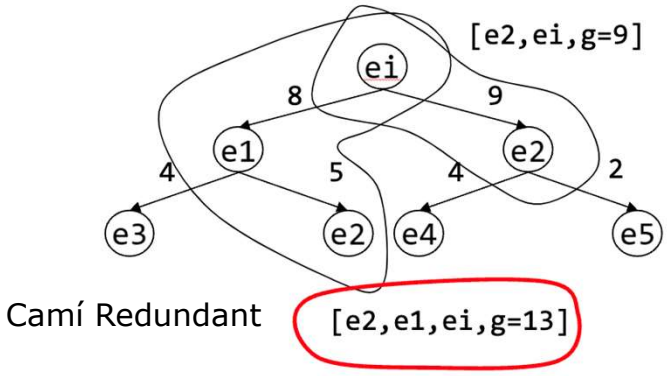
Retorna:
- `[[8,12,11,f=3],[8,12,13,f=3]`

*** Pista: Feu servir la funció "update_f" de la classe Path.

4.2 Cerca A*

Remove_redundant_paths

Sessió de Teoria



Els sub-camins d'un camí òptim són també òptims

El camí òptim no pot ser redundant

Sessió de Pràctica

Comproveu si el node expandit ja s'ha explorat abans.
Si és així i el cost anterior és pitjor, actualitzeu el nou cost.
Si no és així, elimineu el nou camí de la llista de camins expandits

Input:

- Llista de Paths expandits
- Llista de Paths de l'arbre explorat.
- Diccionari d'estacions visitades amb cost.

Retorna:

- Llista de Paths expandits sense camins redundants
- Llista de Paths de l'arbre explorat sense camins redundants
- Diccionari d'estacions visitades amb costos actualitzats

Exemple de crida:

- `remove_redundant_paths([[12,8],[12,11],[12,13]],[],{12:0})`

Retorna:

- `[[12,8,7],[12,8,9]],[[12,8,9],[12,13],[12,11,10,5],[12,8,7],[12,11,10,2]],{12:0,8:12,11:2,6,...}`

4.2 Cerca A*

Insert_cost_f

Sessió de Teoria

```
Funció CERCA_A* (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=Inserció_ordenada_f(E,Cua(Llista));
3.   Ffinsque;
4.   Si (Llista<>NIL) Retornar(Cap(Llista));
5.   Sinó Retornar("No existeix Solucio");
Ffuncio
```

Cerca A* ⟶ Inserció_Ordenada_**f**(E,Cua(Llista));

Sessió de Pràctica

Inseriu els camins expandits ordenats en funció del cost total estimat a la llista de camins explorats. El millor cost total estimat va davant de tot de la llista.

Input:
- Llista de Paths expandits
- Llista de Paths de l'arbre explorat

Retorna:
- Llista de camins explorats incloent els camins expandits tots ordenats segons el cost total estimat (f)

Exemple de crida:
- `Insert_cost_f([[12,11,10]],[[12,8],[12,13]])`

Retorna:
- `[[12,11,10,f=8.6],[12,8,f=12.6],[12,13,f=15.6]]`

4.2 Cerca A*

A_star

Sessió de Teoria

```
Funció CERCA_A* (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=Inserció_ordenada_f(E,Cua(Llista));
3.   Ffinsque;
4.   Si (Llista<>NIL) Retornar(Cap(Llista));
5.   Sinó Retornar("No existeix Solucio");
Ffuncio
```

Sessió de Pràctica

Trobar la ruta òptima entre dues estacions fent servir l'estratègia de **cerca d'A*:**

Input:

- ID Estació origen
- ID Estació destí
- Mapa
- Elecció de Criteri (Adjacència per defecte)

Retorna:

- El *Path q*ue representa la ruta òptima

Exemple de crida:

- `AStar(14,7,Map,1)`

Retorna:

- `[14,13,8,7]`

4.3 Cerca A* millorada

Astar_improved (coordenades enlloc d'estacions)

Nou Problema

Posició Inicial (X_i,Y_i) →  → Posició Objectiu (X_o,Y_o)

Simplificacions del problema

- L'usuari camina en línia recta a una **velocitat igual a 5**
- L'usuari **pot anar caminant a** qualsevol de les estacions del mapa i també a les coordenades de destí
- L'usuari **pot anar caminant des de** les coordenades d'origen i des de qualsevol de les estacions a les coordenades de destí
- L'usuari **no pot caminar d'una estació de metro a una altra.**

Sessió de Pràctica

Implementació del l'A* on la posició de l'usuari be donada en coordenades i no pas amb estacions

Input:
- Coordenades d'origen
- Coordenades de destí
- Mapa

Returns:
- El *Path* òptim en **temps** per a la ruta entre les coordenades d'origen i les coordenades de destí

Calling:
- `Astar_improved([100,50],[60,180],Mapa)`

Returns:
- `[0,10,11,-1]`

5. Entrega de la Part 2

Per a l'avaluació d'aquesta segona 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 author a l'inici de l'arxiu).

L'entrega s'ha de fer abans del dia **17/03/2023 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. Si no ho poseu bé la nota serà un 0.

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. Per evitar que el codi entri en bucles infinits hi ha un **límit de temps** per a cada exercici, per tant si les vostres funcions triguen massa les considerarà incorrectes.

Recordeu el que es diu a la **guia docent** sobre **copiar o deixar copiar ....**

Sense perjudici d'altres mesures disciplinàries que s'estimin oportunes, i d'acord amb la normativa acadèmica vigent, les **irregularitats comeses per l'alumnat** que puguin conduir a una variació de la qualificació es qualificaran amb un zero (0). Les activitats d'avaluació qualificades d'aquesta forma i per aquest procediment no seran recuperables. Si és necessari superar qualsevol d'aquestes activitats d'avaluació per aprovar l'assignatura, aquesta assignatura quedarà suspesa directament, sense oportunitat de recuperar-la en el mateix curs. Aquestes irregularitats inclouen, entre d'altres:

- còpia total o parcial d'una pràctica, informe, o qualsevol altra activitat d'avaluació;
- deixar copiar;
- ús no autoritzat i/o no referenciat de la IA (p. ex,Copilot, ChatGPToequivalents) per a resoldre exercicis, pràctiques i/o qualsevol altra activitat avaluable;
- presentar un treball de grup no fet íntegrament pels membres del grup;
- presentar com a propis materials elaborats per un tercer, encara que siguin traduccions o adaptacions, i en general treballs amb elements no originals i exclusius de l'alumnat.
- tenir dispositius de comunicació (com telèfons mòbils, smart watches, etc.) accessibles durant les proves d'avaluació teòric-pràctiques individuals (exàmens).

En resum: **copiar, deixar copiar o plagiar** en qualsevol de les activitats d'avaluació equival a un SUSPENS amb **nota inferior o igual a 3,0.**