



ESCOLA TÈCNICA SUPERIOR
D'ENGINYERIA
Universitat Rovira i Virgili



Pràctica 1

Intel·ligència Artificial

Ricard Tamargo López
Joel Chacón López

Curs: 2024-2025

Professor: Roger Mallol Parera

Data de lliurament: 16/03/2025

Index

Introducció.....	2
Tasques.....	3
Formalització del problema.....	3
Definició de les heurístiques.....	4
Implementació Best First i A*.....	5
Best-First.....	5
A*.....	5
Estructura comuna.....	6
Proves.....	7
Conclusions generals.....	11
Anàlisi de Hill Climbing.....	12

Introducció

En aquesta primera pràctica, s'ha plantejat la implementació de diferents algorismes de cerca informada per trobar el camí més ràpid per desplaçar-se per un mapa representat per una matriu de caselles. Cada casella del mapa té una altura associada, o bé pot ser un precipici insalvable. L'objectiu principal és estudiar i comparar el rendiment dels algorismes Best-first i A*, utilitzant tres heurístiques diferents per guiar la cerca.

El mapa inicial, que es mostra a la **Figura 1**, està format per una matriu de 10x10 caselles. Cada casella té un valor d'alçada, i algunes estan marcades com a precipicis, els quals són infranquejables. Els moviments permesos són horitzontals i verticals, i el temps necessari per desplaçar-se entre dues caselles ve determinat per la diferència d'alçades. Concretament:

- Si la diferència d'alçades és positiva o zero, el temps és de 1 + (diferència d'alçades).
- Si la diferència és negativa, el temps es redueix a 0.5 unitats.
- No es permet el moviment a caselles amb precipicis.









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

Figura 1. Mapa inicial

Tasques

Formalització del problema

En aquest problema, es tracta de representar un mapa amb diferents alçades, on l'objectiu és trobar el camí més eficient entre un punt d'inici i un punt de destí. Cada estat del problema està representat per la classe **Node.java**, que conté els següents atributs principals:

- **(x, y)**: Coordenades del node dins de la matriu que representa el mapa.
- **altura**: Valor de l'alçada del node en el mapa.
- **cost**: Cost acumulat per arribar fins a aquest node. Es calcula sumant els costos de cada moviment des del node inicial.
- **pare**: Referència al node des del qual s'ha arribat a aquest. Aquest atribut s'utilitza per a la reconstrucció del camí una vegada s'ha trobat la solució.

Els operadors representen els moviments permesos dins de la matriu, i es troben definits a la classe **AlgorismeBase.java**. Aquests moviments són:

```
public static final int[][] MOVIMENTS = {  
    {1, 0}, {0, 1}, {-1, 0}, {0, -1} // abaix, dreta, adalt, esquerra  
};
```

Figura 2. Definició dels moviments

No obstant, el cost de cada moviment varia en funció de la diferència d'alçada entre el node actual i el node al qual es vol moure:

- **Si l'altura augmenta o es manté igual**: el cost s'incrementa proporcionalment segons la fórmula:
 - $1 + (\text{alçada_destí} - \text{alçada_actual})$, ja que pujar requereix més esforç.
- **Si l'altura disminueix**: el cost serà 0,5, ja que baixar és "més fàcil"

Definició de les heurístiques

En aquest problema, s'han implementat tres heurístiques diferents per estimar el cost restant des d'un node fins al destí. Cada una d'aquestes heurístiques està implementada a la seva respectiva classe (**Heuristica1.java**, **Heuristica2.java**, **Heuristica3.java**).

1. Heurística 1: Diferència d'alçades

Aquesta heurística compara l'alçada del node actual amb l'alçada del node final. Es calcula mitjançant la fórmula següent.:

$$h(n) = |\text{altura actual} - \text{altura destino}|$$

Com podem veure s'enfoca exclusivament en el desnivell entre el node actual y el node destí, és a dir que no té en compte la distància horitzontal o vertical de la matriu, lo qual pot ser admissible si el cost del problema depèn en gran mesura en la diferencia d'altura i no sobreestima el cost real d'arribar a la meta.

2. Heurística 2: Distància Euclidiana

Aquesta heurística calcula la distància en línia recta (euclidiana) entre el node actual i el node final. Es calcula mitjançant la fórmula següent:

$$h(n) = \sqrt{(x_{\text{destí}} - x_{\text{actual}})^2 + (y_{\text{destí}} - y_{\text{actual}})^2}$$

Aquesta heurística assumeix que el camí més curt entre dos punts és una línia recta, ignorant les alçades i els obstacles. Com que la distància euclidiana sempre és igual o menor que la distància real (que pot incloure desviacions degudes als precipicis o alçades), aquesta heurística és admissible, ja que mai sobreestima el cost real.

3. Heurística 3: Distància Manhattan amb Penalització per Alçada

L'última heurística que hem proposat combina la distancia de Manhattan amb una penalització per diferències d'altura, per tal de reflectir de millor forma el cost real del moviment.

$$h(n) = |x_{\text{destí}} - x_{\text{actual}}| + |y_{\text{destí}} - y_{\text{actual}}| + \max(0, \text{alturameta} - \text{alturaactual})/2.0$$

Aquesta heurística té en compte tant la distància horitzontal com la diferència d'alçades, afegint una penalització proporcional si el destí està a una alçada superior. Tot i que aquest enfocament reflecteix millor l'esforç necessari per pujar, la penalització per alçada pot fer que la heurística sobreestimi el cost real en alguns casos, cosa que la fa no admissible. No obstant això, pot ser útil en situacions on el desnivell és un factor crític i es vol prioritzar la reducció de l'esforç en pujar.

Implementació Best First i A*

S'han implementat dos algorismes de cerca informada: **Best-First** i **A***. Tots dos algorismes comparteixen una estructura similar, però es diferencien en com utilitzen la informació heurística per guiar la cerca.

Best-First

L'algorisme Best-First utilitza una llista de nodes pendents (pends) i un conjunt de nodes visitats (visitats) per explorar el mapa. En cada iteració, es selecciona el node amb el valor heurístic més baix, és a dir, el que sembla més proper al destí segons l'heurística, i s'exploren els seus veïns. La cerca es basa exclusivament en l'heurística, sense tenir en compte el cost acumulat per arribar a cada node. Això fa que Best-First sigui més ràpid en alguns casos, però no garanteix trobar la solució òptima.

- **Funcionament:**

1. S'inicialitza la llista de nodes pendents amb el node inicial.
2. En cada iteració, es selecciona el node pendent amb el valor heurístic més baix.
3. Si el node seleccionat és el destí, es retorna com a solució.
4. S'exploren els veïns del node actual i s'afegeixen a la llista de pendents si no han estat visitats.
5. Es repeteix el procés fins que es troba una solució o es recorre tot el mapa.

- **Característiques clau:**

1. Utilitza una heurística per guiar la cerca.
2. No té en compte el cost acumulat, cosa que pot fer que no trobi la solució òptima.
3. És eficient en termes de memòria, ja que només manté una llista de nodes pendents i un conjunt de nodes visitats.

A*

L'algorisme A*, per altra banda, combina el cost acumulat des de l'inici (cost) amb l'heurística per estimar el cost total fins al destí. Això permet que A* trobi sempre la solució òptima, sempre i quan l'heurística sigui admissible. En cada iteració, es selecciona el node amb el valor més baix de la suma cost + heurística.

- **Funcionament:** funciona igual que Best-First, l'única diferència és que combina el cost acumulat i l'heurística per guiar la cerca

- **Característiques clau:**

1. Combina el cost acumulat i l'heurística per guiar la cerca.
2. Garanteix trobar la solució òptima si l'heurística és admissible.
3. És més lent que Best-First en alguns casos, ja que explora més nodes per assegurar l'optimalitat.

Estructura comuna

Tots dos algorismes comparteixen una estructura comuna, implementada a la classe abstracta **AlgorismeBase.java**. Aquesta classe proporciona el mètode `getGermans`, que genera els nodes veïns d'un node donat, tenint en compte els moviments permesos i el cost associat a cada moviment. El cost es calcula en funció de la diferència d'alçades entre el node actual i el veí, seguint les regles especificades a l'enunciat.

Moviments permesos: Es poden fer moviments horitzontals i verticals, però no diagonals.

Càlcul del cost:

- Si la diferència d'alçades és positiva o zero, el cost és $1 + (\text{alçada_destí} - \text{alçada_actual})$.
- Si la diferència és negativa, el cost és 0.5.

Proves

S'han de tenir en compte els diferents resultats que es mostren per el terminal

1. Quantitat de nodes visitats

Mesura quants estats han estat explorats per cada algorisme abans de trobar la solució, per lo qual un nombre menor indica una cerca més eficient.

2. Camí resultant

Indica la seqüència de nodes des de l'inici (0,0) fins al destí (9,9), com més curts sigui el camí millor, en termes de cost total .

3. Cost total de la solució

Representa la suma dels costos de moviment al llarg del camí trobat, per tant un cost menor indica una solució més eficient.

Mapa 1:

- Heurística 1: Comparació d'alçades

Best-First

- Camí: [(0,0), (1,0), (2,0), (3,0), (3,1), (4,1), (4,2), (4,3), (4,4), (3,4), (3,5), (3,6), (3,7), (3,8), (4,8), (5,8), (6,8), (7,8), (8,8), (9,8), (9,9)]
- Cost total: 30.0
- Nodes visitats: 61
- Solució òptima: No
- Anàlisi: L'heurística 1 guia la cerca cap a nodes amb menor diferència d'alçada, però no té en compte la distància horitzontal. Això fa que el camí sigui més llarg i menys eficient. El nombre d'estats visitats és elevat, ja que l'algorisme explora moltes opcions abans de trobar una solució.

A*

- Camí: [(0,0), (0,1), (0,2), (0,3), (0,4), (1,4), (1,5), (2,5), (2,5), (3,5), (4,5), (4,6), (5,6), (5,7), (6,7), (7,7), (8,7), (8,8), (9,8), (9,9)]
- Cost total: 25.0
- Nodes visitats: 91
- Solució òptima: Sí
- Anàlisi: L'algorisme A* troba una solució òptima, però requereix explorar més nodes que Best-First. L'heurística 1, combinada amb el cost acumulat, permet trobar un camí més eficient, tot i que el nombre d'estats visitats és més alt.

- Heurística 2: Distància Euclidiana

Best-First

- Camí: [(0,0), (1,0), (1,1), (2,1), (3,1), (4,1), (4,2), (4,3), (4,4), (5,4), (5,4), (6,4), (6,5), (6,6), (7,6), (7,7), (8,7), (8,8), (9,8), (9,9)]
- Cost total: 26.5
- Nodes visitats: 18
- Solució òptima: No
- Anàlisi: L'heurística 2 guia la cerca cap al destí en línia recta, cosa que redueix el nombre d'estats visitats. Tot i això, el camí no és òptim, ja que no té en compte el cost acumulat ni els obstacles.

A*

- Camí: [(0,0), (1,0), (2,0), (3,0), (4,0), (4,1), (5,1), (5,1), (6,1), (6,2), (7,2), (7,3), (8,3), (8,4), (8,5), (9,5), (9,6), (9,7), (9,8), (9,9)]
- Cost total: 25.0
- Nodes visitats: 90
- Solució òptima: Sí
- Anàlisi: A* troba una solució òptima, però requereix explorar molts nodes. L'heurística 2, combinada amb el cost acumulat, permet trobar un camí eficient, tot i que el nombre d'estats visitats és elevat.

- Heurística 3: Distància Manhattan amb penalització per alçada

Best-First

- Camí: [(0,0), (1,0), (2,0), (3,0), (3,1), (4,1), (4,2), (4,3), (4,4), (5,4), (5,4), (6,4), (7,4), (8,4), (8,5), (8,6), (8,7), (8,8), (9,8), (9,9)]
- Cost total: 26.5
- Nodes visitats: 18
- Solució òptima: No
- Anàlisi: L'heurística 3 combina distància i alçada, cosa que redueix el nombre d'estats visitats. Tot i això, el camí no és òptim, ja que no té en compte el cost acumulat de manera completa.

A*

- Camí: [(0,0), (0,1), (1,1), (2,1), (3,1), (4,1), (5,1), (5,1), (6,1), (6,2), (7,2), (7,3), (8,3), (8,4), (8,5), (9,5), (9,6), (9,7), (9,8), (9,9)]
- Cost total: 25.0
- Nodes visitats: 90
- Solució òptima: Sí
- Anàlisi: A* troba una solució òptima, però requereix explorar molts nodes. L'heurística 3, combinada amb el cost acumulat, permet trobar un camí eficient, tot i que el nombre d'estats visitats és elevat.

```

0 1 2 3 3 -1 0 1 2 2
2 1 2 4 3 2 1 3 3 3
2 2 -1 4 5 3 2 -1 4 4
3 3 -1 4 6 4 2 3 3 3
2 2 3 3 5 3 3 2 3 3
2 1 1 3 4 -1 2 2 3 4
2 0 0 1 2 1 1 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 1 2 4 5
-----
Heurística 1: Comparació d'altures
Best First:
Total de nodes visitats: 61
Camí: [(0,0), (1,0), (2,0), (3,0), (3,1), (4,1), (4,2), (4,3), (4,4), (3,4), (3,5), (3,6), (3,7), (3,8), (4,8), (5,8), (6,8), (7,8), (8,8), (9,8), (9,9)]
Cost total: 30.0
A*:
Total de nodes visitats: 91
Camí: [(0,0), (0,1), (0,2), (0,3), (0,4), (1,4), (1,5), (2,5), (3,5), (4,5), (4,6), (5,6), (5,7), (6,7), (7,7), (8,7), (8,8), (9,8), (9,9)]
Cost total: 25.0
-----
Heurística 2: Distància Euclidiana
Best First:
Total de nodes visitats: 18
Camí: [(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)]
Cost total: 26.5
A*:
Total de nodes visitats: 90
Camí: [(0,0), (1,0), (2,0), (3,0), (4,0), (4,1), (5,1), (6,1), (6,2), (7,2), (7,3), (8,3), (8,4), (8,5), (9,5), (9,6), (9,7), (9,8), (9,9)]
Cost total: 25.0
-----
Heurística 3: Distància Manhattan
Best First:
Total de nodes visitats: 18
Camí: [(0,0), (1,0), (2,0), (3,0), (3,1), (4,1), (4,2), (4,3), (4,4), (5,4), (6,4), (7,4), (8,4), (8,5), (8,6), (8,7), (8,8), (9,8), (9,9)]
Cost total: 26.5
A*:
Total de nodes visitats: 90
Camí: [(0,0), (0,1), (1,1), (2,1), (3,1), (4,1), (5,1), (6,1), (6,2), (7,2), (7,3), (8,3), (8,4), (8,5), (9,5), (9,6), (9,7), (9,8), (9,9)]
Cost total: 25.0

```

Figura 3. Proves amb Mapa1.txt

Mapa 2:

- Heurística 1: Comparació d'alçades

Best-First

- Camí: [(0,0), (1,0), (1,1), (2,1), (3,1), (4,1), (4,2), (4,3), (5,3), (6,3), (6,4), (7,4), (7,5), (8,5), (8,6), (9,7), (9,8), (9,9)]
- Cost total: 24.5
- Nodes visitats: 59
- Solució òptima: No
- Anàlisi: L'heurística 1 guia la cerca cap a nodes amb menor diferència d'alçada, però no té en compte la distància horitzontal. El camí no és òptim, i el nombre d'estats visitats és elevat.

A*

- Camí: [(0,0), (1,0), (1,1), (2,1), (3,1), (4,1), (4,2), (4,3), (5,3), (6,3), (6,4), (7,4), (7,5), (8,5), (8,6), (9,7), (9,8), (9,9)]
- Cost total: 24.5
- Nodes visitats: 77
- Solució òptima: Sí
- Anàlisi: A* troba una solució òptima, però requereix explorar molts nodes. L'heurística 1, combinada amb el cost acumulat, permet trobar un camí eficient.

- Heurística 2: Distància Euclidiana

Best-First

- Camí: [(0,0), (1,0), (1,1), (2,1), (3,1), (4,1), (4,2), (4,3), (4,4), (5,4), (5,5), (6,5), (6,6), (7,6), (7,7), (8,7), (8,8), (9,8), (9,9)]
- Cost total: 30.5
- Nodes visitats: 18
- Solució òptima: No
- Anàlisi: L'heurística 2 guia la cerca cap al destí en línia recta, cosa que redueix el nombre d'estats visitats. Tot i això, el camí no és òptim, ja que no té en compte el cost acumulat ni els obstacles.

A*

- Camí: [(0,0), (1,0), (1,1), (2,1), (3,1), (4,1), (4,2), (4,3), (5,3), (6,3), (6,4), (6,5), (6,6), (7,6), (8,6), (9,6), (9,7), (9,8), (9,9)]
- Cost total: 24.5
- Nodes visitats: 67
- Solució òptima: Sí
- Anàlisi: A* troba una solució òptima, però requereix explorar molts nodes. L'heurística 2, combinada amb el cost acumulat, permet trobar un camí eficient.

- Heurística 3: Distància Manhattan amb penalització per alçada

Best-First

- Camí: [(0,0), (1,0), (1,1), (2,1), (3,1), (4,1), (4,2), (4,3), (5,3), (5,4), (6,4), (7,4), (7,5), (8,5), (8,6), (8,7), (9,7), (9,8), (9,9)]
- Cost total: 27.5
- Nodes visitats: 18
- Solució òptima: No
- Anàlisi: L'heurística 3 combina distància i alçada, cosa que redueix el nombre d'estats visitats. Tot i això, el camí no és òptim, ja que no té en compte el cost acumulat de manera completa.

A*

- Camí: [(0,0), (1,0), (1,1), (2,1), (3,1), (4,1), (4,2), (4,3), (5,3), (6,3), (6,4), (7,4), (7,5), (7,6), (8,6), (9,6), (9,7), (9,8), (9,9)]
- Cost total: 24.5
- Nodes visitats: 65
- Solució òptima: Sí
- Anàlisi: A* troba una solució òptima, però requereix explorar molts nodes. L'heurística 3, combinada amb el cost acumulat, permet trobar un camí eficient.

```

0 -1 2 5 2 -1 -1 -1 3 5
1 1 2 4 5 2 1 3 -1 3
-1 2 -1 4 5 3 2 -1 4 4
2 3 -1 5 2 4 2 3 3 3
-1 2 3 3 6 3 3 2 3 3
2 -1 -1 3 4 1 2 2 3 4
5 2 2 1 2 1 1 1 1 -1
5 1 1 5 1 1 1 1 1 -1
5 3 2 1 -1 2 2 6 1 -1
4 3 2 2 -1 1 1 2 4 2
-----
Heurística 1: Comparació d'altures
Best First:
Total de nodes visitats: 59
Camí: [(0,0), (1,0), (1,1), (2,1), (3,1), (4,1), (4,2), (4,3), (5,3), (6,3), (6,4), (7,4), (7,5), (8,5), (8,6), (9,6), (9,7), (9,8), (9,9)]
Cost total: 24.5
A*:
Total de nodes visitats: 77
Camí: [(0,0), (1,0), (1,1), (2,1), (3,1), (4,1), (4,2), (4,3), (5,3), (6,3), (6,4), (7,4), (7,5), (8,5), (8,6), (9,6), (9,7), (9,8), (9,9)]
Cost total: 24.5
-----
Heurística 2: Distància Euclidiana
Best First:
Total de nodes visitats: 18
Camí: [(0,0), (1,0), (1,1), (2,1), (3,1), (4,1), (4,2), (4,3), (4,4), (5,4), (5,5), (6,5), (6,6), (7,6), (7,7), (8,7), (8,8), (9,8), (9,9)]
Cost total: 30.5
A*:
Total de nodes visitats: 67
Camí: [(0,0), (1,0), (1,1), (2,1), (3,1), (4,1), (4,2), (4,3), (5,3), (6,3), (6,4), (6,5), (6,6), (7,6), (8,6), (9,6), (9,7), (9,8), (9,9)]
Cost total: 24.5
-----
Heurística 3: Distància Manhattan
Best First:
Total de nodes visitats: 18
Camí: [(0,0), (1,0), (1,1), (2,1), (3,1), (4,1), (4,2), (4,3), (5,3), (5,4), (6,4), (7,4), (7,5), (8,5), (8,6), (8,7), (9,7), (9,8), (9,9)]
Cost total: 27.5
A*:
Total de nodes visitats: 65
Camí: [(0,0), (1,0), (1,1), (2,1), (3,1), (4,1), (4,2), (4,3), (5,3), (6,3), (6,4), (7,4), (7,5), (7,6), (8,6), (9,6), (9,7), (9,8), (9,9)]
Cost total: 24.5

```

Figura 4. Proves amb Mapa2.txt

Conclusions generals

Best-First tendeix a trobar solucions més ràpidament, però no sempre són òptimes. El nombre d'estats visitats és menor, però el camí pot ser més llarg. D'altra banda, A* sempre troba solucions òptimes, però requereix explorar més nodes, ja que combina l'heurística amb el cost acumulat per garantir l'optimalitat. Quant a les heurístiques, la 2 i la 3 redueixen el nombre d'estats visitats en comparació amb la 1, però no sempre garanteixen l'optimalitat en Best-First. En general, A* és més fiable per trobar solucions òptimes, mentre que Best-First és més ràpid però menys precís.

Anàlisi de Hill Climbing

Aquest algorisme no explora tots els nodes com fan Best-First o A*, sinó que es queda amb la primera millora que troba. Justificació amb cada heurística:

1. Heurística 1: Diferència d'alçades

Aquesta heurística es basa en la diferència d'alçada entre el node actual i el destí. Hill Climbing podria funcionar bé en mapes on el camí òptim implica pujar o baixar de manera consistent, ja que es mouria cap al veí que minimitza aquesta diferència. Però en mapes amb molts precipicis o on el camí requereix fer desviacions, Hill Climbing podria quedar atrapat en un punt sense poder trobar la solució. Així que no és garantit que trobés una solució en tots els casos.

2. Heurística 2: Distància Euclidiana

Aquesta heurística utilitza la distància en línia recta fins al destí. En mapes simples sense obstacles, Hill Climbing podria funcionar bé, ja que es mouria cap al veí que està més a prop del destí en línia recta. Però en mapes complexos, amb precipicis o camins que no són directes, Hill Climbing podria quedar atrapat en un punt sense trobar la solució, perquè no té en compte els obstacles ni el cost acumulat.

3. Heurística 3: Distància Manhattan amb penalització per alçada

Aquesta heurística combina la distància de Manhattan amb una penalització per la diferència d'alçades. Hill Climbing podria funcionar millor amb aquesta heurística, ja que té en compte tant la distància horitzontal com el desnivell. Però com que Hill Climbing no pot retrocedir ni té memòria dels estats visitats, podria quedar atrapat en situacions on el camí requereix fer moviments que no semblen òptims en un primer moment. Així que tampoc és garantit que trobés una solució en tots els casos.

Indiferentment del mapa utilitzat, el principal problema per a Hill Climbing és la presència de precipicis. Aquests obstacles fan que el camí òptim pugui requerir desviacions o moviments temporals en direccions que no minimitzen immediatament la funció heurística. Com que Hill Climbing no té memòria dels estats visitats i no pot retrocedir, podria quedar atrapat en un òptim local quan es troba amb un precipici o una zona on no pot avançar sense fer un moviment que sembla menys favorable segons l'heurística.

En mapes sense precipicis o amb camins directes, Hill Climbing podria trobar una solució, ja que es mouria cap al veí que minimitza la funció heurística sense trobar obstacles. Però en mapes amb precipicis o camins complexos, Hill Climbing no és garantit que trobés una solució, ja que no pot explorar alternatives ni retrocedir per provar altres camins.