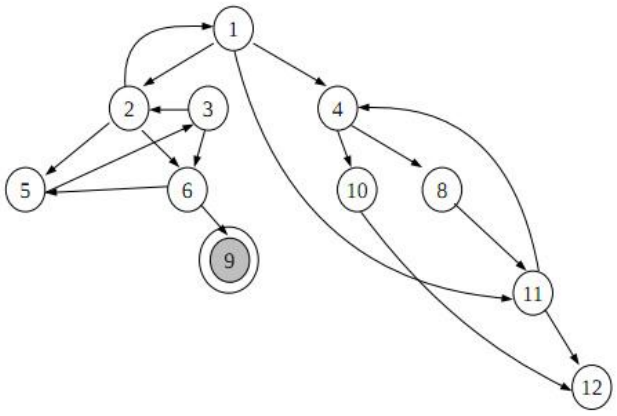
1. Dado el siguiente espacio de estados, aplica la búsqueda en anchura, en profundidad, bidireccional e iterativa (sólo 10 iteraciones para ésta última). Representa, por cada iteración, el estado de las pilas y del grafo en memoria (aplica el borrado de nodos que utilice cada técnica).



BÚSQUEDA EN AMPLITUD (nodo, mejor_padre)

Iteración 0:

Frontera: {(1, -)}

Explorados: { }

Iteración 1:

Frontera: {(2,1), (4,1), (11,1)}

Explorados: {(1, -)}

Iteración 2:

Frontera: {(4,1), (11,1), (5,2), (6,2)}

Explorados: {(1, -), (2,1)}

Iteración 3:

Frontera: {(11,1), (5,2), (6,2), (10,4), (8,4)}

Explorados: {(1, -), (2,1), (4,1)}

Iteración 4:

Frontera: {(5,2), (6,2), (10,4), (8,4), (12,11)}

Explorados: {(1, -), (2,1), (4,1), (11,1)}

Iteración 5:

Frontera: {(6,2), (10,4), (8,4), (12,11), (3,5)}

Explorados: {(1, -), (2,1), (4,1), (11,1), (5,2)}

Iteración 6:

Frontera: {(10,4), (8,4), (12,11), (3,5), (9,6)}

Explorados: {(1, -), (2,1), (4,1), (11,1), (5,2), (6,2)}

Iteración 7:

Frontera: {(8,4), (12,11), (3,5), (9,6)}

Explorados: {(1, -), (2,1), (4,1), (11,1), (5,2), (6,2), (10,4)}

Iteración 8:

Frontera: {(12,11), (3,5), (9,6)}

Explorados: {(1, -), (2,1), (4,1), (11,1), (5,2), (6,2), (10,4), (8,4)}

Iteración 9:

Frontera: {(3,5), (9,6)}

Explorados: {(1, -), (2,1), (4,1), (11,1), (5,2), (6,2), (10,4), (8,4), (12,11)}

Iteración 10:

Frontera: {(9,6)}

Explorados: {(1, -),(2,1),(4,1), (11,1),(5,2),(6,2), (10,4), (8,4),(12,11), (3,5)}

Iteración 11:

Frontera: { }

Explorados: {(1, -), (2,1), (4,1), (11,1), (5,2), (6,2), (10,4), (8,4), (12,11), (3,5), **(9,6)**}

Camino: 1-2-6-9

BÚSQUEDA EN PROFUNDIDAD (nodo, mejor_padre)

Iteración 0:

Frontera: {(1, -)}

Explorados: { }

Iteración 1:

Frontera: {(2,1), (4,1), (11,1)}

Explorados: {(1 -)}

Iteración 2:

Frontera: {(5,2), (6,2), (4,1), (11,1)}

Explorados: {(1, -), (2,1)}

Iteración 3:

Frontera: {(3,5), (6,2), (4,1), (11,1)}

Explorados: {(1, -), (2,1), (5,2)}

Iteración 4:

Frontera: {(6,2), (4,1), (11,1)}

Explorados: {(1, -), (2,1), (5,2), (3,5)}

Iteración 5:

Frontera: {(9,6), (4,1), (11,1)}

Explorados: {(1, -), (2,1), (5,2), (3,5), (6,2)}

Iteración 6:

Frontera: {(4,1), (11,1)}

Explorados: {(1, -), (2,1), (5,2), (3,5), (6,2), **(9,6)**}

Camino: 1-2-6-9

BÚSQUEDA BIDIRECCIONAL usando anchura en los dos sentidos

Desde el nodo inicial:

Iteración 0:

Frontera: {(1, -)}

Explorados: { }

Iteración 1:

Frontera: {(2,1), (4,1), (11,1)}

Explorados: {(1, -)}

Iteración 2:

Frontera: {(4,1), (11,1), (5,2), (6,2)}

Explorados: {(1, -), (2,1)}

Iteración 3:

Frontera: {(11,1), (5,2), (6,2), (10,4), (8,4)}

Explorados: {(1, -), **(2,1)**, (4,1)}

Desde el nodo final:

Iteración 0:

Frontera: {(- ,9)}

Explorados: { }

Iteración 1:

Frontera: {(9, 6)}

Explorados: {(- ,9)}

Iteración 2:

Frontera: {(6,2), (6,3)}

Explorados: {(- ,9), (9, 6)}

Iteración 3:

Frontera: {(6,3), (2,1)}

Explorados: {(- ,9), (9, 6), **(6,2)**}

Camino: 1-2-6-9

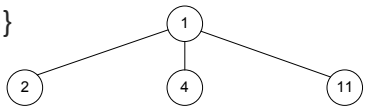
BÚSQUEDA ITERATIVA

Iteración 0: profundidad límite = 2

Frontera: {(1,-)}
Explorados: { }

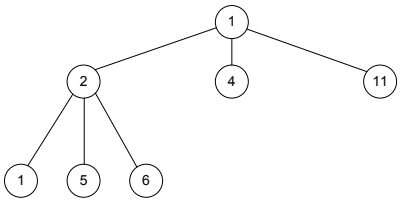
Iteración 1:

Frontera: {(2,1), (4,1), (11,1)}
Explorados: {(1,-)}



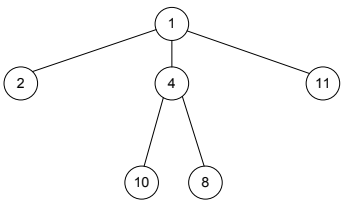
Iteración 2:

Frontera: {(1,-), (5,2), (6,2), (2,1), (4,1), (11,1)}
Explorados: {(1,-), (2,1)}



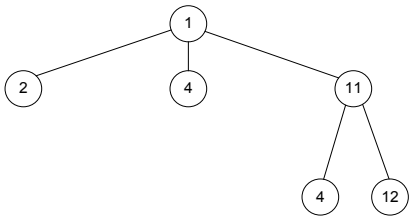
Iteración 3:

Frontera: {(10,4), (8,4), (4,1), (11,1)}
Explorados: {(1,-), (2,1), (4,1)}



Iteración 4:

Frontera: {(4,1), (12,11)}
Explorados: {(1,-), (2,1), (4,1), (11,1)}

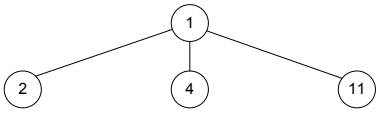


Iteración 5: profundidad límite = 3

Frontera: {(1,-)}
Explorados: { }

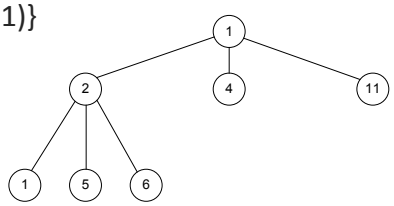
Iteración 6:

Frontera: {(2,1), (4,1), (11,1)}
Explorados: {(1,-)}



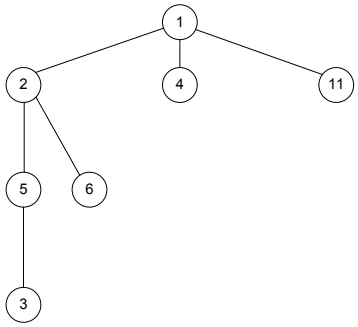
Iteración 7:

Frontera: {(1,-), (5,2), (6,2), (4,1), (11,1)}
Explorados: {(1,-), (2,1)}



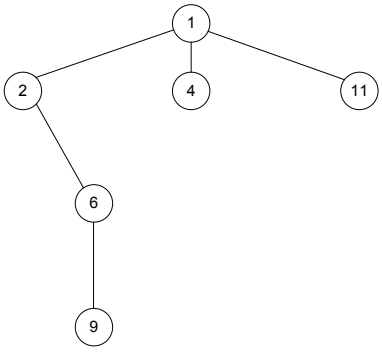
Iteración 8:

Frontera: {(3,5), (6,2), (4,1), (11,1)}
Explorados: {(1,-), (2,1), (5,2)}



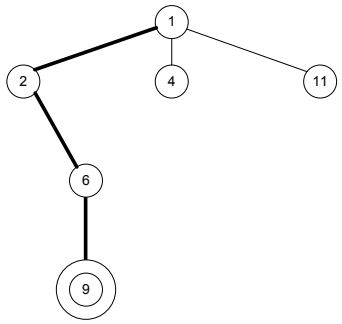
Iteración 9:

Frontera: {(9,6), (4,1), (11,1)}
Explorados: {(1,-), (2,1), (5,2), (6,2)}



Iteración 10:

Frontera: {(9,6), (4,1), (11,1)}
Explorados: {(1,-), (2,1), (5,2), (6,2), **(9,6)**}



El camino es *1-2-6-9*

2. Para los siguientes problemas, describa los siguientes aspectos:

a) Representación de sus estados

b) Reglas de producción

c) Estrategia de control apropiada para encontrar la solución

c) Jarras de agua

REPRESENTACIÓN DE SUS ESTADOS

(x, y) tal que x=0, 1, 2, 3, 4 e y=0, 1, 2, 3

x representa los litros de la jarra de 4 litros

y representa los litros de la jarra de 3 litros

Estado inicial -> (0, 0) : las jarras vacías

Estado final -> (2, 0) : jarra de 4 litros por la mitad y jarra de 3 litros vacía

REGLAS DE PRODUCCIÓN

- (x, y) $\rightarrow$ (4, y) Llenar la jarra de 4 litros: si x<4
- (x, y) $\rightarrow$ (x, 3) Llenar la jarra de 3 litros: si y<3
- (x, y) $\rightarrow$ (0, y) Vaciar la jarra de 4 litros: si x>0
- (x, y) $\rightarrow$ (x, 0) Vaciar la jarra de 3 litros: si y>0
- (x, y) $\rightarrow$ (4, y-(4-x)) Verter agua de la jarra de 3 litros a la de 4 hasta llenarla: si x+y≥4, y>0, x<4
- (x, y) $\rightarrow$ (x-(3-y), 3) Verter agua de la jarra de 4 litros a la de 3 hasta llenarla: si x+y≥3, x>0, y<3

- (x, y) $\rightarrow$ (x+y, 0) Verter todo el agua del cántaro de 3 litros al de 4 litros: si x+y≤4, y>0
- (x, y) $\rightarrow$ (0, x+y) Verter todo el agua del cántaro de 4 litros al de 3 litros: si x+y≤3, x>0

ESTRATEGIA DE CONTROL APROPIADA PARA ENCONTRAR LA SOLUCIÓN

Búsqueda bidireccional en amplitud, ya que conocemos el estado inicial y el final.

d) Puzzle a 8

REPRESENTACIÓN DE SUS ESTADOS

El número de diferentes ordenaciones de 9 casillas sin repetición son precisamente todas sus permutaciones, por lo tanto es 9! = 362.880 . Como el número de permutaciones posibles, 9! se dividen entre dos $^{9!}/_{2}$ con un número de inversiones par y otros tantos con inversiones impar, solo la primera mitad es alcanzable (debido a la imposibilidad de mover dos piezas del puzzle adyacentes). Por lo tanto, el tamaño del espacio de estados del puzzle a 8 es de 181.440 estados posibles.

REGLAS DE PRODUCCIÓN

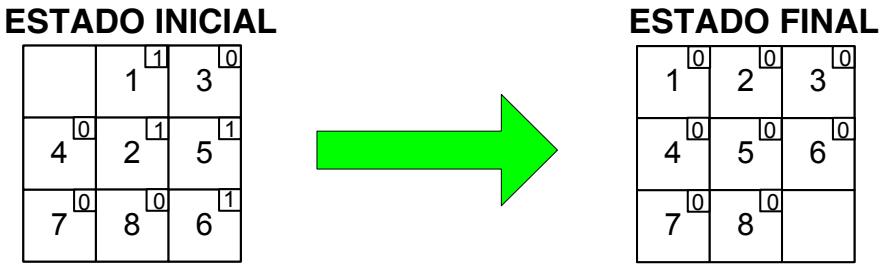
- Si la casilla blanca no está en la fila 1 $\rightarrow$ se puede mover el blanco hacia arriba
- Si la casilla blanca no está en la fila 3 $\rightarrow$ se puede mover el blanco hacia abajo
- Si la casilla blanca no está en la columna 1 $\rightarrow$ se puede mover el blanco a la izquierda
- Si la casilla blanca no está en la columna 3 $\rightarrow$ se puede mover el blanco a la derecha

ESTRATEGIA DE CONTROL APROPIADA PARA ENCONTRAR LA SOLUCIÓN

Algoritmo A*, utilizando como heurística, por ejemplo, la suma de las distancias de Manhattan de todas las fichas que forman un estado concreto en el puzzle. Esta suma es para todas las casillas mal colocadas de las distancias horizontales y verticales a las posiciones de esas casillas en el estado meta.

4. Resuelva el problema del puzzle a 8 con los diferentes métodos de búsqueda. Para las búsquedas primero el mejor y A* utilice como heurística la suma de la distancia de Manhattan de cada ficha hasta su posición objetivo.

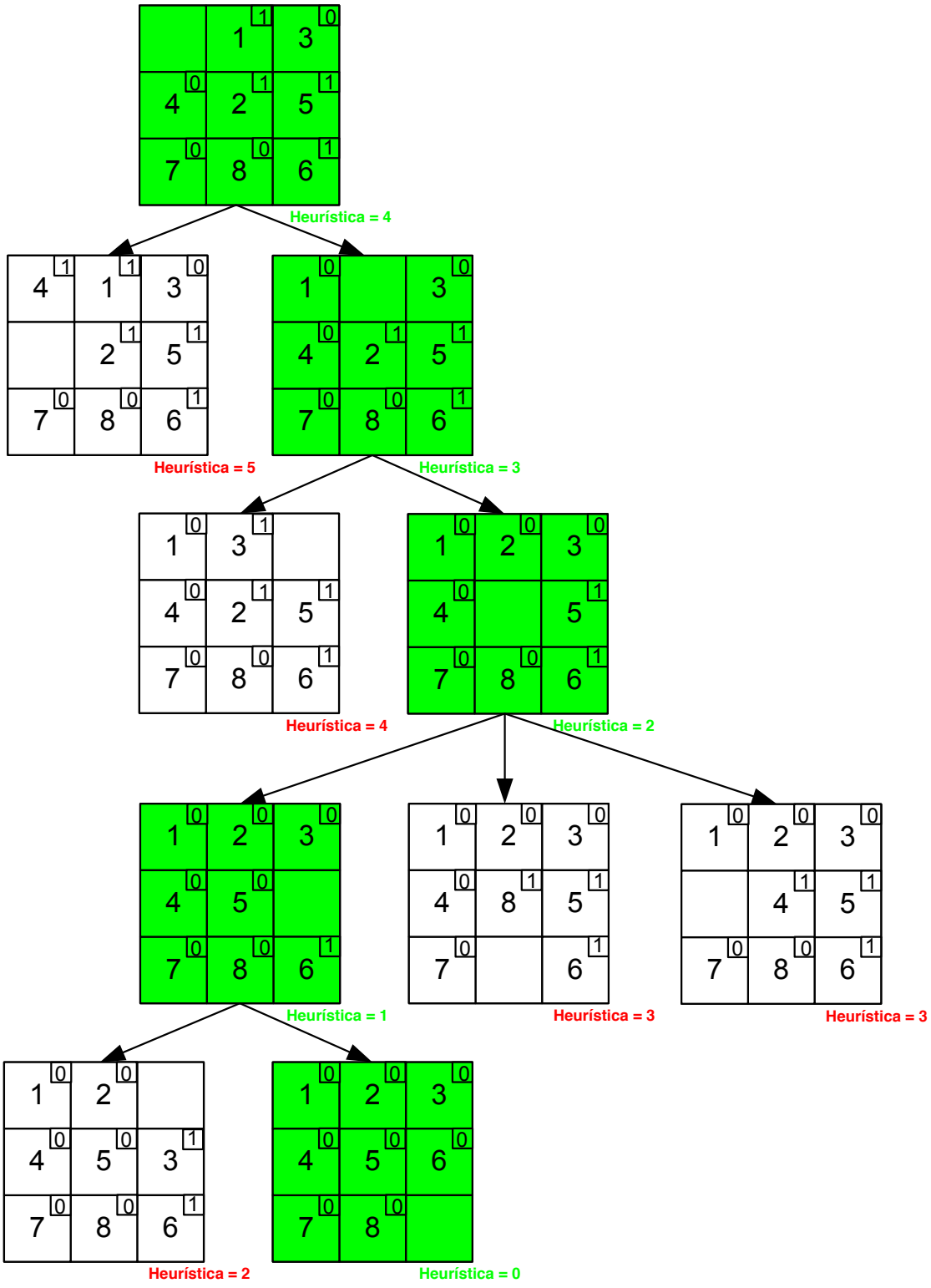
La realización a mano de los algoritmos de búsqueda a ciegas sobre este problema, serían muy costosos e ineficientes, por lo que decidimos solo realizarlos por los métodos de Primero el mejor y A*



ALGORITMO PRIMERO EL MEJOR

Vamos escogiendo en cada iteración, el estado con menor heurística, hasta llegar al nodo objetivo.

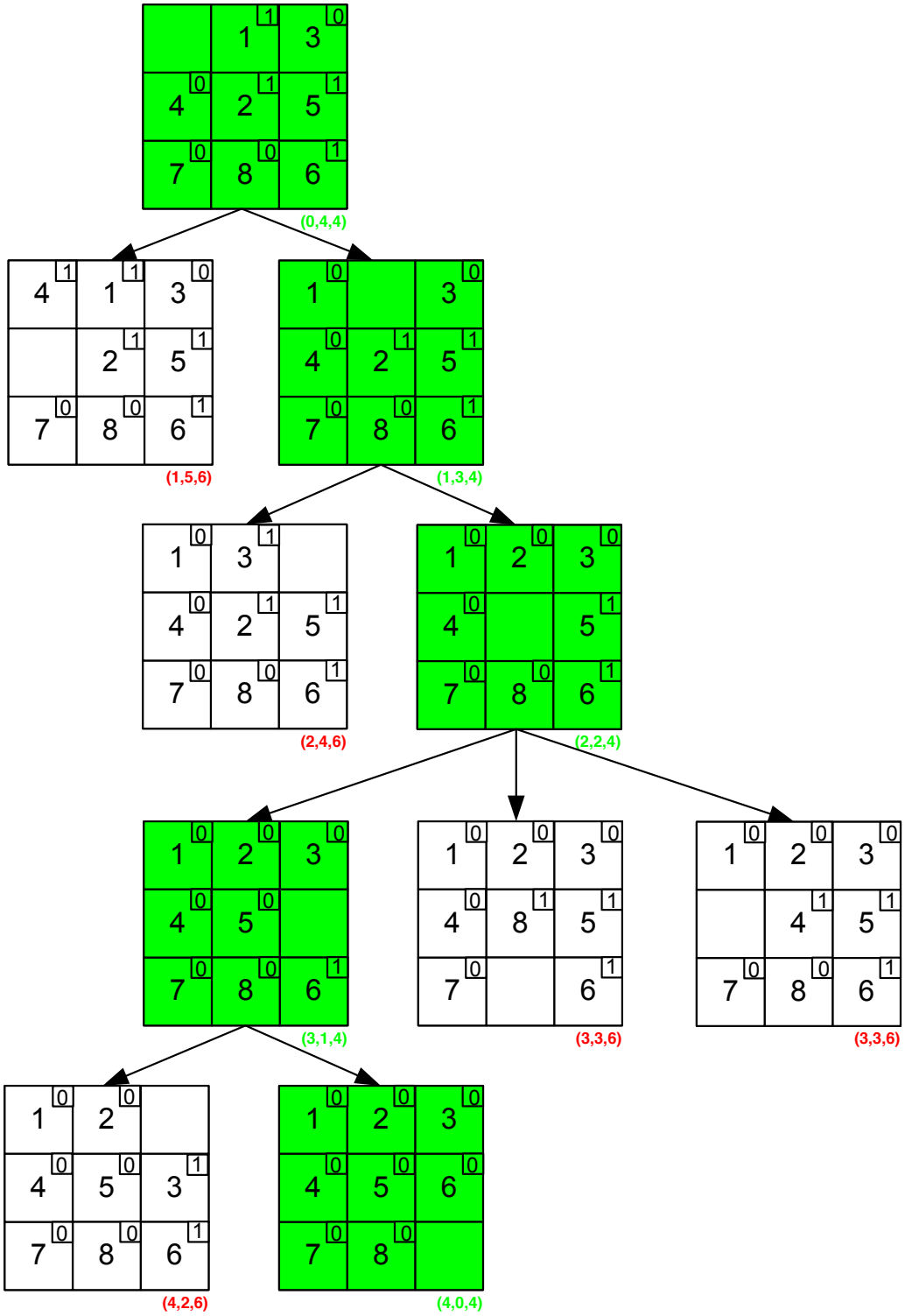
Los nodos los representamos como la figura de cómo se encontraría el puzzle en ese momento. Cada una de las piezas tiene también un recuadro en la esquina superior derecha que indica la distancia de Manhattan desde la posición en la que se encuentra hasta la posición del estado final, cuya suma de esa distancia de cada casilla conforma la función heurística.


Heurística = 4
Heurística = 5
Heurística = 3
Heurística = 4
Heurística = 2
Heurística = 1
Heurística = 3
Heurística = 3
Heurística = 2
Heurística = 0

ALGORITMO A*

La notación que aparece debajo de cada estado sería: (pasos recorridos, heurística, suma de ambos)

En cada iteración vamos escogiendo el estado que la suma de pasos realizados y heurística sea menor.

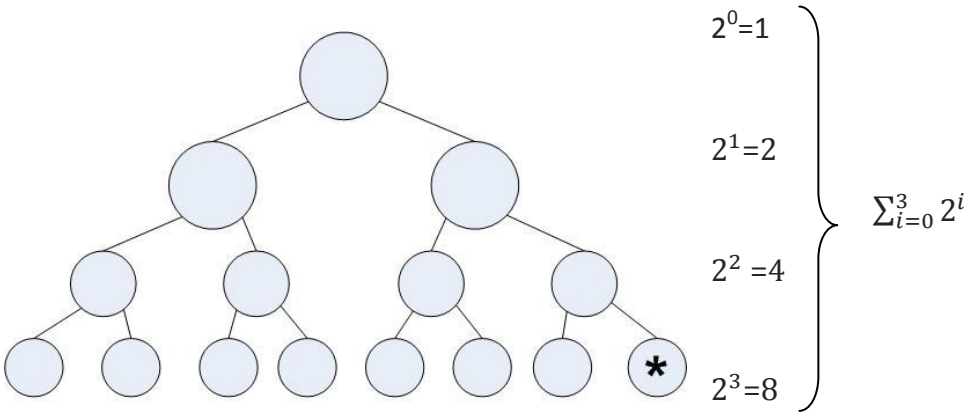


5. Consideremos un árbol finito de profundidad d con un factor de ramificación b (un solo nodo raíz, con profundidad 0, y b sucesores por cada nodo, etc.). Supongamos que el nodo objetivo menos profundo se encuentra a una profundidad g≤d.

a) ¿Cuál es el número mínimo de nodos generados por la búsqueda primero en profundidad con una profundidad límite de d? ¿Y el máximo?

El nº mínimo de nodos creados será g*b+1 (+1 es por el nodo origen), es decir, si encontramos el nodo objetivo en la primera rama que examinemos, se crearán b nodos por cada vez que examinemos uno, y examinaremos g nodos (el nodo objetivo menos profundo está a profundidad g). En el mejor de los casos, g sería 1 (no consideramos que el nodo origen sea el destino), y se crearían b+1 nodos.

Como máximo, el nodo destino podría encontrarse en la última rama que examináramos, por lo que se generarán $\sum_{i=0}^{d} b^i$ nodos. Esto es porque en el peor caso, tendríamos que recorrer hasta la última rama del árbol.



b) ¿Cuál es el número mínimo de nodos generados por la búsqueda primero en anchura? ¿Y el máximo?

En búsqueda en anchura, se generarán $\sum_{i=0}^{g} b^i$ nodos siempre (crear todos los hijos correspondientes a cada nivel de profundidad hasta llegar a profundidad g). En el mejor de los casos (g=1) se crearán b+1 nodos, mientras que en el peor de los casos (g=d), los nodos generados serían $\sum_{i=0}^{d} b^i$

c) ¿Cuál es el número mínimo de nodos generados por el descenso iterativo? ¿Y el máximo? (Considérese que comenzamos con una profundidad límite inicial de 1 y la vamos incrementando una unidad cada iteración).

Como mínimo, se generan $\sum_{i=0}^{g-1}(\sum_{j=0}^{i} b^i) + b * g$+1

b*g+1: se refiere a la última iteración realizada, es decir la iteración en la que la profundidad máxima es g, y por tanto se encuentra el nodo solución. En esa iteración, los nodos creados son los mismos que en una búsqueda en profundidad.

$\sum_{i=0}^{g-1}(\sum_{j=0}^{i} b^i)$: en una iteración en la que la profundidad máxima sea menor a la profundidad donde se encuentra el nodo solución menos profundo, los nodos creados serán los mismos que en el peor de los casos de la búsqueda en profundidad.

Como máximo, en la última iteración (i=g), en vez de encontrar el nodo objetivo el primero lo encontremos el último, así que: $\sum_{i=0}^{g}(\sum_{j=0}^{i} b^i)$. En el peor caso (g=d), se crearían $\sum_{i=0}^{d}(\sum_{j=0}^{i} b^i)$ nodos (es decir, el sumatorio de los nodos creados en el peor caso de la búsqueda en profundidad desde la iteración con profundidad 0 hasta profundidad d).

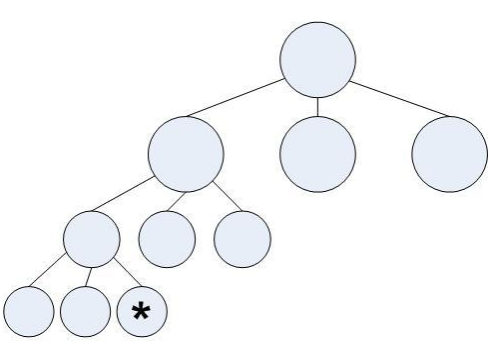
6. Supongamos que mediante la búsqueda primero en anchura se pueden generar 10000 nodos por segundo y que para almacenar cada nodo son necesarios 100 bytes. Con estos datos ¿cuáles son los requisitos de tiempo y memoria para una búsqueda aplicada a un árbol de búsqueda en profundidad d y un factor de ramificación de 5? Muestre los resultados del análisis en una tabla.

Profundidad	Nodos	Tiempo	Memoria
1	$\sum_{i=0}^{1} 5^i$ = 6	6/10000= 600 ns	6*100 = 600B
2	$\sum_{i=0}^{2} 5^i$ = 31	31/10000= 3100ns	31*100 = 3100B
3	$\sum_{i=0}^{3} 5^i$ = 156	156/10000= 15.6 ms	15600B ≃ 15KB
4	$\sum_{i=0}^{4} 5^i$ = 781	78.1 ms	78100B ≃ 76KB
5	$\sum_{i=0}^{5} 5^i$ = 3906	390.6 ms	≃ 381KB
6	$\sum_{i=0}^{6} 5^i$ = 19531	1953.1 ms ≃ 1.9 s	≃ 1907KB
7	$\sum_{i=0}^{7} 5^i$ = 97656	≃ 9.8 s	≃9537KB ≃ 9.3MB
8	$\sum_{i=0}^{8} 5^i$ = 488281	≃ 48.8 s	≃ 46.57MB
9	$\sum_{i=0}^{9} 5^i$ = 2441406	≃ 244 s ≃ 4.07 min	≃ 232.8MB
10	$\sum_{i=0}^{10} 5^i$= 12207031	≃ 20.35 min	≃1164.15MB≃1.14GB
15	$\sum_{i=0}^{15} 5^i$=3.815E+10	≃1059 horas ≃44.1 días	≃3552.7GB≃3.47TB
20	$\sum_{i=0}^{20} 5^i$=1.192E+14	≃3311369horas≃378años	≃ 10742TB

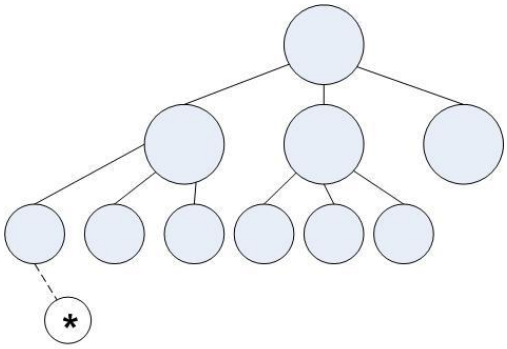
7. Razona situaciones y dibuja los espacios de búsqueda asociados (árboles o grafos con nodos iniciales y finales y camino encontrado a la solución), en los que:

a) La búsqueda en profundidad encuentre la solución antes que la búsqueda en anchura

La búsqueda en profundidad encontrará la solución antes que en anchura si el nodo destino se encuentra a alta profundidad y en una de las primeras ramas que examinemos



Búsqueda en profundidad



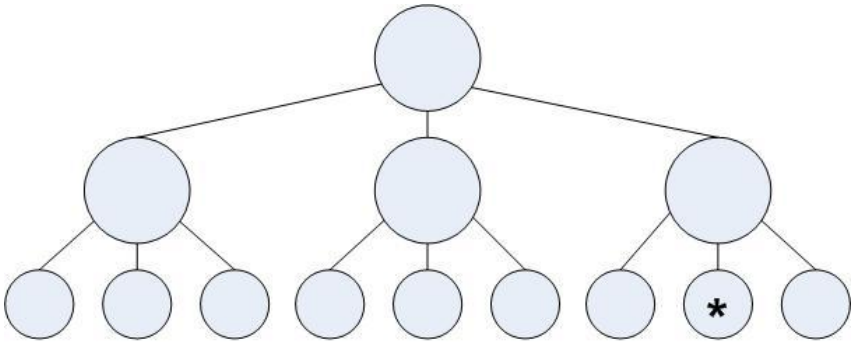
Búsqueda en anchura

En ambos casos se han realizado las mismas iteraciones, mientras que en la izquierda ya se ha conseguido llegar al nodo solución, en la derecha no lo ha conseguido (necesitaría dos iteraciones más).

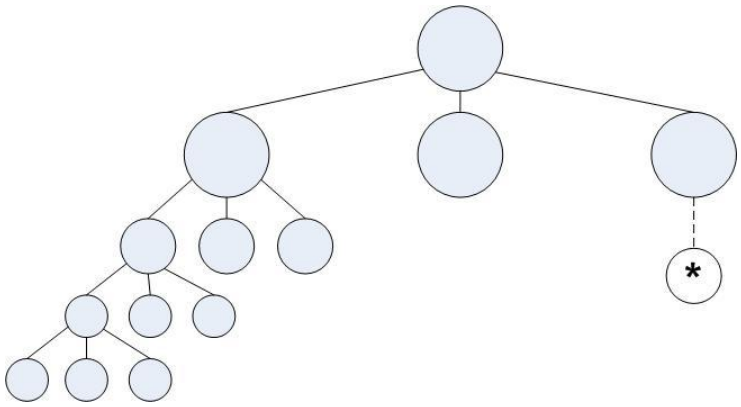
Búsqueda en anchura va creando todos los hijos de los nodos que están a un mismo nivel, entonces si la solución se encuentra a una profundidad alta, tardará más en llegar a él.

b) La búsqueda en anchura encuentre la solución antes que la búsqueda en profundidad

Si el nodo solución se encuentra a baja profundidad y en una de las últimas ramas que exploraríamos con profundidad.



Búsqueda en anchura



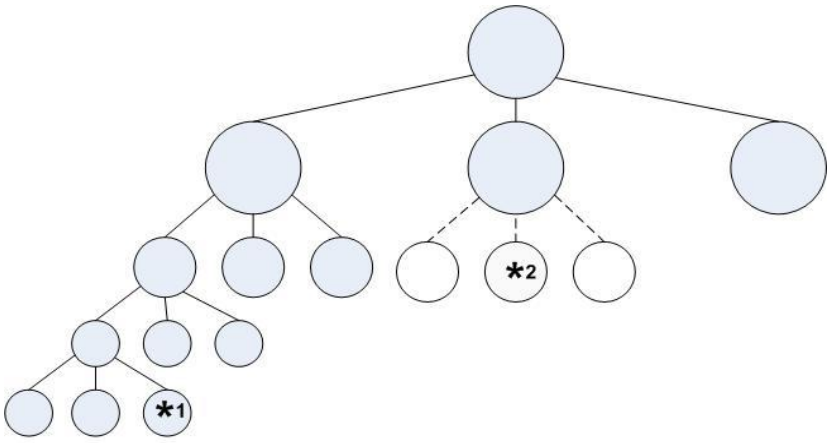
Búsqueda en profundidad

En ambos dibujos se han realizado las mismas iteraciones, mientras que en búsqueda en anchura ya hemos conseguido encontrar el nodo solución, en profundidad no.

Esto se debe a que profundidad busca hasta abajo en una rama y luego pasa a otra. Si la solución se encuentra en una de las últimas ramas que explora, tardará más.

c) La búsqueda en profundidad no encuentre la solución óptima

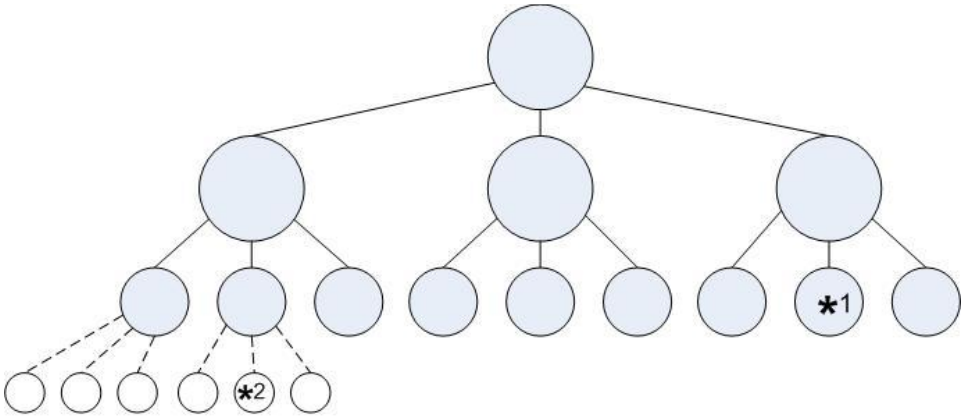
Si hay varios nodos destino, puede ser que la primera rama que examinemos en la que se encuentre un nodo solución, éste no sea el menos profundo. Es decir, si examinamos antes una rama con un nodo solución más profundo que la rama que tiene el nodo solución menos profundo.



Como se puede ver, tenemos dos nodos solución. La búsqueda en profundidad podría comenzar examinando las ramas de la izquierda, encontrando el nodo *1, mientras que la solución óptima sería *2.

d) La búsqueda en anchura no encuentre la solución óptima

La búsqueda en anchura siempre encuentra la solución óptima, pues va avanzando conforme a distancia (primero explora todos los caminos de distancia 1, luego todos los de distancia 2,…, no puede encontrar un camino de distancia 4 y que haya uno de 2 porque ya los había explorado antes).



Como el método de búsqueda en anchura va creando los hijos de todos los nodos de una misma profundidad, siempre encontrará el nodo $*1$ antes que el $*2$.

e) Búsqueda en profundidad sea mejor que Primero el mejor

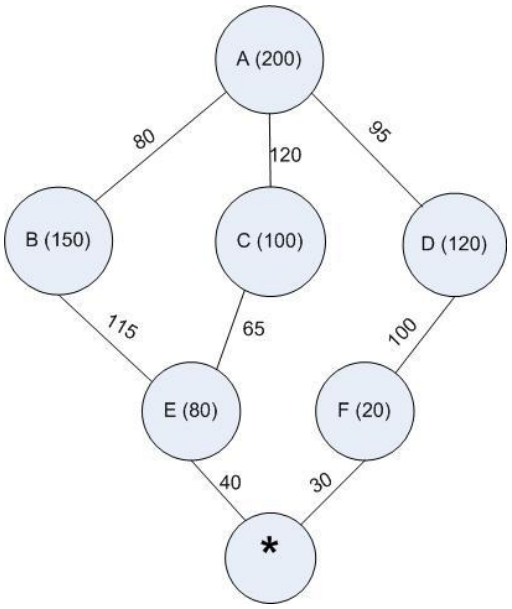
Si la heurística que tenemos es mala, el método de búsqueda Primero el mejor no garantiza que funcione bien, por lo que el método en profundidad puede ser mejor.

f) Búsqueda en anchura sea mejor que A*

Al igual que antes, si no tenemos una buena heurística, el método A* no funciona como debiera, así que la búsqueda en anchura podría ser mejor.

g) A* no encuentre la solución óptima

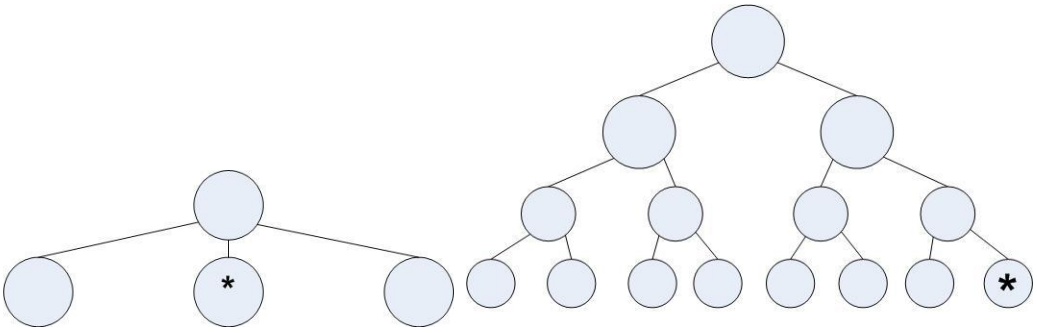
A* podrá no encontrar la solución óptima si tenemos una mala heurística, si la heurística es pesimista, es decir, que el valor de la heurística es mayor que el de la distancia real entre el nodo actual y el objetivo.



En este caso, por ejemplo (en el que en el camino se encuentra la distancia real, y dentro del nodo la heurística), podemos ver que la heurística del nodo E sería pesimista, pues es mayor que la distancia real que existe entre él y el objetivo, por lo que no podríamos asegurar que el método A* con esta heurística encontrase el camino óptimo.

h) La búsqueda en profundidad y la búsqueda en anchura tengan un funcionamiento equivalente

Si el nodo objetivo se encuentra a profundidad 1, o si estuviese en el peor de los casos (última rama examinada en ambos casos).



Nodo solución en profundidad 1 Nodo solución en peor de los casos

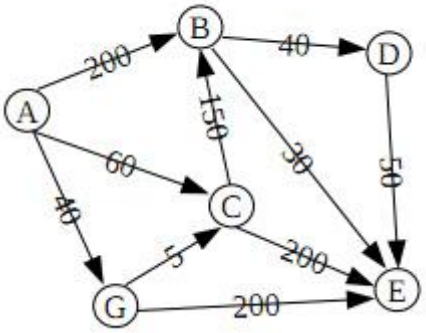
i) La búsqueda de profundidad iterativa sea peor que la búsqueda en profundidad

Si solo hay un nodo objetivo, siempre que no haya ramas infinitas, pues el funcionamiento es el mismo, exceptuando que en la iterativa se establece un límite de profundidad que se va incrementando poco a poco. Si no hay ramas

infinitas, será igual que la búsqueda en profundidad pero con un gran gasto en tiempo añadido (todas las iteraciones en las que la profundidad límite es menor a la profundidad en la que se encuentra el nodo objetivo).

Si hay varios nodos solución, aunque tarde más, la búsqueda en profundidad iterativa siempre va a encontrar entre ellos la solución óptima.

8. Dado el grafo de la figura, determinar el mejor camino para ir del nodo A al E mediante la utilización del algoritmo A*



Nodo	*Distancia Heurística a E*
A	150
B	30
C	70
D	50
G	50
E	0

La notación que aparece para los nodos es: (nodo, mejor padre, distancia recorrida, heurística, suma de ambos)

Iteración 0:

Frontera: {(A,-,0,150,150)}
Explorados: { }

Iteración 1:

Frontera: {(G,A,40,50,90), (C,A,60,70,130), (B,A,200,30,230)}
Explorados: {(A,-,0,150,150)}

Iteración 2:

Frontera: {(C,G,45,70,115), (B,A,200,30,230), (E,G,240,0,240)}
Explorados: {(A,-,0,150,150), (G,A,40,50,90)}

Iteración 3:

Frontera: {(B,C,195,30,225), (E,G,240,0,240) }
Explorados: {(A,-,0,150,150), (G,A,40,50,90), (C,G,45,70,115)}

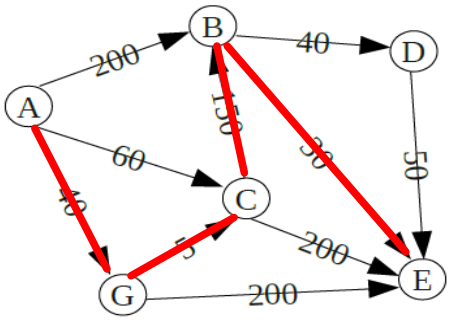
Iteración 4:

Frontera: {(E,B,225,0,225), (D,B,235,50,285)}
Explorados: {(A,-,0,150,150), (G,A,40,50,90), (C,G,45,70,115), (B,C,195,30,225) }

Iteración 5:

Frontera={(D,B,235,50,285)}
Explorados={(A,-,0,150,150), (G,A,40,50,90), (C,G,45,70,115), (B,C,195,30,225), **(E,B,225,0,225)**}

El camino será *A-G-C-B-E*



9. Sea el siguiente grafo, en el que los arcos tienen un coste y los nodos una estimación heurística de su distancia al nodo Z (Z es el nodo objetivo y A es el nodo inicial).

a) Sin ningún conocimiento a priori (sin conocer la estructura del grafo, sus pesos…) ¿qué podrías hacer para asegurarte de que A* encuentra el camino mínimo hasta el nodo solución?

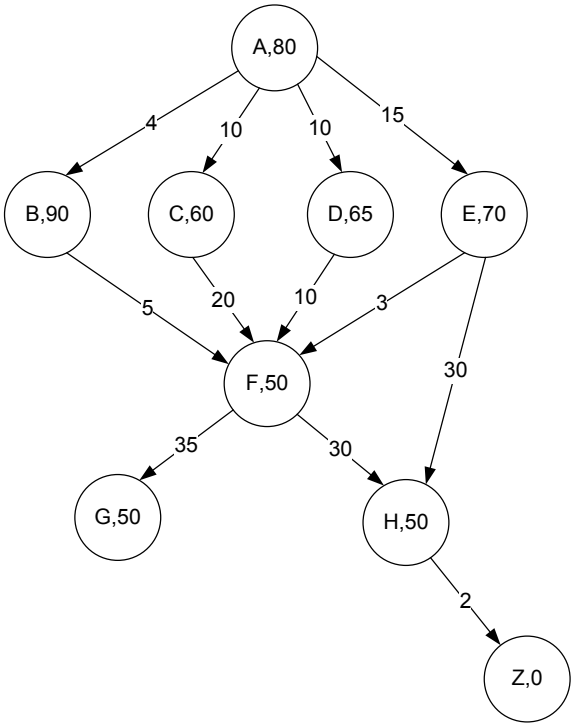
Asegurarnos de que la información heurística sea menor o igual a la información real, ya que este método es óptimo si:

h(x) ≤ h'(x) , es decir, información heurística ≤ información real

b) Observando el grafo, pero sin aplicar A* ¿puedes asegurar si este método encontrará o no el camino mínimo entre A y Z?

No podemos asegurarlo, dado que la heurística es pesimista, por lo tanto la heurística es mala. Solo podemos asegurar que este método encontrará el camino mínimo si su heurística es menor o igual al coste real.

c) Aplica el algoritmo A*. Dibuja en cada etapa del algoritmo el subgrafo parcial creado y la situación de las listas ABIERTA y CERRADA



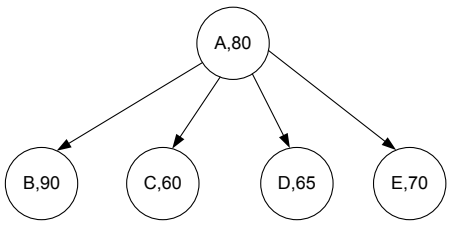
La notación que aparece para los nodos es: (nodo, mejor padre, dato real, heurística, suma de ambos)

Iteración 0:

Frontera: {(A,-,0,80,80)}
Explorados: { }

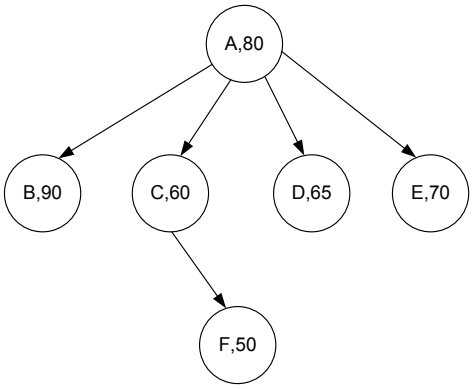
Iteración 1:

Frontera: {(B,A,4,90,94), (C,A,10,60,70), (D,A,10,65,75), (E,A,15,70,85)}
Explorados: {(A,-,0,80,80)}



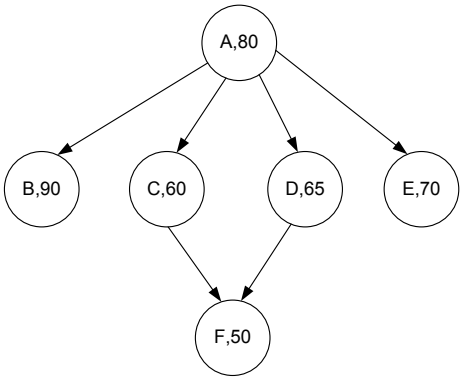
Iteración 2:

Frontera: {(B,A,4,90,94), (D,A,10,65,75), (E,A,15,70,85), (F,C,30,50,80)}
Explorados: {(A,-,0,80,80), (C,A,10,60,70)}



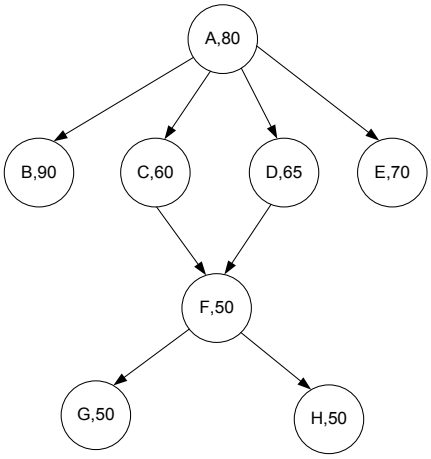
Iteración 3:

Frontera: {(B,A,4,90,94), (E,A,15,70,85), (F,D,20,50,70)}
Explorados: {(A,-,0,80,80), (C,A,10,60,70), (D,A,10,65,75)}



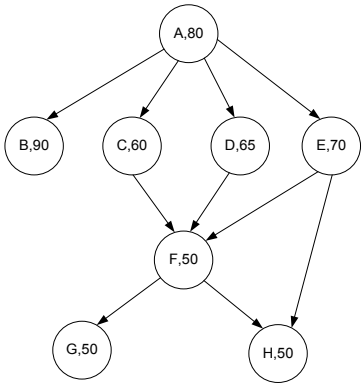
Iteración 4:

Frontera: {(B,A,4,90,94), (E,A,15,70,85), (G,F,55,50,105), (H,F,50,50,100)}
Explorados: {(A,-,0,80,80), (C,A,10,60,70), (D,A,10,65,75),(F,D,20,50,70)}



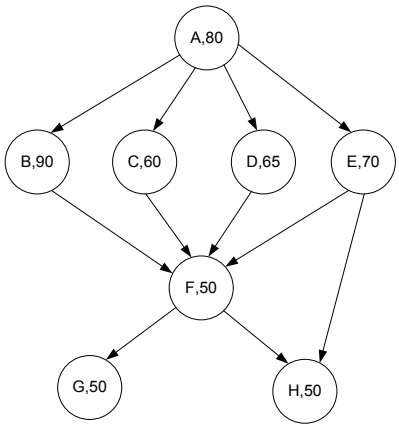
Iteración 5:

Frontera: {(B,A,4,90,94), (G,F,53,50,103), (H,E,45,50,95)}
Explorados: {(A,-,0,80,80), (C,A,10,60,70), (D,A,10,65,75), (F,E,18,50,68), (E,A,15,70,85)}



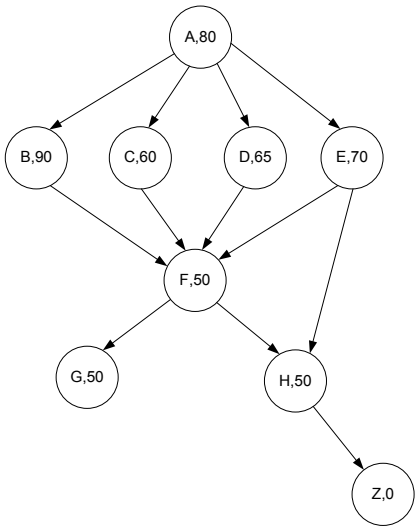
Iteración 6:

Frontera: {(G,F,44,50,94), (H,F,39,50,89)}
Explorados: {(A,-,0,80,80), (C,A,10,60,70), (D,A,10,65,75), (F,B,9,50,59), (E,A,15,70,85), (B,A,4,90,94)}



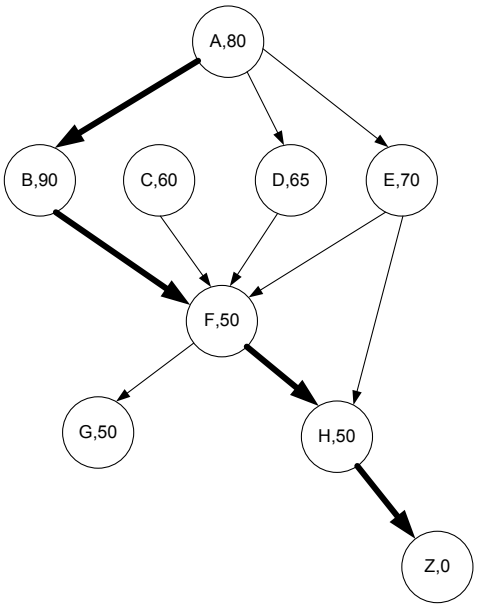
Iteración 7:

Frontera: {(G,F,44,50,94), (Z,H,41,0,41)}
Explorados: {(A,-,0,80,80), (C,A,10,60,70), (D,A,10,65,75), (F,B,9,50,59), (E,A,15,70,85), (B,A,4,90,94), (H,F,39,50,89)}



Iteración 8:

Frontera: {(G,F,44,50,94)}
Explorados: { (A,-,0,80,80), (C,A,10,60,70), (D,A,10,65,75), (F,B,9,50,59), (E,A,15,70,85), (B,A,4,90,94), (H,F,39,50,89), **(Z,H,41,0,41)**}



El camino será *A-B-F-H-Z*

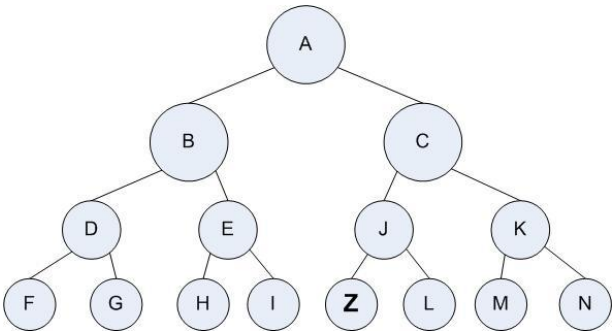
10. Responde justificadamente a estas dos cuestiones:

a) ¿Por qué la complejidad temporal de todas las búsquedas ciegas, exceptuando la búsqueda bidireccional, es la misma?

Porque, en el peor de los casos, habrá que recorrer todos los nodos del grafo, por lo que ambos tardarían lo mismo.

b) ¿Por qué la complejidad espacial de la búsqueda en profundidad es menor que la de la búsqueda en anchura?

Porque, en el peor de los casos, en la búsqueda en anchura tendremos almacenados en memoria todos los nodos, mientras que en profundidad, cuando exploramos una rama, llegamos al final de la misma y no hemos encontrado ahí la solución, podemos ir borrando de memoria nodos innecesarios.

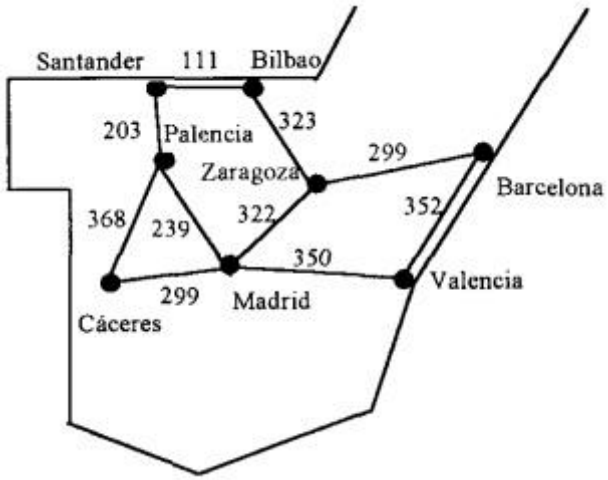


Considerando el nodo Z como nodo objetivo, según el dibujo, la búsqueda en profundidad examinaría primero las ramas hijas del nodo B (A-B-D-F, A-B-D-G,…), no encontrando el nodo solución, por lo que antes de proseguir con la búsqueda, esos nodos pueden ser eliminados de memoria, pues no nos serán útiles en el resto de la búsqueda.

Ejercicios que no son de la relación:

El siguiente problema se ha extraído del libro "Problemas resueltos de Inteligencia Artificial", editorial "Addison-Wesley", edición 1998, página 73, ejercicio2.7

I) Dado el siguiente mapa de carreteras en el que los caminos entre cada dos ciudades están etiquetados con sus distancias en kilómetros.



a) Describa el grafo correspondiente a la búsqueda del camino más corto entre Palencia y Barcelona. Para ello señale cuál es el algoritmo adecuado y aplíquelo. Indique y describa la utilización del método que le permita encontrar el camino que recorra el menor número de ciudades para trasladarse entre las capitales anteriormente señaladas.

Para encontrar el camino más corto, el algoritmo A*. Se escoge en cada iteración el nodo que minimice la función g(x) + h(x)

{ g(x) : distancia recorrida real

h(x) : heurística }

Para que funcione bien, la heurística ha de ser pesimista (incluso podríamos utilizar en todos los casos heurística 0).

Para recorrer el menor nº de ciudades podemos usar la búsqueda en amplitud, ya que es un método óptimo.

b) Téngase en cuenta el siguiente cuadro de distancias aéreas estimadas desde cada ciudad a Barcelona:

	Bilbao	Cáceres	Madrid	Palencia	Santander	Valencia	Zaragoza
Barcelona	502	850	550	580	605	303	275

Utilizando como función heurística la distancia aérea estimada a la meta, que llamaremos "d", describa el grafo de búsqueda resultante de la aplicación del método heurístico que garantice encontrar la solución óptima. Justifique su elección frente a otros algoritmos de "búsqueda informada".

La notación que usamos para los nodos es: (nodo, mejor padre, g(x), h(x), suma)

Iteración 0:

Frontera: {(Palencia, - , 0, 580, 580)}

Explorados: { }

Iteración 1:

Frontera: {(Santander, Palencia, 203, 605, 808), (Cáceres, Palencia, 368, 850, 1218), (Madrid, Palencia, 239, 550, 789)}

Explorados: {(Palencia, - , 0, 580, 580)}

Iteración 2:

Frontera: {(Santander, Palencia, 203, 605, 808), (Cáceres, Palencia, 368, 850, 1218),(Zaragoza, Madrid, 561, 275, 836),(Valencia, Madrid, 589, 303, 892)}

Explorados: {(Palencia, - , 0, 580, 580), (Madrid, Palencia, 239, 550, 789)}

Iteración 3:

Frontera: {(Cáceres, Palencia, 368, 850, 1218), (Zaragoza, Madrid, 561, 275, 836), (Valencia, Madrid, 589, 303, 892), (Bilbao, Santander, 314, 502, 816)}

Explorados: {(Palencia, - , 0, 580, 580), (Madrid, Palencia, 239, 550, 789), (Santander, Palencia, 203, 605, 808)}

Iteración 4:

Frontera: {(Cáceres, Palencia, 368, 850, 1218), (Zaragoza, Madrid, 561, 275, 836), (Valencia, Madrid, 589, 303, 892)}

Explorados: {(Palencia, - , 0, 580, 580), (Madrid, Palencia, 239, 550, 789), (Santander, Palencia, 203, 605, 808), (Bilbao, Santander, 314, 502, 816)}

Iteración 5:

Frontera: {(Cáceres, Palencia, 368, 850, 1218), (Valencia, Madrid, 589, 303, 892), (Barcelona, Zaragoza, 860, 0, 860)}

Explorados: {(Palencia, - , 0, 580, 580), (Madrid, Palencia, 239, 550, 789), (Santander, Palencia, 203, 605, 808), (Bilbao, Santander, 314, 502, 816), (Zaragoza, Madrid, 561, 275, 836)}

Iteración 6:

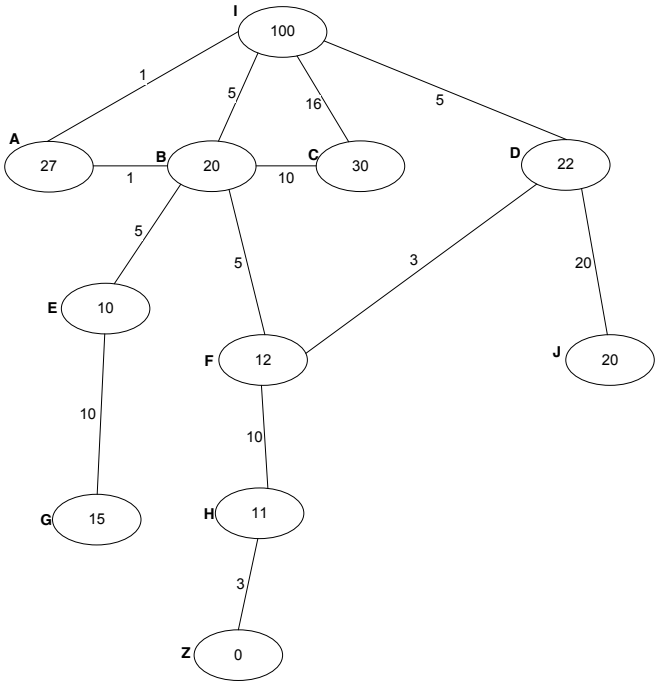
Frontera: {(Cáceres, Palencia, 368, 850, 1218), (Valencia, Madrid, 589, 303, 892)}

Explorados: {(Palencia, - , 0, 580, 580), (Madrid, Palencia, 239, 550, 789), (Santander, Palencia, 203, 605, 808), (Bilbao, Santander, 314, 502, 816), (Zaragoza, Madrid, 561, 275, 836), **(Barcelona, Zaragoza, 860, 0, 860)**}

Camino: Palencia - Madrid - Zaragoza - Barcelona

El siguiente problema se ha extraído del libro "Problemas resueltos de Inteligencia Artificial", editorial "Addison-Wesley", edición 1998, página 62, ejercicio2.4

II) Aplicar el algoritmo A* al siguiente grafo. El nodo inicial es I y hay un solo nodo meta que en este caso es Z. A cada arco se le ha asociado su coste y a cada nodo la estimación de la menor distancia desde ese nodo al nodo meta



La notación que usamos para los nodos es: (nodo, mejor padre, real, heurística, suma de ambos)

Iteración 0:

Frontera: {(I,-,0,100,100)}
Explorados: { }

Iteración 1:

Frontera: {(A,I,1,27,28), (B,I,5,20,25), (C,I,16,30,46), (D,I,5,22,27)}
Explorados: {(I,-,0,100,100)}

Iteración 2:

Frontera: {(A,I,1,27,28), (C,B,16,30,45), (D,I,5,22,27), (E,B,10,10,20), (F,B,10,12,22)}
Explorados: {(I,-,0,100,100), (B,I,5,20,25)}

Iteración 3:

Frontera: {(A,I,1,27,28), (C,B,16,30,45), (D,I,5,22,27), (F,B,10,12,22), (G,E,20,15,35)}
Explorados: {(I,-,0,100,100), (B,I,5,20,25), (E,B,10,10,20)}

Iteración 4:

Frontera: {(A,I,1,27,28), (C,B,16,30,45), (D,I,5,22,27), (G,E,20,15,35), (H,F,20,11,31)}
Explorados: {(I,-,0,100,100), (B,I,5,20,25), (E,B,10,10,20), (F,B,10,12,22)}

Iteración 5:

Frontera: {(A,I,1,27,28), (C,B,16,30,45), (G,E,20,15,35), (H,F,18,11,29), (J,D,25,20,45)}

Explorados: {(I,-,0,100,100), (B,I,5,20,25), (E,B,10,10,20), (F,D,8,12,20), (D,I,5,22,27)}

Iteración 6:

Frontera: {(C,B,12,30,42), (G,E,17,15,32), (H,F,17,11,28), (J,D,25,20,45)}

Explorados: {(I,-,0,100,100), (B,A,2,20,22), (E,B,7,10,17), (F,B,7,12,19), (D,I,5,22,27), (A,I,1,27,28)}

Iteración 7:

Frontera: {(C,B,12,30,42), (G,E,17,15,32), (J,D,25,20,45), (Z,H,20,0,20)}

Explorados: {(I,-,0,100,100), (B,A,2,20,22), (E,B,7,10,17), (F,B,7,12,19), (D,I,5,22,27), (A,I,1,27,28), (H,F,17,11,28)}

Iteración 8:

Frontera: {(C,B,12,30,42), (G,E,17,15,32), (J,D,25,20,45)}

Explorados: {(I,-,0,100,100), (B,A,2,20,22), (E,B,7,10,17), (F,B,7,12,19), (D,I,5,22,27), (A,I,1,27,28), (H,F,17,11,28), **(Z,H,20,0,20)**}

El camino será *I-A-B-F-H-Z*