Ejercicios 4: Búsqueda heurística

Departamento de Informática / Department of Computer Science
Universidad Carlos III de Madrid

Inteligencia Artificial
Grado en Ingeniería Informática
2019/20

Tabla de contenidos

Notas sobre el formato de la solución

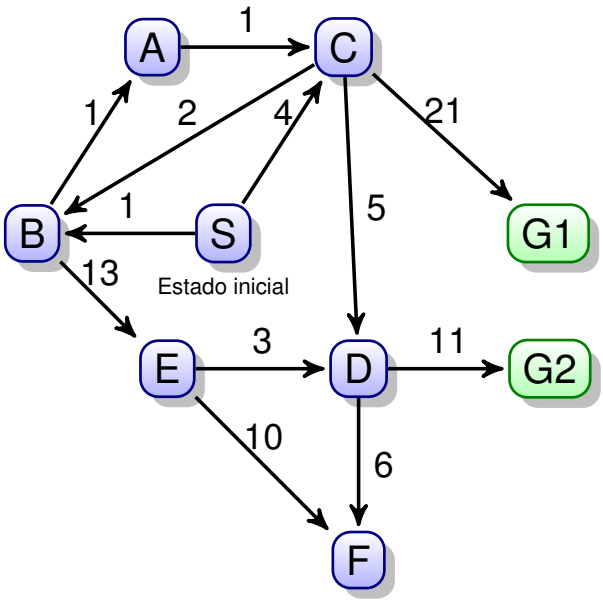
- Para definir un estado hay que escribir toda la información que lo caracteriza
- Para simular la ejecución de un algoritmo no basta indicar la solución, sino que hay que dejar claro el orden de expansión de los nodos. Se puede usar uno cualquiera de los siguientes formatos (en algún ejercicio se especifica cuál):
  - 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 sea un estado, etiquetado con el orden en que ha sido generado y expandido, así como el resto de parámetros (heurística, coste acumulado, función de evaluación, etc.). Si es posible, cada arco tiene que etiquetarse con el nombre (y parámetros si hay) de la acción utilizada.
- Para definir una heurística, si es posible debe escribirse una fórmula que sea capaz de calcularla a partir del estado. Nota: en el ejercicio 1 tal cosa no es posible, se usa una tabla.

Ejercicio 1: Búsqueda en Grafo con costes

Tenemos el problema descrito en el grafo de la figura, donde los vértices representan estados, los arcos acciones con sus costes asociados, el estado inicial es S y el objetivo es llegar a uno de los estados objetivo {G1,G2}.

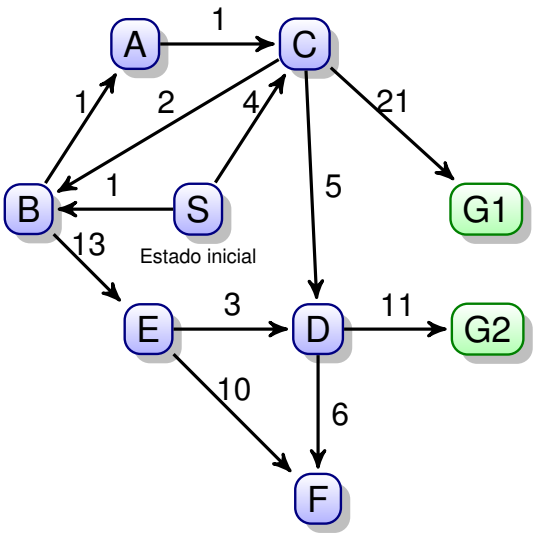
- Defina dos heurísticas admisibles, en la que una sea más informada que otra.
- En ambos casos, aplique búsqueda en escalada y A*. Represente la ejecución en forma de tabla, indicando en cada línea el nodo expandido, nodos generados, y lista abierta obtenida. Indique la solución obtenida. Compare los resultados.

  Nota: como criterio de desempate use el mejor valor de h, y secundariamente el orden alfabético.



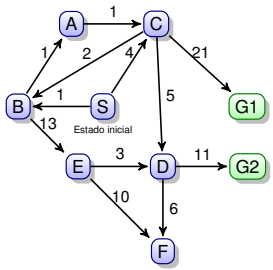
Grafo con costes y heurística, heurísticas

- Algunas posibles heurísticas:
  - $H_1(n)$. Usamos como heurística para n el número mínimo de operadores que hay que usar para llegar desde a n a un estado objetivo, sin tener en cuenta la dirección de las flechas (así podemos dar valor a F).
  - $H_2(n)$. Usamos como heurística para n el triple del valor anterior $H_2(n) = H_1(n) * 3$. Estaría garantizado que es admisible si todos los costes fueran como mínimo 3. En este caso podemos razonar que llegar al nodo G1 cuesta como mínimo 21 y al nodo G2, 11.



Estado	H_1(n)	H_2(n)
S	2	6
A	2	6
B	2	6
C	1	3
D	1	3
E	2	6
F	2	6
G1	0	0
G2	0	0

Grafo con costes y heurística, escalada



H1(n):

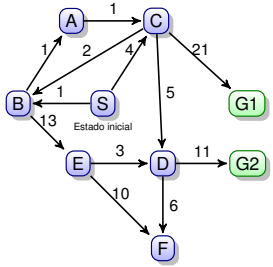
ESCALADA

ABIERTA	Exp. (min H)	Gen. (g,h)
	S	B(1,2),C(4,1)
C(4,1),B(1,2)	C(4,1)	B(4+2,2)[1],G1(4+21,0)

(1): este nodo B no se genera al expandir C, porque hay un nodo B predecesor de C.

Solución: S - C - G1, Longitud: 2, Coste: 25

Grafo con costes y heurística, escalada



H2(n):

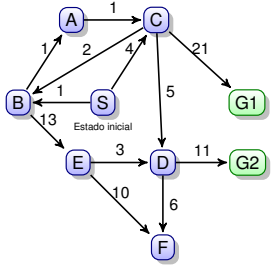
ESCALADA

ABIERTA	Exp. (min H)	Gen. (g,h)
	S	B(1,6),C(4,3)
C(4,3),B(1,6)	C(4,3)	B(4+2,6)[1],G1(4+21,0)

(1): este nodo B no se genera al expandir C, porque hay un nodo B predecesor de C.

Solución: S - C - G1, Longitud: 2, Coste: 25

Grafo con costes y heurística, A*



H1(n):

A*

ABIERTA	Exp. (min F)	Gen. (g,h,f)
	S	B(1,2,3),C(4,1,5)
B(1,2,3),C(4,1,5)	B(1,2,3)	A(1+1,2,4),E(1+13,2,16)
A(2,2,4),C(4,1,5),E(14,2,16)	A(2,2,4)	C(2+1,1,4)[1]
C(3,1,4),E(14,2,16)	C(3,1,4)	B(3+2,2,8)[2],D(3+5,1,9),G1(3+21,0,24)[3]
D(8,1,9),E(14,2,16),G1(24,0,24)	D(8,1,9)	F(8+6,2,16),G2(8+11,0,19)
E(14,2,16),F(14,2,16), G2(19,0,19),G1(24,0,24)	E(14,2,16)	D(14+3,1,18)[2], F(14+10,2,26)[4]
F(14,2,16),G2(19,0,19),G1(24,0,24)	F(14,2,16)	
G2(19,0,19),G1(24,0,24)	G2(19,0,19)	META

(1): el nodo C recién generado tiene menor valor de *f* que el que está en ABIERTA , de manera que reemplaza al previo.

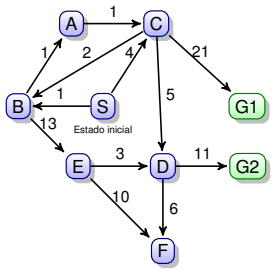
(2): el nodo recién generado tiene mayor valor de *f* que el que está ya cerrado, de manera que se descarta (se reabriría en caso contrario).

(3): en A* sólo se termina cuando se expande un nodo objetivo.

(4): el nodo recién generado tiene mayor valor de *f* que el que está en ABIERTA , de manera que se descarta.

Solución: S-B-A-C-D-G2, Longitud: 5, Coste: 19

Grafo con costes y heurística, A*



H2(n):

A*

ABIERTA	Exp. (min F)	Gen. (g,h,f)
	S	B(1,6,7),C(4,3,7)
C(4,3,7),B(1,6,7)	C(4,3,7)	B(4+2,7,13)[1],D(4+5,3,12),G1(4+21,0,25)[2]
B(1,6,7),D(9,3,12),G1(25,0,25)	B(1,6,7)	A(1+1,6,8),E(1+13,6,14)
A(2,6,8),D(9,3,12)	A(2,6,8)	C(2+1,3,6)[3]
E(14,6,20),G1(25,0,25)		
C(3,3,6),D(9,3,12)	C(3,3,6)	B(3+2,9,14)[4],D(3+5,3,11)[5],G1(3+21,0,24)[5]
E(14,6,20),G1(25,0,25)		
D(8,3,11),E(14,6,20),G1(24,0,24)	D(8,3,11)	F(8+6,6,20), G2(8+11,0,19)
G2(19,0,19), E(14,6,20),	G2(19,0,19)	META
F(14,6,20),G1(24,0,24)		

(1): el nodo B recién generado tiene mayor valor de *f* que el que está en ABIERTA , de manera que se descarta.

(2): en A* sólo se termina cuando se expande un nodo objetivo.

(3): el nodo C recién generado tiene menor valor de *f* que el C que se había cerrado, de manera que se reabre.

(4): el nodo recién generado tiene mayor valor de *f* que el análogo ya cerrado, de manera que se descarta (se reabriría en caso contrario).

(5): el nodo generado tiene menor *f* que el que está en ABIERTA , de modo que lo reemplaza

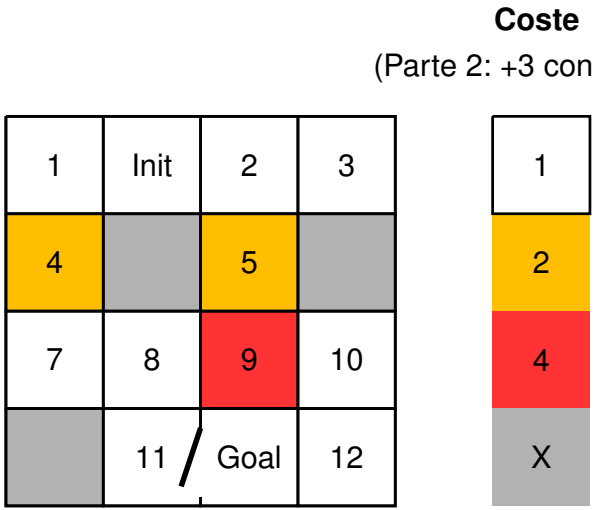
Solución: S-B-A-C-D-G2, Longitud: 5, Coste: 19

Grafo con costes y heurística, comparación

- Hemos escogido H_1 y H_2 usando el grafo y calculando mentalmente. Esto no podemos hacerlo de manera general (además requiere una búsqueda).
- Escalada no llega a la solución óptima (en coste), pero expande pocos nodos dado el criterio que hemos usado en H_1 y H_2.
- A* con H_1 encuentra el óptimo, pero para ello expande E y F, porque la heurística de dichos estados subestima mucho el coste real.
- A* con H_2 no expande todo el subárbol que habría a partir de E. Al añadir más información el algoritmo se hace más eficiente.

Ejercicio 2: Laberinto con costes

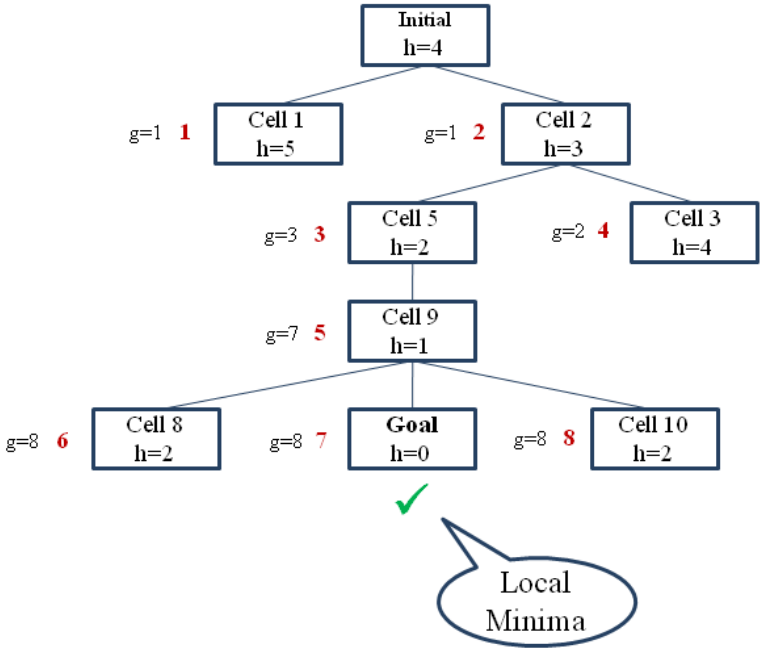
- En el laberinto de la figura, el objetivo es encontrar un camino con coste mínimo desde la casilla “Init” hasta la casilla “Goal”. Se indica el coste de entrar en cada casilla (en las casillas grises no se puede entrar).
- Los operadores, en orden, son: arriba, izquierda, abajo, derecha
- Utilizando como función heurística la distancia de Manhattan , aplicar búsqueda en escalada y A*. Comparar los resultados.
- Asuma que la acción que lleva desde la casilla 11 y Goal tiene un coste extra (quizás abrir una puerta) de 3. Indique cómo cambiaría la ejecución de A*.



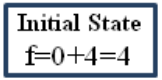
Coste

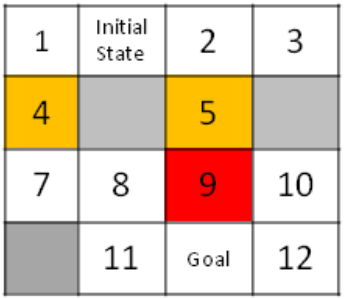
(Parte 2: +3 con puerta)

Laberinto con costes, escalada

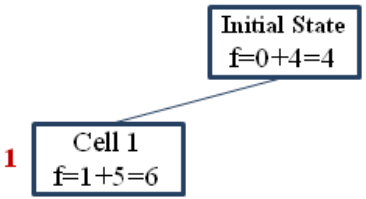


Laberinto con costes, A*





Laberinto con costes, A*



1	Initial State	2	3
4		5	
7	8	9	10
	11	Goal	12

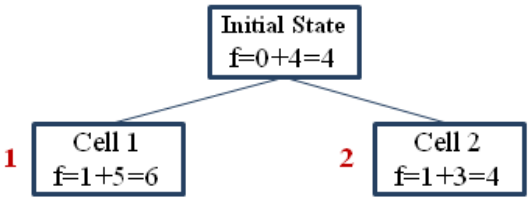
Wall

Cost 1

Cost 4

Cost 2

Laberinto con costes, A*



1	Initial State	2	3
4		5	
7	8	9	10
	11	Goal	12

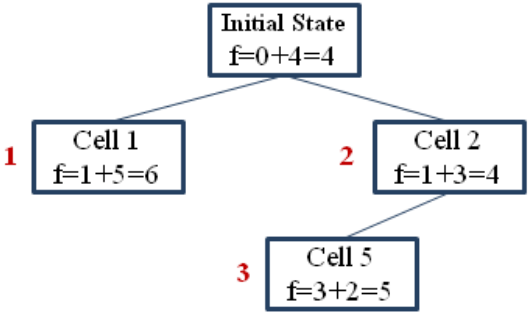
Wall

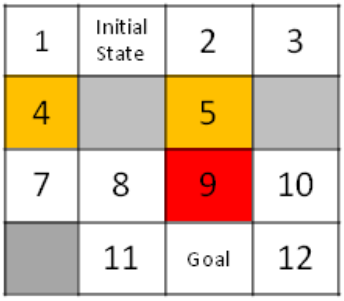
Cost 1

Cost 4

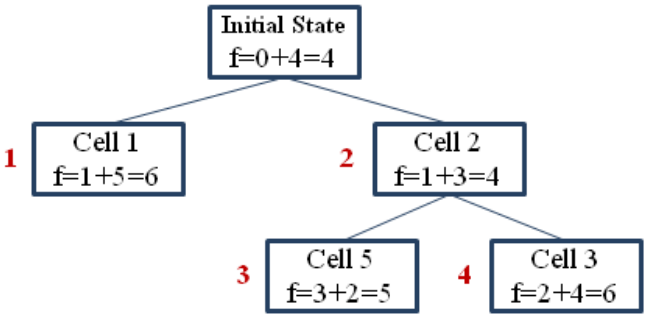
Cost 2

Laberinto con costes, A*

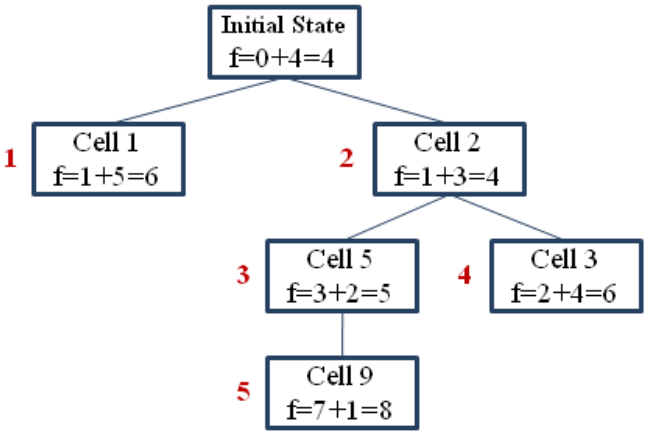




Laberinto con costes, A*

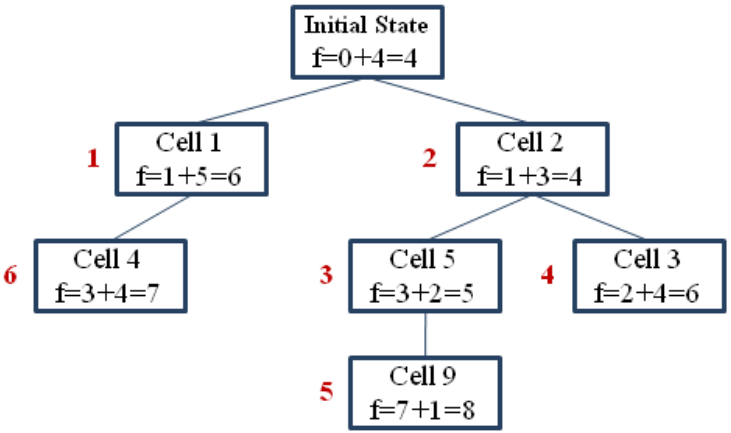


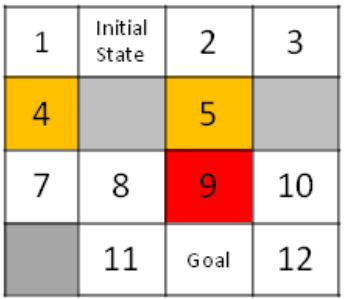
Laberinto con costes, A*



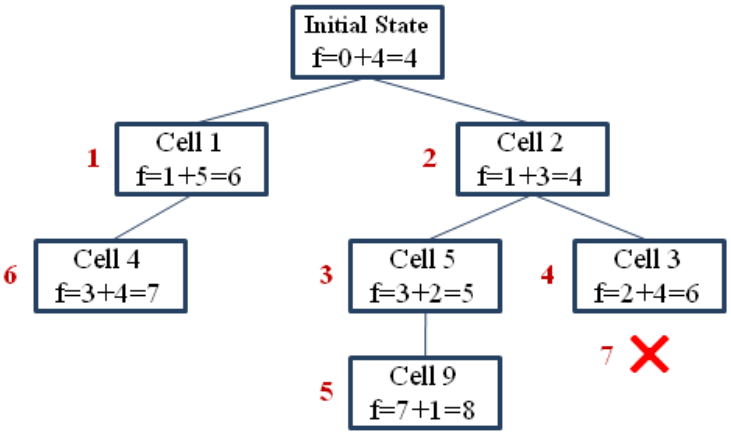
Laberinto con costes, A*

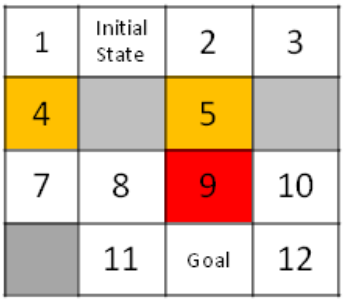
Solucionando empates en *f* tomando el *g* más pequeño. Si elegimos solucionarlos utilizando el *h* más pequeño, la celda Cell3 se expandiría antes que Cell1



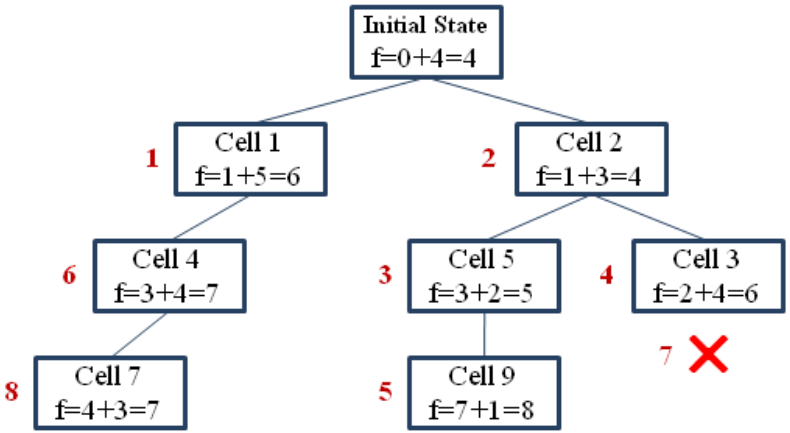


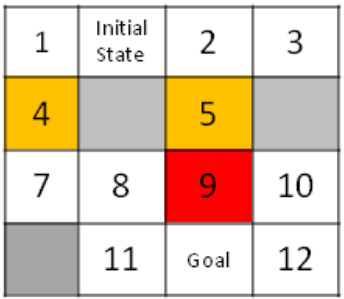
Laberinto con costes, A*



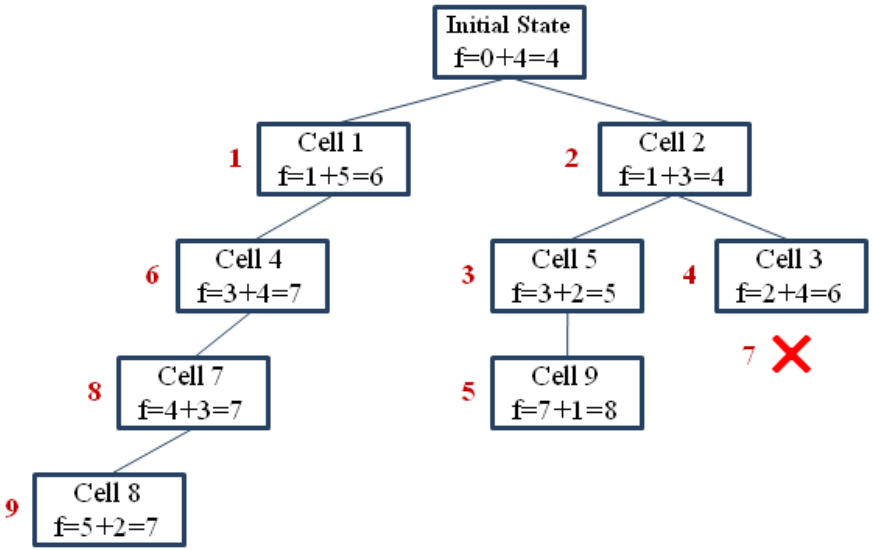


Laberinto con costes, A*

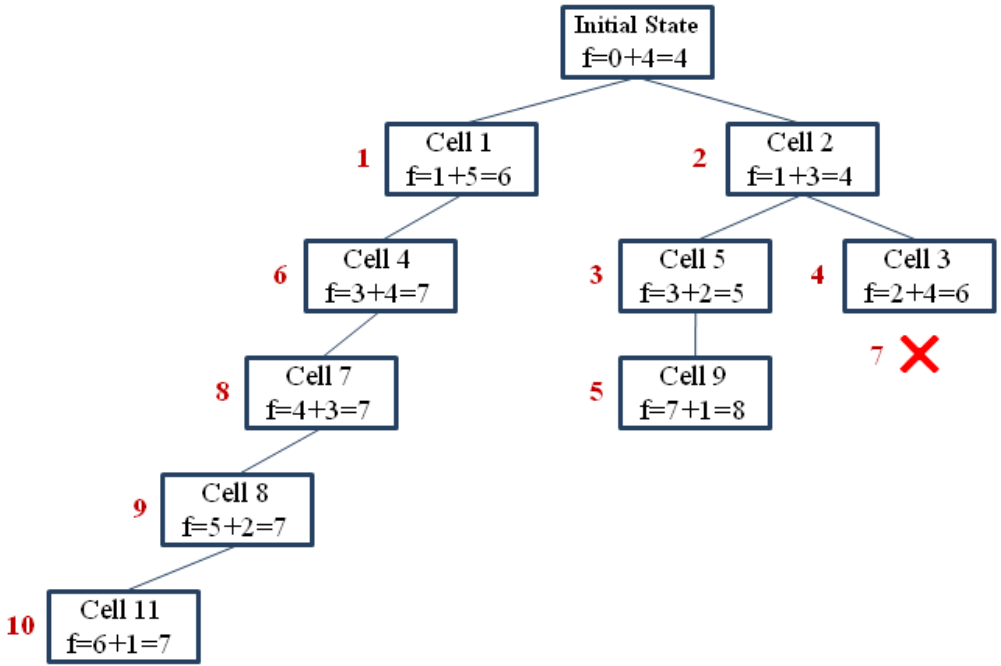




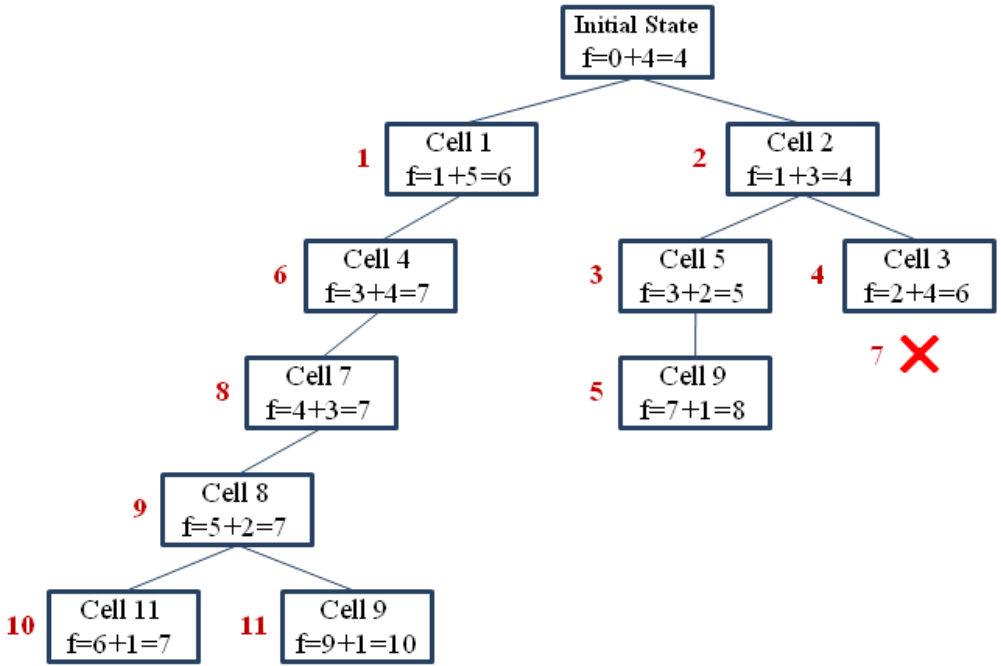
Laberinto con costes, A*



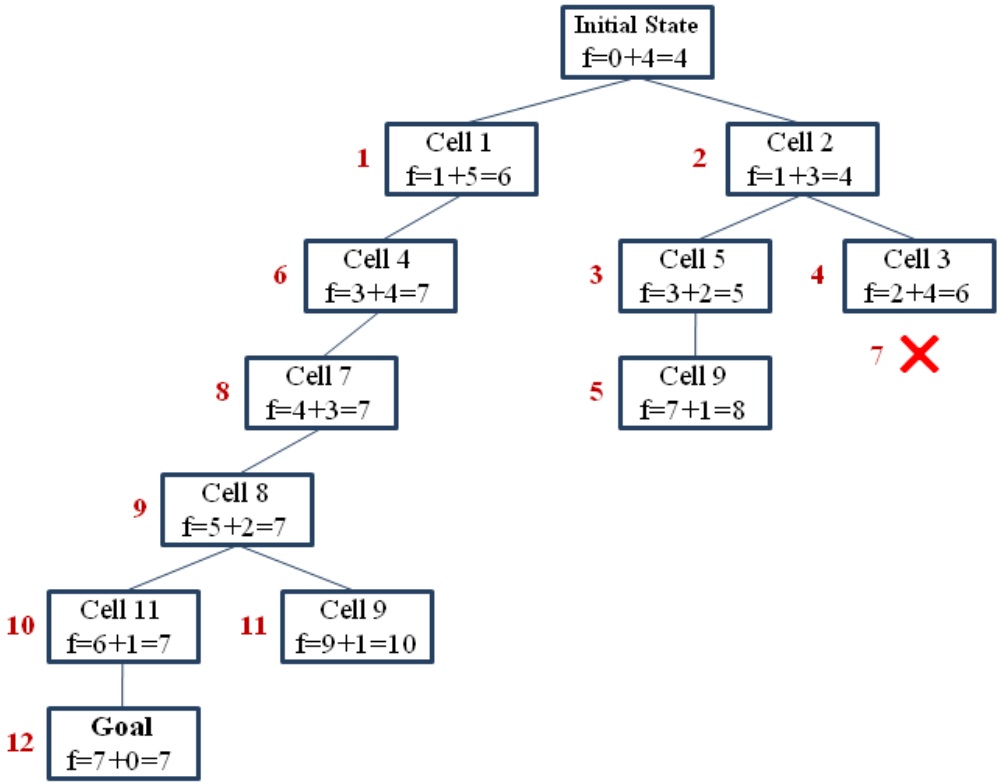
Laberinto con costes, A*



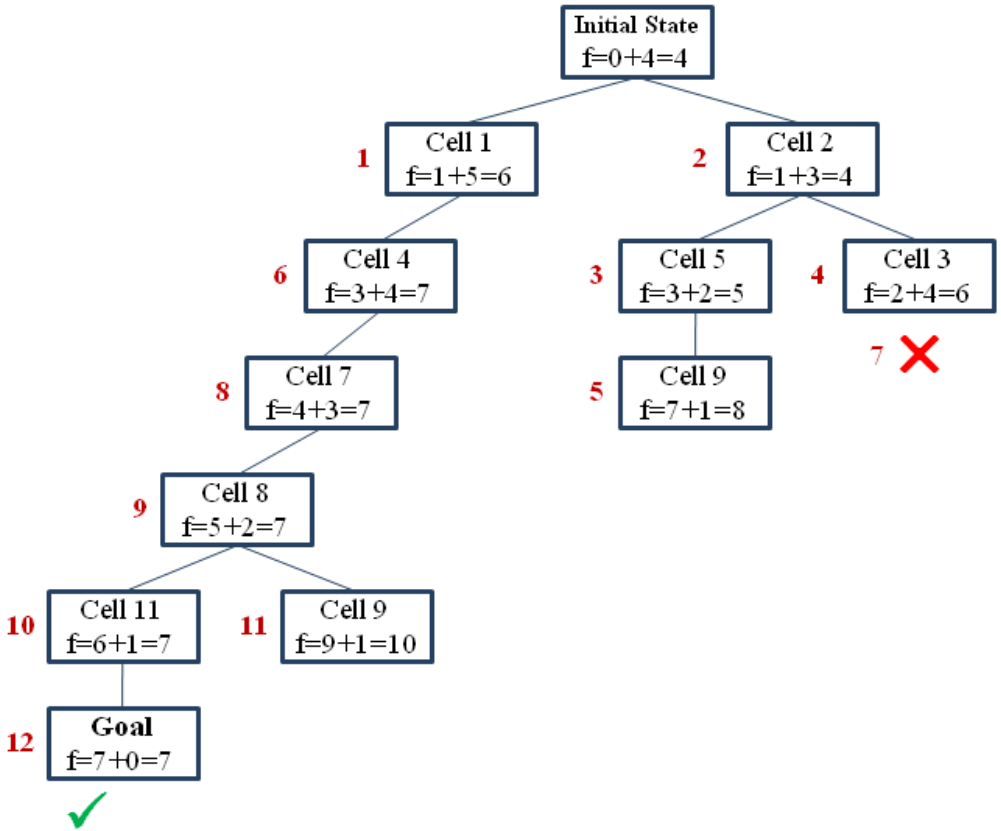
Laberinto con costes, A*



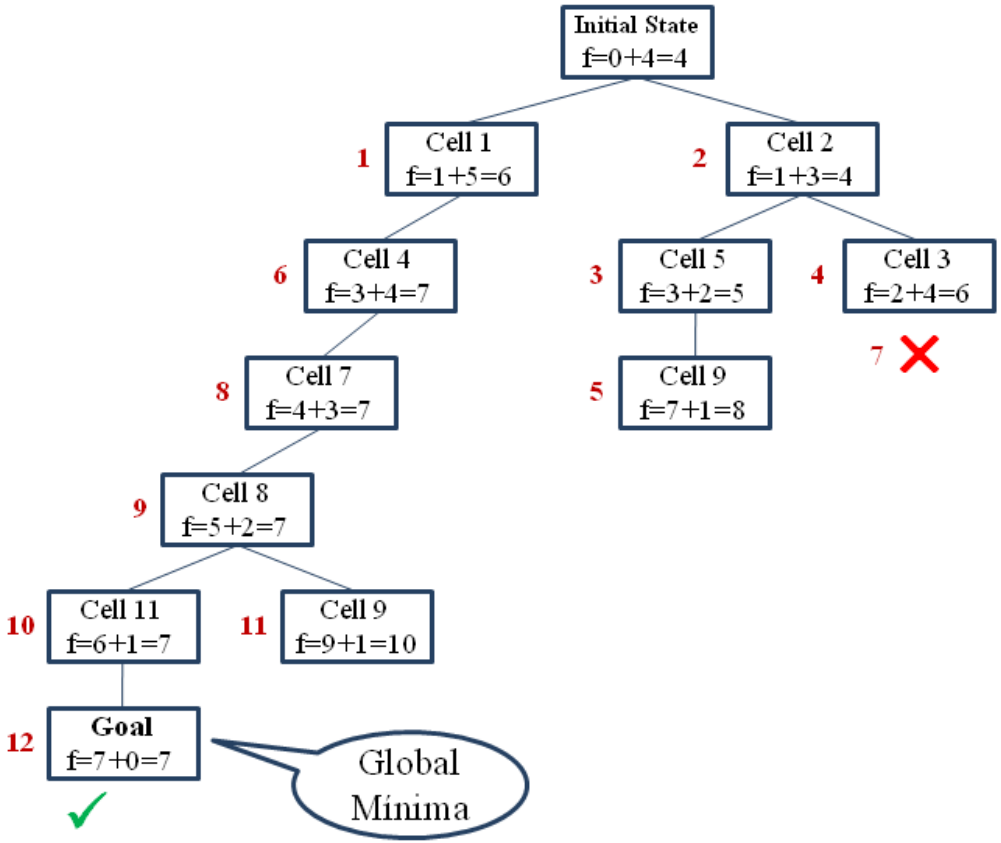
Laberinto con costes, A*



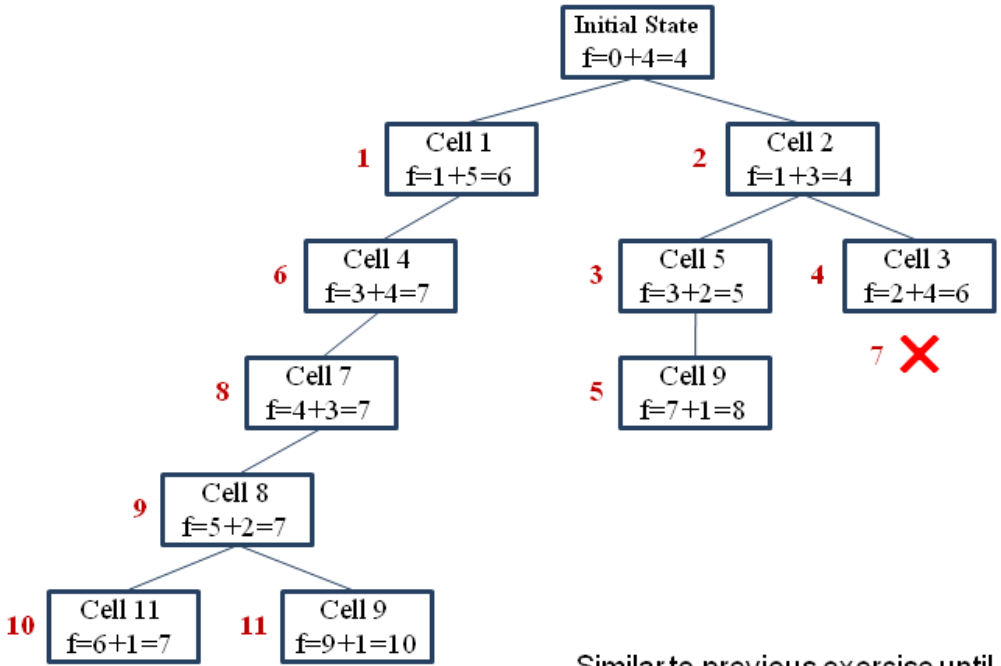
Laberinto con costes, A*



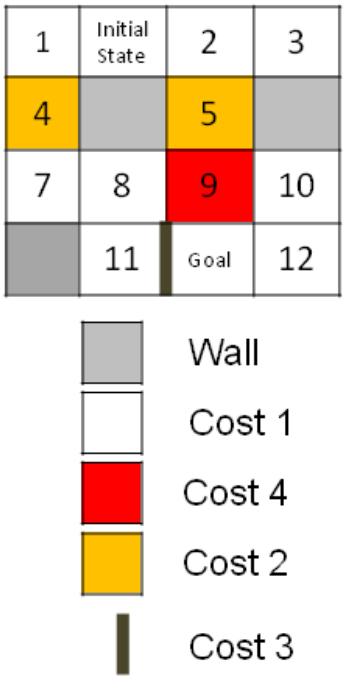
Laberinto con costes, A*



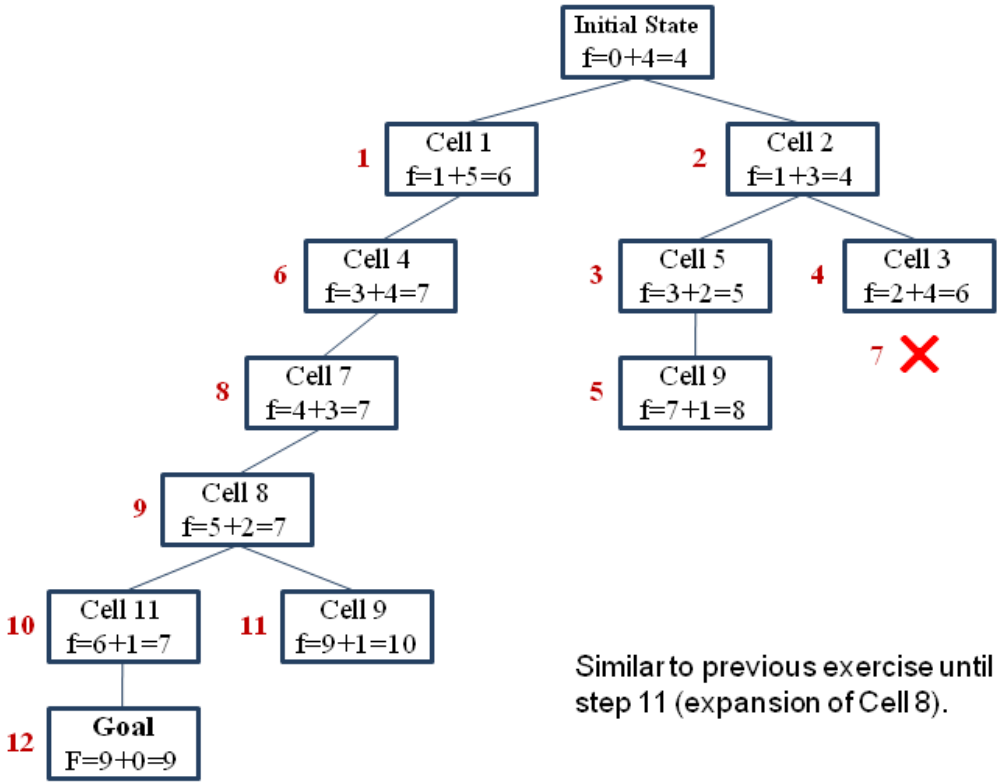
Laberinto con costes y puerta,A*



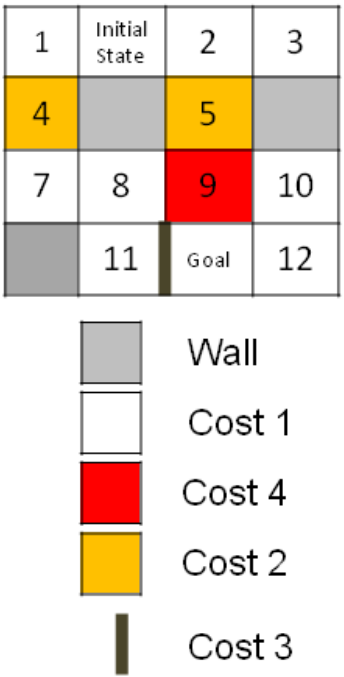
Similar to previous exercise until step 11 (expansion of Cell 8).



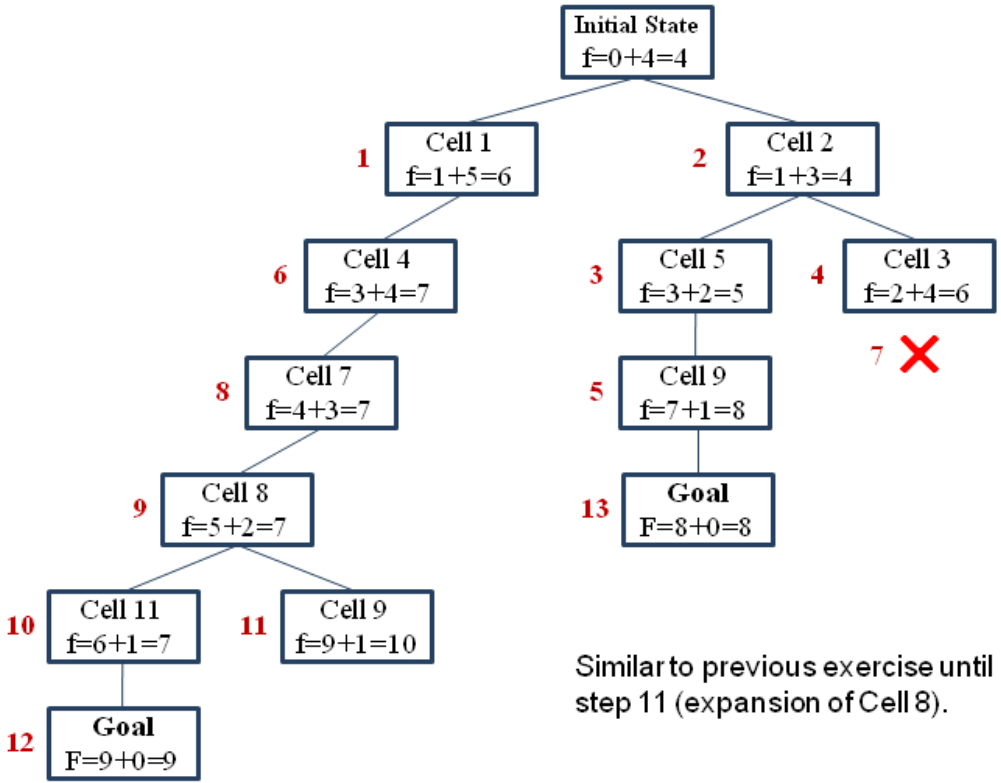
Laberinto con costes y puerta,A*



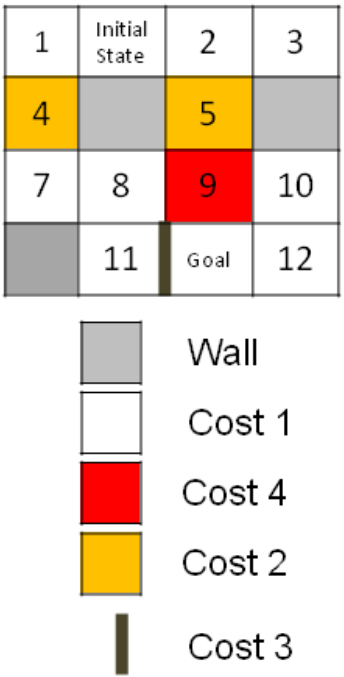
Similar to previous exercise until step 11 (expansion of Cell 8).



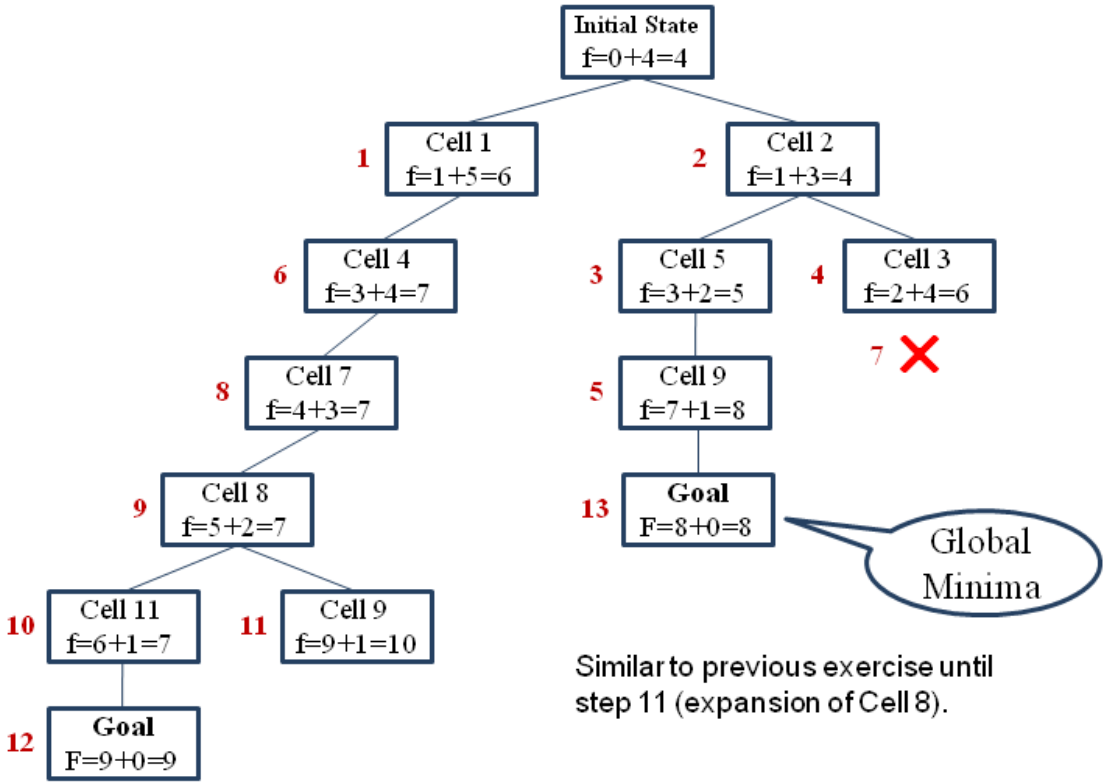
Laberinto con costes y puerta,A*



Similar to previous exercise until step 11 (expansion of Cell 8).



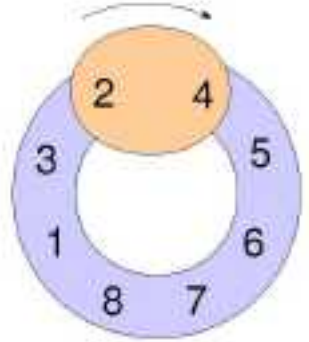
Laberinto con costes y puerta,A*



Ejercicio 3: TopSpin

- Juego de permutaciones con base giratoria que permite intercambiar dos números
- Estado meta: números ordenados con el par (1,2) sobre la base giratoria
- Operadores (en orden de aplicación):
  1. Mover números a la derecha
  2. Mover números a la izquiera
  3. Rotar base giratoria
- **Se pide**:
  1. Representar el problema y los operadores
  2. Simular la búsqueda en profundidad con profundidad máxima 3, con profundidad máxima 4, y amplitud
  3. Definir una heurística para este juego
  4. Aplicar búsqueda en escalada (hill-climbing), utilizando la heurística definida

TOPSPIN: Situación inicial del problema



TopSpin, representación

- Representación de los **estados**:
  - Tenemos que ser capaces de distinguir el orden de las cifras
  - Una forma adecuada es una lista de números $\{n_1, \cdots n_8\}$, donde $n_i \in \{1, \cdots 8\}$ son variables
  - Hay que indicar qué dos números pueden intercambiarse, consideraremos que son los dos primeros: n_1 y n_2

  Estado inicial (I): 24567813 Estado final : 12345678

TopSpin, representación

- Representación de los **estados**:
  - Tenemos que ser capaces de distinguir el orden de las cifras
  - Una forma adecuada es una lista de números $\{n_1, \cdots n_8\}$, donde $n_i \in \{1, \cdots 8\}$ son variables
  - Hay que indicar qué dos números pueden intercambiarse, consideraremos que son los dos primeros: n_1 y n_2

  Estado inicial (I): 24567813 Estado final : 12345678

- Representación de los **operadores**:
  - Operador D (Derecha): Sin parámetros ni precondiciones:

$$n_1\, n_2\, n_3\, n_4\, n_5\, n_6\, n_7\, n_8 \xrightarrow{D} n_8\, n_1\, n_2\, n_3\, n_4\, n_5\, n_6\, n_7$$

TopSpin, representación

- Representación de los **estados**:
  - Tenemos que ser capaces de distinguir el orden de las cifras
  - Una forma adecuada es una lista de números $\{n_1, \cdots n_8\}$, donde $n_i \in \{1, \cdots 8\}$ son variables
  - Hay que indicar qué dos números pueden intercambiarse, consideraremos que son los dos primeros: n_1 y n_2

  Estado inicial (I): 24567813 Estado final : 12345678

- Representación de los **operadores**:
  - Operador D (Derecha): Sin parámetros ni precondiciones:

$$n_1\, n_2\, n_3\, n_4\, n_5\, n_6\, n_7\, n_8 \xrightarrow{D} n_8\, n_1\, n_2\, n_3\, n_4\, n_5\, n_6\, n_7$$

  - Operador I (Izquierda): Sin parámetros ni precondiciones:

$$n_1\, n_2\, n_3\, n_4\, n_5\, n_6\, n_7\, n_8 \xrightarrow{I} n_2\, n_3\, n_4\, n_5\, n_6\, n_7\, n_8\, n_1$$

TopSpin, representación

- Representación de los **estados**:
  - Tenemos que ser capaces de distinguir el orden de las cifras
  - Una forma adecuada es una lista de números $\{n_1, \cdots n_8\}$, donde $n_i \in \{1, \cdots 8\}$ son variables
  - Hay que indicar qué dos números pueden intercambiarse, consideraremos que son los dos primeros: n_1 y n_2

  Estado inicial (I): 24567813 Estado final : 12345678

- Representación de los **operadores**:
  - Operador D (Derecha): Sin parámetros ni precondiciones:

$$n_1\, n_2\, n_3\, n_4\, n_5\, n_6\, n_7\, n_8 \xrightarrow{D} n_8\, n_1\, n_2\, n_3\, n_4\, n_5\, n_6\, n_7$$

  - Operador I (Izquierda): Sin parámetros ni precondiciones:

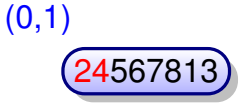
$$n_1\, n_2\, n_3\, n_4\, n_5\, n_6\, n_7\, n_8 \xrightarrow{I} n_2\, n_3\, n_4\, n_5\, n_6\, n_7\, n_8\, n_1$$

  - Operador R (Rotar): Sin parámetros ni precondiciones:

$$n_1\, n_2\, n_3\, n_4\, n_5\, n_6\, n_7\, n_8 \xrightarrow{R} n_2\, n_1\, n_3\, n_4\, n_5\, n_6\, n_7\, n_8$$

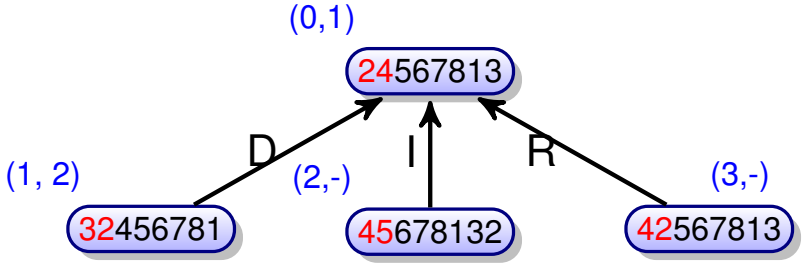
TopSpin, profundidad max=3

Nota: Etiquetamos los nodos con el paso en que se generan seguido del paso en que se expanden (si se hace)



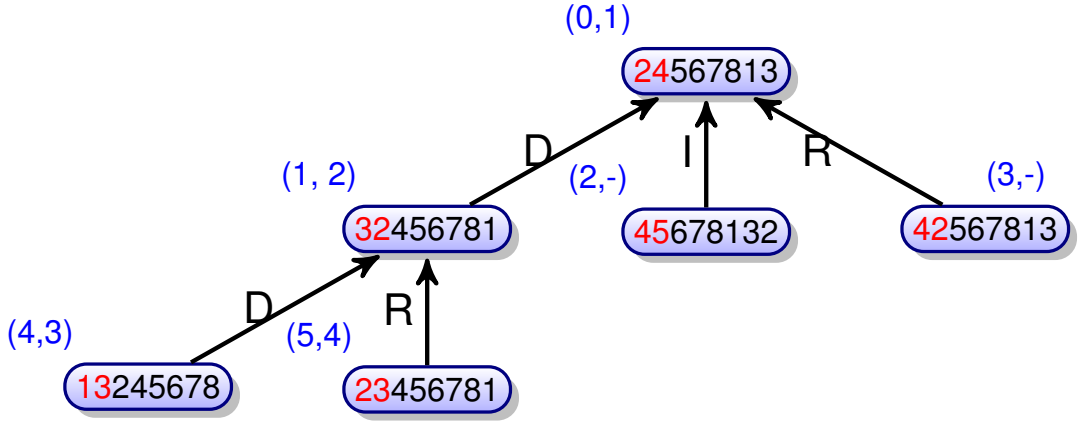
TopSpin, profundidad max=3

Nota: Etiquetamos los nodos con el paso en que se generan seguido del paso en que se expanden (si se hace)



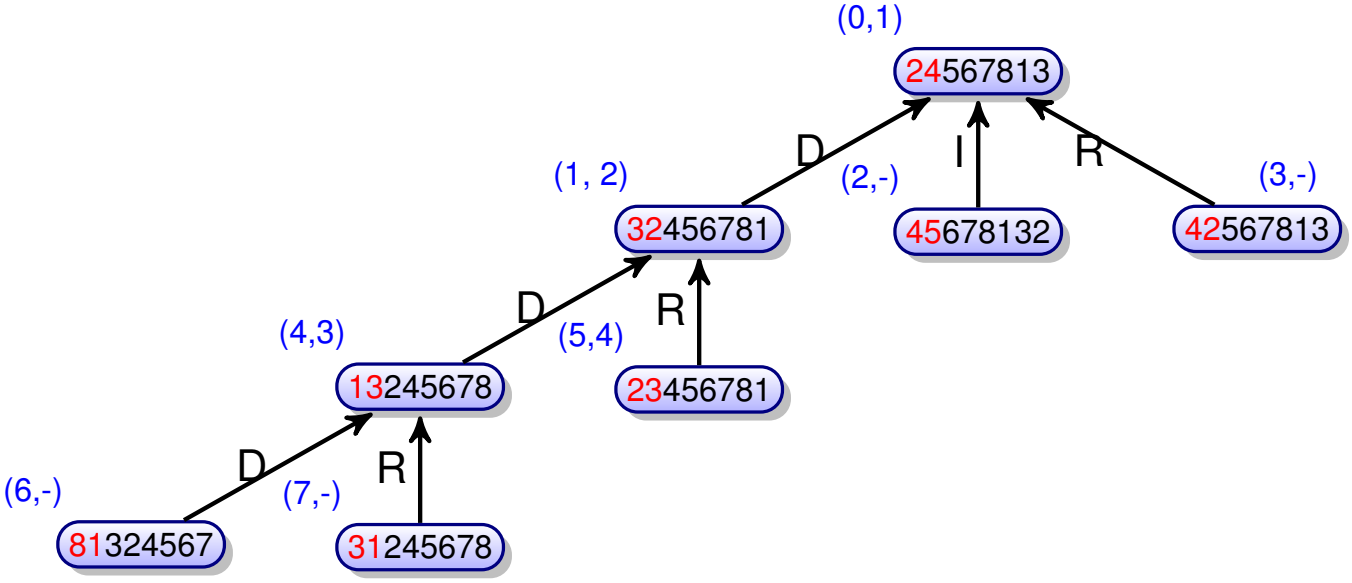
TopSpin, profundidad max=3

Nota: Etiquetamos los nodos con el paso en que se generan seguido del paso en que se expanden (si se hace)



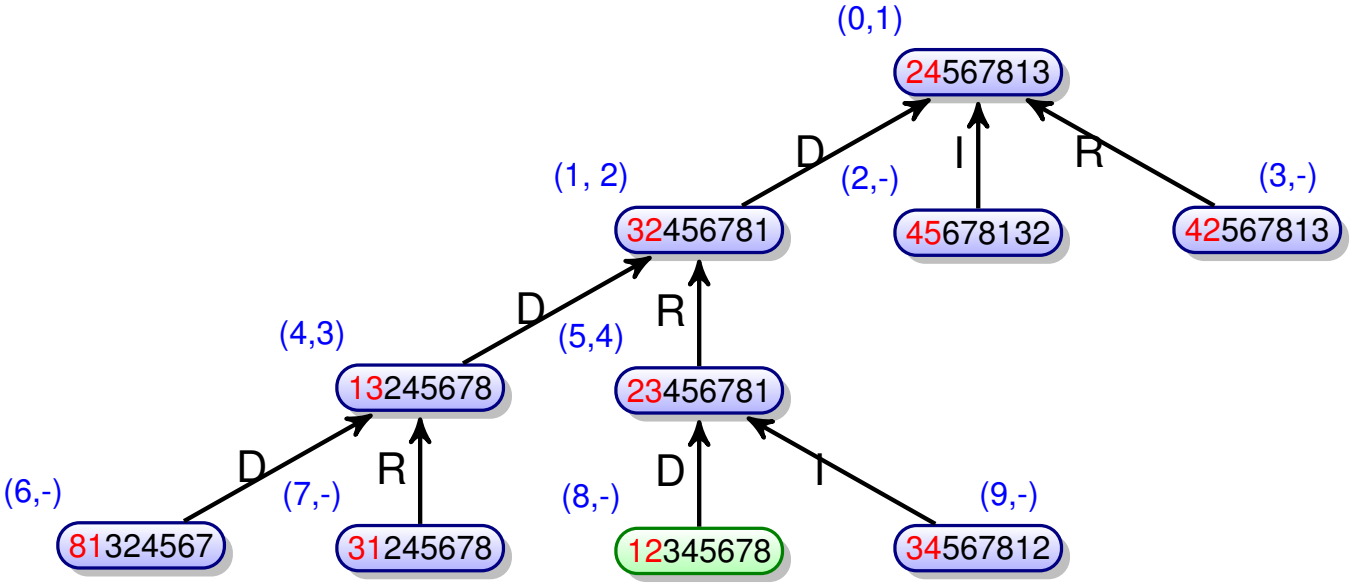
TopSpin, profundidad max=3

Nota: Etiquetamos los nodos con el paso en que se generan seguido del paso en que se expanden (si se hace)



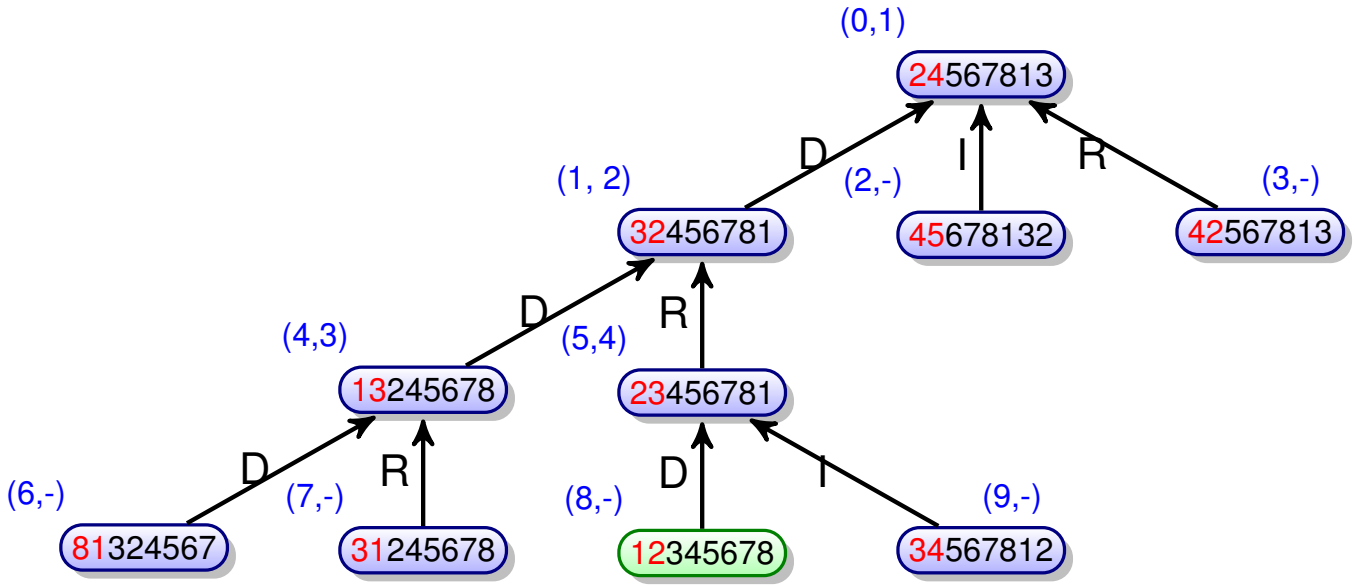
TopSpin, profundidad max=3

Nota: Etiquetamos los nodos con el paso en que se generan seguido del paso en que se expanden (si se hace)



TopSpin, profundidad max=3

Nota: Etiquetamos los nodos con el paso en que se generan seguido del paso en que se expanden (si se hace)



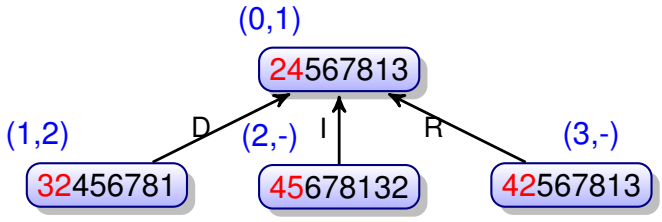
- Longitud solución=3
- Nodos generados=9
- Nodos expandidos=4

TopSpin, profundidad max=4

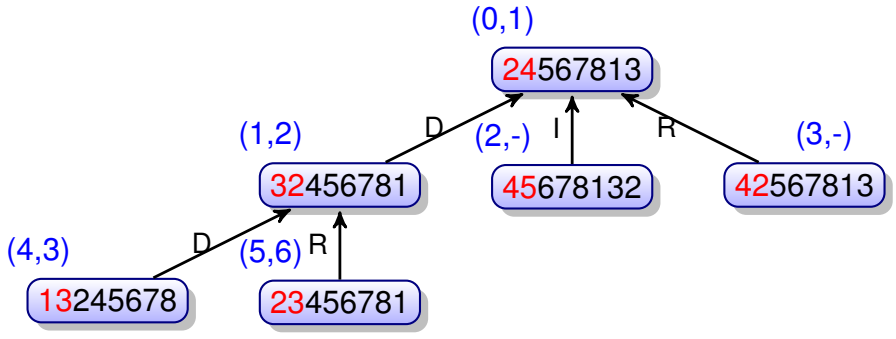
(0,1)

24567813

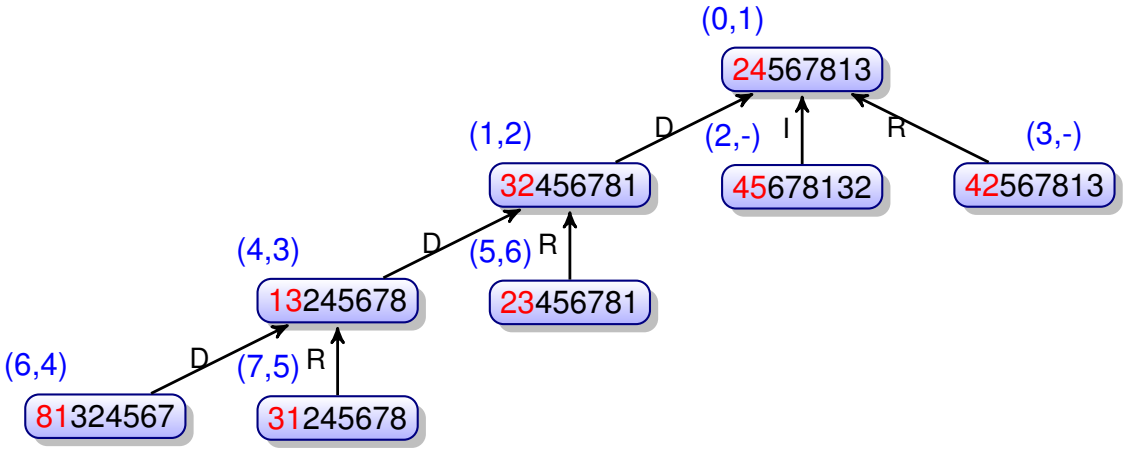
TopSpin, profundidad max=4



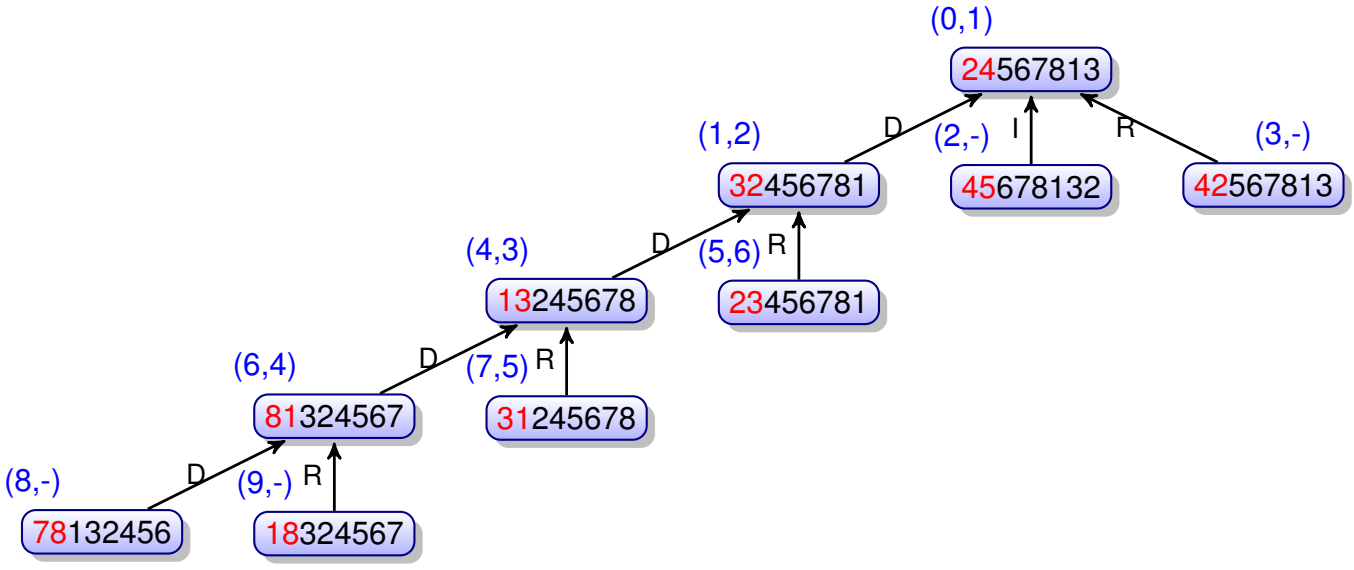
TopSpin, profundidad max=4



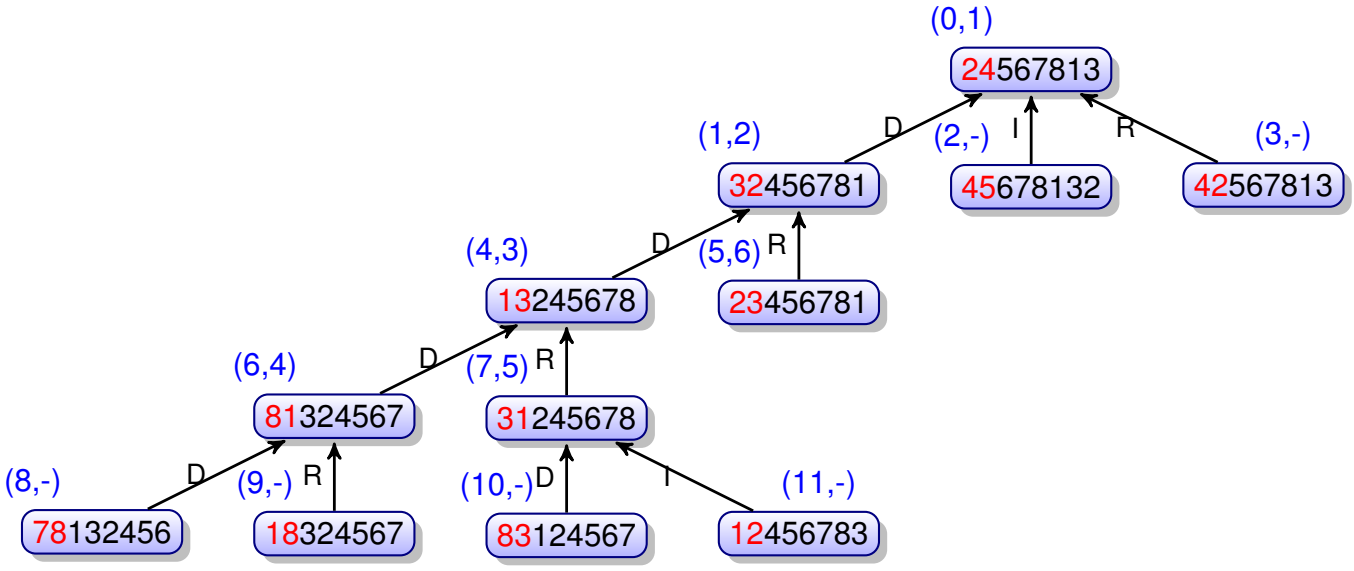
TopSpin, profundidad max=4



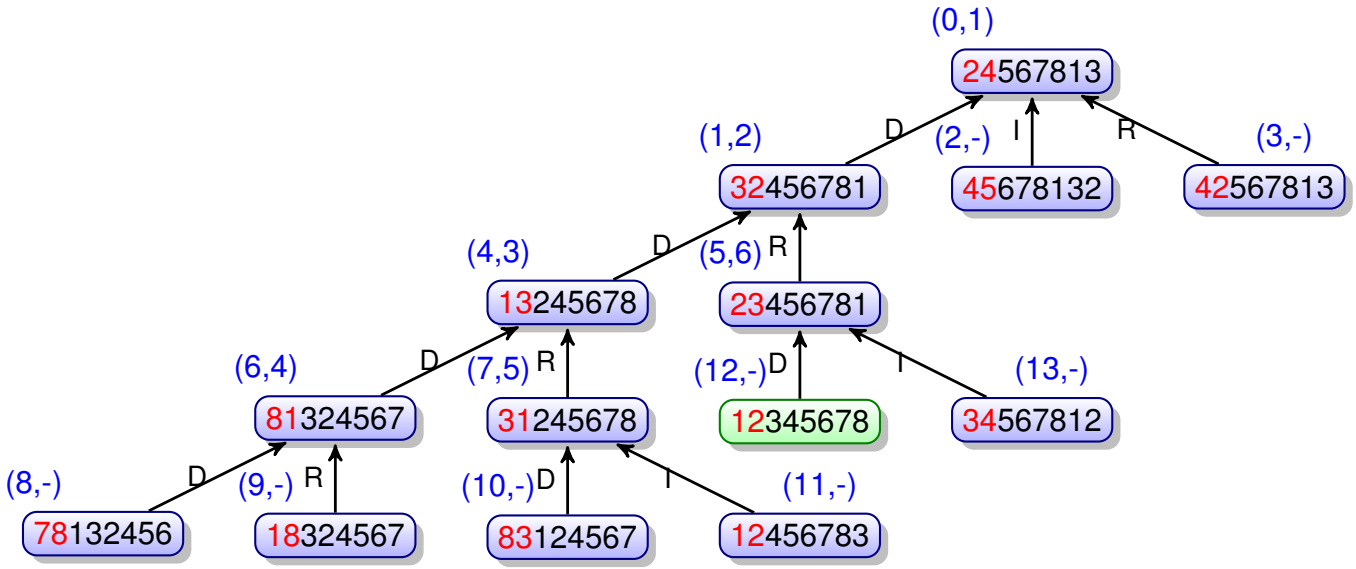
TopSpin, profundidad max=4



TopSpin, profundidad max=4



TopSpin, profundidad max=4

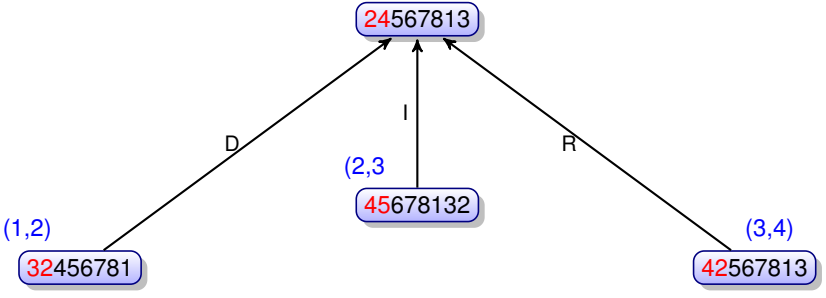


- Longitud solución=3
- Nodos generados=13
- Nodos expandidos=6

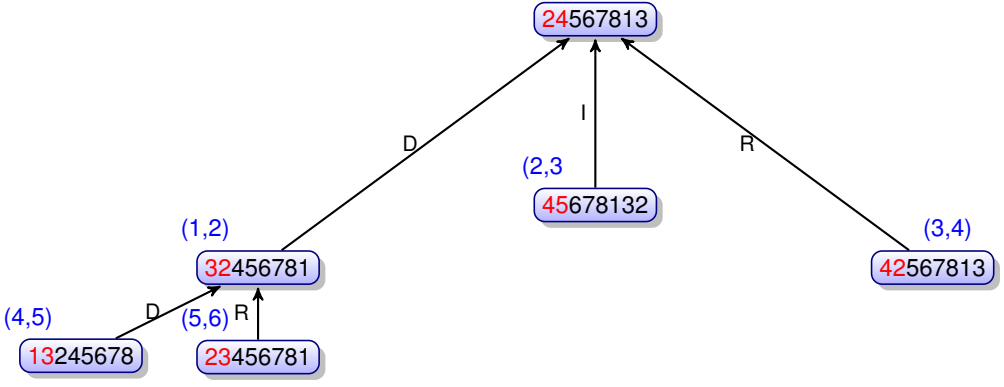
TopSpin, amplitud

24567813

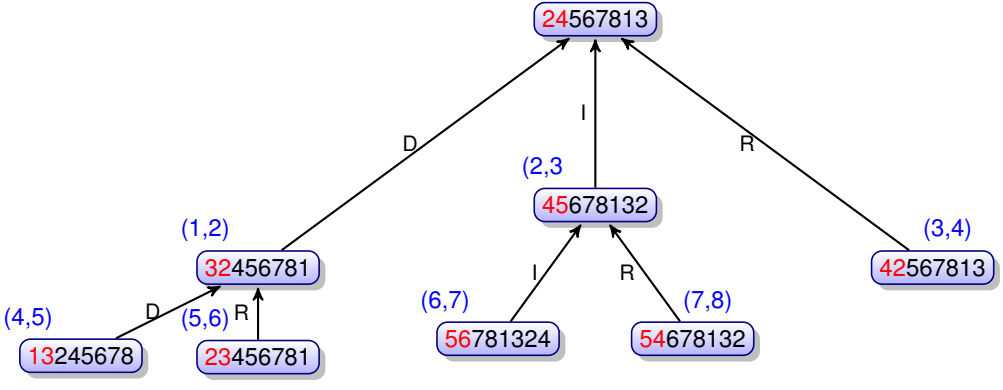
TopSpin, amplitud



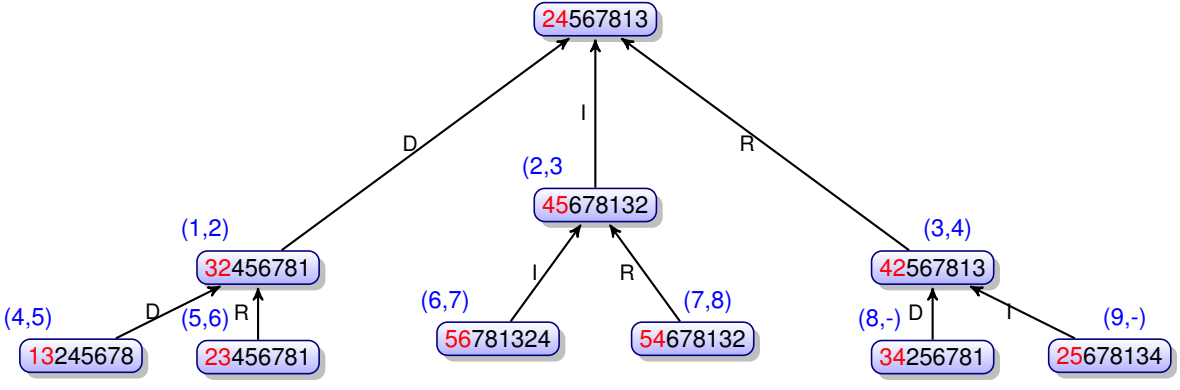
TopSpin, amplitud



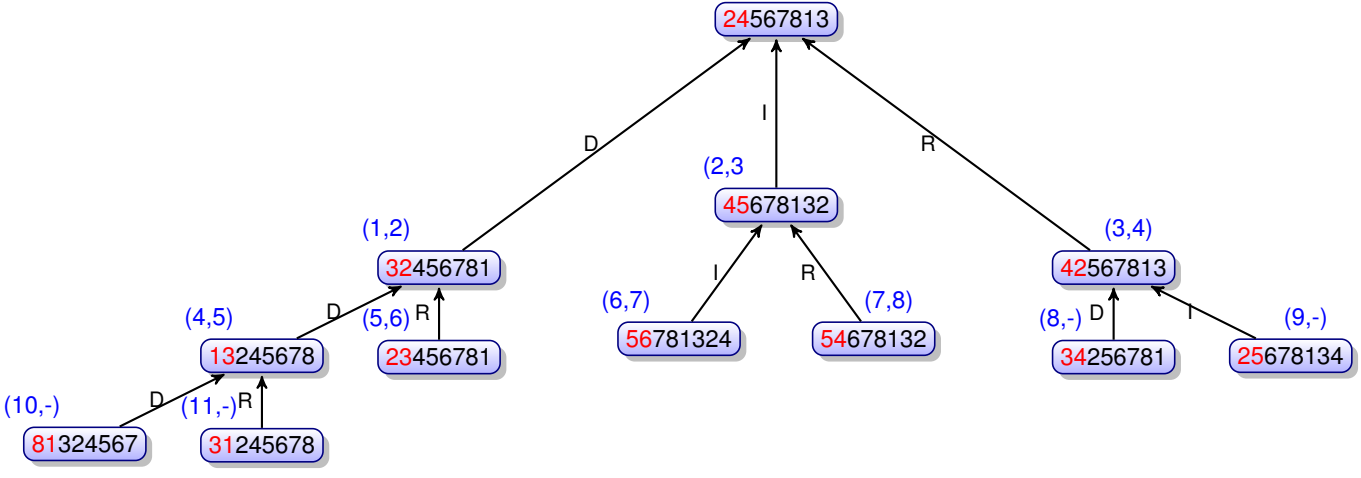
TopSpin, amplitud



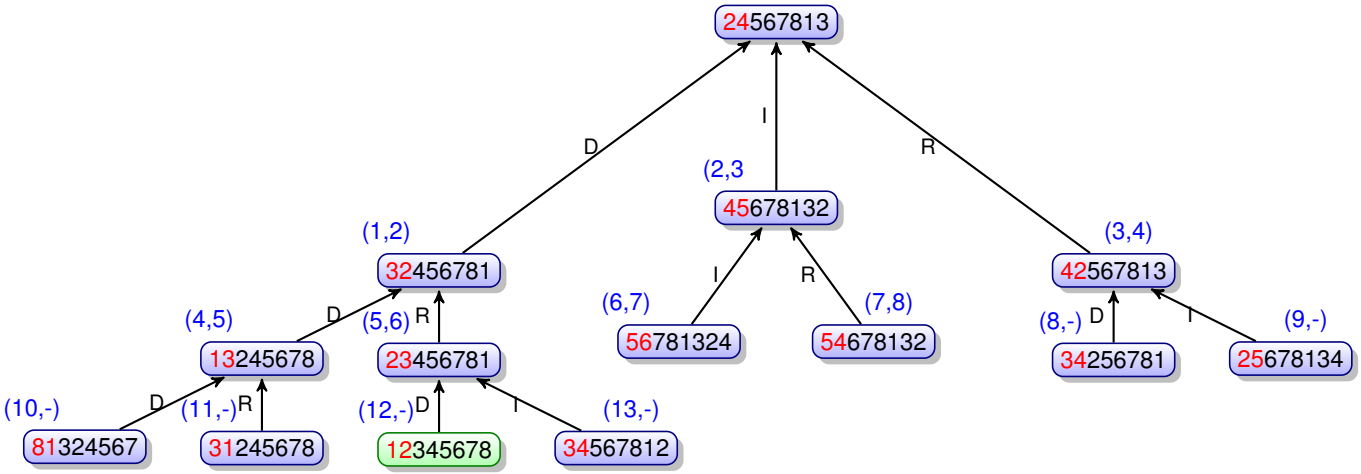
TopSpin, amplitud



TopSpin, amplitud



TopSpin, amplitud



- Longitud solución=3
- Nodos generados=13
- Nodos expandidos=6

TopSpin, definición de heurística

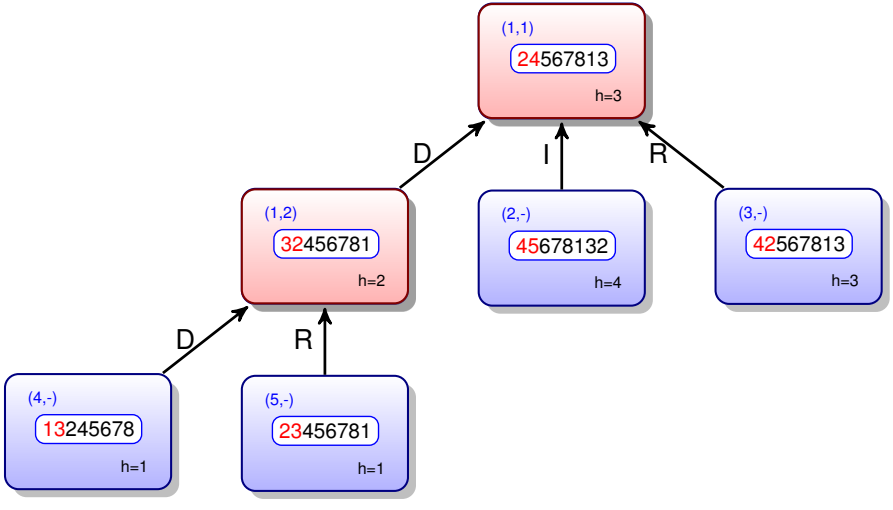
Estado inicial (I): 24567813

- Distancia circular: número mínimo de posiciones para que un número i alcance su posición final, considerando ambas direcciones
  $d_I(1) = 2 \quad d_I(2) = 1 \quad d_I(3) = 3 \quad d_I(4) = 2$
  $d_I(5) = 2 \quad d_I(6) = 2 \quad d_I(7) = 2 \quad d_I(8) = 2$
- 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

$$h(I) = 3$$

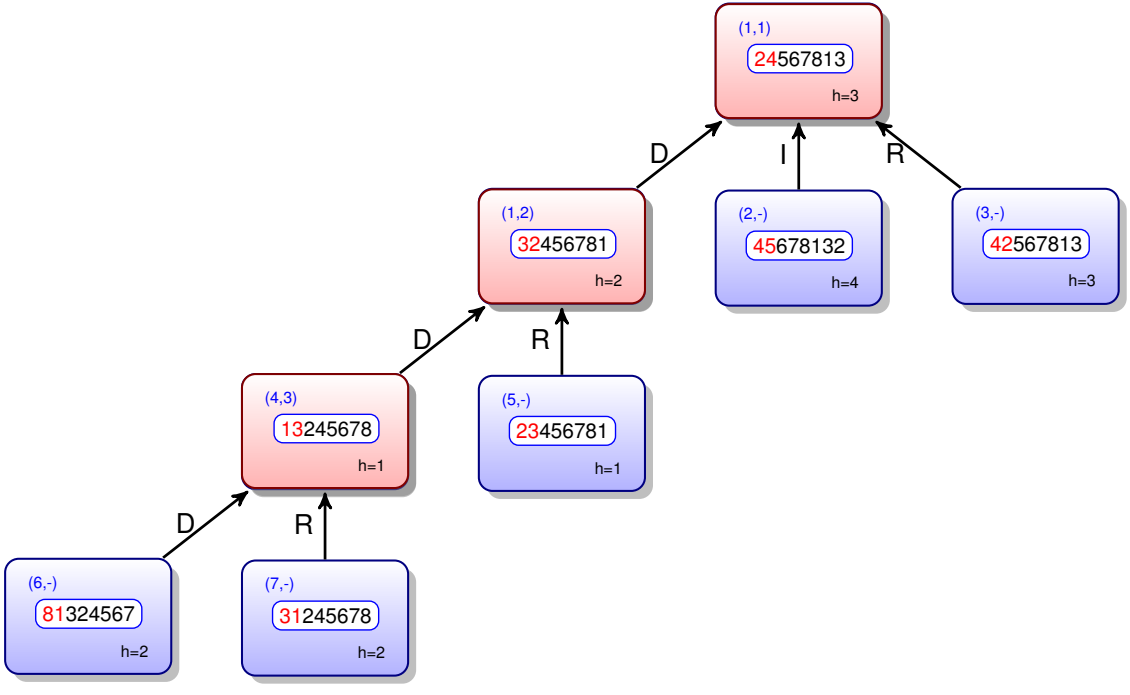
TopSpin: Escalada

Nota: se escoge el mejor sucesor (incluso si este no mejora el nodo actual)



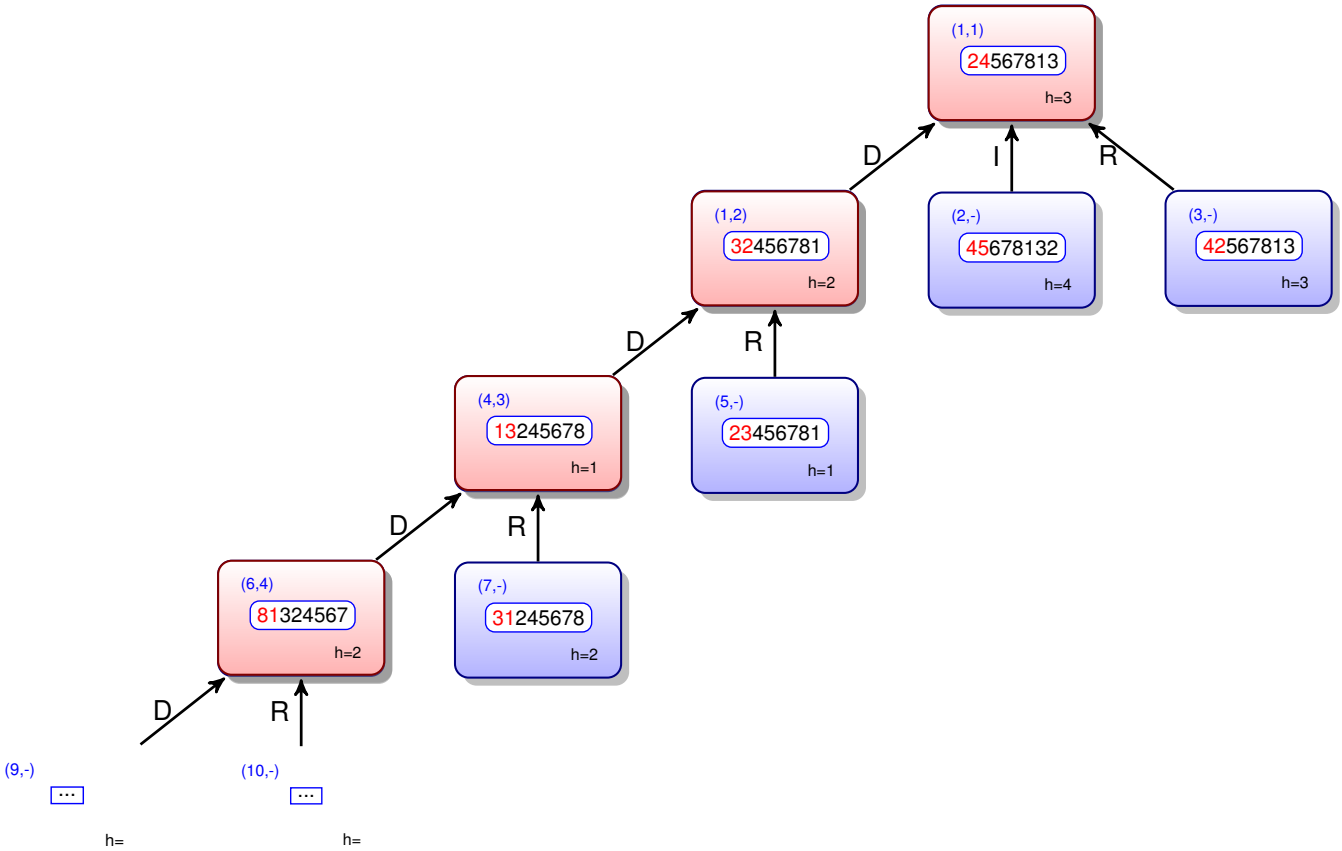
TopSpin: Escalada

Nota: se escoge el mejor sucesor (incluso si este no mejora el nodo actual)



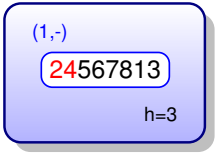
TopSpin: Escalada

Nota: se escoge el mejor sucesor (incluso si este no mejora el nodo actual)



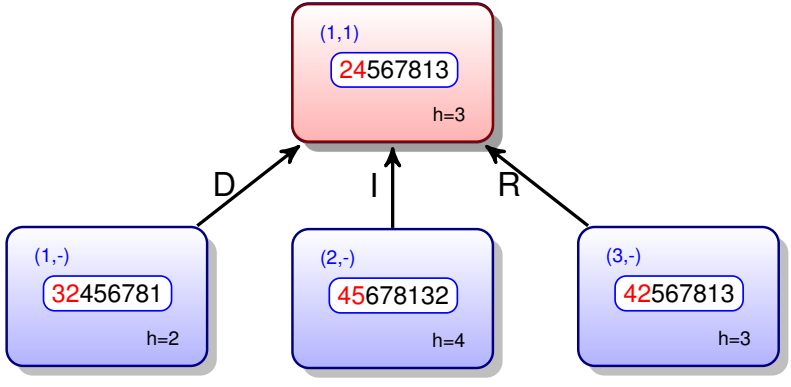
TopSpin, Escalada (variante)

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).



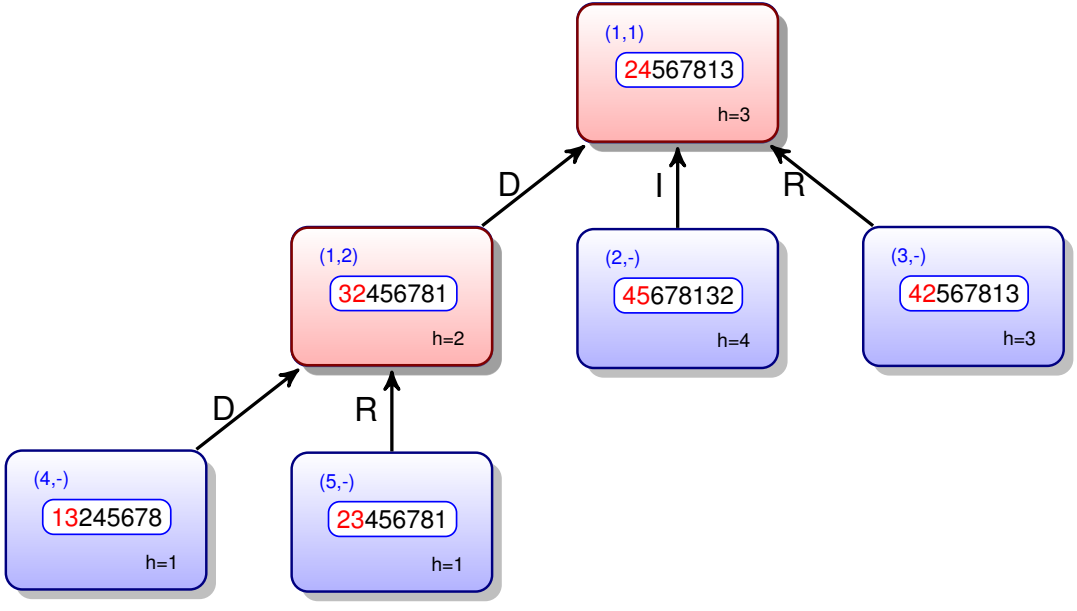
TopSpin, Escalada (variante)

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).



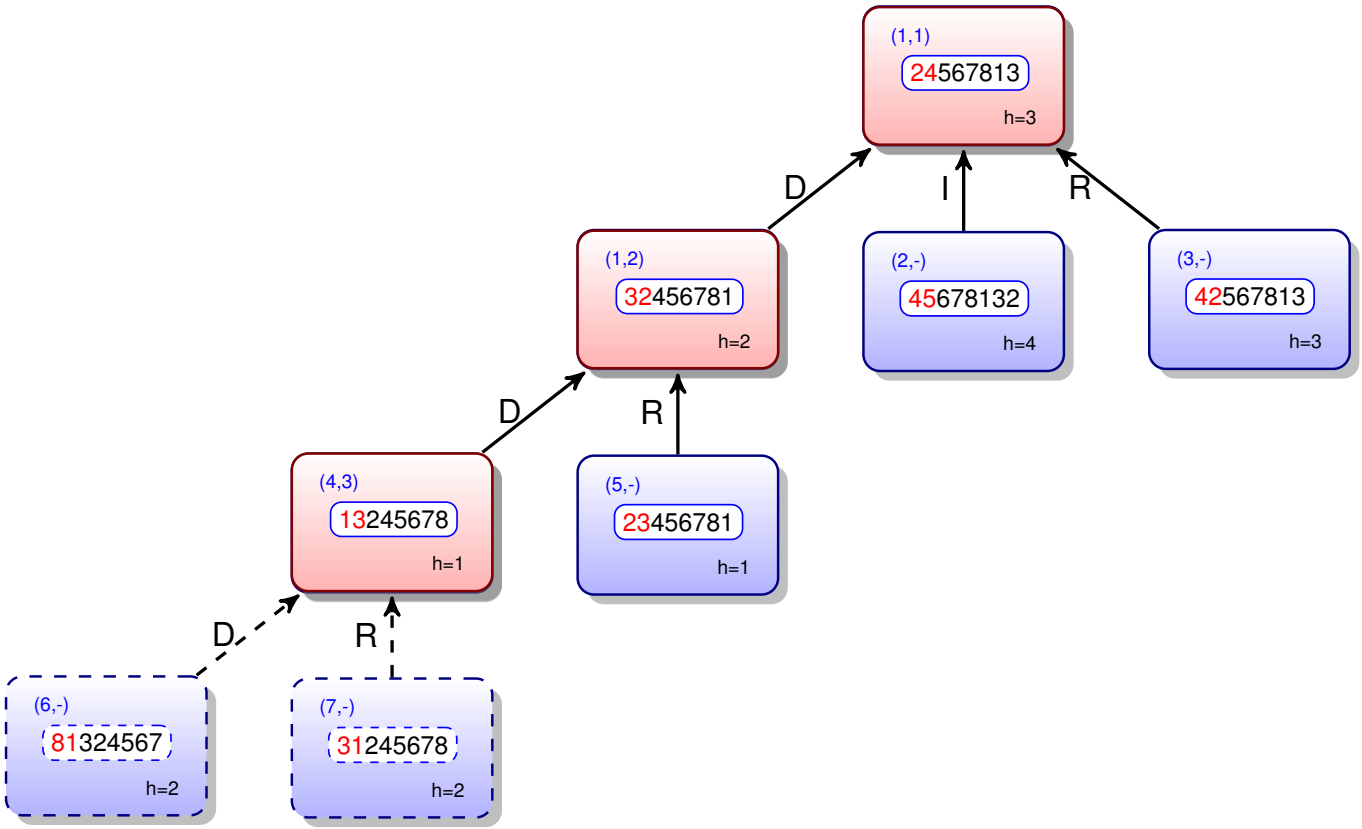
TopSpin, Escalada (variante)

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).



TopSpin, Escalada (variante)

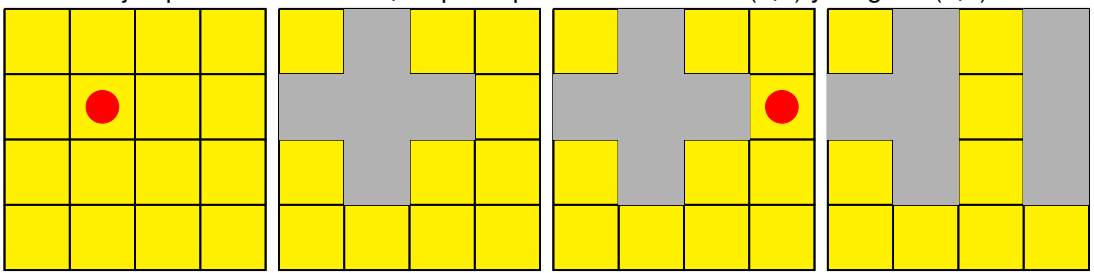
Escalada con restricción: se escoge el mejor sucesor pero solo si mejora el nodo actual. Si no se puede, $\rightarrow$ STOP (no se encuentra la solución).



Ejercicio 4: Apaga la luz

- El juego consiste en un grid de bombillas NxN. Cuando el juego empieza, un número cualquiera de bombillas están encendidas. Presionar una bombilla cambia su estado y el estado de las cuatro bombillas adyacentes.
- La meta es **apagar** todas las bombillas, *preferiblemente presionando el menor número de veces posible*
- **Se pide**:
  1. Caracterice el espacio de estados de este problema y su tamaño
  2. Representar los estados y operadores de este problema
  3. Definir una función de evaluación $f(n) = g(n) + h(n)$ para el algoritmo A*

Ejemplo: todo iluminado, se pulsa primero la bombilla (2,2) y luego la (2,4)



Apaga la luz, representación

- Espacio de estados: matriz NxN con valores binarios para representar encendida (1) o apagada (0). Hay $2^{N \times N}$ estados posibles.
- Podemos tener como estado un conjunto de tuplas (x,y,v), donde v es 0 si la bombilla está apagada o 1 si está encendida.
- Operadores: Press(x,y): cambiar el estado de la bombilla en la celda Cell y las cuatro celdas adyacentes.

Press(x,y):	
(x,y,?v0), (x,y-1,?vN) ,(x+1,y,?vE), (x-1,y,?vW), (x,y+1,?vS) (x,y,(1-?v0)), (x,y-1,(1-?vN)), (x+1,y,(1-?vE)), (x-1,y,(1-?vW)), (x,y+1,(1-?vS))	$\rightarrow$

Nota: Asumimos que se modifican las tuplas de las casillas

- Heurísticas:
  - $h_1(n) \rightarrow$ número de bombillas encendidas. No admisible, porque en una sola acción puede cambiar cinco bombillas.
  - $h_2(n) \rightarrow$ número de bombillas encendidas $/5$ (redondeado hacia arriba).
  - $g(n)$: número de cambios desde el estado inicial
- Función de evaluación: $f(n) = g(n) + h_2(n)$

Ejercicio 5: Asignación de tareas

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.

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

	Tareas			
Trabajador	T1	T2	T3	T4
W1	20	5	3	10
W2	10	3	6	20
W3	15	4	4	15
W4	10	6	8	25

Asignación de tareas

La solución a este ejercicio se deja para el alumno.

Ejercicio 6: Problema del viajante

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.

- Represente el problema para resolverlo mediante búsqueda
- Defina una heurística y argumente si es admisible
- Simule la ejecución de búsqueda en escalada y A*, comenzando desde C1

	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	-

Problema del viajante

La solución a este ejercicio se deja para el alumno.