

# Ejercicios Búsqueda heurística

## Escalada, A-Estrella

---

Inteligencia Artificial Colmenarejo  
Curso 2022-2023

---

### BÚSQUEDA NO INFORMADA (CLASE ANTERIOR)

- Resolución de un grafo **sin costes** mediante búsqueda no informada.
  - Búsqueda en amplitud.
  - Búsqueda en profundidad.
- Resolución de un grafo **con costes** mediante búsqueda no informada.
  - Búsqueda en amplitud (con costes).
  - Búsqueda en profundidad (con costes).
  - Búsqueda mediante el algoritmo de Dijkstra.
- **Representación** de un problema de búsqueda.

### BÚSQUEDA INFORMADA (HOY):

dos algoritmos de resolución de grafos

- Búsqueda en **escalada** (Hill Climbing)
- Búsqueda **A\***

Así como el concepto de **heurística, admisibilidad**

## Representación del problema

Para representar un problema es necesario seguir los mismos parámetros que con búsqueda no informada, pero teniendo en cuenta la presencia de heurística y costes:

- Heurística: Coste estimado hasta el estado objetivo, si es admisible asegura óptimos. No admisible puede obtener subóptimos.
- Costes: Coste necesario para llegar hasta el estado actual

## Espacio de búsqueda del problema

Ni la heurística ni los costes modifican la cantidad de estados por lo que el espacio de búsqueda se calcula del mismo modo al visto en temas anteriores.

## Consideraciones:

- Para definir un estado hay que escribir toda la información que lo caracteriza.
  - En los ejercicios de grafos todo el estado se representa con una sola letra (A, B) junto a su coste y heurística, si lo hubiera.
  - En otros ejercicios es posible necesitar estructuras como tuplas, listas, u objetos para almacenar TODA la información que identifique a dicho instante dentro del problema.
- El formato de los estados y acciones tiene que escribirse de forma genérica (con variables).
- El estado inicial o final se escriben con valores particulares para estas variables.

## Aplicación de algoritmos de Búsqueda

- Hay que simular la ejecución de un algoritmo no basta indicar la solución mientras se deja claro el orden de expansión de los nodos. Se puede usar cualquier formato mientras el profesor lo comprenda. Por ejemplo, se proponen los siguientes (y se ven en los ej.):
  - Una tabla en la que cada línea muestre la acción tomada, nodo expandido, y el estado del algoritmo (lista abierta con todos los detalles necesarios).
  - Un árbol donde cada nodo contiene un estado, etiquetado con el orden en que ha sido generado y expandido. Si es posible, cada arco tiene que etiquetarse con el nombre (y parámetros si hay) de la acción utilizada para expandir el nodo.
- Control de estados repetidos (convención), los estados repetidos pueden ignorarse cuando el coste de llegar al estado repetido es mayor que el coste del estado original, en caso contrario si debe explorarse el estado repetido.
- Máximos locales, el espacio de búsqueda puede contener soluciones al problema que parezcan la mejor opción a la vista del resto de estados posibles, algoritmos puramente greedy como la búsqueda que solo consideren la heurística y no estén “modificados” para evitar estos máximos locales pueden no asegurar una solución al problema. En caso de que no se especifique lo lógico es asumir que el algoritmo solicitado es en su versión más básica (sin modificaciones). En cualquier caso si se tienen dudas sobre como resolver un problema lo mejor es especificar la suposición que se necesite de forma escrita para que el profesor lo comprenda.

El TopSpin es un juego de permutaciones que consiste en una ruleta de números que puede girar y una base giratoria que permite intercambiar dos números adyacentes.

El objetivo del juego es ordenar los números colocando la base giratoria en el par (1,2)

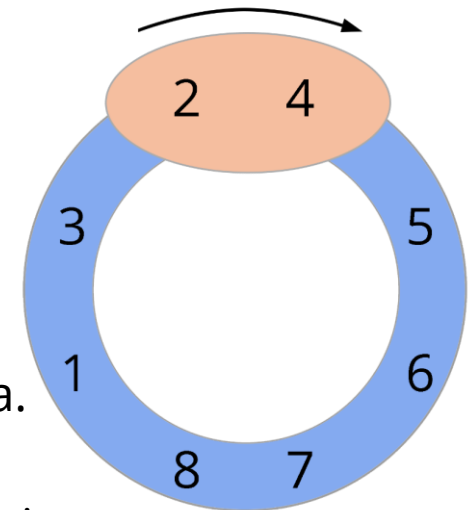
Los operadores serían (en orden de aplicación):

1. Mover números a la derecha
2. Mover números a la izquierda
3. Rotar base giratoria

Partiendo del estado inicial de la imagen mostrada.

### Preguntas:

- Definir una heurística de distancia para el TopSpin
- Hacer búsqueda en escalada (Hill Climbing) y en profundidad con la heurística definida



Para representar el estado podemos dar la cadena de números en sentido horario empezando por el primer número de la base giratoria. Por ejemplo el estado meta quedaría así: **12 - 345678**

## Definir una heurística de distancia para el TopSpin

Encontrar métrica: **distancia circular**.

Contar el número mínimo de posiciones que hay que girar para que el número 'i' llegue a su posición correcta.

Por ejemplo, en el estado **24 - 567813** tendríamos las distancias:

1. Distancia 2
2. Distancia 1
3. Distancia 3
4. Distancia 2
5. Distancia 2
6. Distancia 2
7. Distancia 2
8. Distancia 2

**¿Heurística?**

Con esta medida se pueden generar varias heurísticas: la distancia máxima, la suma de distancias, etc.

Heurística 1: suma de distancias circulares de todos los números ¡NO ADMISIBLE!  
Porque dicha suma da un número mucho mayor del necesario en algunos casos simples

El motivo es que un solo movimiento puede mover ocho números a la vez

Heurística 2, admisible: máximo de las distancias circulares de todos los números

En este ejercicio se utilizará la distancia máxima, teniendo en el ejemplo propuesto un valor de 3.

24 - 567813

$h(I) = 3$

## Hacer búsqueda en escalada y en profundidad con la heurística definida

$h=3$  24 - 567813

Para la búsqueda en escalada tenemos que expandir **siempre** el nodo que mejore la **heurística** actual. Nota: se escoge el mejor sucesor (incluso si este no mejora el nodo actual)

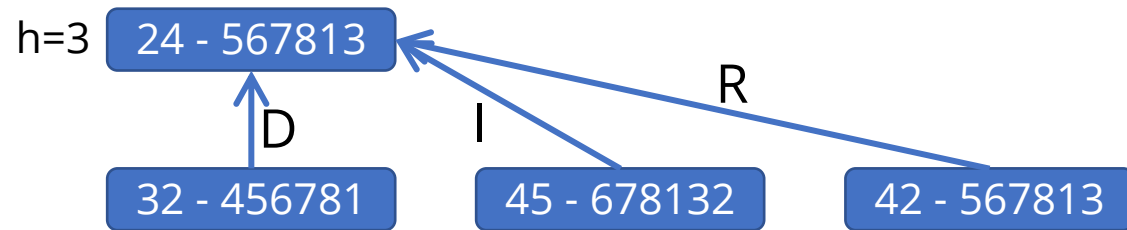
Algunas modificaciones para mejorarlo pueden permitir heurísticas iguales a la actual, o que sean peores (aunque la mejor de las restantes)

Por ejemplo: Escalada con restricción: se escoge el mejor sucesor pero solo si mejora el nodo actual. Si no se puede,  $\rightarrow$  STOP (no se encuentra la solución).



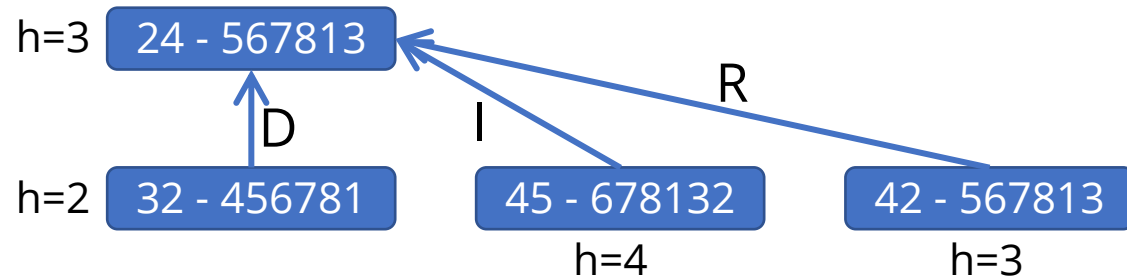
# Solución en escalada

**Hacer búsqueda en escalada y en profundidad con la heurística definida**



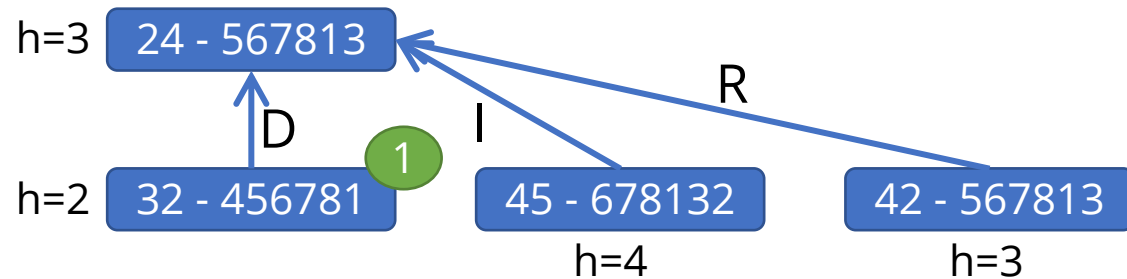
# Solución en escalada

**Hacer búsqueda en escalada y en profundidad con la heurística definida**



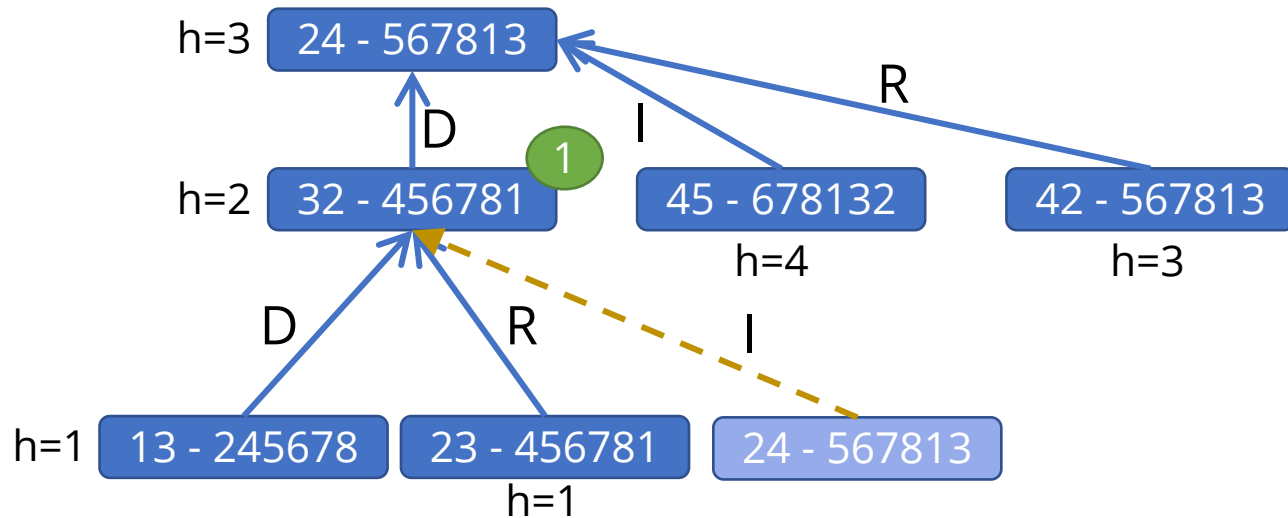
# Solución en escalada

**Hacer búsqueda en escalada y en profundidad con la heurística definida**



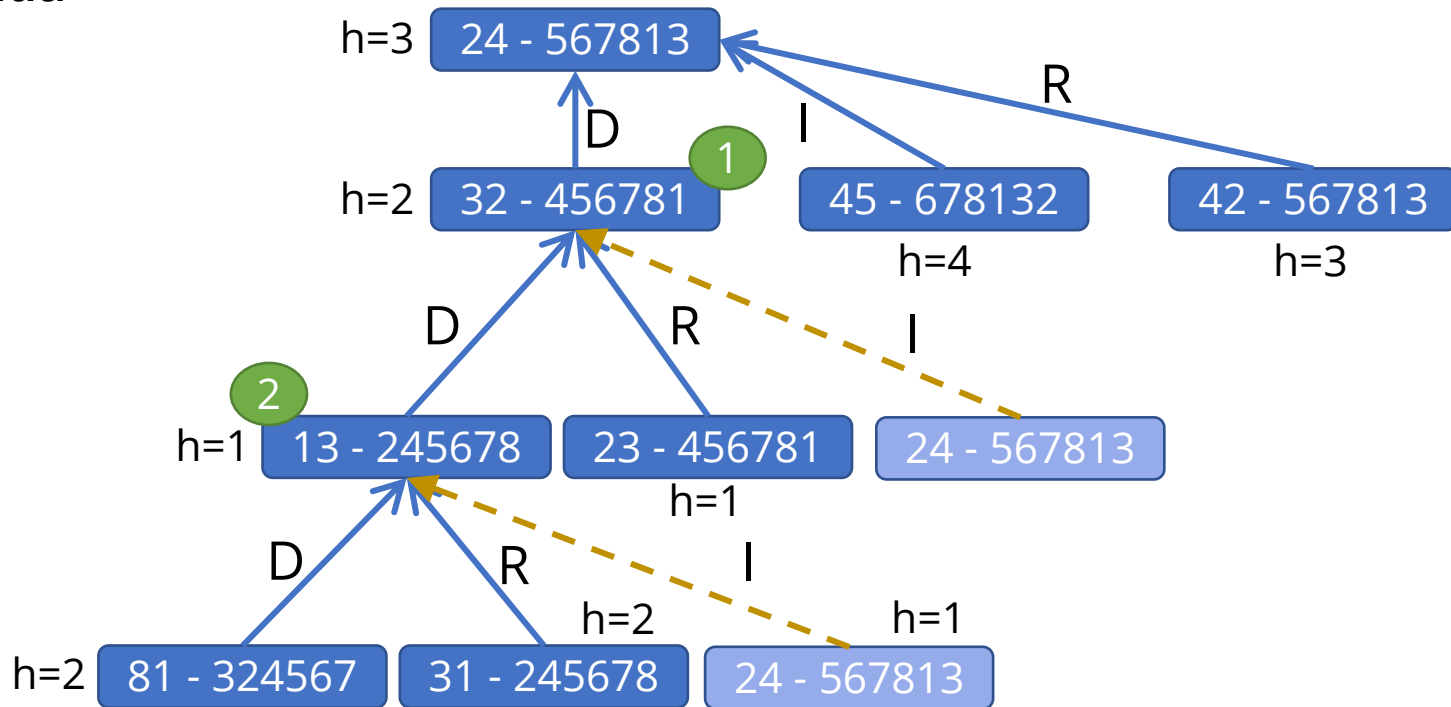
# Solución en escalada

**Hacer búsqueda en escalada y en profundidad con la heurística definida**



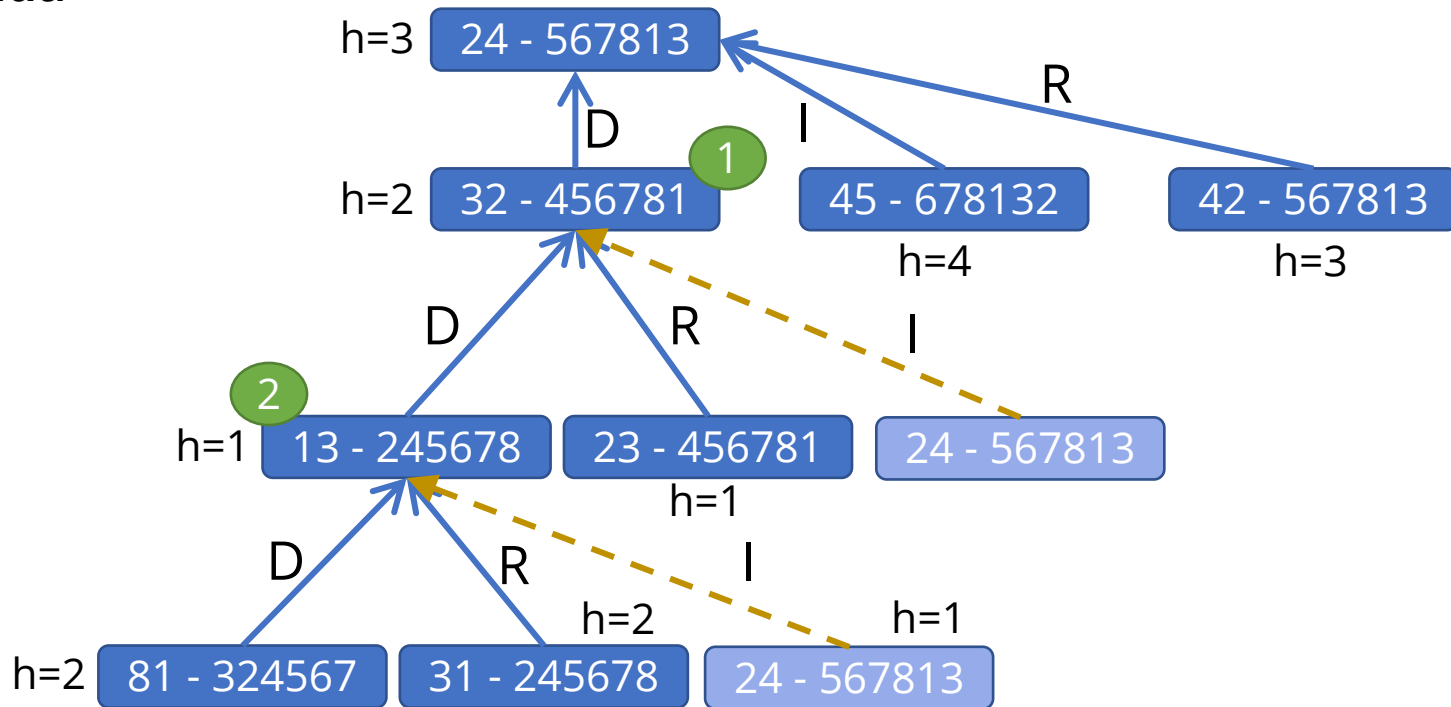
# Solución en escalada

Hacer búsqueda en escalada y en profundidad con la heurística definida



# Solución en escalada

**Hacer búsqueda en escalada y en profundidad con la heurística definida**



El algoritmo acaba sin lograr una solución ya que no encuentra un nodo sucesor mejor que el nodo actual.

Tampoco es capaz de reconsiderar el camino elegido volviendo hacia atrás.

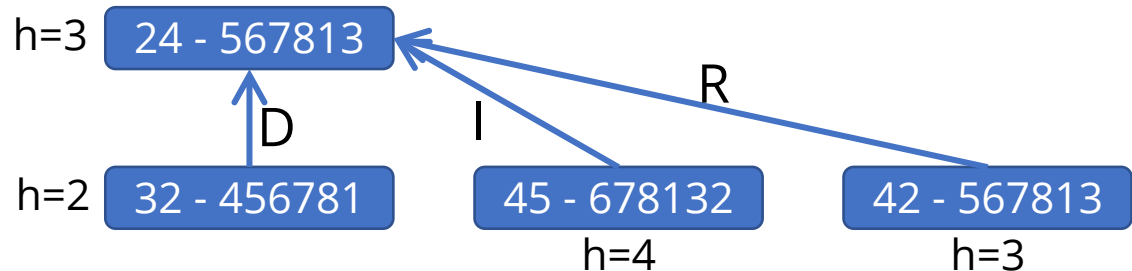
## Hacer búsqueda en escalada y en profundidad con la heurística definida

$h=3$  24 - 567813

Para la búsqueda en profundidad, se aplica el mismo procedimiento que el visto con anterioridad. La única diferencia es que se aprovecha la heurística para seleccionar la mejor opción entre nodos del mismo nivel.

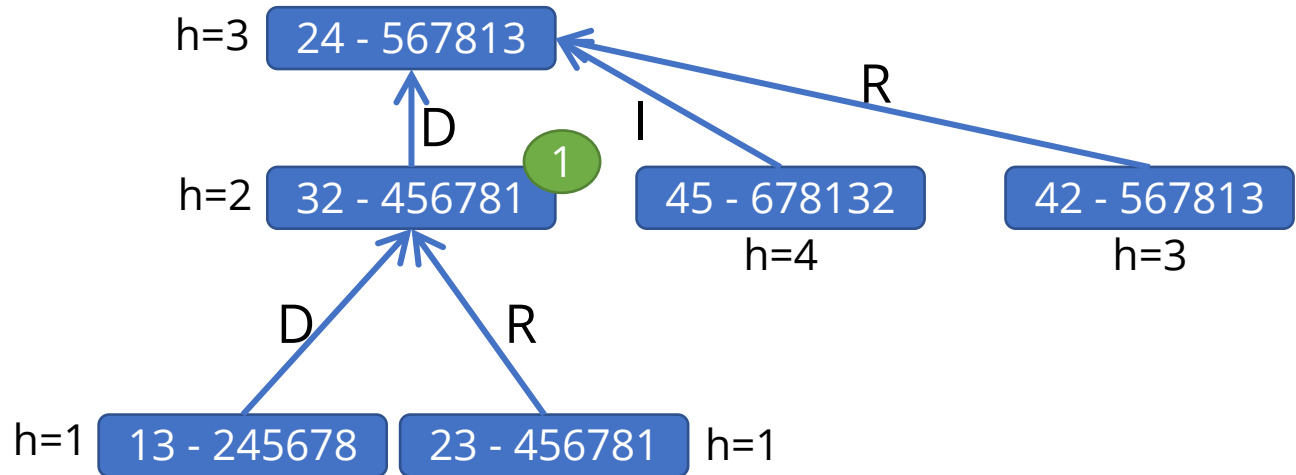
Por supuesto también se podría aplicar una amplitud con heurística siguiendo la misma modificación.

**Hacer búsqueda en escalada y en profundidad con la heurística definida**

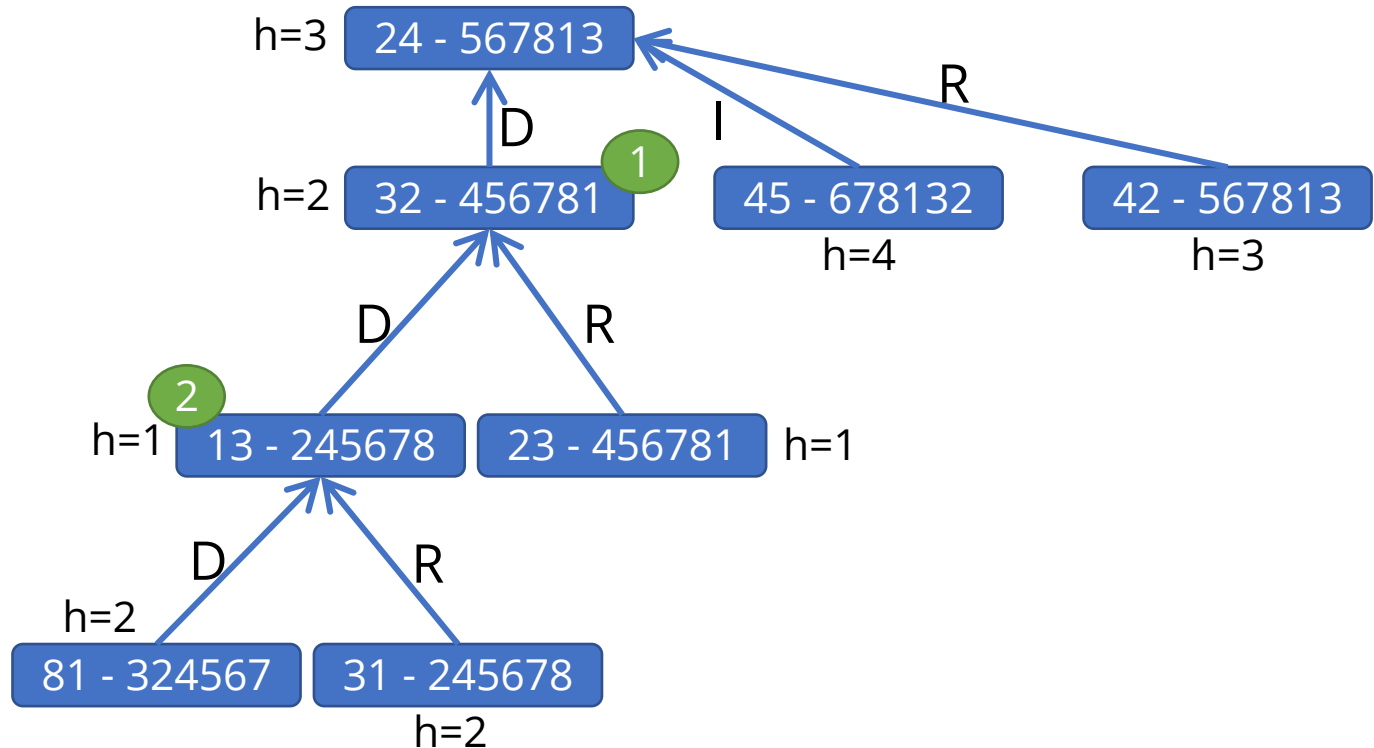




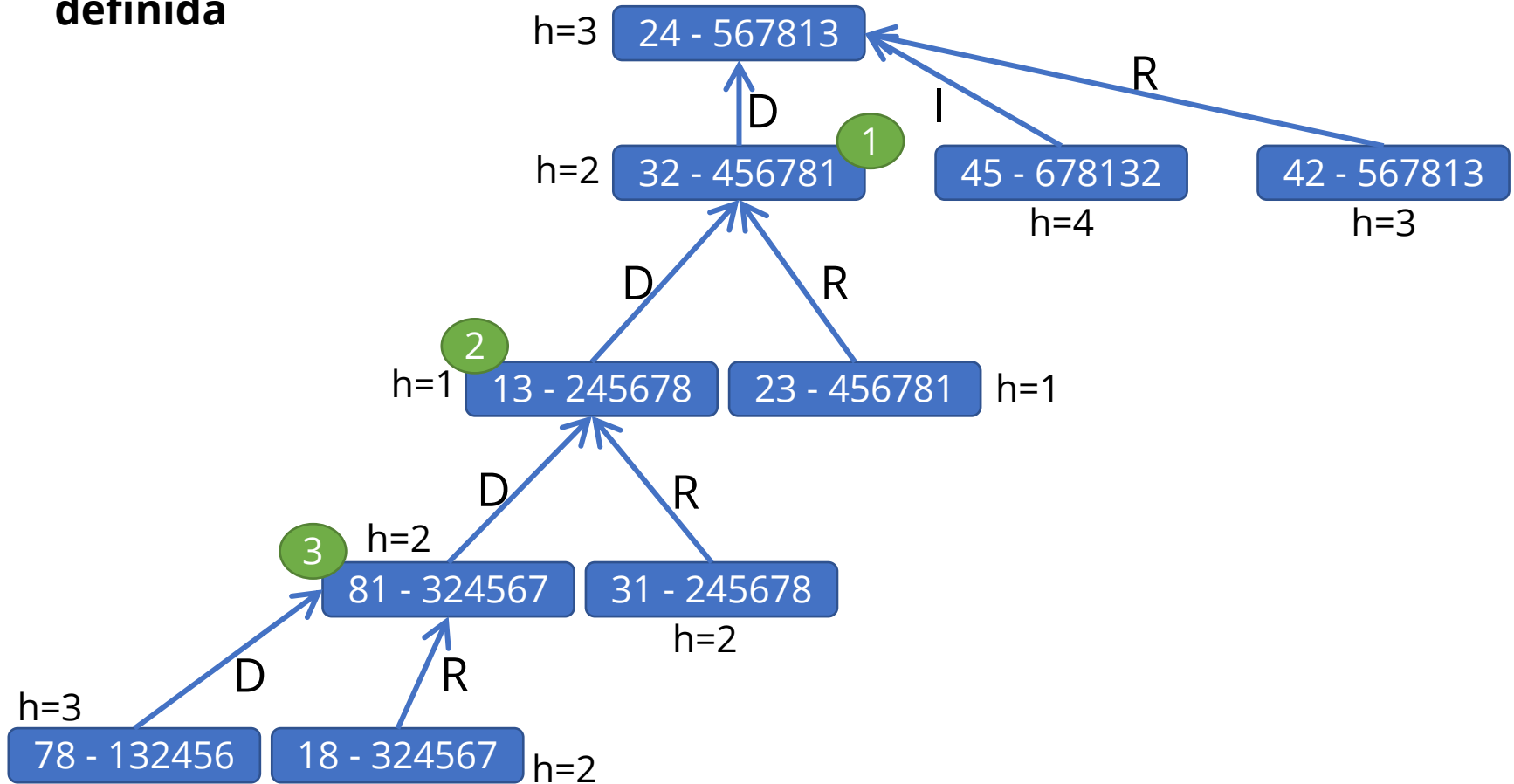
**Hacer búsqueda en escalada y en profundidad con la heurística definida**



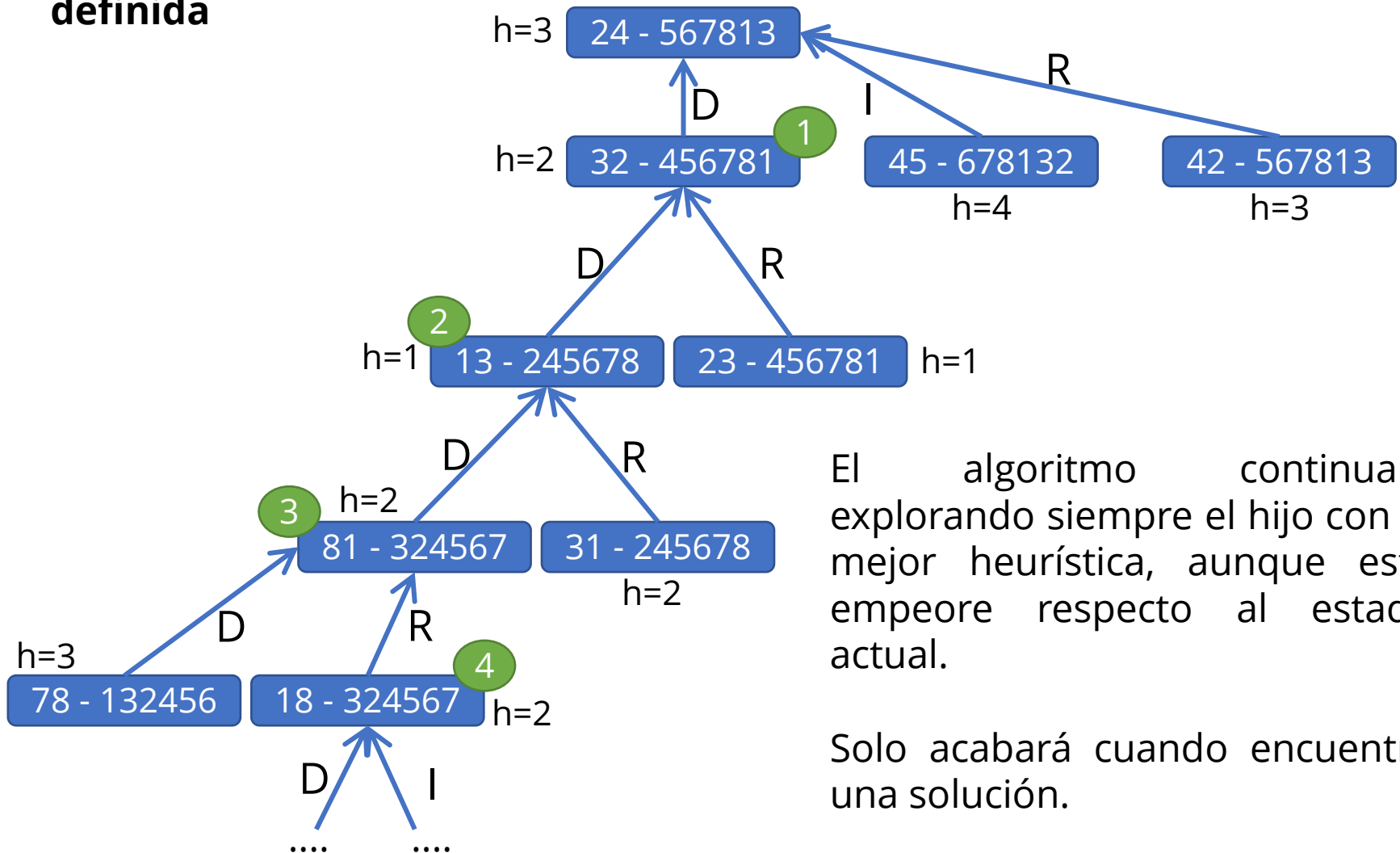
Hacer búsqueda en escalada y en profundidad con la heurística definida



Hacer búsqueda en escalada y en profundidad con la heurística definida



**Hacer búsqueda en escalada y en profundidad con la heurística definida**



El algoritmo continuará explorando siempre el hijo con la mejor heurística, aunque esta empeore respecto al estado actual.

Solo acabará cuando encuentre una solución.

El objetivo consiste en encontrar el camino con menor coste desde el origen hasta la meta, utilizando los operadores:

1. Mover arriba
2. Mover izquierda
3. Mover abajo
4. Mover derecha

Los números identifican a la celda mientras que el color indica el coste.

Las celdas grises son muros infranqueables, las blancas tienen coste 1, las amarillas 2 y las rojas 4.

1	I	2	3
4		5	
6	7	8	9
	10	M	11

Utilizar distancia Manhattan como heurística para solucionar el problema mediante búsqueda en escalada y búsqueda A\*.

## Solución Búsqueda escalada

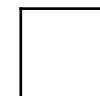
Inicio  
 $h = 4$     $g = 0$

$h(n)$  = la distancia estimada hasta la meta  
 $g(n)$  = el coste de ir de Inicio a la casilla actual

1	I	2	3
4		5	
6	7	8	9
	10	M	11



Muro



Coste 1

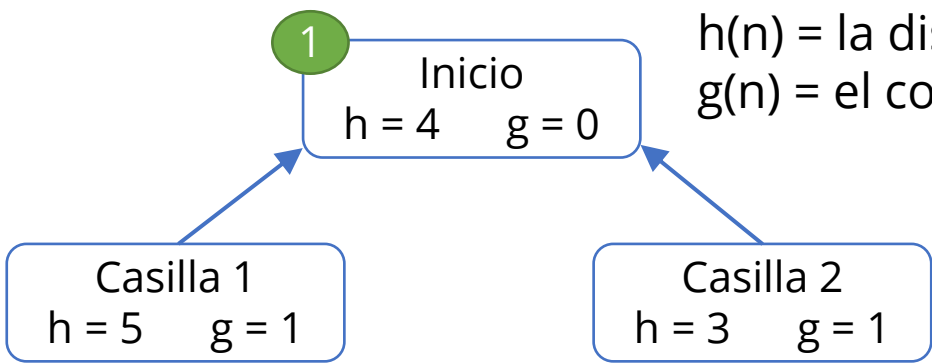


Coste 2







Coste 4

# Solución Búsqueda escalada

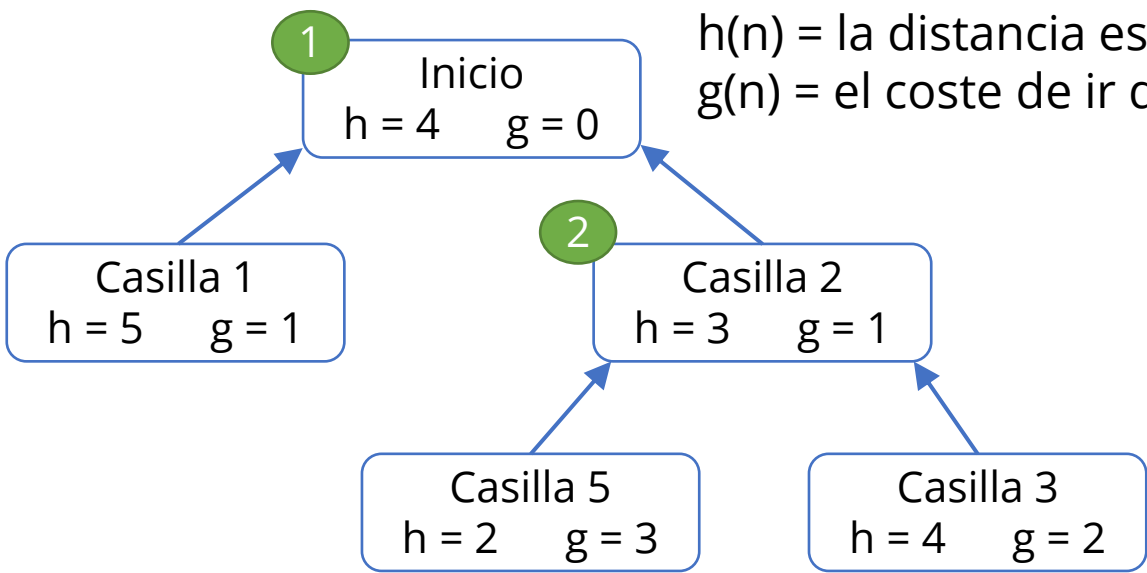


$h(n)$  = la distancia estimada hasta la meta  
 $g(n)$  = el coste de ir de Inicio a la casilla actual

1	I	2	3
4		5	
6	7	8	9
	10	M	11

-  Muro
-  Coste 1
-  Coste 2
-  Coste 4

# Solución Búsqueda escalada



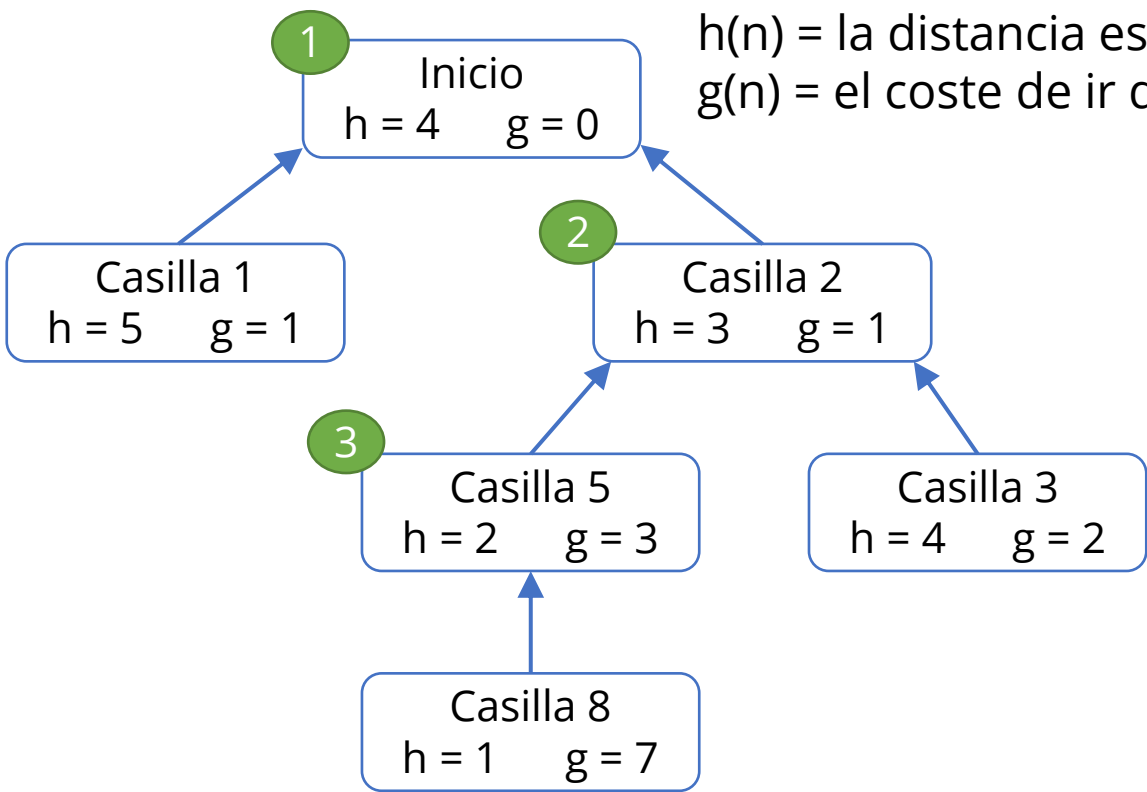
$h(n)$  = la distancia estimada hasta la meta  
 $g(n)$  = el coste de ir de Inicio a la casilla actual

1	I	2	3
4		5	
6	7	8	9
	10	M	11

- Muro
- Coste 1
- Coste 2
- Coste 4







# Solución Búsqueda escalada



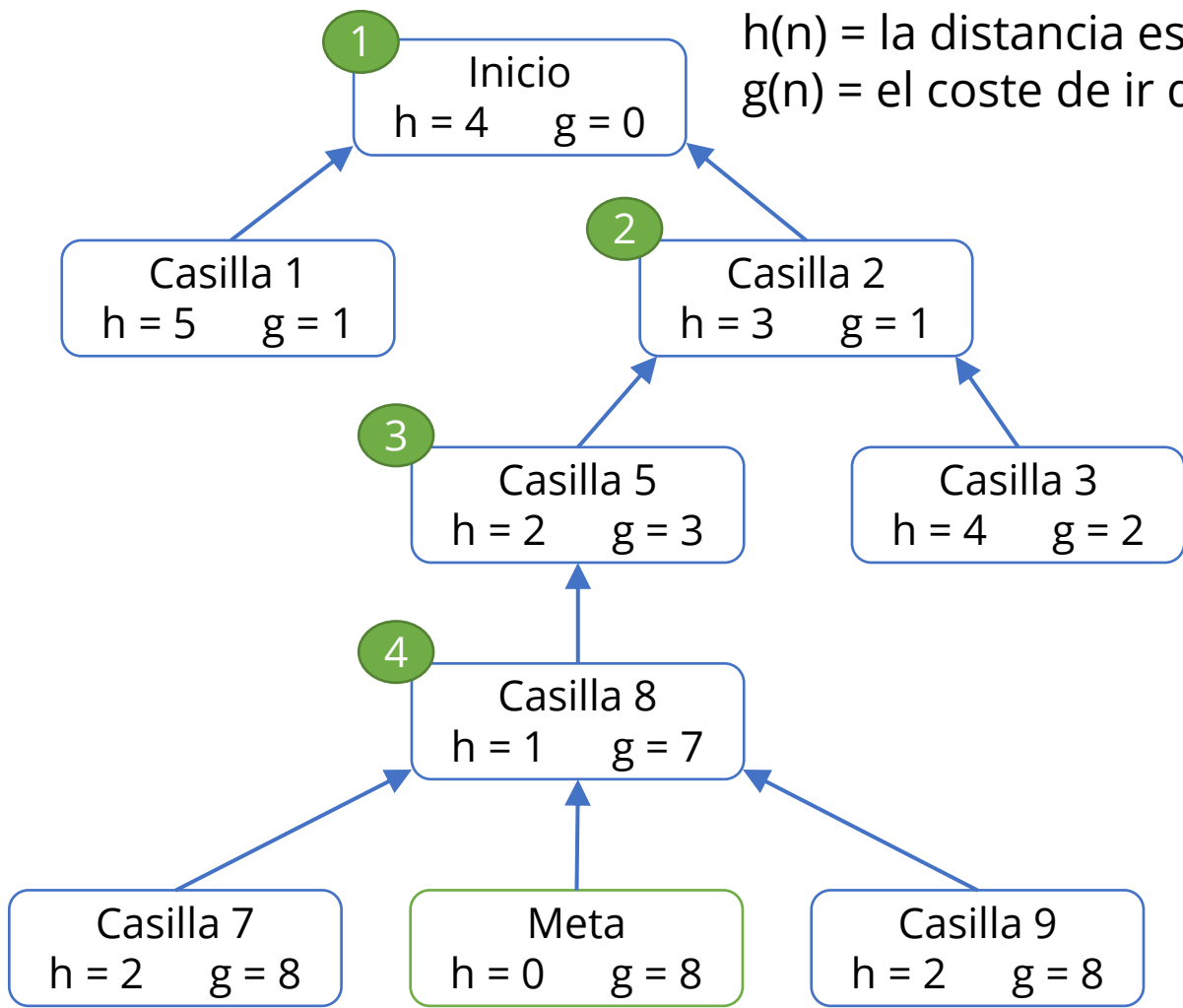
$h(n)$  = la distancia estimada hasta la meta  
 $g(n)$  = el coste de ir de Inicio a la casilla actual

1	I	2	3
4		5	
6	7	8	9
	10	M	11


-  Muro
-  Coste 1
-  Coste 2
-  Coste 4


# Solución Búsqueda escalada


$h(n)$  = la distancia estimada hasta la meta  
 $g(n)$  = el coste de ir de Inicio a la casilla actual




1	I	2	3
4		5	
6	7	8	9
	10	M	11

 Muro

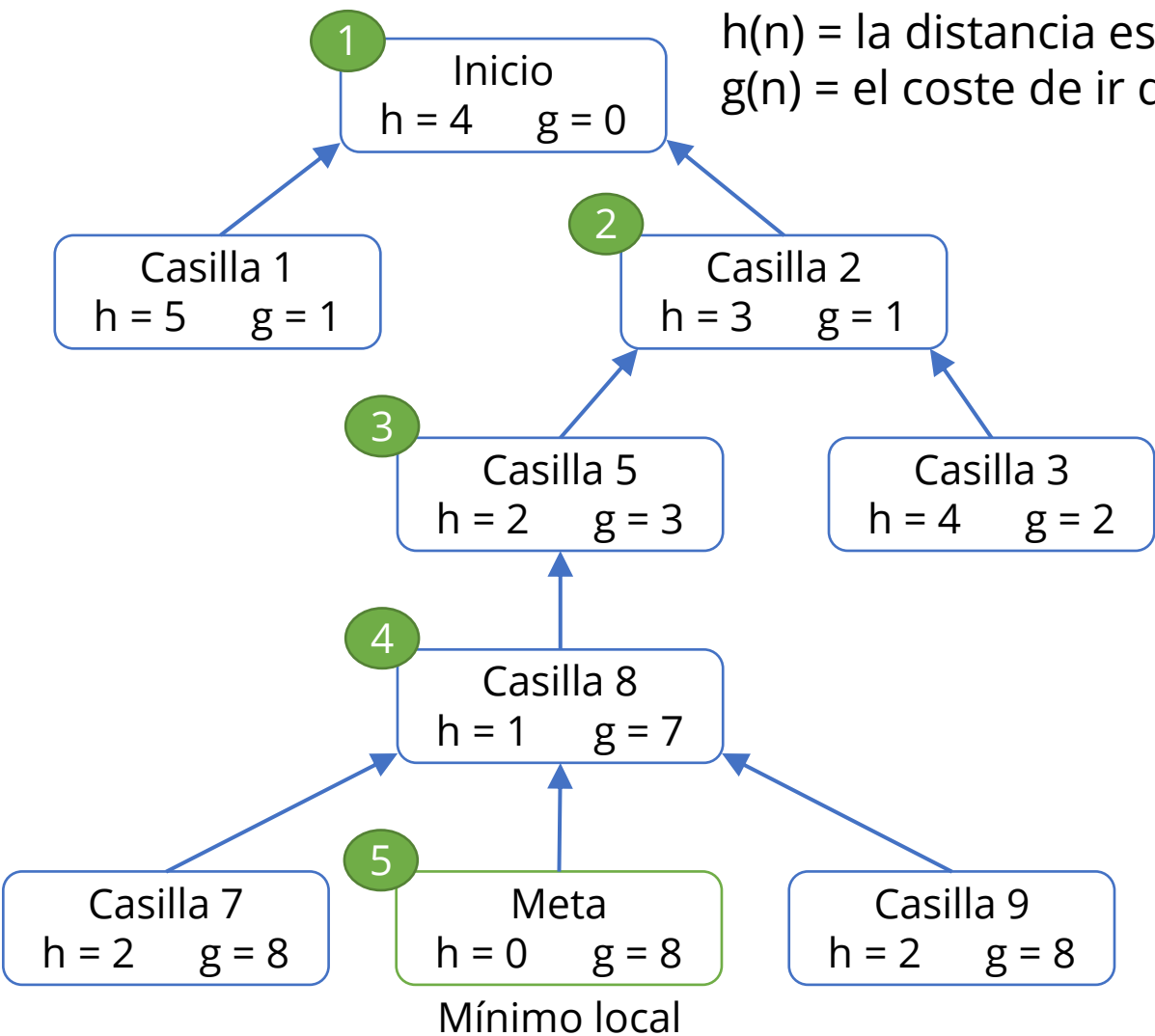
 Coste 1

 Coste 2


 Coste 4


# Solución Búsqueda escalada


$h(n)$  = la distancia estimada hasta la meta  
 $g(n)$  = el coste de ir de Inicio a la casilla actual




1	I	2	3
4		5	
6	7	8	9
	10	M	11

 Muro

 Coste 1

 Coste 2


 Coste 4


# Solución Búsqueda A-Estrella


Inicio  
g=0, h=4, f=4


h(n) = la distancia estimada hasta la meta  
g(n) = el coste de ir de Inicio a la casilla actual  
f(n) = g(n) + h(n)

1	I	2	3
4		5	
6	7	8	9
	10	M	11

 Muro

 Coste 2

 Coste 1

 Coste 4

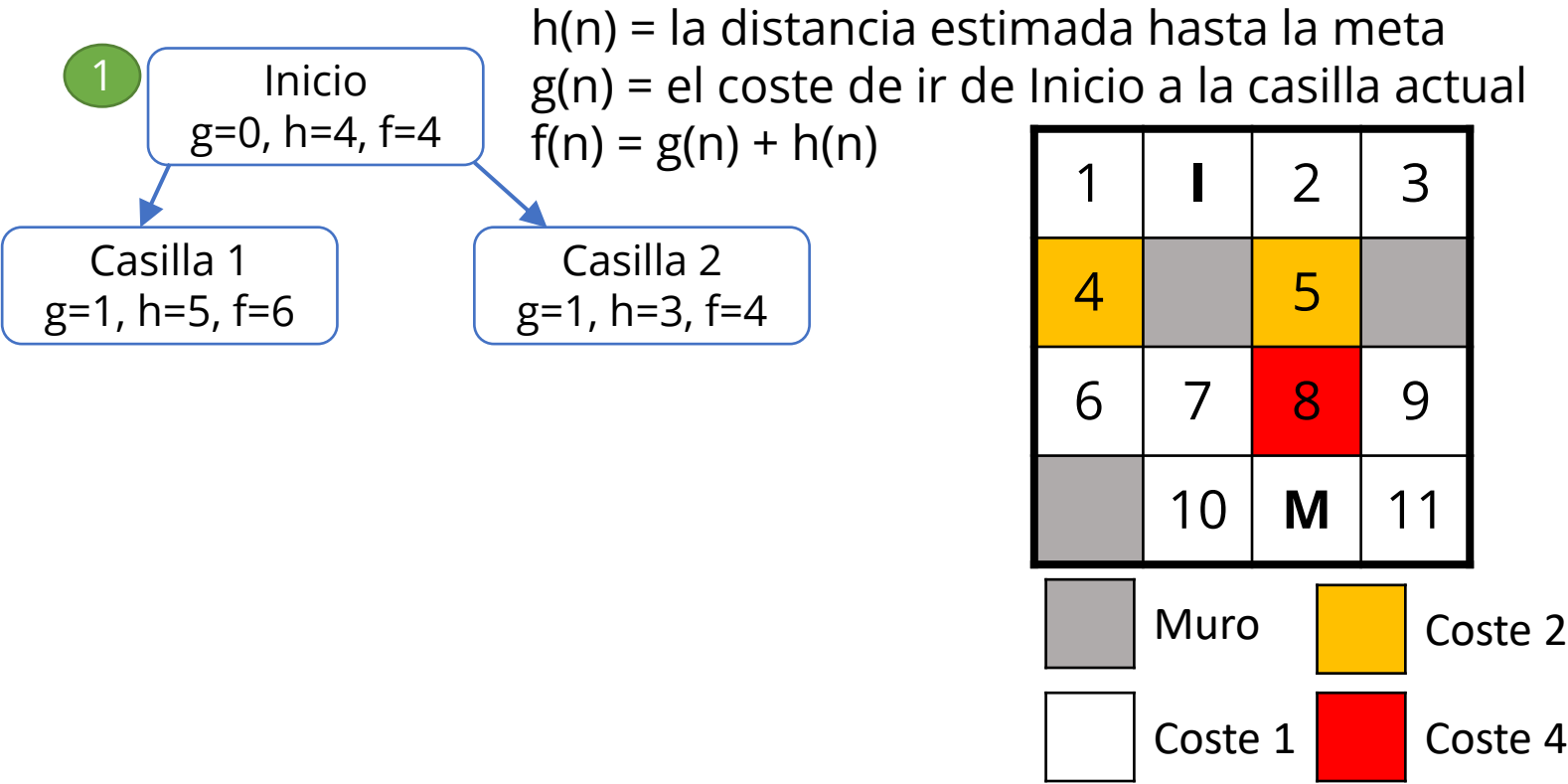
La búsqueda en A\* utiliza una función de evaluación compuesta de heurística y coste.

Los nodos generados se añaden a una lista de nodos abiertos que se ordena por dicha función de evaluación. En cada iteración se selecciona el mejor nodo de dicha lista y se expande generando nuevos nodos.

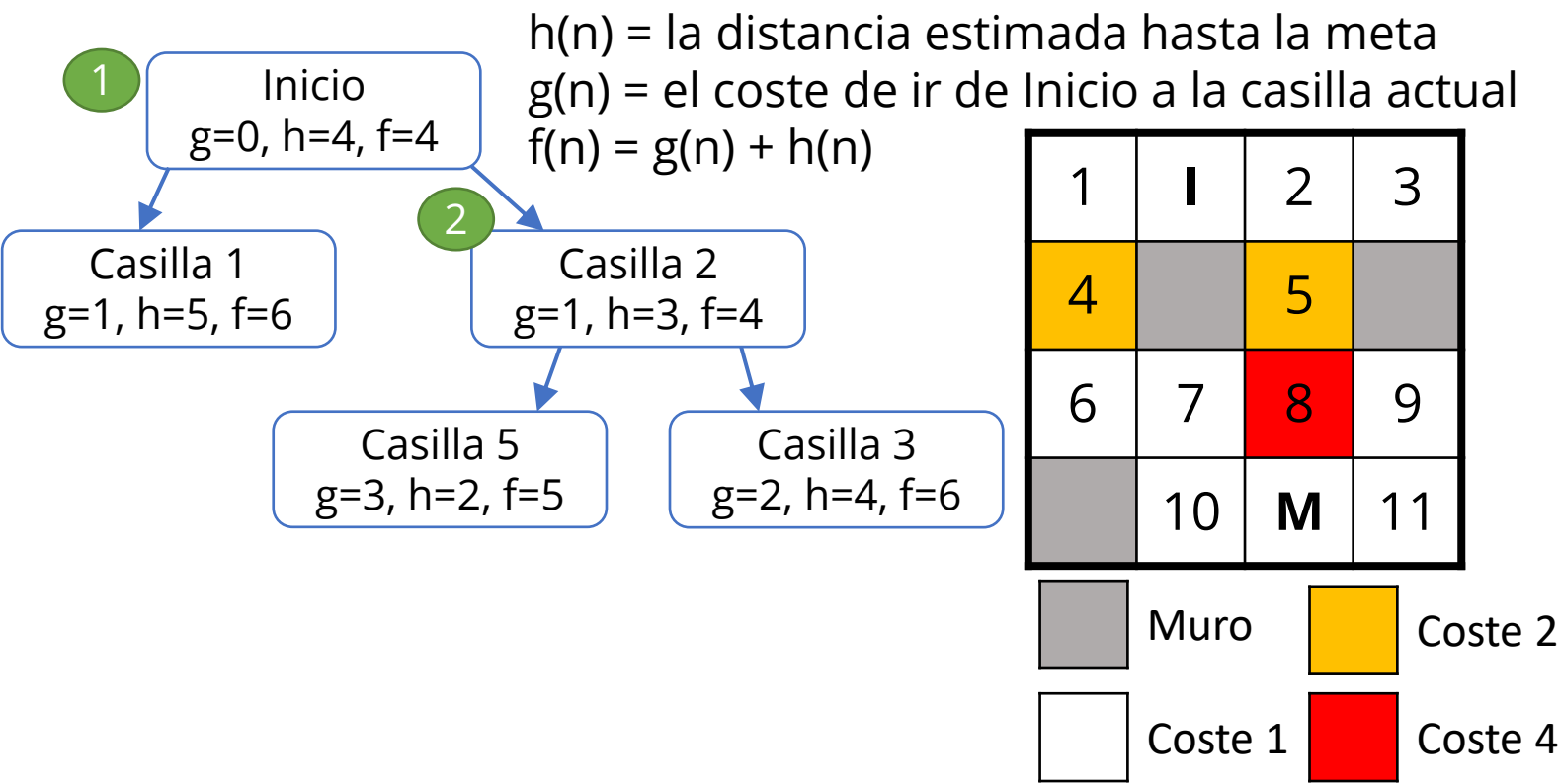
Lista de nodos abiertos (ordenada por evaluación)

Inicio

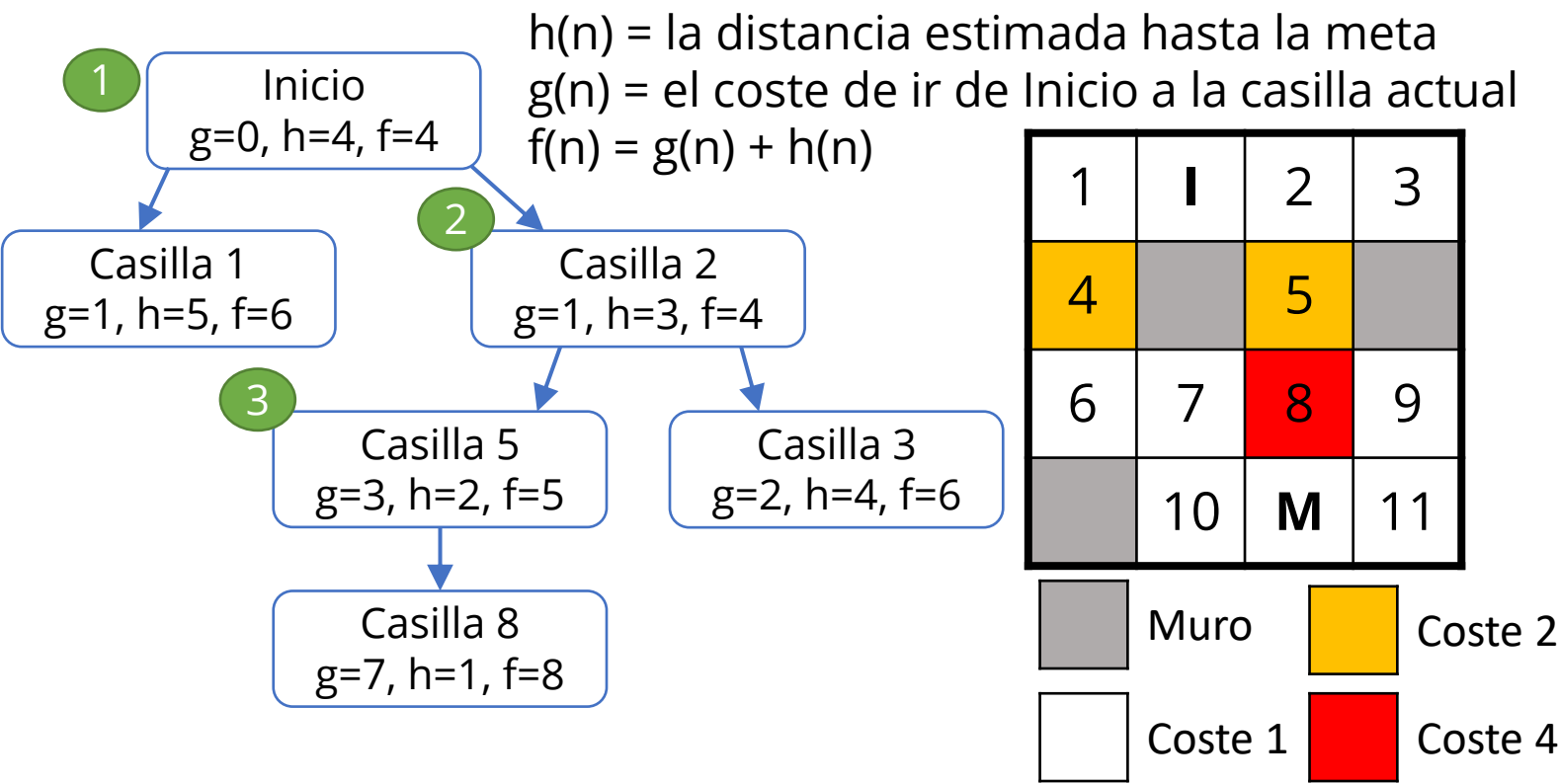
# Solución Búsqueda A-Estrella



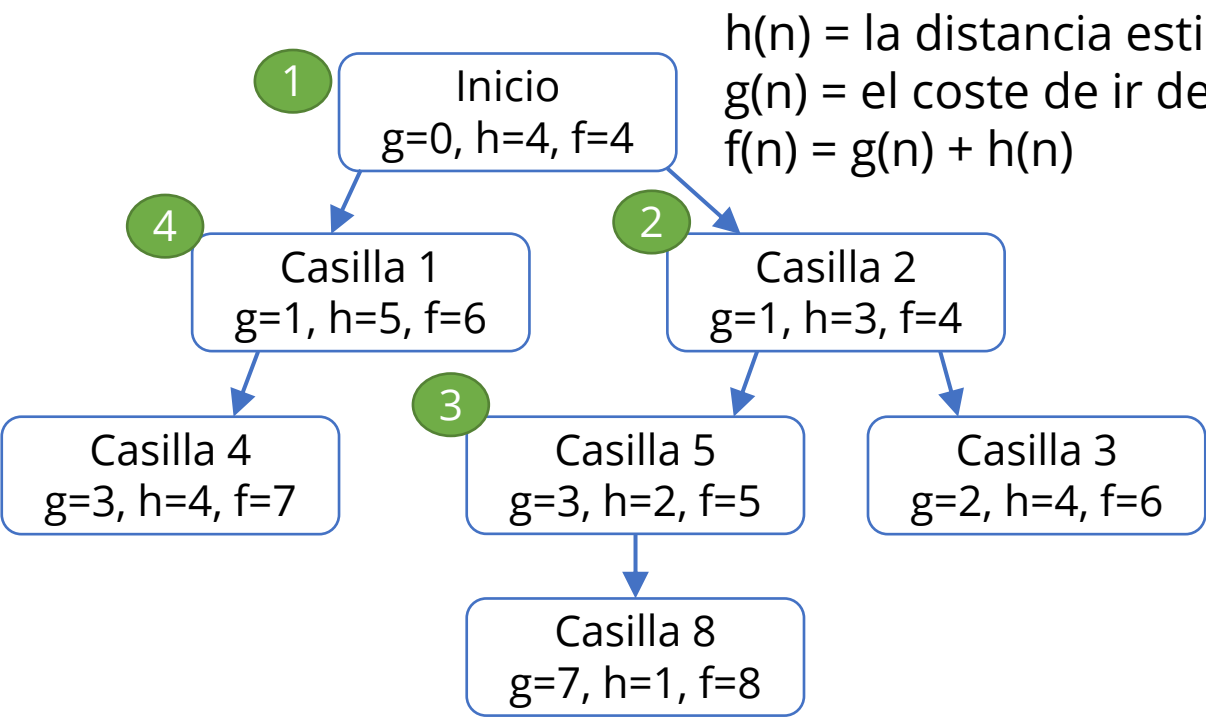
# Solución Búsqueda A-Estrella




# Solución Búsqueda A-Estrella





# Solución Búsqueda A-Estrella




1	I	2	3
4		5	
6	7	8	9
	10	M	11

 Muro

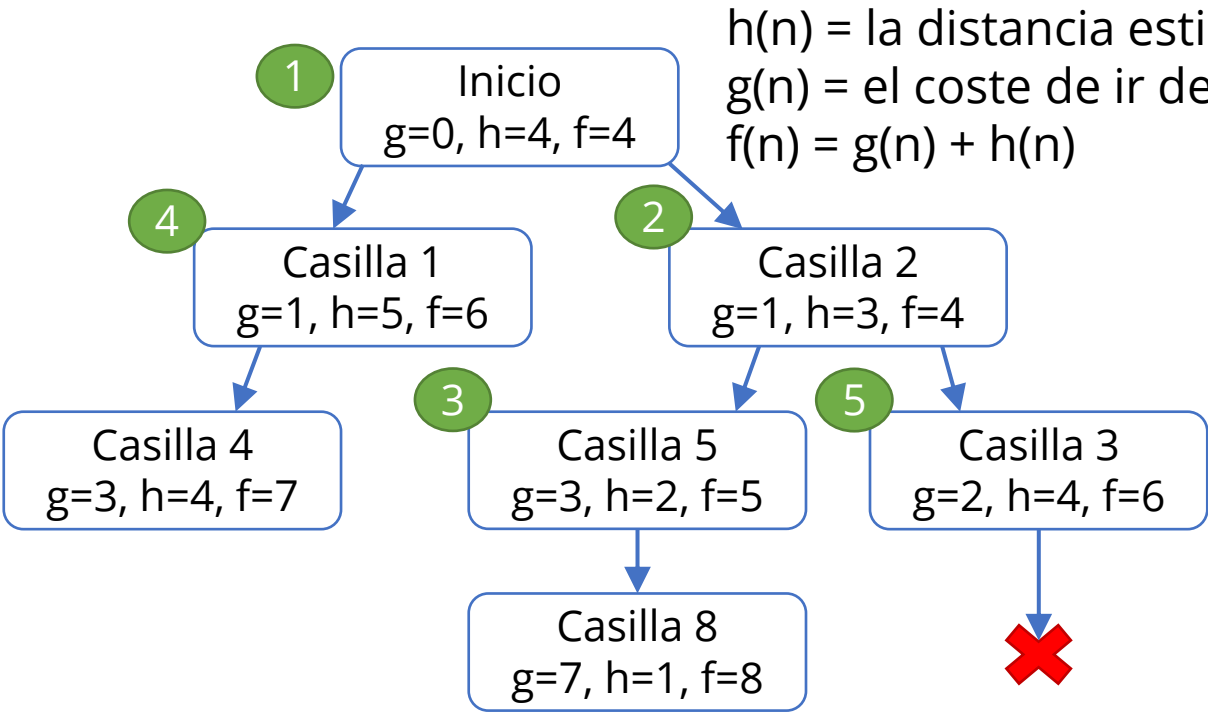
 Coste 2

 Coste 1


 Coste 4





# Solución Búsqueda A-Estrella




1	I	2	3
4		5	
6	7	8	9
	10	M	11

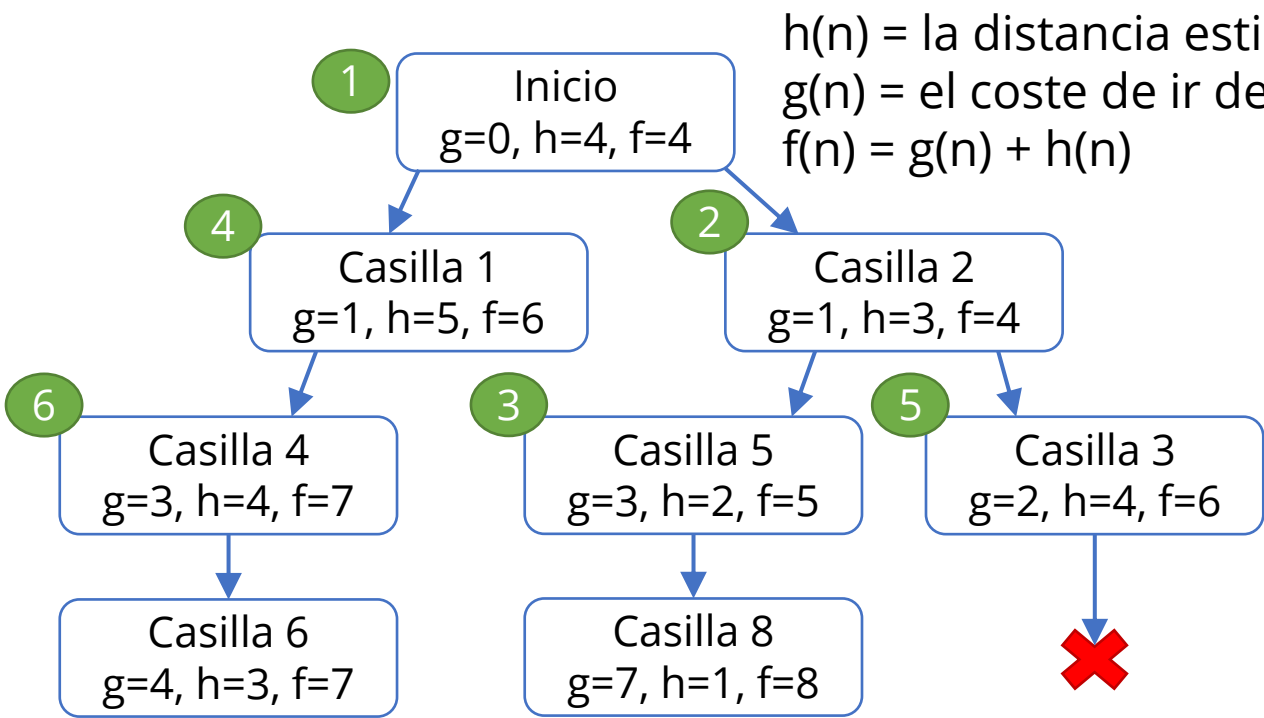
 Muro

 Coste 2


 Coste 1


 Coste 4


# Solución Búsqueda A-Estrella




1	I	2	3
4		5	
6	7	8	9
	10	M	11

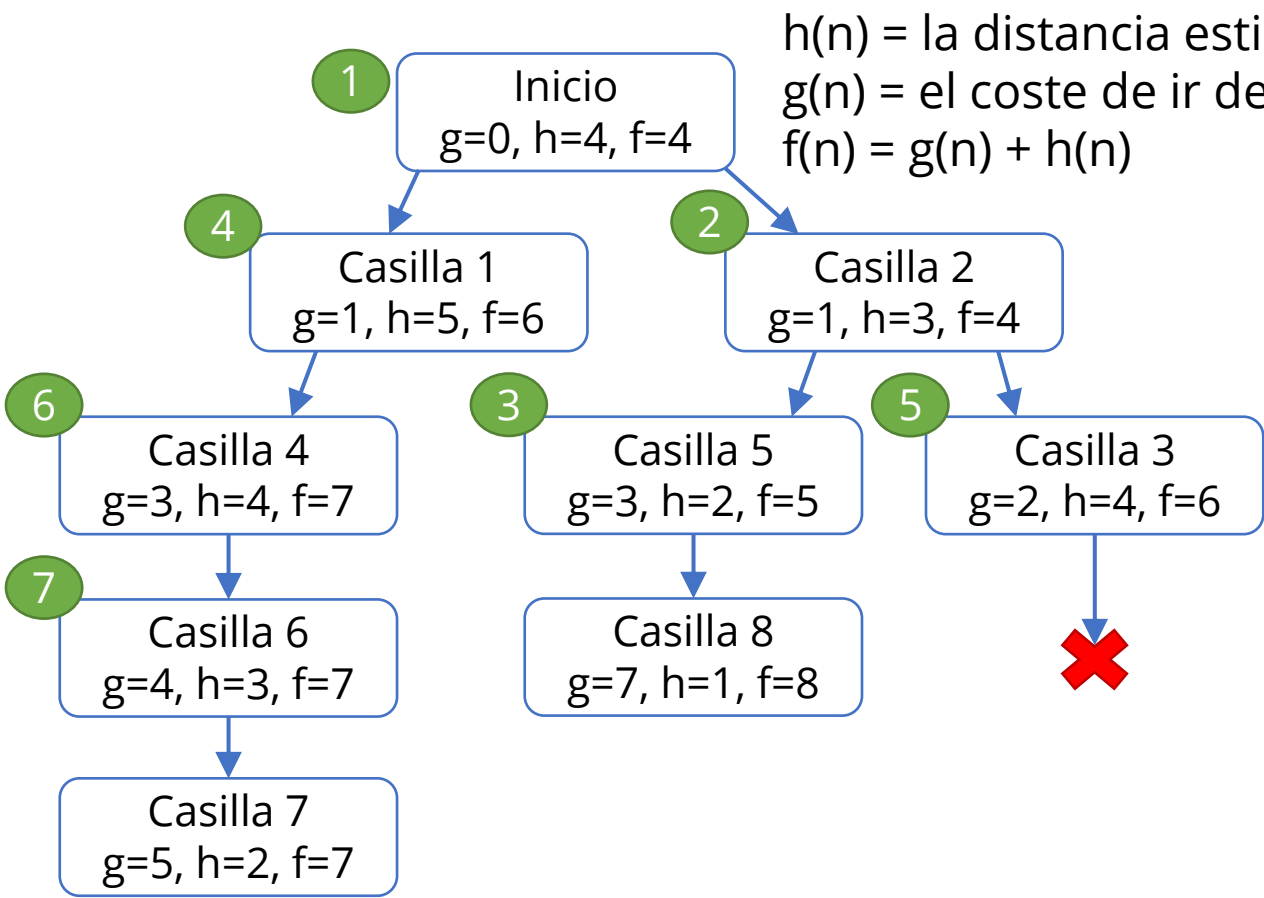
 Muro

 Coste 2


 Coste 1


 Coste 4


# Solución Búsqueda A-Estrella




1	I	2	3
4		5	
6	7	8	9
	10	M	11

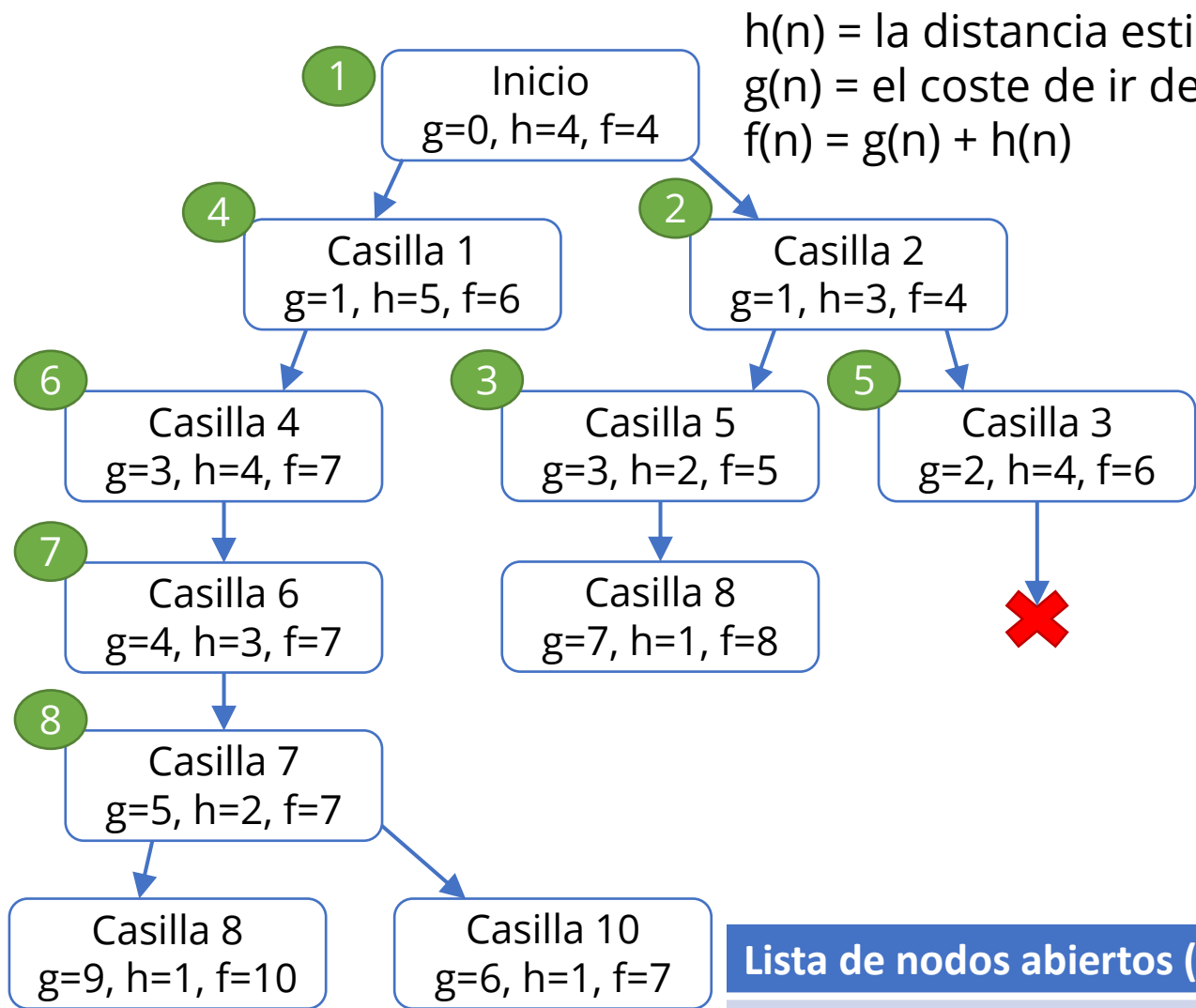
 Muro

 Coste 2

 Coste 1

 Coste 4

# Solución Búsqueda A-Estrella



1	I	2	3
4		5	
6	7	8	9
	10	M	11

 Muro

 Coste 2

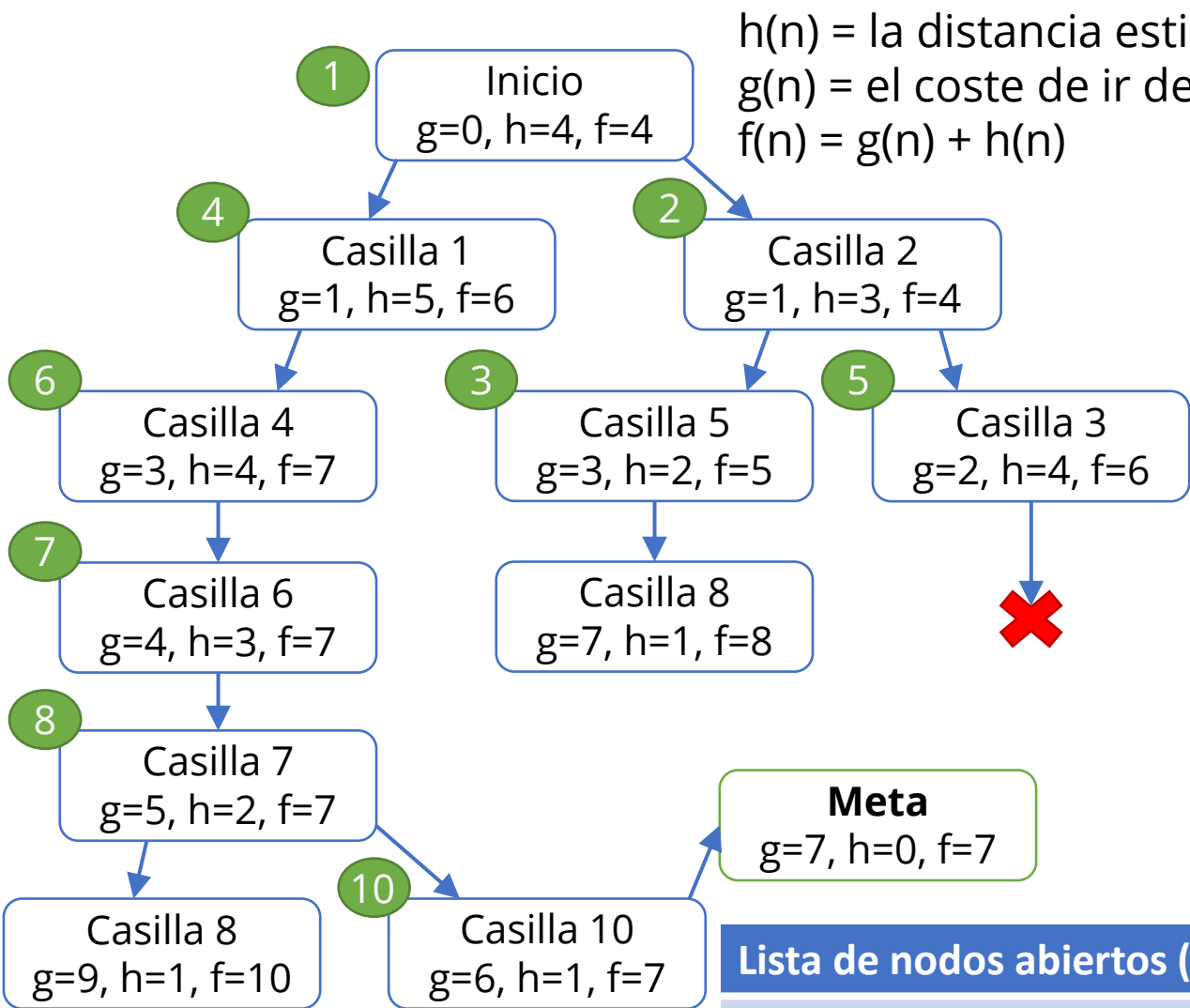
 Coste 1

 Coste 4


Lista de nodos abiertos (ordenada por evaluación)


10,8(f=8),8(f=10)


# Solución Búsqueda A-Estrella




1	I	2	3
4		5	
6	7	8	9
	10	M	11

 Muro

 Coste 2

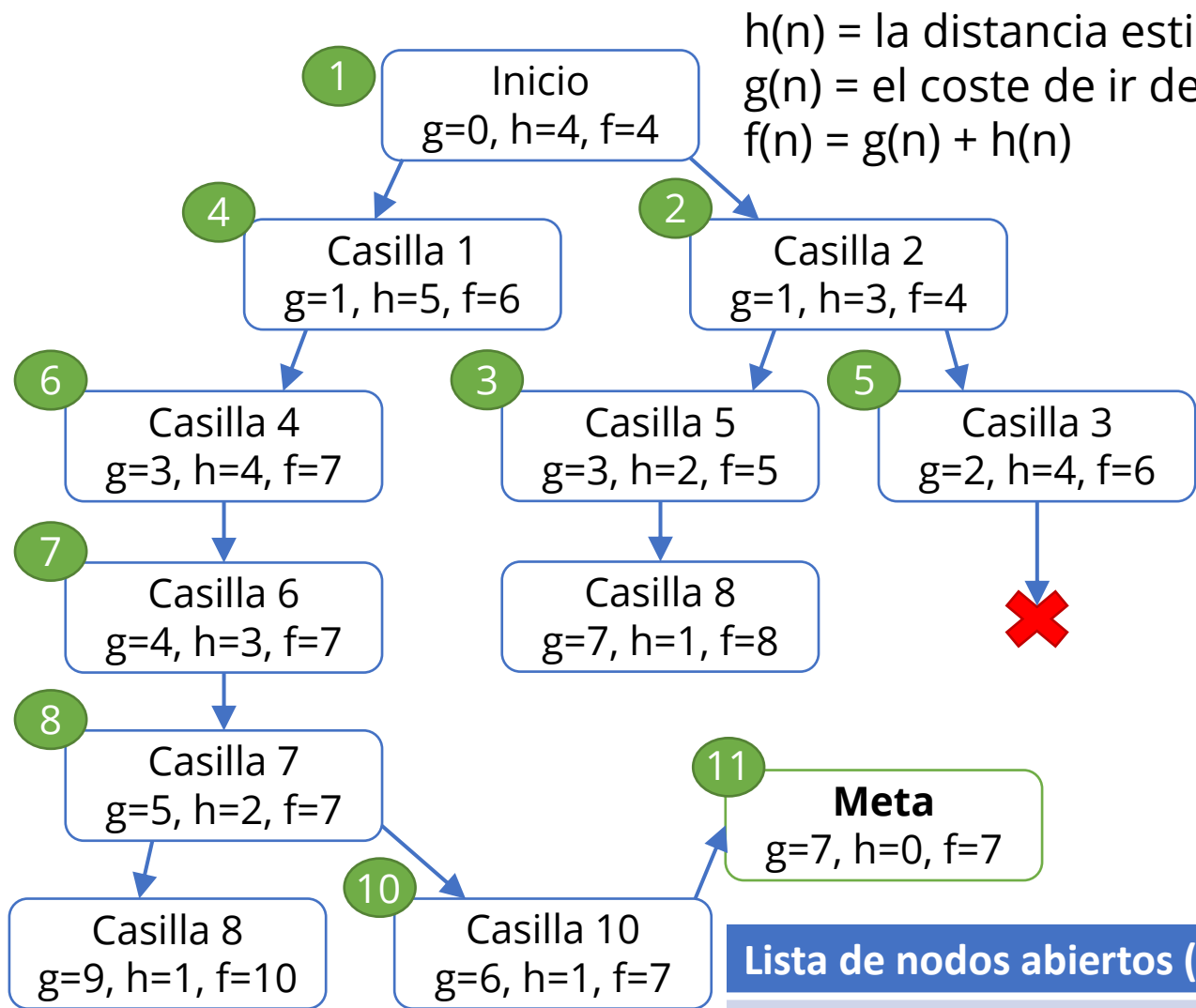
 Coste 1

 Coste 4


Lista de nodos abiertos (ordenada por evaluación)


Meta,8(f=8),8(f=10)


# Solución Búsqueda A-Estrella




1	I	2	3
4		5	
6	7	8	9
	10	M	11

 Muro

 Coste 2

 Coste 1

 Coste 4

Lista de nodos abiertos (ordenada por evaluación)

Se ha llegado a un nodo meta

El objetivo consiste en encontrar el camino con menor coste desde el origen hasta la meta, utilizando los operadores:

1. Mover arriba
2. Mover izquierda
3. Mover abajo
4. Mover derecha

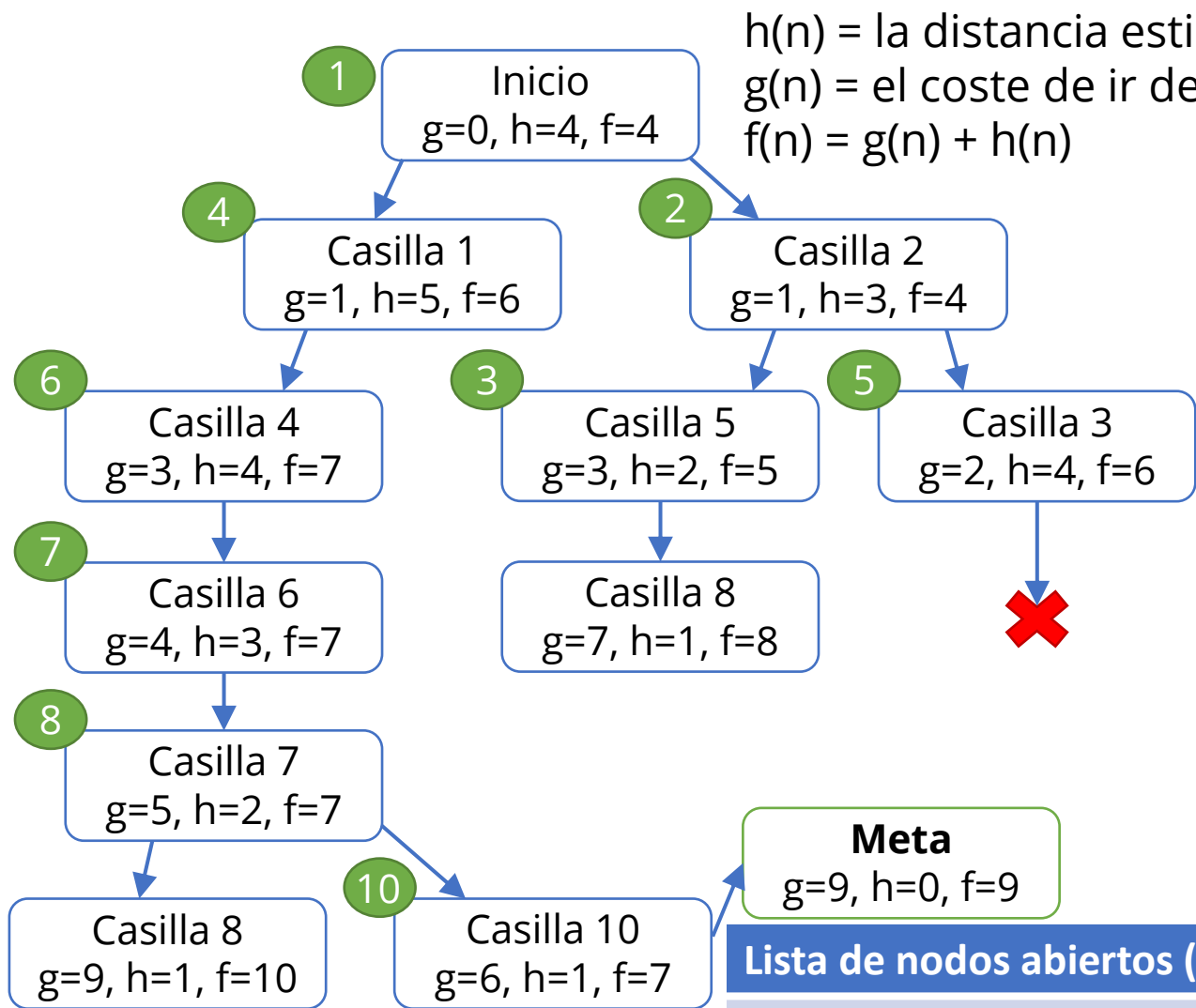
Los números identifican a la celda mientras que el color indica el coste.

Las celdas grises son muros infranqueables, las blancas tienen coste 1, las amarillas 2 y las rojas 4.

1	I	2	3
4		5	
6	7	8	9
	10	M	11

En el problema anterior, considerar que hay una puerta entre la casilla 11 y la casilla meta con coste 3. Utilizando misma función de heurística, ejecutar el algoritmo A\*.

# Solución Búsqueda A-Estrella



1	I	2	3
4		5	
6	7	8	9
	10	M	11

Muro

Coste 2

Coste 1

Coste 4

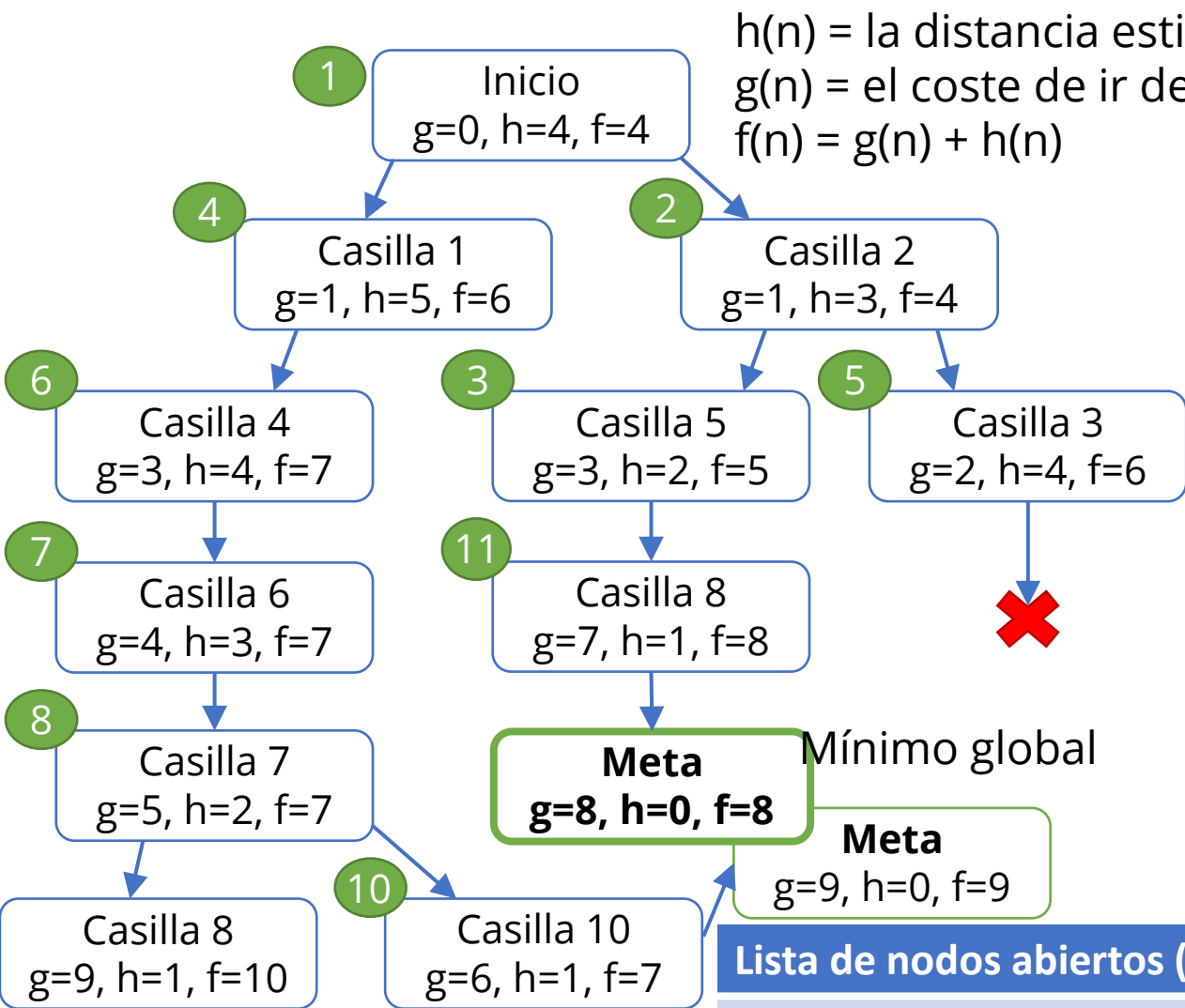
Coste 3

Lista de nodos abiertos (ordenada por evaluación)

8(f=8), **Meta**, 8(f=10)



# Solución Búsqueda A-Estrella



1	I	2	3
4		5	
6	7	8	9
	10	M	11

Muro

Coste 2

Coste 1

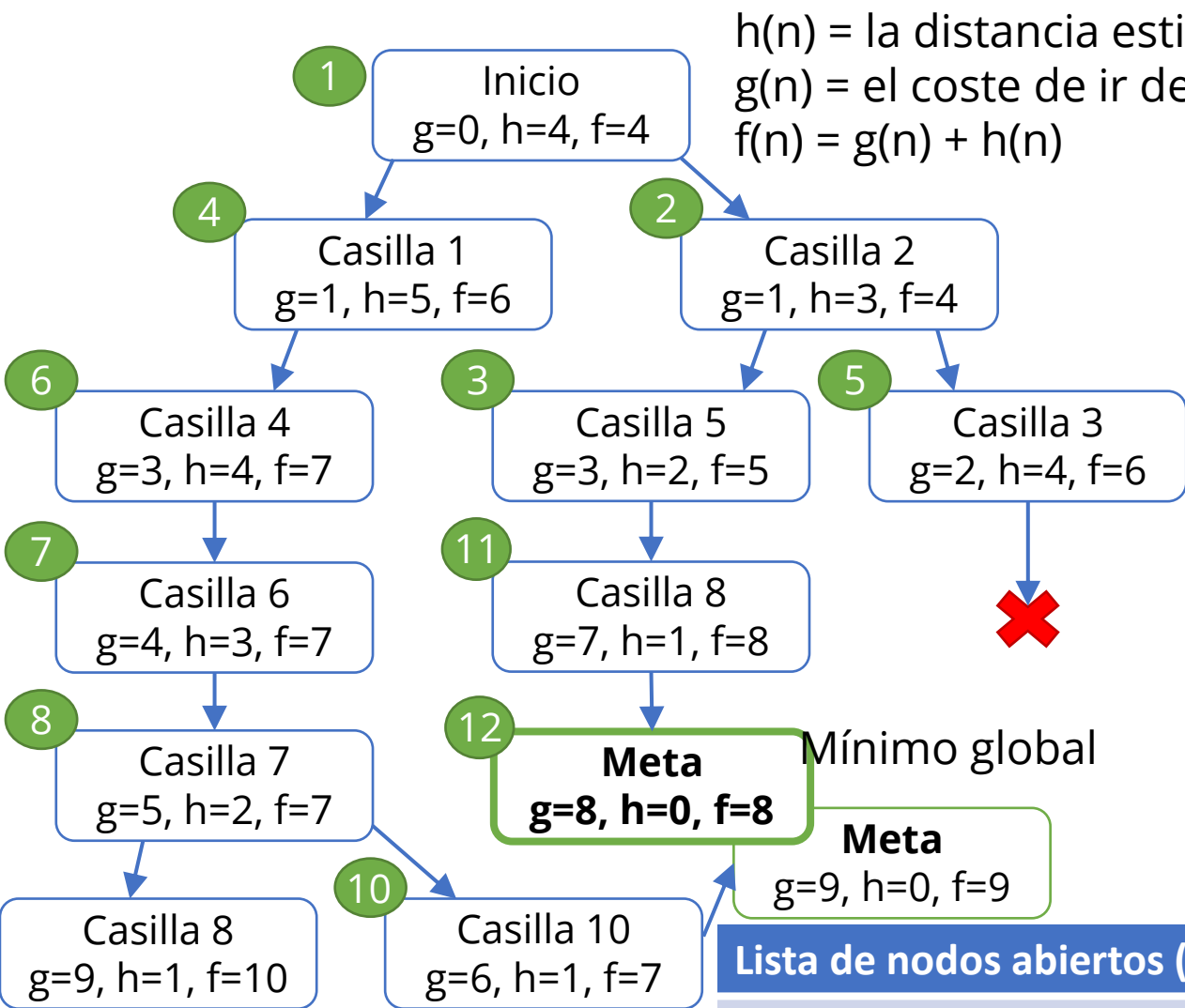
Coste 4

Coste 3


Lista de nodos abiertos (ordenada por evaluación)


Meta(f=8), Meta(f=9), 8(f=10)


# Solución Búsqueda A-Estrella





1	I	2	3
4		5	
6	7	8	9
	10	M	11

 Muro

 Coste 2

 Coste 1

 Coste 4

 Coste 3

Lista de nodos abiertos (ordenada por evaluación)

Se ha llegado a un nodo meta

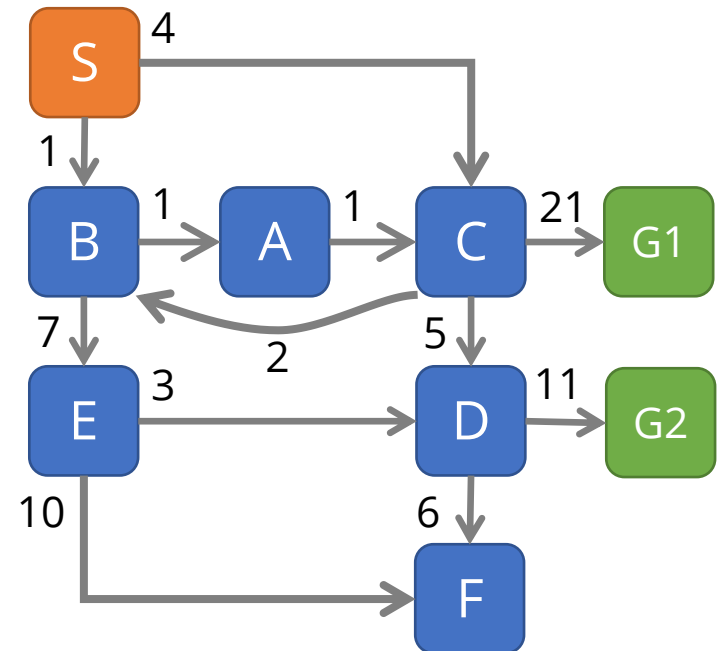
Para este grafo.

- donde los vértices representan estados,
- los arcos son acciones con sus costes asociados,
- el estado inicial es S,
- y el objetivo es llegar a cualquiera de los estados objetivo {G1,G2}.

### Preguntas:

- Proponer 2 heurísticas admisibles, ¿Cuál es más informada?
- Hacer Búsqueda en Escalada.
- Hacer Búsqueda en A\*.

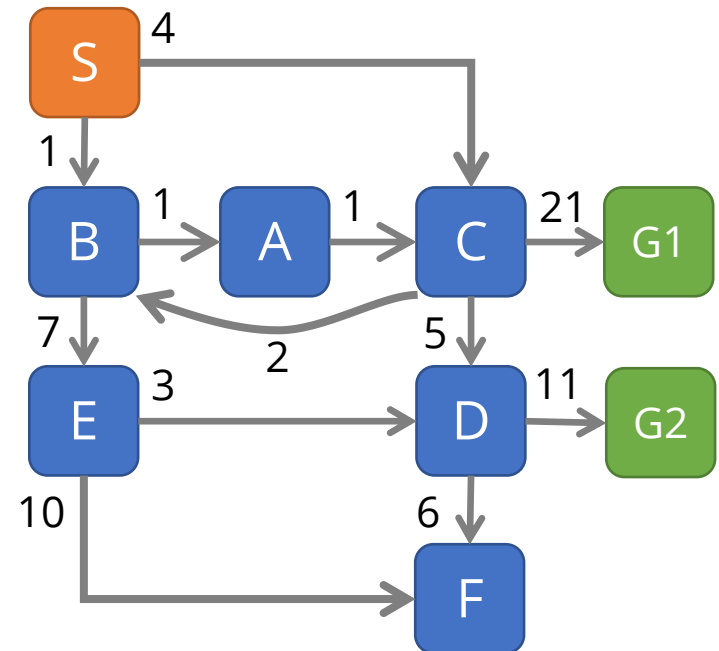
Nota: Para resolver empates, usar la heurística y en segundo lugar el orden alfabético.



Nota: Para resolver empates, usar la heurística y en segundo lugar el orden alfabético..

**H1:**

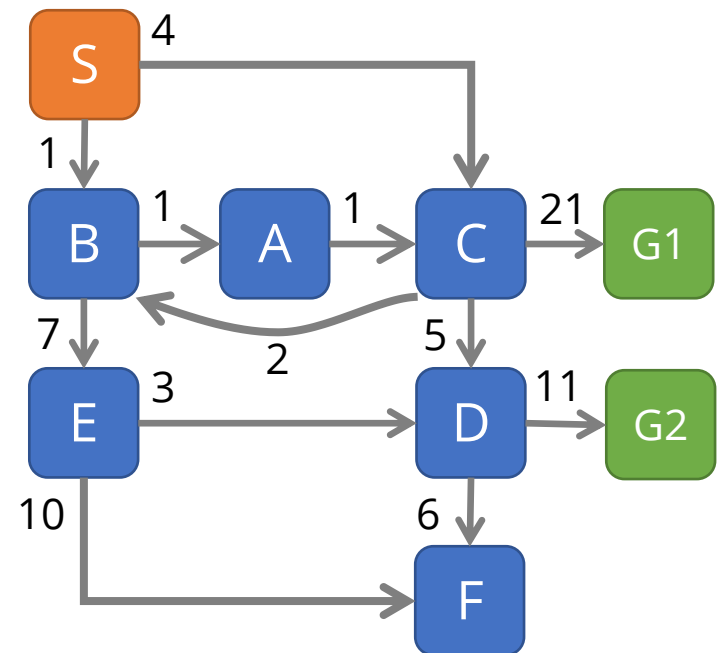
**H2:**



Nota: Para resolver empates, usar la heurística y en segundo lugar el orden alfabético.

**H1:** Saltos mínimos para llegar a un estado meta

**H2:** Dado que el coste medio es aproximadamente 3,8, podemos multiplicar los saltos mínimos por 3.



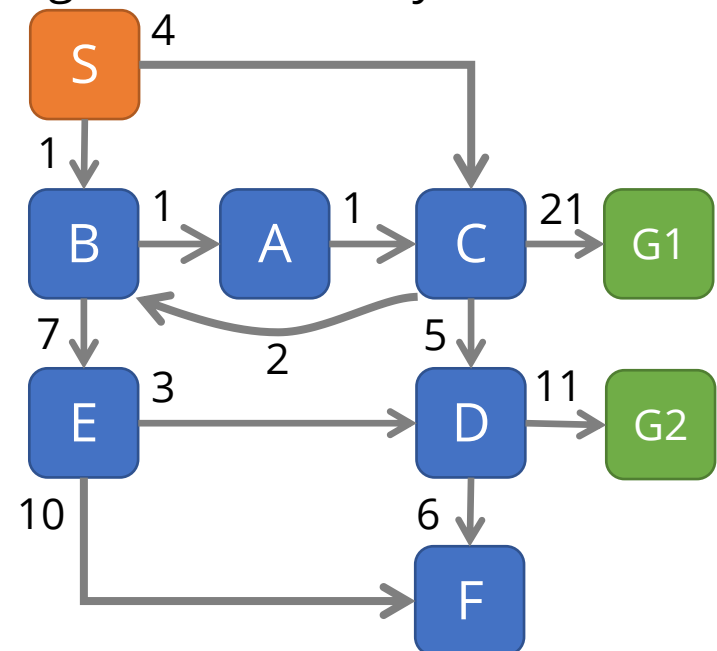
Nota: Para resolver empates, usar la heurística y en segundo lugar el orden alfabético..

**No podemos garantizar que H2 sea admisible porque hay saltos menores que 3, podría haber un camino en el que estuviéramos sobreestimando.**

Igualmente, dentro de este problema pequeño sí podemos ver que lo es, ya que los dos costes más caros son los que llegan a las metas y estos son mucho mayores que 3

**H1:** Saltos mínimos para llegar a un estado meta

**H2:** Dado que el coste medio es aproximadamente 3,8, podemos multiplicar los saltos mínimos por 3.

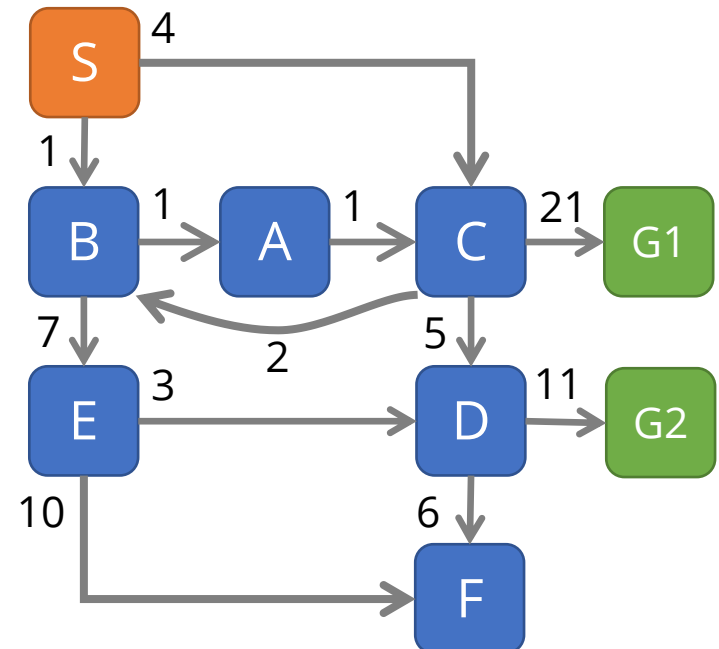


Nota: Para resolver empates, usar la heurística y en segundo lugar el orden alfabético..

H1: Saltos mínimos para llegar a un estado meta

H2: Dado que el coste medio es aproximadamente 3,8, podemos multiplicar los saltos mínimos por 3.

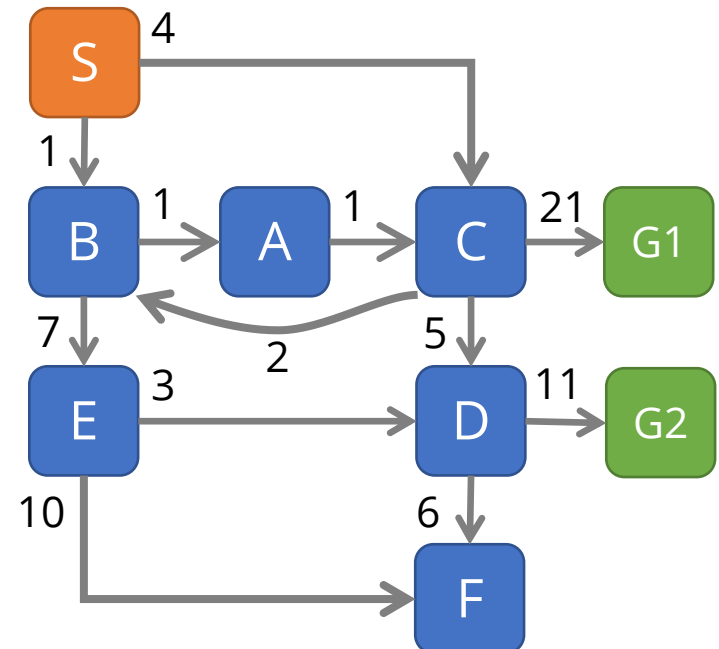
Estado	H1	H2
S	2	6
A	2	6
B	2	9
C	1	3
D	1	3
E	2	6
F	2	6
G1	0	0
G2	0	0



Nota: Para resolver empates, usar la heurística y en segundo lugar el orden alfabético..

Ciclo	Expande	Genera	Abiertos
0	-	-	S

**H1:** Saltos mínimos para llegar a un estado meta





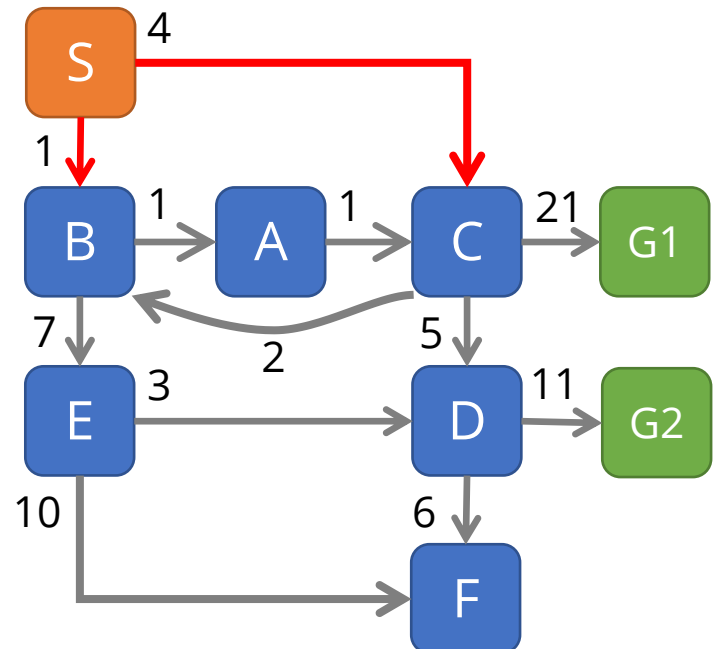
Nota: Para resolver empates, usar la heurística y en segundo lugar el orden alfabético..

Ciclo	Expande	Genera	Abiertos
0	-	-	S
1	S	C,B	C,B

Es importante tener en cuenta el coste y la heurística, porque se podría llegar a un mismo estado con distintos costes

Ciclo	Expande	Genera	Abiertos
0	-	-	S
1	S	C(4,1),B(1,2)	C(4,1),B(1,2)

**H1:** Saltos mínimos para llegar a un estado meta



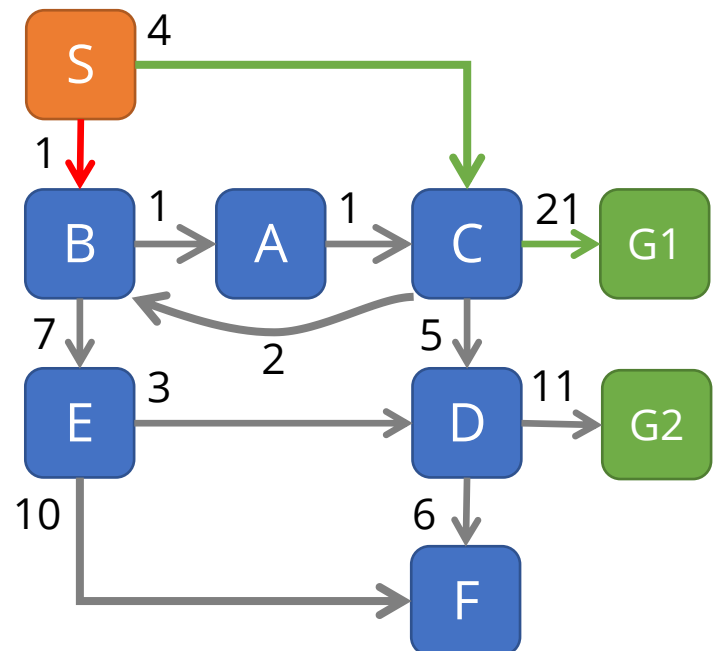
Nota: Para resolver empates, usar la heurística y en segundo lugar el orden alfabético..

Ciclo	Expande	Genera	Abiertos
0	-	-	S
1	S	C(4,1),B(1,2)	C(4,1),B(1,2)
2	C(4,1)	B(6,2), G1(25,0)	B(1,2), G1(25,0)

El nodo B generado es peor al que ya tenemos en la lista de abiertos, por tanto no es necesario guardarlo porque ya tenemos una forma mejor de llegar al mismo

Ciclo	Expande	Genera	Abiertos
3	G1(25,0)		

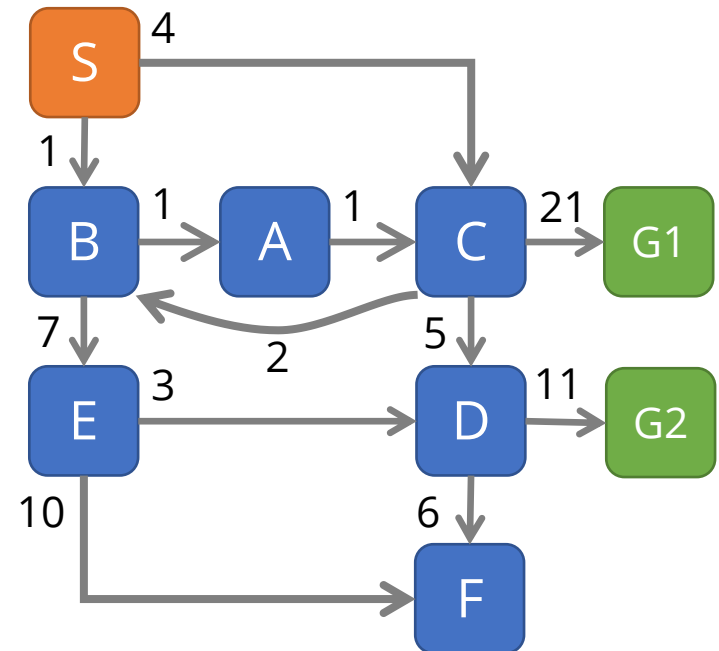
**H1:** Saltos mínimos para llegar a un estado meta



Nota: Para resolver empates, usar la heurística y en segundo lugar el orden alfabético..

Ciclo	Expande	Genera	Abiertos
0	-	-	S

**H2:  $H1 * 3$**

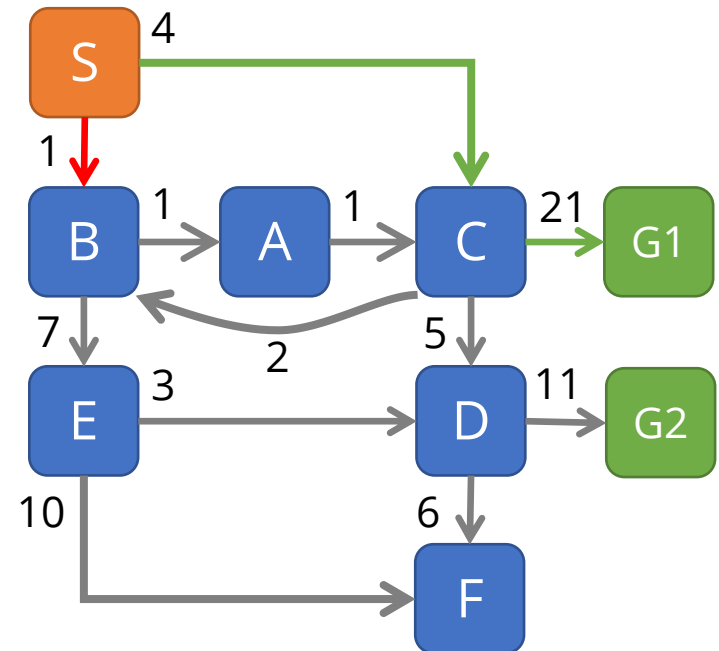


Nota: Para resolver empates, usar la heurística y en segundo lugar el orden alfabético..

Ciclo	Expande	Genera	Abiertos
0	-	-	S

Se va a llegar a la misma solución, ya que la heurística sigue la misma progresión, simplemente esta multiplicada por 3. Esto no siempre es así, sucede en este caso concreto.

**H2:  $H1 \times 3$**

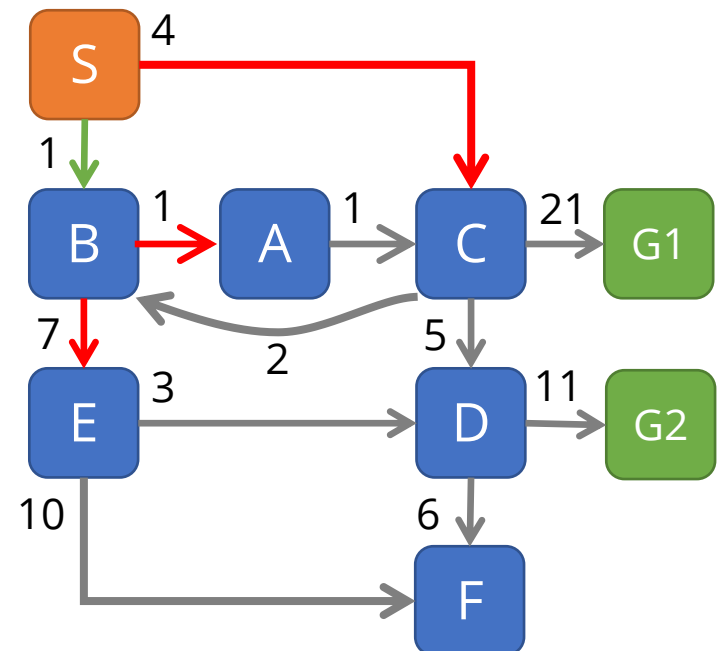


Nota: Para resolver empates, usar la heurística y en segundo lugar el orden alfabético..

Ciclo	Expande	Genera	Abiertos
0	-	-	S
1	S	C(4,1,5), B(1,2,3)	B(1,2,3), C(4,1,5)
2	B(1,2,3)	A(...),E(...)	C(4,1,5)...

En A\* tenemos que incluir la función  $f=g+h$  (coste+heurística), que es la que va a ordenar los nodos

**H1:** Saltos mínimos para llegar a un estado meta



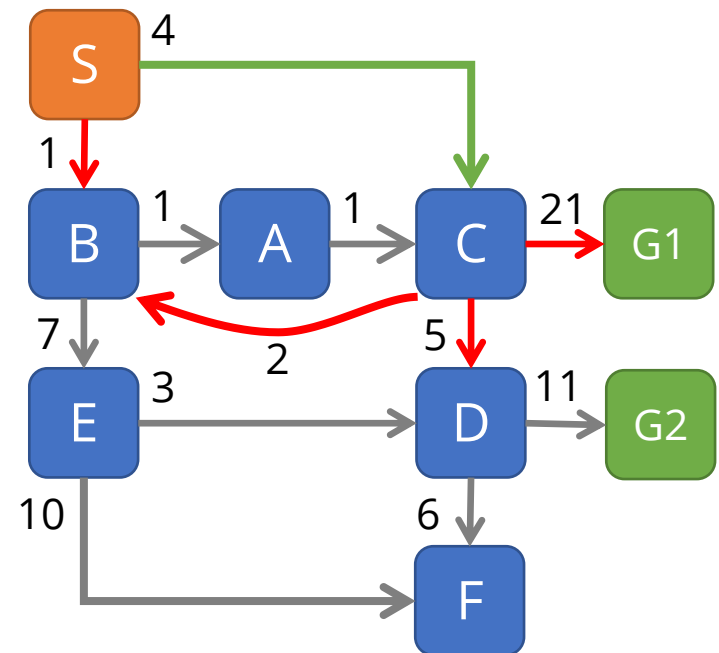
Nota: Para resolver empates, usar la heurística y en segundo lugar el orden alfabético..

Ciclo	Expande	Genera	Abiertos
0	-	-	S
1	S	C(4,1,5), B(1,2,3)	B(1,2,3), C(4,1,5)
2	B(1,2,3)	A(...),E(...)	C(4,1,5)...

**H2:** H1\*3, tiene la misma f por lo que se expande según la heurística

Ciclo	Expande	Genera	Abiertos
0	-	-	S
1	S	C(4,3,7), B(1,6,7)	C(4,3,7), B(1,6,7)
2	C(4,3,7)	B(...),D(...), G1(...)	B(1,6,7)...

**H1:** Saltos mínimos para llegar a un estado meta

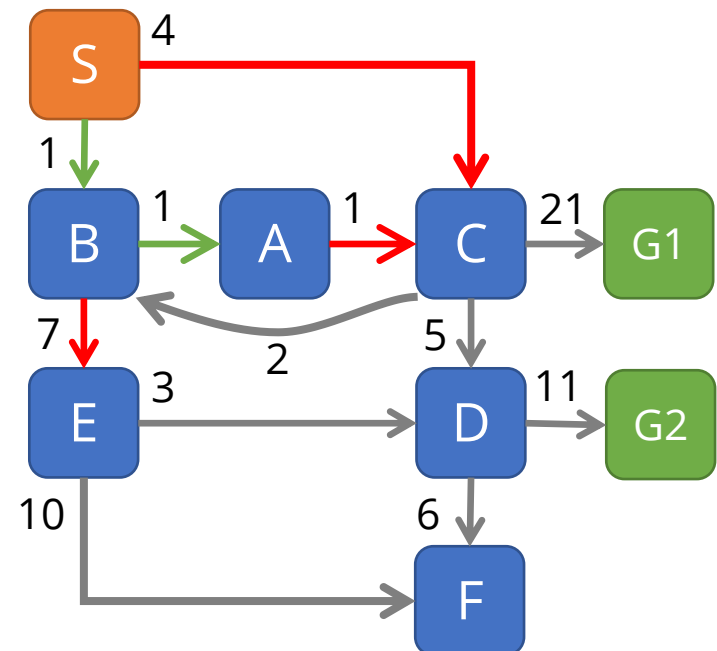


Nota: Para resolver empates, usar la heurística y en segundo lugar el orden alfabético..

Ciclo	Expande	Genera	Abiertos
0	-	-	S
1	S	C(4,1,5), B(1,2,3)	B(1,2,3), C(4,1,5)
2	B(1,2,3)	A(2,2,4), E(8,2,10)	A(2,2,4), C(4,1,5), E(8,2,10)
3	A(2,2,4)	C(3,1,4)	C(3,1,4), C(4,1,5), E(8,2,10)

Aunque ya tenemos C, si que hay que insertarlo porque se llega con mejor función de evaluación, es mas al ser C(4,1,5) peor no será necesario visitarlo porque se va a visitar antes C(3,1,4)

**H1:** Saltos mínimos para llegar a un estado meta

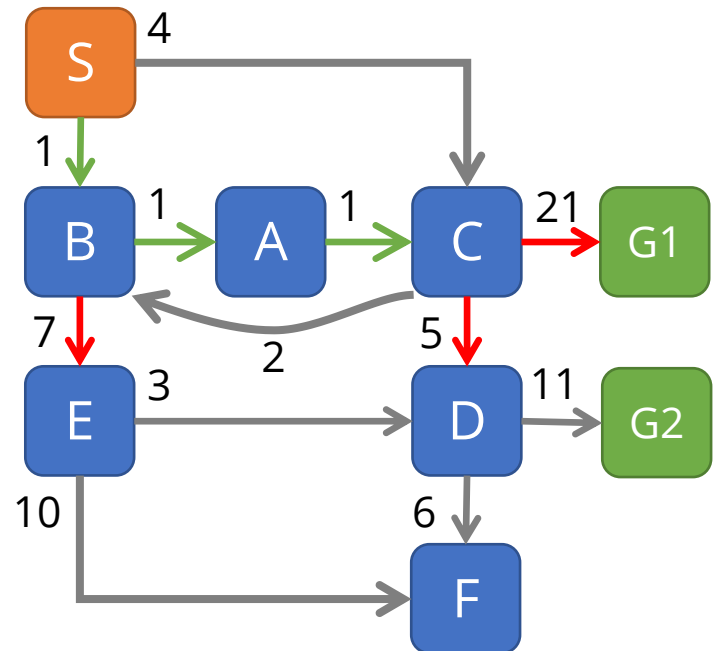


Nota: Para resolver empates, usar la heurística y en segundo lugar el orden alfabético..

Ciclo	Expande	Genera	Abiertos
0	-	-	S
1	S	C(4,1,5), B(1,2,3)	B(1,2,3), C(4,1,5)
2	B(1,2,3)	A(2,2,4), E(8,2,10)	A(2,2,4), C(4,1,5), E(8,2,10)
3	A(2,2,4)	C(3,1,4)	C(3,1,4), E(8,2,10)
4	C(3,1,4)	B(5,2,7), D(8,1,9), G1(24,0,24)	D(8,1,9), E(8,2,10), G1(24,0,24)

B ya ha sido expandido anteriormente con menor coste

**H1:** Saltos mínimos para llegar a un estado meta



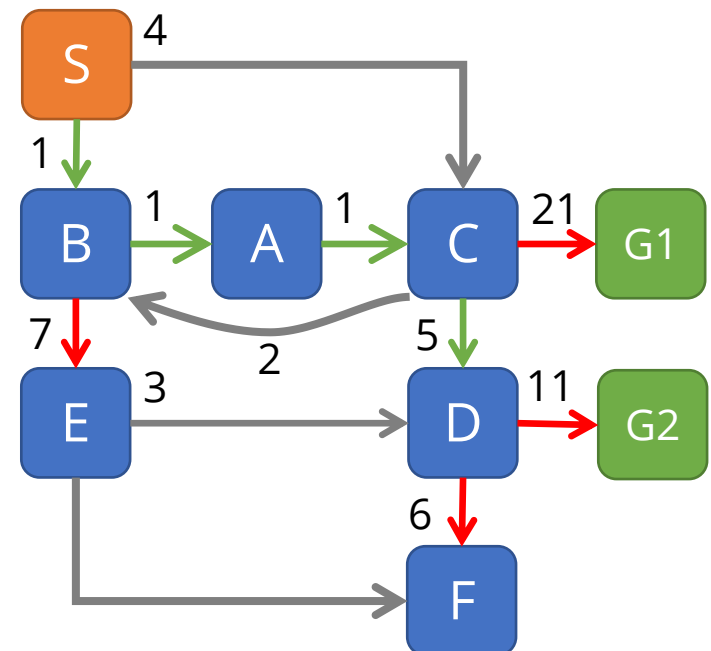


Nota: Para resolver empates, usar la heurística y en segundo lugar el orden alfabético..

Ciclo	Expande	Genera	Abiertos
3	A(2,2,4)	C(3,1,4)	C(3,1,4), E(8,2,10)
4	C(3,1,4)	B(5,2,7), D(8,1,9), G1(24,0,24)	D(8,1,9), E(8,2,10), G1(24,0,24)
5	D(8,1,9)	F(14,2,16), G2(19,0,19)	E(8,2,10), F(14,2,16), G2(19,0,19) G1(24,0,24)

Aunque ya hay un nodo meta (G1) en la lista de abiertos, no podemos asegurar que sea optimo al haber nodos con una mejor función

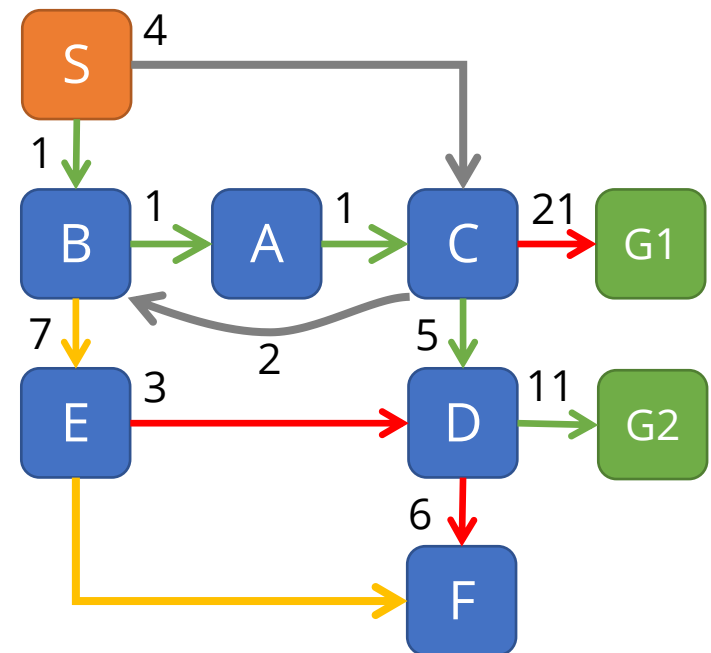
**H1:** Saltos mínimos para llegar a un estado meta



Nota: Para resolver empates, usar la heurística y en segundo lugar el orden alfabético..

Ciclo	Expande	Genera	Abiertos
5	D(8,1,9)	F(14,2,16), G2(19,0,19)	E(8,2,10), F(14,2,16), G2(19,0,19) G1(24,0,24)
6	E(8,2,10)	D(11,1,12), F(18,2,20)	F(14,2,16), G2(19,0,19) G1(24,0,24)
7	F(14,2,16)	-	G2(19,0,19) G1(24,0,24)
8	G2(19,0,19)		

**H1:** Saltos mínimos para llegar a un estado meta



## Ejercicio 5

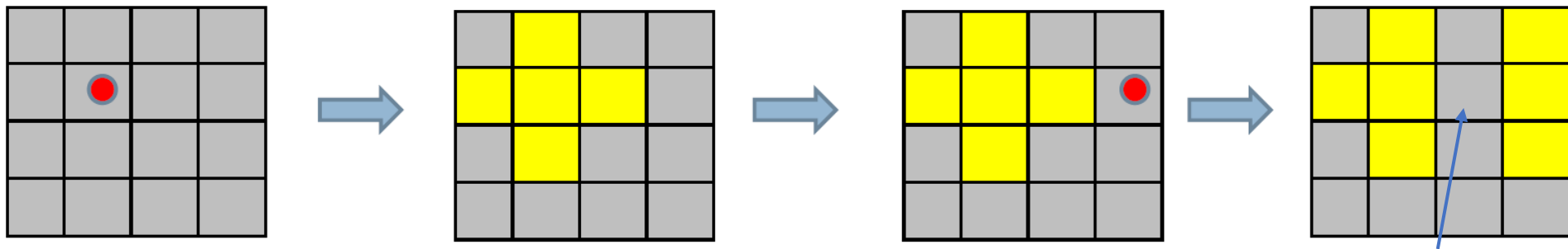
# Enunciado

Pág. 59

El juego consiste en apagar todas las luces de una tabla  $N \times N$ . El objetivo es apagar todas las luces, minimizando el número de acciones. La única acción disponible es pulsar una de las luces, haciendo que tanto esta luz como las cuatro adyacentes inviertan su estado (de apagado al encendido y al revés).

En el estado inicial del juego se tiene un número al azar de estas luces encendidas.

Un ejemplo de la ejecución del juego es:



Se invierte su estado

### Preguntas:

- ¿Cual será la representación de los estados y operadores?
- Definir una función de evaluación (la función del coste + la función de la heurística) para el algoritmo  $A^*$ .

**Estado dentro del grafo:** posición luces en un instante. Dos posibilidades:

1. Matriz NxN de valores binarios para representar encendida (1) o apagada (0).
2. O array de tuplas (x,y,v), siendo v el valor binario.

Estado inicial: luces en cualquier combinación, salvo todas apagadas.

Estado final: todas las luces apagadas. Solo un estado final.

**Espacio de estados:** Hay  $2^{NxN}$  estados posibles, ya que cada una de las NxN luces puede estar en uno de dos estados distintos.

**Operadores:** Press(x,y): cambiar el estado de la bombilla en la celda pulsada y las cuatro celdas adyacentes.

Precondiciones:	$(x, y, ? v0)$	La celda pulsada tenía valor v0 (cualquiera)
	$(x, y - 1, ? vN)$	La celda de arriba
	$(x + 1, y, ? vE)$	La celda de la derecha
	$(x - 1, y, ? vW)$	La celda de izquierda
	$(x, y + 1, ? vS)$	La celda de abajo
Postcondiciones	$(x, y, (1 - ? v0))$	Invierte valor celda pulsada
	$(x, y - 1, (1 - ? vN))$	Invierte valor celda arriba
	$(x + 1, y, (1 - ? vE))$	Invierte valor celda derecha
	$(x - 1, y, (1 - ? vW))$	Invierte valor celda izquierda
	$(x, y + 1, (1 - ? vS))$	Invierte valor celda abajo

---

**Heurísticas:**

El objetivo de la heurística es estimar el costo mínimo desde el nodo actual hasta el nodo objetivo, en este caso, terminar de apagar las luces restantes.

Función de costes:  $g(n)$ : número de cambios desde el estado inicial

Función heurística:

$h_1(n) \rightarrow$  número de bombillas encendidas.

No admisible, porque en una sola acción puede cambiar cinco bombillas.

$h_2(n) \rightarrow$  número de bombillas encendidas/5, (peor caso cambias).

Función de evaluación:

$$f(n) = g(n) + h_2(n)$$

Tenemos un problema de asignación de tareas (t1 a t4) a una serie de trabajadores (w1 a w4). Cada trabajador tarda el tiempo indicado en la tabla en resolver cada tarea.

Tarea→ Trabajador↓	T1	T2	T3	T4
W1	20	5	3	10
W2	10	3	6	30
W3	15	4	4	15
W4	10	6	8	25

### Preguntas:

- Represente el problema para resolverlo mediante búsqueda
- Defina al menos dos heurísticas, y argumente el porque son admisibles y cuál sería preferible usar
- Simule la ejecución de búsqueda en escalada y A\*

### **Estado dentro del grafo:**

En vista de grafo de búsqueda, los nodos son las asignaciones de tareas a trabajadores. Array de tuplas: (tarea-trabajador asignado)

Estado inicial: Todas las tareas sin trabajador asignado. Solo una combinación.

Estado final: Todas las tareas con trabajador asignado. Múltiples estados finales.

### **Espacio de estados:**

Si hay 4 tareas (t1 a t4) y 4 trabajadores (w1 a w4), entonces el espacio de estados contendría 24 (4!) posibles asignaciones de tareas a trabajadores. Cada uno de estos estados representa una solución diferente para el problema de asignación de tareas.

### **Operadores:**

Asignar una tarea a un trabajador: Este operador consiste en asignar una tarea específica a un trabajador específico.

### Heurísticas:

El objetivo de la heurística es estimar el costo mínimo desde el nodo actual hasta el nodo objetivo, en este caso, completar la asignación de tareas restantes.

**Función de costes:**  $g(n)$ : suma de costes de las tareas asignadas hasta el momento

**Función heurística**  $h(n)$ : varias alternativas

$h_4(n)$ : cantidad de tareas pendientes.

- admisible porque siempre subestimaré el costo real, ya que en el peor caso, cada tarea tendrá que ser completada por un trabajador distinto, lo que requerirá el tiempo máximo posible.

$h(n)$ : cantidad de trabajadores disponibles.

- no es admisible porque no tiene en cuenta el tiempo que tardan los trabajadores en completar sus tareas y puede sobreestimar el costo real.

$h_1(n)$ : tiempo medio estimado para completar tareas restantes.

- no es admisible porque puede sobreestimar o subestimar el costo real, dependiendo de la asignación de tareas.

$h_2(n)$ : tiempo máximo estimado para completar tareas restantes.

$h_3(n)$ : tiempo mínimo estimado para completar tareas restantes.



A partir de este punto se presentan ejercicios sin solución que pueden ser resueltos por el alumno para apoyar el estudio de la asignatura. Cualquier duda de los mismos o de ejercicios previos puede ser consultada con el profesor de prácticas en el mail: [damigo@inf.uc3m.es](mailto:damigo@inf.uc3m.es)

Tenemos un problema de asignación de rutas (TSP), en el que hay que generar un camino que visite cinco ciudades (C1 a C5) cuyas distancias aparecen en la tabla.

	C1	C2	C3	C4	C5
C1	-	5	3	10	8
C2	5	-	6	20	10
C3	3	6	-	15	3
C4	10	20	15	-	10
C5	8	10	3	10	-

### Preguntas:

- Represente el problema para resolverlo mediante búsqueda
- Defina al menos dos heurísticas, y argumente el porque son admisibles y cuál sería preferible usar
- Simule la ejecución de búsqueda en escalada y A\*