INTEL·LIGÈNCIA ARTIFICIAL

PRÀCTICA 1:

CERCA INFORMADA

Raúl Martín Morales

Jan Torres Rodríguez

Intel·ligència Artificial

Curs 2024-25

Benet Manzanares

Índex

1. Formalització problema

El problema principal consisteix en trobar el camí més ràpid per moure's des de un node inicial (xIni, yIni) cap a un node final (xFin, yFin) en un mapa representat per una matriu 10 x 10. Cada node té un valor d'altura associat, on alguns poden tenir precipicis de valor –1. Els moviments que es permeten son horitzontals i verticals.

Cal definir els diferents tipus d'estats, que en aquest cas seran els moviments a les diferents caselles del mapa per les que es pot passar. Cadascuna té un valor que determina la seva alçada, delimitant el temps que es triga en arribar a dita casella des d'una altra.

Aquest cost de temps es calcularà segons les següents condicions:

- Si la diferència d'altura entre nodes es positiva o 0, el cost es de 1 + la diferencia d'altures

- Si la diferència d'altura es negativa, el cost serà sempre ½.

- I es prohibirà el pas per nodes amb precipicis.

Un estat esta definit com a un node, que te com atributs la seva posició a la matriu, la seva altura i el camí o distancia recorreguda fins a arribar a aquest node. També hem decidit guardar junt al Node el seu predecessor i els seus adjunts per facilitar el treball en futurs càlculs.

Establert el funcionament bàsic, es tenen l'estat inicial i final com els nodes inicial i final, respectivament. Els atributs a tenir en compte de cada estat serà l'alçada, que determinarà més endavant el temps emprat en arribar-hi, i aquelles caselles a les que s'hi pot arribar des d'una determinada posició, tenint que els moviments han de ser en horitzontal o vertical.

Estats:

Un estat serà tot aquell que indiqui una posició determinada del mapa. Tindrà els següents atributs

- Posició
- Valor d'alçada
- Nº d'estats recorreguts per arribar a l'estat
- Temps acumulat

Coneixent els atributs, l'estat inicial serà aquell en el que es comenci, per exemple:

Estat Inicial

- (0,0)
  - 0
  - 0
  - 0

L'estat final tindrà els atributs per definir fins que no s'hi arribi des d'un altre estat. És a dir:

Estat Final

- (9,9)
  - 5
- No definit
- No definit

Operadors:

Op1: Moviment cap a la dreta (x, y+1)

Op2: Moviment cap a la esquerra (x, y-1)

Op3: Moviment cap a dalt (x-1, y)

Op4: Moviment cap a baix (x+1, y)

Restriccions: No moure's a un precipici i no sortir del domini del mapa.

2. Heurístiques

Per tal de dur una cerca més eficient, hem hagut de dissenyar tres heurístiques diferents que es basen en diferents informacions que estimaran un cost per decidir quin es millor camí.

HEuclidiana.java (Distància Euclidiana + Penalització pel Camí Recorregut)

Aquesta heurística utilitza la formula h(n) = sqrt((xf – x)^2 + (yf - y)^2) + |y.getDist|/ 2

On es calcula la distància en línia recta entre el node inicial i el final i se li suma un factor addicional que te en comtes la distancia recorreguda fins el node(nodes recorreguts), multiplicat per 0,5. Aquesta heurística serà més eficient en terrenys amb pocs precipicis o pocs canvis bruscos de altura. I tindrà un millor resultat quan el camí sigui diagonal.

Admissibilitat: Considerem que sí es admissible, aj qeu només afegeix una petita penalització proporcional a la distancia recorreguda, sempre ens dona una bona estimació del camí més curt.

HManhattan.java (Distància Manhattan + Cost de Temps)

Aquesta heurística utilitza la formula h(n) = |xf– x| + |yf - y| + temps

Es calcula la distancia entre el node actual i el objectiu utilitzant la formula anterior, a més se li suma la diferencia de altura entre aquets nodes, que ve a sent el temps que triga, simulant que pujar costa mes que baixar. Aquesta heurística seria més eficient amb terrenys amb poca variació d'altures.

Admissibilitat: Considerem que és admissible però no sempre pot serà admissible, com que penalitza massa algunes pujades, a vegades pot fer que l'algoritme pensi que un camí és pitjor del que realment és, i fa que A* explori més nodes del necessari.

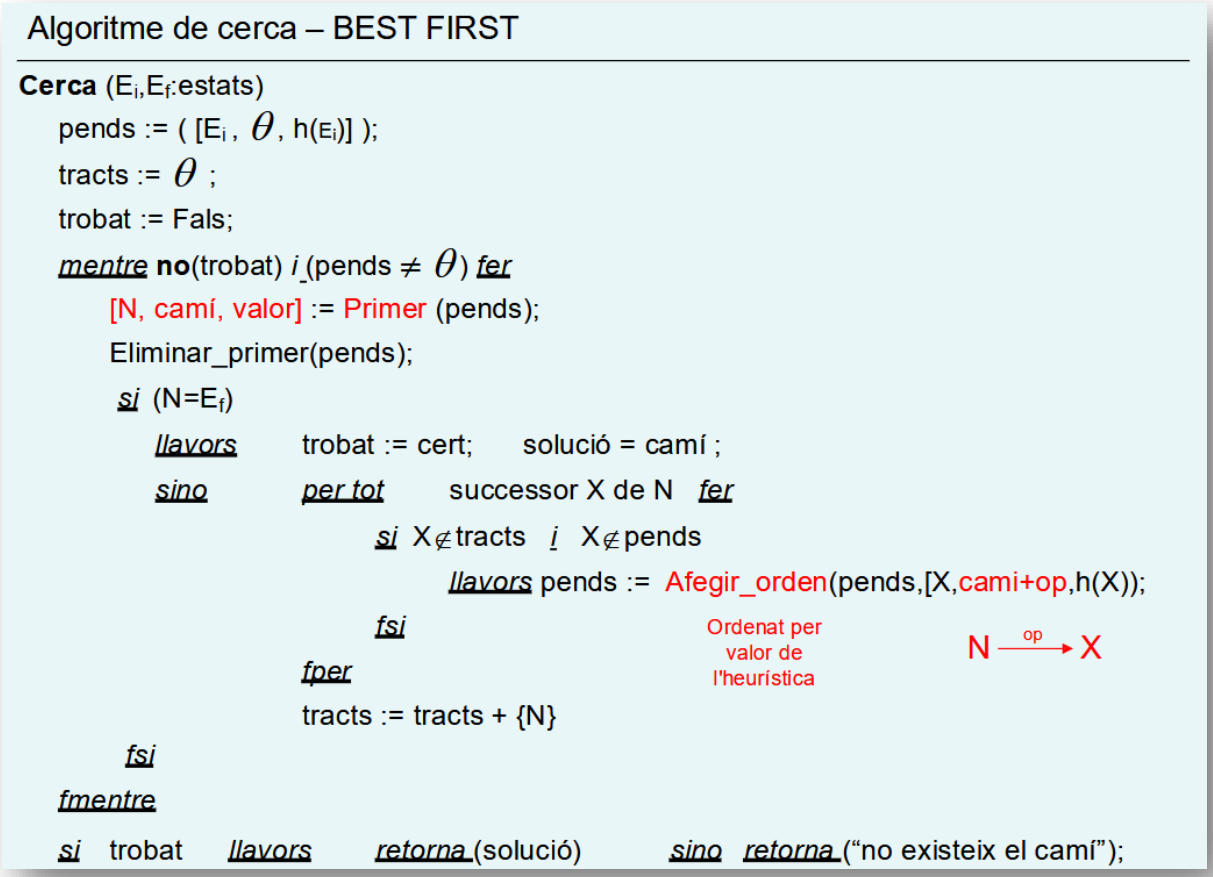
HAltura.java (Distància Manhattan + Penalització per Alçada)

En aquesta heurística utilitzem la formula de Manhattan, a més li sumem un factor de la diferència d'altura per 0,5. Aquesta heurística penalitzarà més els camins amb grans canvis d'altura, ja que afegeix un cost addicional d'aquesta diferencia d'altura. Seria útil en terrenys on l'altura influeix significativament en el cost del camí.

Admissibilitat: Considerem que aquesta heurística no es admissible, ja que penalitza massa camins amb pujades, a vegades pot fer que A* descarti un camí que en realitat era millor.

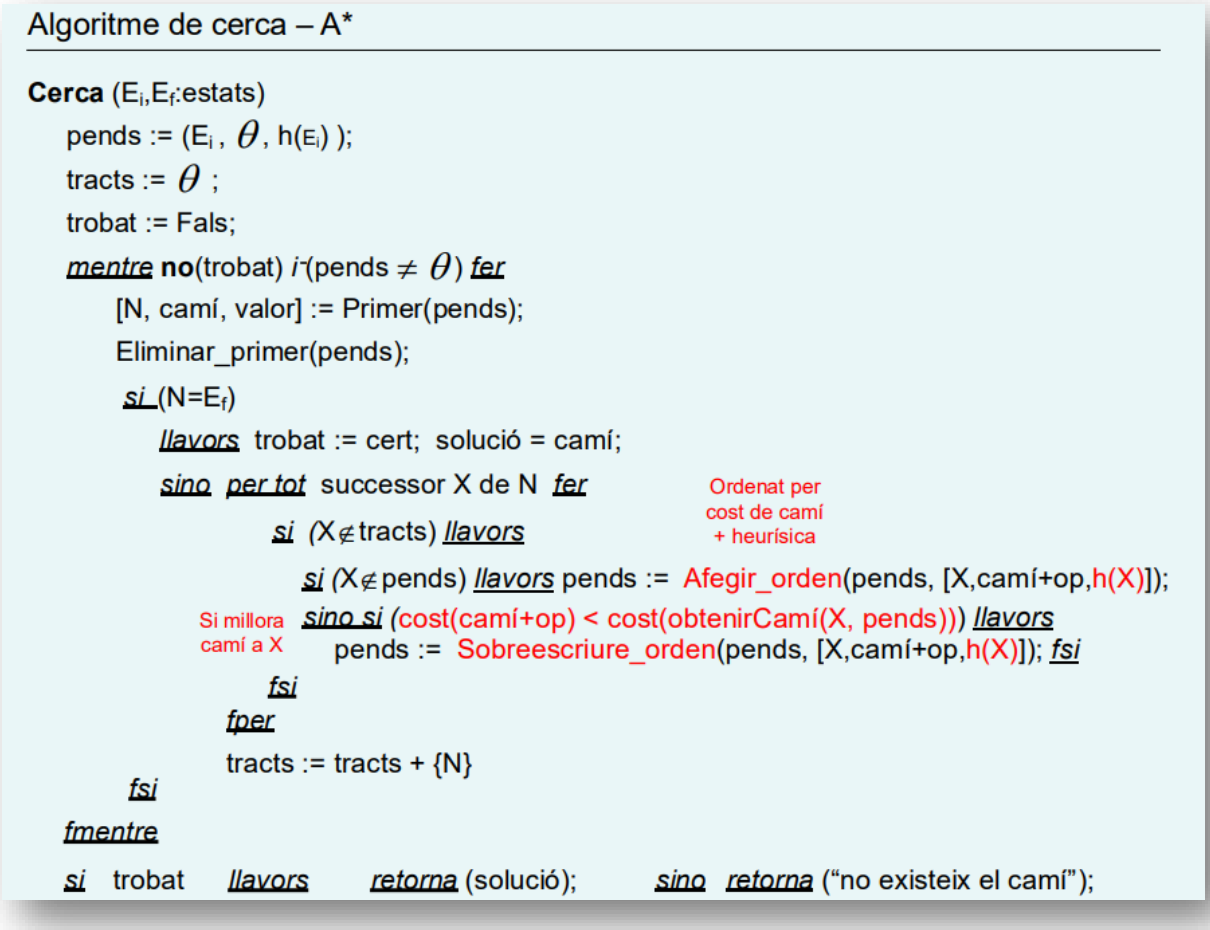
3. Implementació d'Algorismes

Els dos algorismes emprats han estat Best-First i A*. S'ha utilitzat el pseudocodi aportat en les transparències de teoria per a implementar-los. Aquest és el següent:



Pseudocodi 1. Best-First. Tipus d'algoritmes de cerca

Allò més important que s'ha extret del pseudocodi no és només el seu funcionament, sinó també les estructures que necessita per a funcionar correctament. Aquestes són una llista d'elements propis que gestiona els elements pendents a tractar. Aquests contenen l'estat, la distància recorreguda per arribar-hi i el valor de l'heurística d'aquella posició en concret. A més a més es té una llista amb el mateix funcionament que s'encarrega d'emmagatzemar els elements ja tractats.



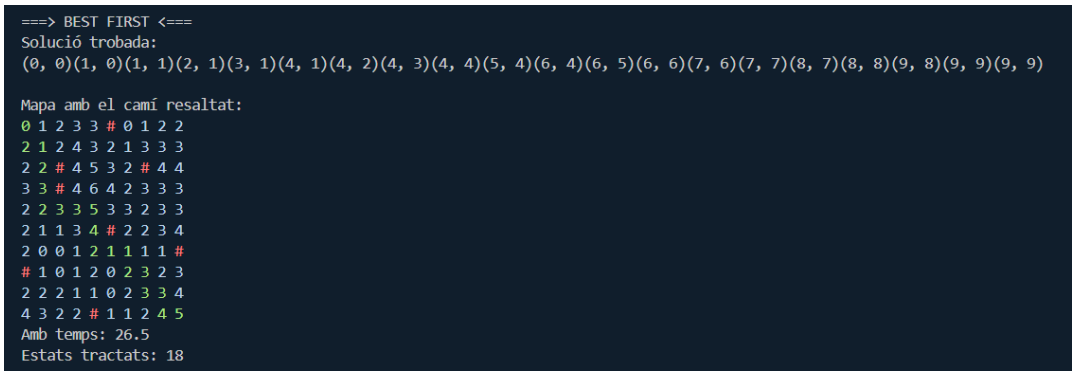
Pseudocodi 2. A. Tipus d'algoritmes de cerca*

Algoritme utilitza una combinació del cost acumulat g(n) i la estimació heurística h(n), on s'obté f(n)= g(n) + h(n), on voldrem el menor cost possible. L'estratègia principal del algoritme en una llista 'pends' que conté els nodes pendents d'explorar, ordenats segons el cost total estimat, y una llista 'tracts' que conté els nodes que ja han sigut processats i per tant no es tornaran a visitar. Per tractar els nodes del voltant, es generant tots el successors 'X' de N i es comprova si ja han estat tractats. Si 'X' no estava en 'pends', s'afegeix amb el seu cost estimat, de manera ordenada, si 'X' ja estava, es revisa si el nou camí generat per aquest node te un millor cost que el que havia establert. I per últim 'N' s'afegeix a 'tracts' per evitar tornar-lo a visitar. I tot això es repetirà fins trobar la solució.

Com a diferència entre els dos algoritmes es que el algoritme Best-First utilitza només la heurística h(n) per decidir quin node explorar, mentre que A* combina la heurística amb el cost del camí. Això fa que A* sigui més eficient per trobar el camí més curt, però alhora explora més nodes que Best-First.

4. Proves dels Algoritmes i Anàlisi d'Heurístiques

Mapa1 Best-First HEuclidiana:



Observem que la solució trobada es de 26,5 i veiem que a trobat una millor solució que la A*, on això sol ser atípic j a que normalment A* hauria de trobar la millor solució sempre, i encara que hagi trobat una millor solució ara, no podem garantir que sempre ho faci, la seva eficiència ha estat condicionada per la Heuristica Euclidiana que le ha guiat en el camí, ha prioritzat el camí en línia recta calculant la distancia entre els dos punts, i no ha explotat gaire opcions, i com BestFirst nomes utilitza la heurística per decidir el següent node, no ha quedat atrapat en rutes subòptimes. S'han tractat 17 estats, molts menys que a A*, indicat que la Euclidiana ha permes a Best First trobar un camí eficient sense analitzar gaire opcions. Si el mapa tingues desnivells es probable que la solució hagués estat pitjor que A*. Considerem que ha trobat una solució optima, ja que ha trobat una millor que A*.

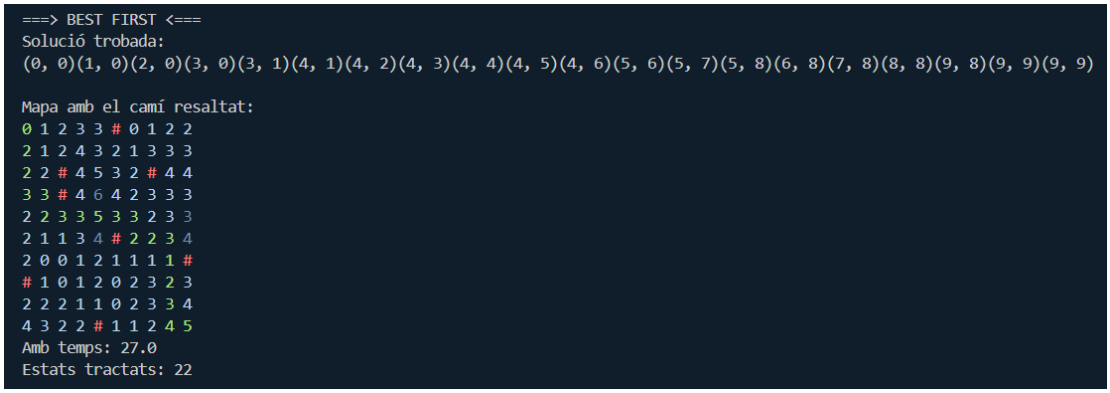
Mapa1 A* HEuclidiana:

```
===> A* <===
Solució trobada:
(0, 0)(1, 0)(1, 1)(1, 2)(1, 3)(2, 3)(3, 3)(3, 4)(4, 4)(4, 5)(4, 6)(5, 6)(6, 6)(7, 6)(7, 7)(7, 8)(8, 8)(8, 9)(9, 9)(9, 9)

Mapa amb el camí resaltat:
0 1 2 3 3 # 0 1 2 2
2 1 2 4 3 2 1 3 3 3
2 2 # 4 5 3 2 # 4 4
3 3 # 4 6 4 2 3 3 3
2 2 3 3 5 3 3 2 3 3
2 1 1 3 4 # 2 2 3 4
2 0 0 1 2 1 1 1 1 #
# 1 0 1 2 0 2 3 2 3
2 2 2 1 1 0 2 3 3 4
4 3 2 2 # 1 1 2 4 5
Amb temps: 27.0
Estats tractats: 74
```

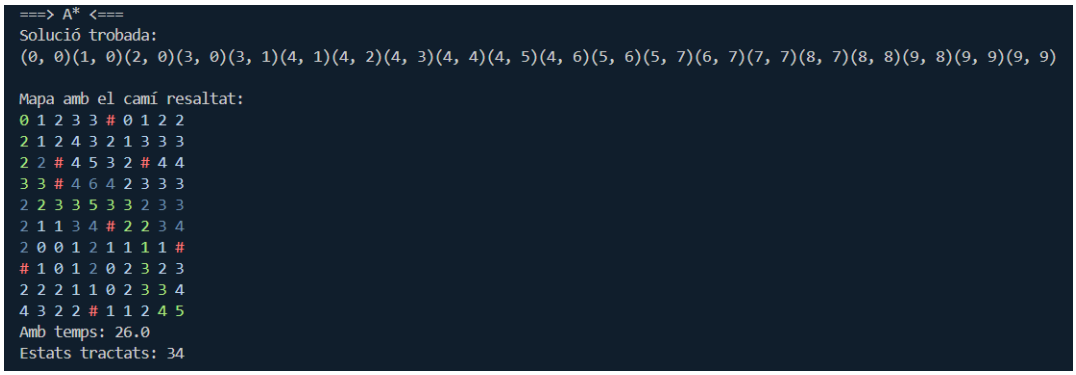
Observem que la solució trobada te un temps de 27, una moca pitjor que en la de BestFirst, com A* sempre garanteix la millor solució respecte la heurística utilitzada, això indica que la Heurística Euclidiana no ah estat una bona heurística per aquest mapa. També s'observa que la Euclidiana explora massa opcions innecessàries a causa d'aquesta heurística. Com prioritza les línies rectes i ignora característiques del mapa, ha fet que A* explori massa estats intentant seguir un camí que no ha estat el ideal. Ha explorat 74 estats, molts mes que a BesrFirst, que explori molts camins abans de prendre la decisió normalment es una bona aplicació, però no ha valgut la pena aquesta vegada. Com el mapa te desnivells i precipicis, cosa que fa que la línia recta no sigui sempre el millor camí. Considerem que la solució trobada no ha sigut la mes optima, i es veu clarament que la heurística ha afectat notòriament.

Mapa1 Best-First HManhattan:



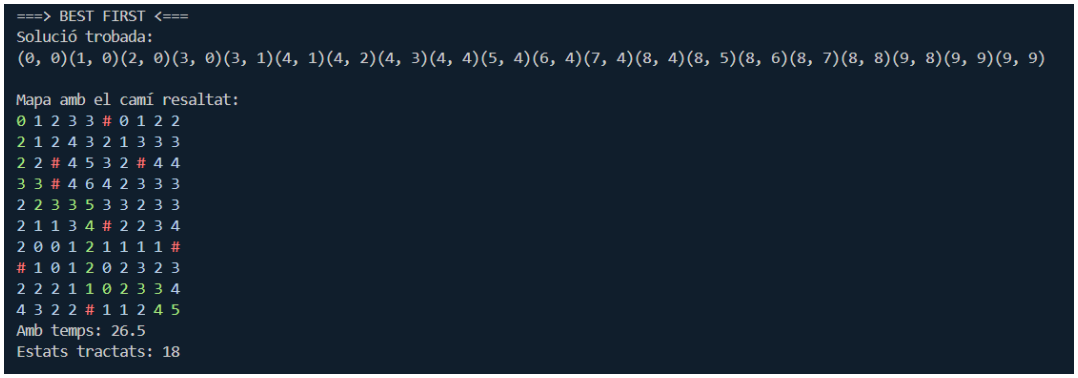
S'observa que el temps ha sigut de 27, i 22 estats tractats, on BestFirst no ha trobat la millor solució possible, ja que el seu camí es ligerament mes costos que el de A*. A causa de que BestFirst si veu un camí millor en un pas ho escull, sense pensar en el cost total del camí. Como el mapa te desnivells i precipicis, BestFirst ha evitat camins amb moltes pujades però sense mirar realment si això li portava un temps total menor. I com la heurística utilitzada no es admissible ja que subestima el cost, utilitzant el mateix càlcul per la heurística que pel cost de temps, aleshores no es considera una solució optima respecte el temps, ja que es deixa portar per la heurística sense considerar el cost real del camí.

Mapa1 A* HManhattan:



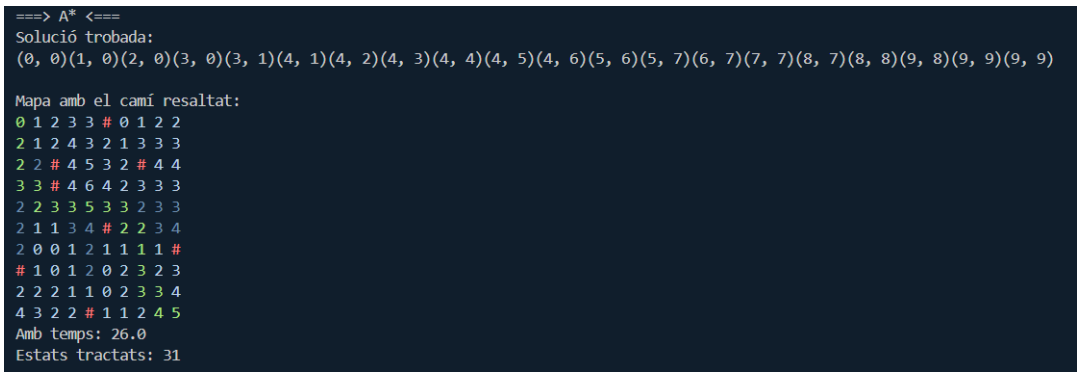
S'observa que hem tractat 34 nodes abans de trobar la solució, això indica que el algoritme ha avaluat més opcions abans de decidir-se per el millor camí, ja que en el cas de la heurística utilitzada fem servir el moviment i les altures. Però com la heurística utilitzada no es admissible per tant no podem dir que la solució trobada es la més optima en termes de temps. A diferencia del BestFirst, ell ha explorat mes camins abans de decidir quin era millor.

Mapa1 Best-First HAltura:



Observem que en aquest cas A* ha trobat una solució lleugerament millor amb 26, cosa que indica que la solució de BestFirst no es optima però es molt positiva. Com Best First nomes segueix la heurística sense tenir en compte el cost acumulat del camí, pot prendre decisions que semblen bones a curt termini, però no sempre son les millors a termes de temps total. La heurística HAltura ha guiat l'algoritme cap a camins que mantenen les altures el mes equilibrada possible, però ha fet que BestFirst nomes tractes 18 estats, un nombre molt baix. Ha sigut eficient en nombre d'estats tractats, però com no considera el cost acumulat, no pot considerar la solució optima. Hi ha desnivells al mapa pel que ha fet que la heurística funcioni bé, però te alguns precipicis cosa que limita les rutes possibles ja que usa Manhattan igualment.

Mapa1 A* HAltura:



Com A* busca sempre la solució amb menor cost si la heurística es admissible, hi ha trobat una millor que Best First, podem confirmar que ha trobat una solució optima respecte el temps, i ha sigut una heurística efectiva en aquest cas. L'ús de estat heurística podria ajudar a minimitzar l'impacte de les pujades i baixades, però sense diferenciar entre elles, i ha evitat exploracions innecessàries i permetre trobar el camí ràpid ja que nomes ha tractat 31 estats, menys estats respecte les altres heurístiques. El mapa ha sigut favorable per aquesta heurística, ha que no ha provocat un nombre excessiu d'estats explorats, el mapa te desnivells i ha funcionat bé.

Mapa2

El segon mapa s'ha creat amb un camí directe des de (0,0) a (9,9) que sembla òptim, però aquest camí està bloquejat per obstacles en posicions claus com (2,2) (4,4) y (6,6). A* trobarà una ruta alternativa rodejant obstacles i les zones de major altura.

Mapa2 BestFirst HEuclidiana:

```
===> BEST FIRST <===
Solució trobada:
(0, 0)(1, 0)(1, 1)(2, 1)(3, 1)(4, 1)(4, 2)(4, 3)(4, 4)(5, 4)(6, 4)(6, 5)(6, 6)(7, 6)(7, 7)(8, 7)(8, 8)(9, 8)(9, 9)(9, 9)

Mapa amb el camí resaltat:
0 1 2 1 0 # 0 1 2 2
0 1 2 2 1 2 1 2 2 2
0 2 # 1 # 2 2 # 2 2
0 2 # 2 2 2 2 2 2 2
0 2 2 3 2 2 2 2 2 2
0 1 # 4 2 # 2 2 2 2
0 0 # 1 2 1 1 1 1 #
# 0 # 0 0 0 2 2 1 1
0 0 # 0 1 0 2 1 3 3
1 0 0 0 # 0 0 1 3 1
Amb temps: 22.0
Estats tractats: 18
```

S'observa que ha trobat una solució lleugerament millor que A*, però això no significa que sigui optima en tots els casos. Atès que A* sempre garanteix el camí mes òptim quan la heurística es admissible, aquest resultat indica que la HEuclidiana ha afavorit a BestFirst en aquest cas. Es podria considerar que la solució es optima respecte el temps en aquest cas en específic. BestFirst ha pogut trobat una ruta mes rapida perquè la distancia euclidiana ha sigut una bona estimació en aquest mapa .Com que el mapa no te obstacles excessius i la diferencia d'altures no ha sigut determinant, ha estat una bona heurística en aquest cas. Només ha tractat 18 estats, cosa que indica que ha trobat una solució molt ràpidament, ha sigut una exploració mes eficient que la de A*. El mapa beneficia la heurística en BestFirst ja que la ruta optima es prou recta i la distancia euclidiana ha sigut una bona aproximació.

Mapa2 A* HEuclidiana:

```
===> A* <===
Solució trobada:
(0, 0)(1, 0)(1, 1)(1, 2)(1, 3)(2, 3)(3, 3)(4, 3)(4, 4)(5, 4)(6, 4)(6, 5)(6, 6)(7, 6)(7, 7)(8, 7)(8, 8)(8, 9)(9, 9)(9, 9)

Mapa amb el camí resaltat:
0 1 2 1 0 # 0 1 2 2
0 1 2 2 1 2 1 2 2 2
0 2 # 1 # 2 2 # 2 2
0 2 # 2 2 2 2 2 2 2
0 2 2 3 2 2 2 2 2 2
0 1 # 4 2 # 2 2 2 2
0 0 # 1 2 1 1 1 1 #
# 0 # 0 0 0 2 2 1 1
0 0 # 0 1 0 2 1 3 3
1 0 0 0 # 0 0 1 3 1
Amb temps: 22.5
Estats tractats: 69
```

Com que A* sempre busca la solució amb menor cost total, en aquest cas no ah pogut millorar el temps trobat per BestFirst, per això considerem que el camí no ha sigut el mes òptim, a causa de la heurística utilitzada. La HEuclidiana ha fet que explores un nombre més alt d'estats (69) sense torbar una millora en el temps total, i en aquest cas explorar més opcions no ha afavorit a A*. Segons el mapa, els obstacles no han creat desviacions significatives, permetent que BFirst trobés ràpidament un camí eficient.

Mapa2 Best-First HManhattan:

```
===> BEST FIRST <===
Solució trobada:
(0, 0)(0, 1)(0, 2)(1, 2)(1, 3)(2, 3)(3, 3)(4, 3)(5, 3)(5, 4)(6, 4)(6, 5)(6, 6)(7, 6)(8, 6)(8, 7)(8, 8)(9, 8)(9, 9)(9, 9)

Mapa amb el camí resaltat:
0 1 2 1 0 # 0 1 2 2
0 1 2 2 1 2 1 2 2 2
0 2 # 1 # 2 2 # 2 2
0 2 # 2 2 2 2 2 2 2
0 2 2 3 2 2 2 2 2 2
0 1 # 4 2 # 2 2 2 2
0 0 # 1 2 1 1 1 1 #
# 0 # 0 0 0 2 2 1 1
0 0 # 0 1 0 2 1 3 3
1 0 0 0 # 0 0 1 3 1
Amb temps: 23.5
Estats tractats: 18
```

Observem que la solució trobada per BestFirst no es la mes optima en termes de temps, ja que comparant amb A*, trobem una millor solució en A*. Best First ha trobat decisions locals basades en la heurística, sense tenir en compte el cost acumulat del camí. La heurística l'ha fet prendre decisions immediates sense verificar el camí serà realment òptim, en aquest cas ha escollit un camí que li semblava millor a en un principi però a la llarga es pitjor. Nomes ha tractat 18 estats, que indica que BestFirst ha trobat la solució ràpidament sense explorar gaire opcions, tot i acabar mas ràpid no ha sigut el millor camí. El mapa te desnivells i obstacles que obliga a explorar camins alternatius, la heurística ha ajudat a evitar pujades innecessàries, però Best Fist no ha fet la millor selecció global.

Mapa2 A* HManhattan:

```
===> A* <===
Solució trobada:
(0, 0)(0, 1)(0, 2)(1, 2)(1, 3)(1, 4)(1, 5)(2, 5)(3, 5)(3, 6)(4, 6)(5, 6)(5, 7)(6, 7)(7, 7)(8, 7)(8, 8)(9, 8)(9, 9)(9, 9)

Mapa amb el camí resaltat:
0 1 2 1 0 # 0 1 2 2
0 1 2 2 1 2 1 2 2 2
0 2 # 1 # 2 2 # 2 2
0 2 # 2 2 2 2 2 2 2
0 2 2 3 2 2 2 2 2 2
0 1 # 4 2 # 2 2 2 2
0 0 # 1 2 1 1 1 1 #
# 0 # 0 0 0 2 2 1 1
0 0 # 0 1 0 2 1 3 3
1 0 0 0 # 0 0 1 3 1
Amb temps: 22.0
Estats tractats: 35
```

Com que A* sempre garanteix la solució amb menor cost total, això confirma que aquesta es l'òptima respecte el temps. La heurística ha penalitzat les pujades i afavoreix les baixades, aleshores ha ajudat a A* a trobar la millor solució evitant pujades innecessàries. Ha tractat 35 estats, el doble que BestFirst , que això ha assegurat que trobés el millor camí. El mapa ha permès que A* trobi una solució millor explorant més estats, i la Heurística ha fet que sigui mes eficient per aquest mapa.

Mapa2 Best-First HAltura:

```
===> BEST FIRST <===
Solució trobada:
(0, 0)(0, 1)(1, 1)(2, 1)(3, 1)(4, 1)(5, 1)(6, 1)(7, 1)(8, 1)(9, 1)(9, 2)(9, 3)(8, 3)(8, 4)(8, 5)(9, 5)(9, 6)(9, 7)(9, 8)(9, 9)(9, 9)

Mapa amb el camí resaltat:
0 1 2 1 0 # 0 1 2 2
0 1 2 2 1 2 1 2 2 2
0 2 # 1 # 2 2 # 2 2
0 2 # 2 2 2 2 2 2 2
0 2 2 3 2 2 2 2 2 2
0 1 # 4 2 # 2 2 2 2
0 0 # 1 2 1 1 1 1 #
# 0 # 0 0 0 2 2 1 1
0 0 # 0 1 0 2 1 3 3
1 0 0 0 # 0 0 1 3 1
Amb temps: 24.0
Estats tractats: 20
```

Encara que A* trobi una solució millor, no significa que la solució trobada per BestFirst no sigui optima. Ha seguit una estratègia basada en la heurística, cosa que ha fet que el camí escollit sigui pitjor. La solució no es la millor però es pot considerar optima respecte els estats tractats, però no segons el temps ja que A* ha estat millor. Només ha tractat 20 estats, cosa que indica que BestFirts ha trobat la solució mes ràpidament però sense explorar prou opcions, tot i això l'algoritme li ha fet arribar a un camí mes costos. El mapa te desnivells i obstacles, cosa que ha fet que la heurística HAltura sigui útil, evitant pujades costoses, però no ha trobat la millor solució global. BestFirst ha seguit el camí que era millor a simple vista però a la llarga no ha sigut eficient.

Mapa2 A* HAltura:

```
===> A* <===
Solució trobada:
(0, 0)(0, 1)(0, 2)(0, 3)(1, 3)(2, 3)(3, 3)(3, 4)(4, 4)(5, 4)(6, 4)(6, 5)(6, 6)(6, 7)(7, 7)(8, 7)(9, 7)(9, 8)(9, 9)(9, 9)

Mapa amb el camí resaltat:
0 1 2 1 0 # 0 1 2 2
0 1 2 2 1 2 1 2 2 2
0 2 # 1 # 2 2 # 2 2
0 2 # 2 2 2 2 2 2 2
0 2 2 3 2 2 2 2 2 2
0 1 # 4 2 # 2 2 2 2
0 0 # 1 2 1 1 1 1 #
# 0 # 0 0 0 2 2 1 1
0 0 # 0 1 0 2 1 3 3
1 0 0 0 # 0 0 1 3 1
Amb temps: 22.5
Estats tractats: 73
```

A comparació con BestFirst, A* ha trobat una millor solució en termes de temps, com que A* sempre garanteix la solució amb menor cost total, això confirma que aquesta es la optima. HAltura ha penalitzat pujades i afavorit les baixades, tenint en compte la distancia Manhattan. A* ha pogut trobar la solució optima explorant mes opcions que BestFirst, ha tractat 73 estats, molt mes que BestFirst, però això ha assegurant que trobes la millor solució. Segons el mapa, els obstacles i desnivells han fet que calgués explorar camins alternatius, i la heurística ha ajudat a evitar pujades innecessàries, fent que A* trobi la millor ruta. Es pot observar com a diferencia que el BestFist, A* ha preferit anar por un camí alternatiu que semblava pitjor al principi però finalment ha tingut un millor resultat.

5. Algoritme Hill Climbing

Hill Climbing es un algoritme que segueix sempre el millor successor, sense mantenir una llista de nodes pendents, això implica diferents punts:

-No es complet, lo que pot quedar-se atrapat en un màxim local sense explorar totes les opcions.
-No es òptim, ja que no garantia la millor solució.
-No permet retrocedir, si la millor opció requereix un pas cap enrere abans de avançar , el algoritme es detindrà.
-Consumeix menys memòria, ja que no emmagatzema la llista de nodes pendents, però això afecta negativament a un risc de bloqueig durant la cerca.

Anàlisi de Hill Climbing amb les nostres heurístiques:

HEuclidiana:

Com aquesta heurística té en compte el cost acumulat fins al node actual, introdueix una penalització addicional en funció del camí recorregut.

Això pot fer que sobreestimi el cost d'alguns camins, fent que Hill Climbing descarti opcions viables abans d'explorar-les.

A més, pot quedar-se bloquejat en zones on la heurística doni un valor molt alt, sense considerar alternatives que podrien portar a la solució de manera més eficient.

HManhattan:

Si el camí més curt requereix rodejant un obstacle abans d'avançar, Hill Climbing no explorarà aquella opció i es quedarà bloquejat en un punt sense sortida.

Aquesta heurística pot ser efectiva en terrenys plans o amb petites variacions d'altura, però si hi ha obstacles o desnivells pronunciats, esdevé poc fiable.

Per aquest motiu, només podria funcionar en escenaris molt concrets i no és adequada per als nostres mapes.

HAltura:

Si el camí òptim requereix baixar una muntanya abans de pujar, Hill Climbing es detindrà immediatament en el primer punt on el successor tingui un cost més alt.

Això fa que es pugui quedar atrapat en zones elevades sense explorar possibles descensos que portin a la solució.

Per tant, aquesta heurística no seria fiable per als nostres mapes, ja que tenen molts desnivells i requeririen moviments de baixada que Hill Climbing no consideraria.

Com a conclusió podem dir-ne que Hill Climbing per els nostres mapes no seria un algorisme adequat per la seva naturalesa de seleccionar sempre el millor successor que limita a que explori camins que requereixen retrocedir temporalment.

Això fa que sigui especialment inefectiu en terrenys amb molts desnivells, on s'hagués de baixar per poder arribar a la solució.

Encara que podria funcionar en terrenys plans, on els nostres mapes no compleix aquets cas, i no garantiria igualment trobar la millor solució, ni trobar-ne una.