Departamento de Sistemas Informáticos y Computación
Escuela Técnica Superior de Ingeniería Informática
Universitat Politècnica de València

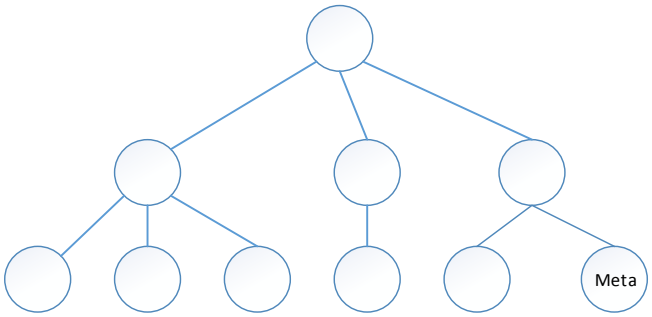
BOLETÍN DE EJERCICIOS
SISTEMAS INTELIGENTES

Bloque 1: Búsqueda

Septiembre 2019

CUESTIONES

1) Dado el espacio de estados de la figura, si realizáramos una búsqueda en profundidad con *backtracking* (expandiendo primero nodos más a la izquierda). ¿Cuál sería el máximo número de nodos que se almacenarían en memoria simultáneamente (OPEN+CLOSED)?



A. 7
B. 5
C. 10
D. 4

---

2) En el problema del puzzle, con h1 la heurística descolocadas y h2 distancias de Manhattan, si tenemos h3=min(h1,h2) y h4=abs(h1-h2), ¿cuáles de estas dos heurísticas, h3 y h4, serían admisibles?

A. Solo h3
B. Solo h4
C. Ambas
D. Ninguna de las dos

---

3) ¿Cuál sería la ordenación (de mayor a menor) en cuanto a número de nodos generados en el peor de los casos de las siguientes estrategias: anchura, profundización iterativa (PI), y profundidad limitada (PL) a nivel 5 por el usuario, para un problema con factor de ramificación b=10 y profundidad de la solución d=5?

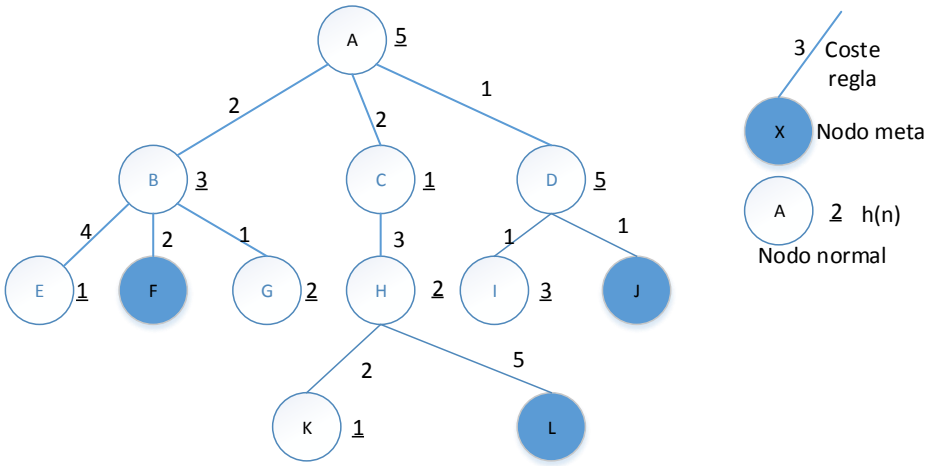
A. Anchura>PI>PL
B. Anchura>PL>PI
C. PI>Anchura>PL
D. PL>Anchura>PI

---

4) Decir cuál de las siguientes afirmaciones es FALSA para una heurística consistente:

A. f(n) es no decreciente
B. Garantiza la mejor solución en búsqueda en grafo incluso sin re-expandir
C. Nunca genera un nodo n1 igual a otro nodo n2 ya generado y donde f(n1)<f(n2)
D. El valor de h(n) de un nodo puede ser menor que el de su nodo padre.

5) Para el espacio de estados de la figura y dada una búsqueda voraz, ¿cuál es el nodo meta que se elegirá en primer lugar como solución?



A. J
B. L
C. I
D. F

---

6) Dado el espacio de estados de la pregunta 5, si aplicamos un algoritmo de profundización iterativa (expandiendo en primer lugar por la izquierda), ¿cuántos nodos se generarán en total?

A. 8
B. 7
C. 10
D. 12

---

7) Dado un problema de búsqueda en el que todos sus operadores tienen el mismo coste, indica cuál de las siguientes afirmaciones es ***correcta***:

A. Un algoritmo de búsqueda con una heurística admisible devolverá la solución más corta
B. Un estrategia en profundidad devolverá siempre la solución de menor coste
C. La estrategia en anchura devolverá la solución más corta pero no la solución de menor coste
D. La estrategia de coste uniforme devolverá la solución de menor coste pero no la solución más corta

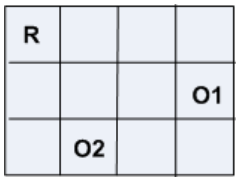
---

8) Dados cuatro métodos de búsqueda: M1 aplica un algoritmo en anchura, M2 aplica un algoritmo de coste uniforme, M3 aplica un algoritmo A con una heurística admisible y M4 aplica un algoritmo A con una heurística no admisible, indica cuál es la respuesta ***incorrecta***:

A. M1, M2 y M3 garantizan que encontrarán la solución óptima independientemente del coste de las acciones
B. M2 y M3 garantizan que encontrarán la solución óptima independientemente del coste de las acciones
C. M3 expandirá menos nodos que M2
D. M4 podría encontrar la solución óptima

---

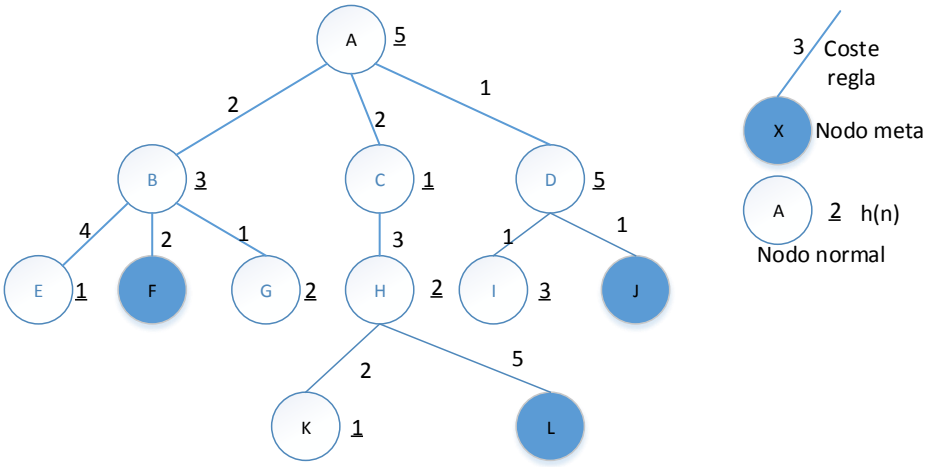
9) La figura muestra un tablero donde R es un robot cuyo objetivo es desplazarse a la posición donde se encuentra el objeto O1 y luego desplazarse a la posición del objeto O2. El robot solo puede moverse horizontal o verticalmente. La figura muestra una instancia concreta de este problema, pudiendo estar R, O1 y O2 en cualquiera de las casillas del tablero. Sea n un nodo de un árbol de búsqueda que representa una situación particular de R, O1 y O2, y manh(x,y) la distancia de Manhattan entre x e y, donde x, y ∈ {R,O1,O2}. Indica la afirmación que es ***correcta***:



A. h(n)=manh(R,O1)+manh(R,O2) es una heurística admisible para este problema
B. h(n)=manh(R,O1)+manh(O1,O2) es una heurística admisible para este problema
C. h(n)=manh(R,O1)*2 es una heurística admisible para este problema
D. No se puede definir una heurística admisible para este problema

---

10) Para el espacio de estados de la figura y dada una búsqueda de tipo A (f(n)=g(n)+h(n)) ¿cuántos nodos es necesario generar para encontrar la solución?



A. 6
B. 8
C. 10
D. 12

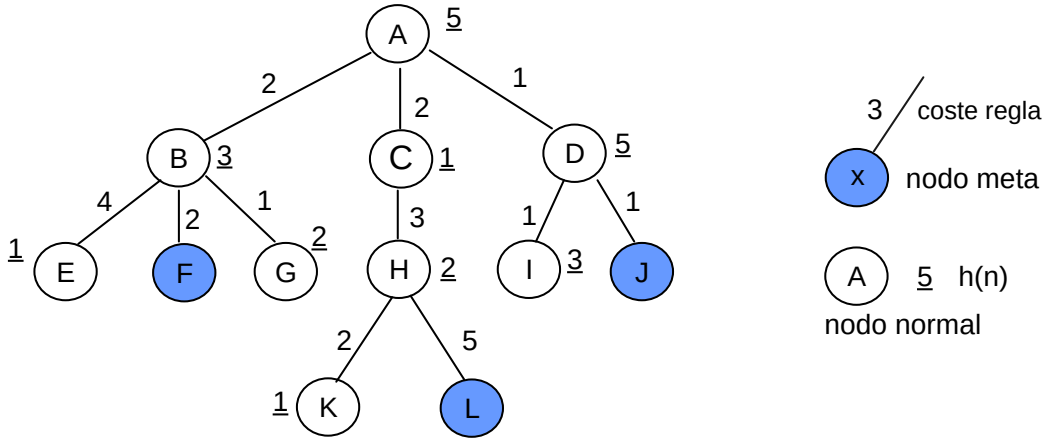
---

11) Para el espacio de estados de la pregunta 10, indica cuál es la afirmación ***correcta***:

A. La aplicación de un algoritmo de tipo A devuelve la solución óptima
B. La función h(n) es consistente (monótona)
C. Una estrategia de coste uniforme devolverá la misma solución que un algoritmo de tipo A
D. Ninguna de las anteriores

---

12) Para el espacio de estados de la figura y dada una búsqueda de tipo A (f(n)=g(n)+h(n)), indica cuál de las siguientes afirmaciones es CORRECTA:



A. La solución que encuentra la búsqueda de tipo A es el nodo J.
B. Es necesario generar un total de 10 nodos para encontrar una solución con un algoritmo de tipo A.
C. La función heurística h(n) no es admisible.
D. Ninguna de las anteriores.

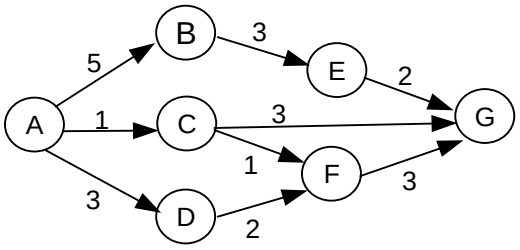
---

13) Dados cuatro métodos de búsqueda: M1 aplica un algoritmo anchura, M2 aplica un algoritmo de coste uniforme, M3 aplica un algoritmo en profundidad, M4 aplica un algoritmo de profundización iterativa, si los costes de las acciones son todos iguales indica cuál es la respuesta INCORRECTA:

A. M1, M2, M3 y M4 encontrarán la solución óptima si existe.
B. M1 y M2 garantizan que encontrarán la solución óptima.
C. M4 encontrará la solución óptima.
D. M1 tendrá un coste espacial mayor que M4.

---

14) Dado el grafo de la figura, donde se señala el coste de los arcos, indica cuál de las siguientes afirmaciones es CORRECTA:



A. La aplicación de un algoritmo de Búsqueda en Anchura encontrará el camino A-D-F-G
B. La aplicación de un algoritmo de Coste Uniforme devolverá una solución de coste 5
C. Un algoritmo de Búsqueda en Anchura y Coste Uniforme encontrarán la misma solución
D. Ninguna de las anteriores.

---

15) Si se aplica un algoritmo de Profundización Iterativa sobre el grafo de la figura 10, ¿cuántas iteraciones serían necesarias hasta encontrar la solución?:

A. 2
B. 3
C. 4
D. Ninguna de las anteriores.

---

16) Sea una búsqueda de tipo A ($f(n)=g(n)+h(n)$) donde la función $h(n)$ es admisible y consistente. El algoritmo devuelve una solución desde el nodo inicial *A* al nodo objetivo *G* que atraviesa un nodo n1. Indica cuál de las siguientes afirmaciones es INCORRECTA:

A. $f(A) <= f(n1) <= f(G)$
B. $f(G)=h^*(A)$
C. $h^*(A) < h(n1)$
D. $f(G)=g(G)$.

---

17) Supongamos dos funciones de evaluación para un mismo problema $f1(n)=g(n)+h1(n)$ y $f2(n)=g(n)+h2(n)$ tales que $\forall n\ h1(n) \leq h2(n) \leq h^*(n)$. Dado un algoritmo de tipo A que utilice estas funciones, indica cuál de las siguientes afirmaciones es CORRECTA:

A. Solo una de las dos funciones encontrará la solución óptima.
B. El algoritmo que utilice $f1(n)$ expandirá menos nodos que el que utilice $f2(n)$
C. El algoritmo que utilice $f1(n)$ expandirá más nodos que el que utilice $f2(n)$
D. Ninguna de las dos funciones desarrollará una búsqueda completa.

---

18) Dados 3 algoritmos de búsqueda, M1 implementa una búsqueda de coste uniforme, M2 es algoritmo de tipo A con una heurística admisible y M3 implementa una búsqueda voraz, ¿cuál de las siguientes afirmaciones es INCORRECTA?:

A. M1 y M2 encontrarán la solución de coste óptimo
B. Se garantiza que M3 encontrará la solución más rápidamente que M1 y M2
C. No se puede garantizar que M3 encontrará la solución óptima
D. M1 expandirá más nodos que M2

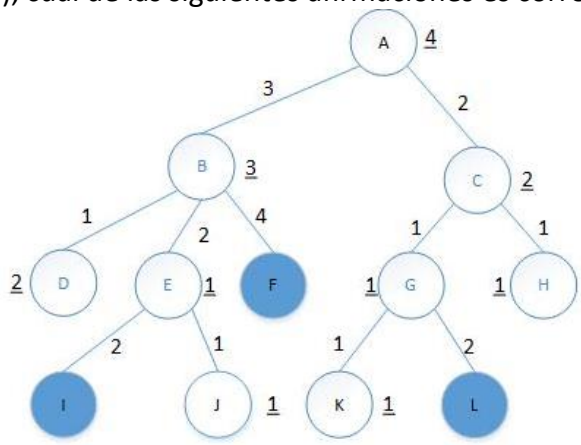
---

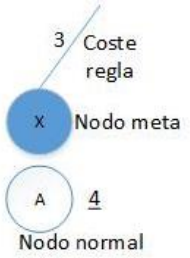
19) Sean dos funciones de evaluación $f1(n)=g(n)+h1(n)$ y $f2(n)=g(n)+h2(n)$, tales que $h1(n)$ es admisible y $h2(n)$ no lo es, indica la respuesta correcta:

A. El uso de ambas funciones en un algoritmo de tipo A garantiza encontrar la solución óptima
B. Se garantiza que $f2(n)$ generará un menor espacio de búsqueda que $f1(n)$
C. Sólo si $h1(n)$ es una heurística consistente, $f1(n)$ generará un menor espacio de búsqueda que $f2(n)$
D. Existe algún nodo n para el que $h2(n)>h^*(n)$

---

20) Para el espacio de estados de la figura y dada una búsqueda en anchura (expandiendo por la izquierda), cuál de las siguientes afirmaciones es correcta:





A. Devuelve el nodo I
B. Genera 8 nodos
C. Expande 4 nodos
D. Ninguna de las tres anteriores

---

21) Para el árbol de estados de la pregunta anterior, y suponiendo una búsqueda de tipo A ($f(n)=g(n)+h(n)$), ¿cuál de las siguientes afirmaciones es FALSA?:

A. Es admisible
B. Devuelve el nodo L
C. Expande 3 nodos
D. Genera 7 nodos

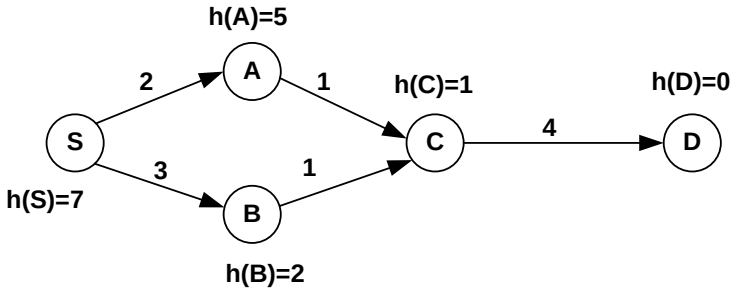
---

22) Asumiendo que todos los nodos de un espacio de búsqueda tienen más de un hijo ¿en cuál de las siguientes estrategias el orden de generación de los nodos nunca puede ser el mismo que el orden de expansión de los mismos?

A. Anchura
B. Coste uniforme
C. Profundidad
D. Búsqueda voraz

---

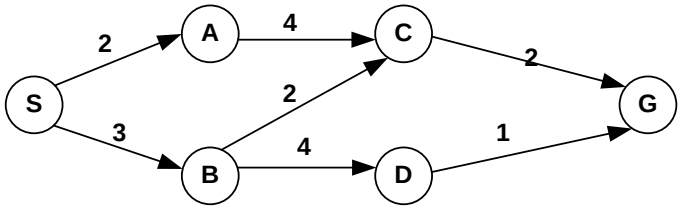
23) Dado el espacio de estados de la figura, donde S es el estado inicial, D el nodo meta, y se indican los costes de cada arco y la estimación $h(n)$ en cada nodo, marca la opción correcta:



A. La aplicación de un algoritmo A (tree search, con control de nodos repetidos en OPEN) no obtendrá la senda óptima.
B. La aplicación de un algoritmo A (graph search, con control de nodos repetidos en CLOSED, tal que un nuevo nodo se descarta si ya existe en CLOSED) obtendrá la senda óptima.
C. La respuesta A no es cierta, debido a que $h(n)$ no es admisible
D. La respuesta B no es cierta, debido a que $h(n)$ no es consistente

---

24) Dado el espacio de estados de la figura, el número de nodos que genera una búsqueda (Tree-Search) de coste uniforme donde, a igualdad de $f(n)$, se expande el nodo alfabéticamente menor es:



A. Mayor que si se realizase una búsqueda en anchura
B. Menor que si se realizase una búsqueda en anchura
C. Menor que si se realizase una búsqueda en profundidad
D. Ninguna de las anteriores es cierta

---

25) ¿Cuál de las siguientes afirmaciones es CORRECTA para una heurística consistente h(n) en una búsqueda de tipo f(n)=g(n)+h(n)?

A. No devuelve la solución óptima
B. El valor heurístico del padre puede ser igual al del hijo
C. Nunca genera un nodo 'n1' igual a otro nodo 'n2' ya generado y donde f(n1)<f(n2)
D. Nunca genera un nodo que ya esté en la lista CLOSED

---

26) Sea una búsqueda de tipo f(n)=g(n)+h(n) con h(n) admisible, dos nodos solución *G1* y *G2* donde *G1* es una solución óptima y *G2* no lo es y un nodo n1 que pertenece al camino solución de *G1*. Indica cuál es la afirmación INCORRECTA:

A. $g(G1) \leq f(G2)$
B. $f(n1) \leq g(G2)$
C. $h^*(n1)+g(n1)=f(G1)$
D. Ninguna de las anteriores

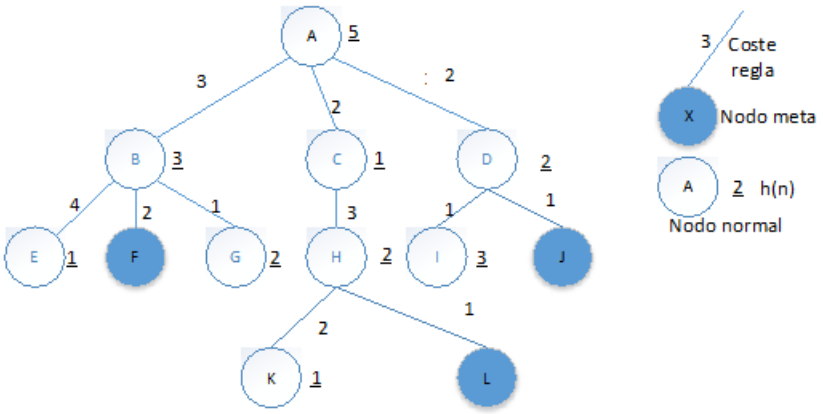
---

27) Respecto al número de nodos generados en el peor de los casos para una búsqueda en Profundización iterativa que encuentra la solución en el nivel d y una búsqueda en Profundidad limitada m=d para un mismo problema, ¿cuál de las siguientes afirmaciones es CORRECTA?

A. Profundidad limitada generará más nodos que Profundización iterativa
B. Profundización iterativa generará más nodos que Profundidad limitada
C. Las dos búsquedas generarán el mismo número de nodos
D. Ninguna de las anteriores

---

28) Para el espacio de estados de la figura y dada una búsqueda de tipo A (f(n)=g(n)+h(n)) ¿cuántos nodos es necesario generar, incluyendo el nodo raíz, para encontrar la solución?



A. 7
B. 8
C. 10
D. 12

---

29) Se desea realizar una búsqueda A* en CLIPS. Para ello, las reglas no deben contener la instrucción `retract` en la parte derecha porque:

A. Al borrar los hechos no podemos calcular el valor de g(n) necesario para una búsqueda A*.
B. No permitiría explorar caminos alternativos al elegido en primer lugar
C. No permitiría encontrar la solución óptima
D. Ninguna de las anteriores

---

30) Dado un algoritmo de búsqueda de tipo A, ($f(n)=g(n)+h(n)$), señala la afirmación **CORRECTA**:

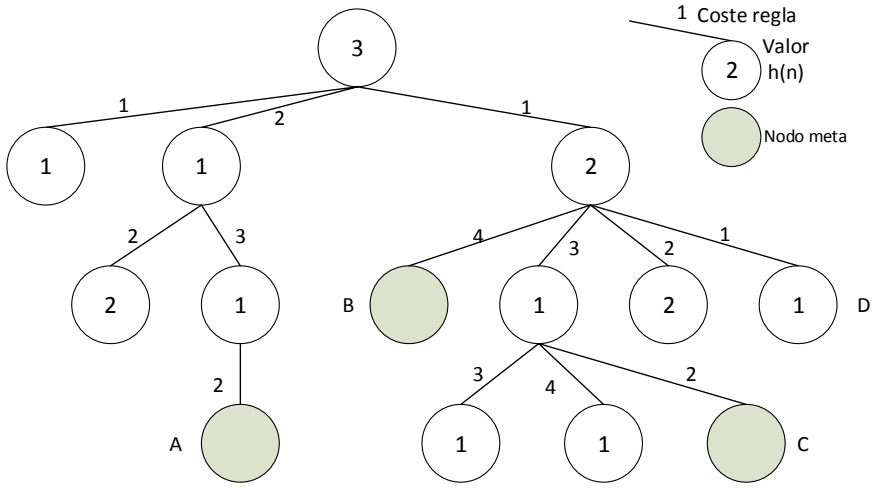
A. Si $h(n)$ es consistente (y admisible), expandirá siempre menos nodos que una búsqueda no informada
B. Con $h(n)$ consistente (y admisible), expandirá siempre menos nodos que no siendo consistente.
C. Encuentran siempre la misma solución, independientemente de si $h(n)$ es admisible o no.
D. Ninguna de las anteriores

---

31) Sea un problema de búsqueda en el que los costes de los operadores son distintos. La aplicación de un algoritmo GRAPH-SEARCH de Coste Uniforme con control de nodos repetidos devuelve una solución de coste 'c' en un nivel de profundidad 'd'. Indica cuál de las siguientes afirmaciones es CIERTA.

A. No es necesario especificar un nivel de profundidad máxima para el espacio de búsqueda con el fin de evitar que el algoritmo entre en un bucle infinito.
B. Una solución con coste c' tal que c' > c solo se encontrará en un nivel d' tal que d' > d.
C. Un algoritmo de Profundización Iterativa sobre el mismo problema devolverá siempre la solución óptima.
D. Un algoritmo de Anchura sobre el mismo problema devolverá siempre la solución óptima.

32) Si se aplica un algoritmo de tipo A en el espacio de estados de la figura siguiente, ¿qué nodo meta se elegirá en primer lugar como solución?



A. A
B. B
C. C
D. D

---

33) Dado el espacio de búsqueda de la figura anterior, una búsqueda A que utilice dichos valores, sería:

A. Admisible y consistente
B. Admisible y no consistente
C. No admisible pero si consistente
D. Ni admisible ni consistente

---

34) Sea un algoritmo A con una heurística $h(n)$. El número de nodos generados para la obtención de la solución ante un estado inicial concreto (indicar la respuesta CORRECTA):

A. Dependerá del factor efectivo de ramaje de la heurística $h(n)$ y de la profundidad de la solución óptima.
B. Dependerá del coste de aplicación de cada regla.
C. Si $h(n)$ es admisible, nunca será mayor que el número de nodos generados con una heurística $h'(n)$ tal que $h'(n)=h^*(n)$
D. Ninguna de las anteriores

---

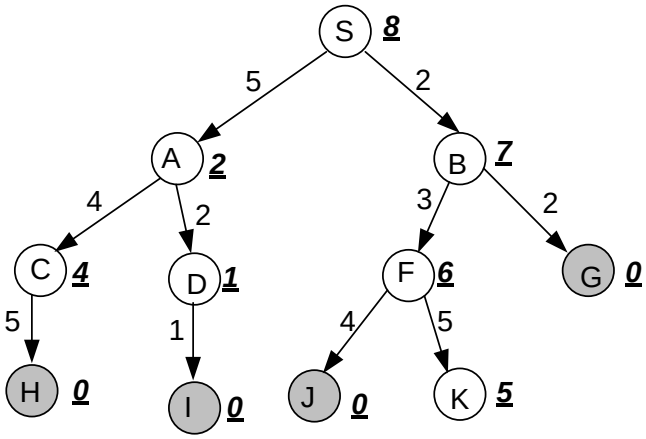
35) En la aplicación de una búsqueda A con método GRAPH-SEARCH, se ha encontrado el camino óptimo hasta cada nodo expandido si:

A. La función heurística es admisible.
B. Se efectúa un control de nodos repetidos en la lista OPEN.

C. La función heurística es consistente.
D. Ninguna de las anteriores.

---

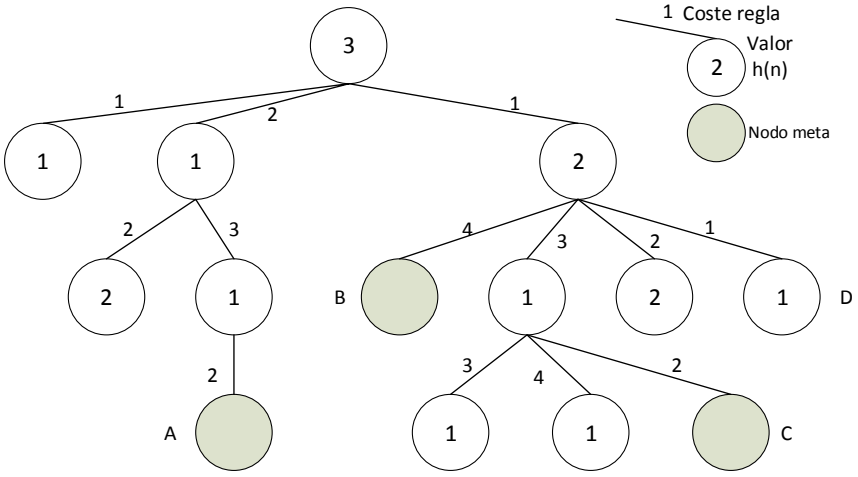
36) Dado el siguiente espacio de búsqueda, donde se indica el coste en las ramas y la estimación h(n) a la derecha de cada nodo, indica la respuesta CORRECTA.



A. Un algoritmo A generará menos nodos que una búsqueda de coste uniforme.
B. Un algoritmo A obtendrá la senda óptima.
C. Una expansión en anchura, empezando por la izquierda, generará igual o menor número de nodos que coste uniforme.
D. Ninguna de las anteriores es cierta.

---

37) Si se aplica una búsqueda voraz en el espacio de búsqueda de la figura, ¿qué nodo meta se elegirá en primer lugar como solución y cuántos nodos se generarán para encontrar dicha solución?



A. Nodo A y se generan 7 nodos
B. Nodo B y se generan 8 nodos
C. Nodo B y se generan 11 nodos
D. Nodo C y se generan 14 nodos

---

38) Dado el espacio de búsqueda de la figura anterior, indica la respuesta **INCORRECTA**:

A. La función h(n) es admisible
B. La función h(n) es consistente
C. Un algoritmo en anchura encontraría la misma solución que un algoritmo de tipo A
D. Un algoritmo en profundidad encontraría la misma solución que un algoritmo voraz

---

39) Sean tres niveles de un árbol de búsqueda para un problema, d1, d2 y d3, donde d1 < d2 < d3, tal que una solución se encuentra en el nivel d1, otra solución en el nivel d2 y otra solución en el nivel d3 (solo hay una solución en cada uno de los niveles). Indica la afirmación **CORRECTA**:

A. La complejidad temporal de un algoritmo de Anchura es $\mathbf{O(b^{d2})}$ y la de un algoritmo de Profundización Iterativa es $\mathbf{O(b^{d1})}$
B. La complejidad temporal de un algoritmo limitado en Profundidad, con máxima profundidad m=d1, es $\mathbf{O(b^{d1+1})}$
C. Asumiendo que se selecciona máxima profundidad m=d3, un algoritmo limitado en Profundidad siempre encontrará antes la solución del nivel d1 o d2.
D. Asumiendo que se selecciona máxima profundidad m=d1, la complejidad temporal de un algoritmo limitado en profundidad y un algoritmo de profundización iterativa es $\mathbf{O(b^{d1})}$

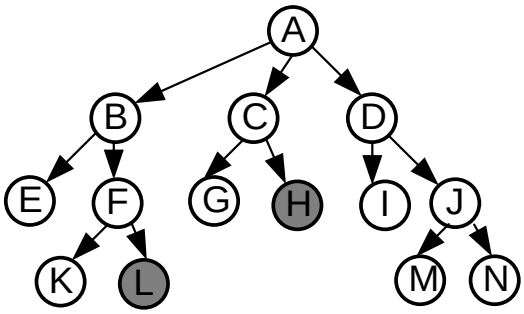
---

40) Sea la aplicación de un algoritmo A* para la resolución de un problema y sea *G* el nodo solución encontrado. Indica la sentencia que es **FALSA**:

A. Si h(n) es consistente entonces ∀n1, n2 tal que n2 es un hijo de n1 se cumple siempre h(n2)>=h(n1)
B. ∀n1, n2, tal que n1 y n2 son nodos del camino solución a *G*, se cumple siempre g(n1)+h*(n1)=g(n2)+h*(n2)
C. ∀n, tal que n es un nodo del camino solución a *G*, se cumple siempre f(n) <= g(G)
D. Se cumple siempre que f(G)=g(G)

---

41) Considerando el siguiente árbol de búsqueda, ¿en qué orden se generarían los nodos y qué nodo meta se encuentra mediante un procedimiento de búsqueda por profundidad iterativa?



A. ABCDEFKL y encuentra nodo meta L
B. ABCDEFGHIJKL y encuentra nodo meta H
C. AABCDABCDEFGH y encuentra nodo meta H
D. ABCDEFGH y encuentra nodo meta H

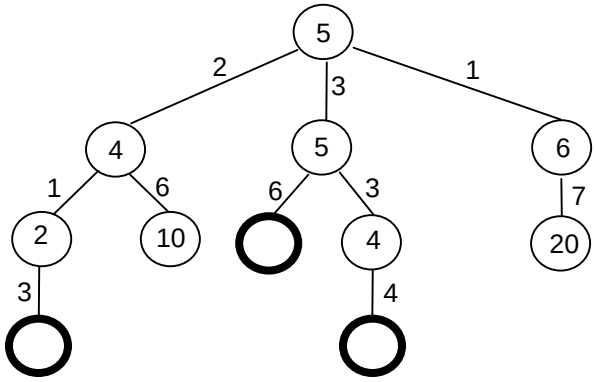
---

42) Dados dos algoritmos A* para un mismo problema, A1 con heurística $h1(n)$ y A2 con heurística $h2(n)$, tal que $\forall n$, $h^*(n) \geq h2(n) > h1(n)$.

A. Es seguro que A1 tardará menos que A2
B. Es seguro que A1 expandirá menos nodos que A2
C. La solución encontrada por A2 será mejor que la encontrada por A1
D. Ninguna de las anteriores es cierta

---

43) Sea el árbol de la figura donde los nodos de trazo grueso son nodos meta, el valor dentro del nodo es el valor de la función heurística aplicada a cada nodo y el valor de los arcos es el coste del operador correspondiente. Indica la respuesta **CORRECTA**:



A. La heurística es admisible y consistente
B. La heurística no es admisible ni consistente
C. Aplicando un algoritmo de tipo A se encuentra la solución óptima
D. Ninguna de las opciones anteriores es correcta

---

44) Sea un problema de búsqueda donde los operadores tiene distinto coste. Existe un nodo solución, $G1$, en el nivel $d1$ del árbol de búsqueda y un nodo solución, $G2$, que es óptimo y se encuentra en un nivel $d2$, tal que $d2 > d1$. Indica la respuesta **CORRECTA**:

A. La complejidad temporal de anchura respecto al número de nodos generados es $O(b^{d1})$
B. Una estrategia en profundidad nunca devolverá la solución $G1$
C. Una estrategia por profundidad iterativa nunca devolverá la solución $G1$.
D. Una estrategia de coste uniforme devolverá siempre la solución $G2$

---

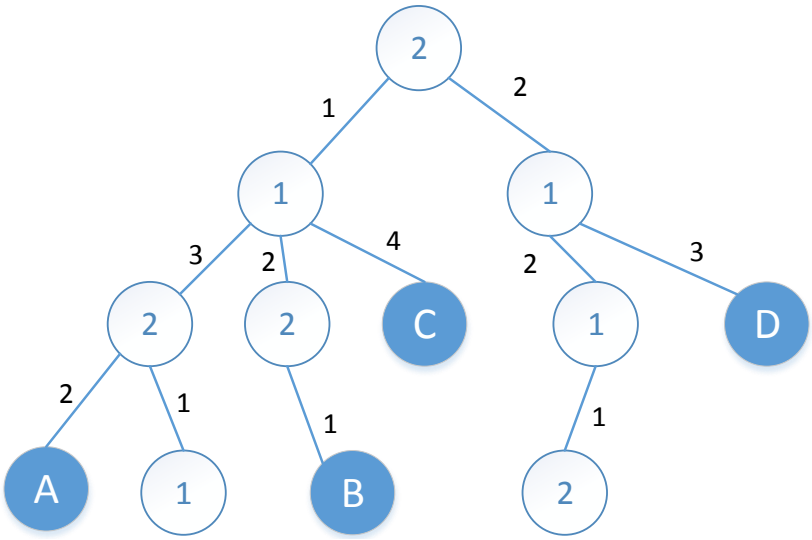
45) En el árbol de búsqueda que se genera con un algoritmo de tipo A* tenemos dos nodos, $n1$ y $n2$, que se corresponden con dos estados repetidos. Se sabe, además, que $n1$ es un nodo que

se encuentra en el camino óptimo a un nodo solución, *G*, mientras que n2 no está en el camino óptimo a *G*. Indica la respuesta **INCORRECTA**:

A. Se cumple siempre $f(n1) \leq f(G)$
B. Se cumple siempre $g(n1) < g(n2)$
C. Se cumple siempre $h(n1) < h(n2)$
D. Se cumple siempre $h(n2) \leq h^*(n2)$

---

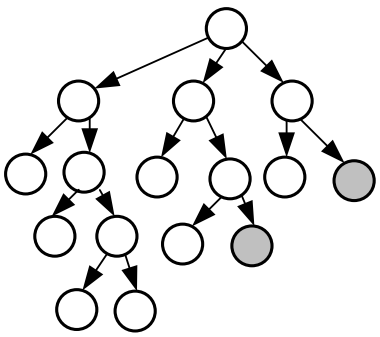
46) Sea el siguiente árbol de búsqueda, donde el valor dentro del nodo denota el valor de una heurística para dicho nodo, y el valor junto a una flecha el coste de dicho operador. Los nodos etiquetados como A, B, C y D son nodos meta. Si se realiza una búsqueda de tipo A, que solución se obtendría en primer lugar.



A. A
B. B
C. C
D. D

---

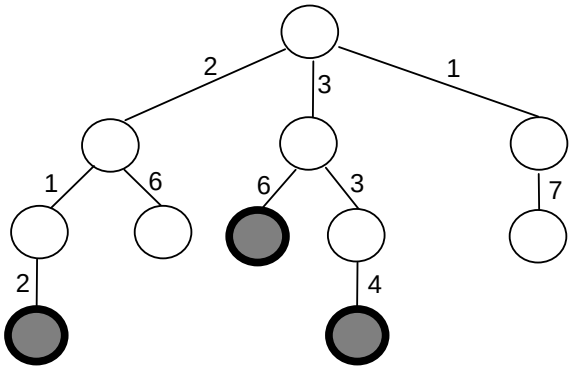
47) Considerando el siguiente árbol de búsqueda, ¿cuántos nodos como máximo se almacenan en memoria, aplicando un procedimiento de búsqueda en profundidad iterativa? (Asúmase que a igual profundad se elige el nodo más a la izquierda)



A. 6
B. 3
C. 4
D. 5

---

48) Dado el árbol de la figura, donde los nodos sombreados son nodos objetivo, indica la respuesta **CORRECTA**:



A. La aplicación de una estrategia en anchura devuelve la misma solución que coste uniforme.
B. La aplicación de una estrategia en anchura devuelve la misma solución que una estrategia de profundidad a nivel máximo de profundidad m=2.
C. La aplicación de una estrategia en anchura devuelve la misma solución que una estrategia de profundidad a nivel máximo de profundidad m=3.
D. La aplicación de una estrategia por coste uniforme devuelve la misma solución que profundización iterativa.

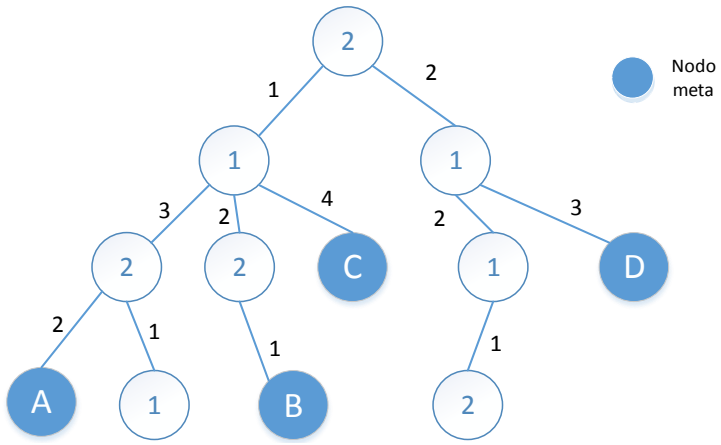
---

49) La aplicación de una heurística admisible, $h1$, a un problema devuelve un nodo solución $G1$ y el número de nodos que expande es $n1$. La aplicación de una heurística admisible, $h2$, al mismo problema, donde $h2$ domina a $h1$, devuelve un nodo solución $G2$ y expande un número de nodos igual a $n2$. Indica la respuesta **CORRECTA**:

A. Se cumple que $g(G1) < g(G2)$
B. Se cumple $h1(G1) < h2(G2)$
C. Se cumple que $n1 < n2$
D. Ninguna de las respuestas anteriores es correcta.

---

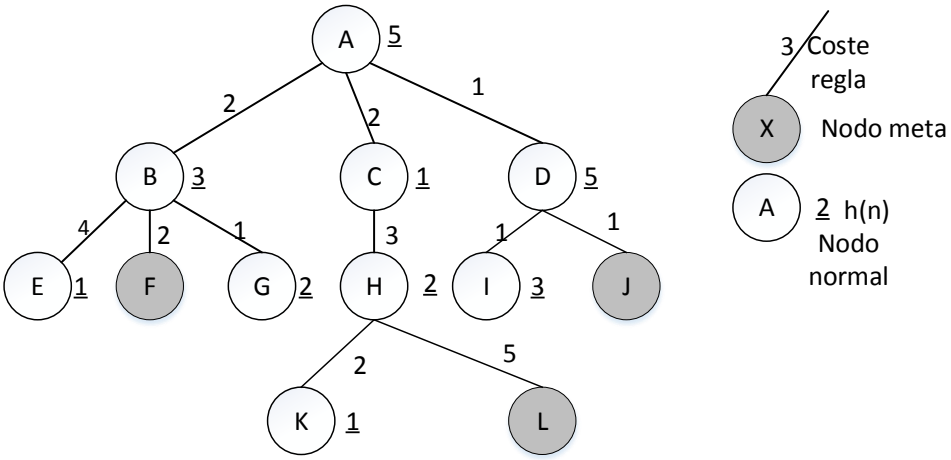
50) Dado el árbol de la siguiente figura, ¿cuántos nodos se generarían (incluyendo nodo inicial) si se aplicara un algoritmo A? (en caso de igualdad de $f(n)$, se expande el nodo más a la izquierda).



A. 6
B. 8
C. 9
D. 10

---

51) Para el espacio de estados de la figura y dada una búsqueda voraz, ¿cuál es el nodo meta que se elegirá en primer lugar como solución? (en caso de igualdad de f(n) se expande el nodo más a la izquierda)



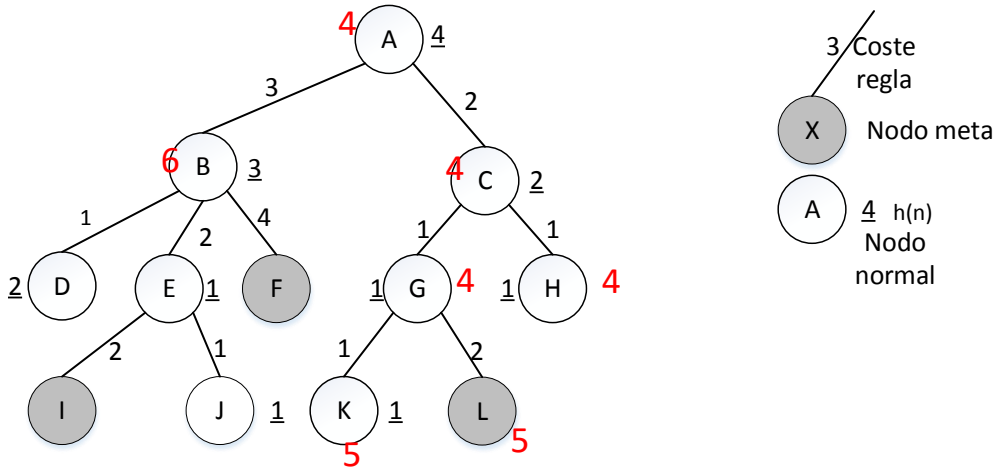
A. L
B. J
C. I
D. F

---

52) Sean dos funciones de evaluación $f1(n)=g(n)+h1(n)$ y $f2(n)=g(n)+h2(n)$, tales que $h1(n)$ es admisible y $h2(n)$ no lo es. Indica la respuesta **CORRECTA**:

A. El uso de ambas funciones en un algoritmo de tipo A garantiza encontrar la solución óptima
B. Sólo si $h1(n)$ es una heurística consistente, $f1(n)$ generará un menor espacio de búsqueda que $f2(n)$
C. Existe algún nodo n para el que $h2(n)>h^*(n)$
D. Se garantiza que $f2(n)$ generará un menor espacio de búsqueda que $f1(n)$

53) Para el espacio de estados de la figura y dada una búsqueda tipo A ($f(n)=g(n)+h(n)$), ¿cuál de las siguientes afirmaciones es **FALSA**:



A. Es admisible
B. Expande 3 nodos
C. Genera 7 nodos
D. Devuelve el nodo L

54) Dado un problema de búsqueda en el que todos sus operadores tienen el mismo coste, indica cuál de las siguientes afirmaciones es **CORRECTA**:

A. Un algoritmo de búsqueda con una heurística admisible devolverá la solución más corta
B. La estrategia en anchura devolverá la solución más corta pero no la solución de menor coste
C. La estrategia de coste uniforme devolverá la solución de menor coste pero no la solución más corta
D. Una estrategia en profundidad devolverá siempre la solución de menor coste

PROBLEMAS

Ejercicio 1

Dados los siguientes tipos de métodos de búsqueda en grafos:

M1) Tipo A*, con f(n)=g(n)+h1(n)
M2) Tipo A*, con f(n)=g(n)+h2(n) / h2(n) > h1(n)
M3) Tipo A, no A*

Responder, justificando la respuesta:

a) Determinar, si es posible, quién expandirá más nodos:
   - Método M1 o el método M2?
   - Método M1 o el método M3?

b) Determinar, si es posible, que solución será mejor:
   - La que obtenga M1 o la que obtenga M2?
   - La que obtenga M1 o la que obtenga M3?

c) Sobre el coste total del proceso de búsqueda, compárase:
   - El del método M1 con el del método M2
   - El del método M1 con el del método M3

Solución:

a)

a.1) M1 porque la función heurística está menos informada y por tanto se generará un árbol con más nodos, más similar a una búsqueda en anchura.
a.2) No se puede saber ya que se está comparando un algoritmo A* y uno de tipo A que no es admisible. M3 generará un árbol de búsqueda más estrecho que M1, pero se desconoce la profundidad de la solución encontrada; además, M3 no garantiza admisibilidad ni completitud.

b)

b.1) Igual, ambos son A*.
b.2) M1 obtiene la óptima, M3 no lo garantiza. En el caso particular que M3 tuviera una h(n) acotada ($h(n) \leq h^*(n)+e$), tendría la garantía de encontrar una solución acotada por (Coste-Óptimo + e)

c)

c.1) Se generan más nodos con M1 pero por el contrario h1(n) es más fácil de calcular que h(2). Dependerá por tanto del coste del cálculo de las funciones heurísticas.
c.2) No se puede saber. h(3) está muy informada, por ser un algoritmo de tipo A y no ser admisible, y por tanto costará más de calcular que h1(n). Por otra parte, no se conoce la relación exacta entre los nodos expandidos por M1 y M3 (ver a.2)

Ejercicio 2

Responder a cada una de las siguientes cuestiones. Justifíquese la respuesta.

a) Dadas las siguientes funciones de evaluación:
f1(n) =g(n) + h1(n)
f2(n) = g(n) + h2(n)
f3(n) = g(n) + h3(n)
f4(n) = g(n) (Búsqueda de Coste Uniforme)
y conociendo que $h1(n) \leq h2(n) \leq h^*(n) \leq h3(n)$,

a.1) Ordenar de mejor a peor las anteriores funciones, siendo el criterio principal la admisibilidad y el secundario la generación de nodos.

a.2) Ordenar de mejor a peor, considerando en primer lugar la generación de nodos y en segundo lugar la admisibilidad.

b) Suponiendo una función f5(n) = h2(n), ¿se puede asegurar que garantiza que encuentra la solución óptima con un número de nodos menor que f2(n)?

Solución:

a.1) Por admisibilidad se agruparían primero f1, f2 y f4 y después f3 (que no es admisible). Dentro de las admisibles, de mejor a peor por expansión de nodos, sería f2, f1 y f4. El orden total queda por tanto f2, f1, f4 y f3.

a.2) No se puede establecer un orden en cuanto a la generación de nodos. Sabemos que existe un orden de menor a mayor número de nodos generados: f2, f1, f4. Pero no sabemos donde estaría f3. Aunque genera un árbol de búsqueda más estrecho que los otros, no tiene la garantía de completitud.

b) En este caso sería un algoritmo búsqueda voraz, donde no se aplica factor de coste, y por tanto no se garantiza la obtención de la solución óptima. De hecho, existe bastante más probabilidad de no encontrar la óptima y encontrar otra solución ya que el proceso de búsqueda va sólo guiado por la función heurística.

Ejercicio 3

Dado un algoritmo admisible A* con f(n)=g(n) + h(n), siendo el coste de aplicación de cada regla 1, si cambiara g(n) y el coste de cada regla fuera variable, ¿Seguiría siendo admisible?

Solución

Si siendo el coste de cada operador 1 se cumple que h(n) <= h*(n), entonces para que se siga cumpliendo esta relación el coste de los operadores debería ser >=1, ya que sino no hay garantía de que siga siendo menor la estimación. Dicho de otro modo, si cambia el coste de los operadores, cambia entonces h*(n) (coste real de la solución), y para asegurar que se sigue cumpliendo h(n) <= h*(n), h*(n) debe crecer, y eso sólo sucede si el coste de los operadores es mayor o igual que el coste anterior.

Ejercicio 4

Suponer que tenemos dos algoritmos A*, con heurísticas h1(n) y h2(n) respectivamente para un mismo problema.

a) Suponiendo $\forall$n h1(n) <= h2(n), ¿cuál de los dos algoritmos encontrará la mejor solución (coste óptimo)? ¿Cuál generará un menor espacio de búsqueda?
b) Suponiendo que h1(n) da lugar a un A* monótono y h2(n) no, ¿cuál de los dos algoritmos tendrá un factor de ramificación menor?

Solución:

a) Ambos encontrarán la mejor solución porque son los dos A*. Y expandirá menos nodos el algoritmo de h2 ya que la función está más informada.

b) El algoritmo de h1 tendrá un factor de ramificación menor. La razón es que si A* es monótono se garantiza que cuando se expande un nodo se cumple g(n)=g*(n), o sea, que ese es el camino óptimo desde el nodo raíz hasta n (dicho de otro modo, si se encuentra otro nodo igual a n, no se expandirá porque ya se ha expandido el nodo de menor coste). Sin embargo, si el algoritmo no es A* monótono no se garantiza esta propiedad, por lo que podría volver a expandirse un nodo ya expandido para el cual se ha encontrado un valor de g(n) menor. Por tanto, como se pueden generar más nodos con un A* no monótono, el factor de ramificación también será mayor.

Ejercicio 5

Dadas dos heurísticas admisibles (h1,h2), cuál de las siguientes funciones heurísticas composición de las dos anteriores es admisible y generará menos nodos en el proceso de búsqueda: (i) max (h1,h2), (ii) min (h1,h2), (iii) h1+h2.

Solución:

La heurística h1+h2 no hay garantías de que sea admisible y será la que genere menos nodos porque está mucho más informada. Las otras dos heurísticas seguirán siendo admisibles y max (h1,h2) generará menos nodos ya que devuelve la función más informada entre h1 y h2. Ordenando las funciones de menor a mayor número de nodos generados tenemos: h1+h2, max (h1, h2), min (h1, h2).

Ejercicio 6

Dados los siguientes Algoritmos A de búsqueda en grafos, donde h*(n) representa el coste del camino optimo a meta:

1) f1(n) = g(n) + h1(n) / h1(n)≤h*(n)
2) f2(n) = g(n) + h2(n) / h2(n)>h1(n)
3) f3(n) = g(n) + h3(n) / h1(n) ≤ h3(n) ≤ (1+a) h*(n), siendo a una constante positiva.
4) f4(n) = g(n) + h4(n) / h4(n) ≤h1(n)

Responder muy brevemente (1-2 líneas por cuestión), justificando la respuesta:

a) Cuáles de los anteriores métodos es admisible?
b) Quién tendrá mayor factor de penetrabilidad, f1 o f4?
c) Qué podemos esperar sobre el coste de la solución que obtenga f3(n)? Y de la que obtenga f2?
d) Ante un problema muy complejo, qué método sería más adecuado, en cada uno de estos casos:
    d.1) Se requiere una respuesta en un tiempo muy corto, siendo esto prioritario al coste de la misma o, incluso, al poder encontrarla.
    d.2) Se requiere una respuesta en un tiempo corto, pero con un buen coste.
    d.3) Se requiere una respuesta de coste óptimo.
e) Para un problema, en general, es posible conocer:
    e.1) El valor de h*(n)?
    e.2) Si una función heurística h(n) cumple h(n) ≤h*(n)?
    e.3) Si una función h(n) cumple la condición de monotonía?

Solución:

a) f1 y f4 son ambas admisibles por tener un h(n)<= h*(n). De las otras dos funciones no se puede garantizar la admisibilidad.
b) f1 ya que está más informada que f4 y por tanto generará menos nodos, por lo que el factor de penetrabilidad será mayor.
c) El coste de la solución que obtenga f3 estará acotado respecto al coste de la solución óptima por un factor (1+a). Sobre la solución que obtenga f2 no puede decirse nada, puede ser la óptima, una solución no óptima o no encontrar la solución.
d) d.1 – Mejor f2, ya que puede estar sobreinformada y obtener soluciones muy rápidas
   d.2 - Mejor f3, está sobreinformada pero acotada por un factor (1+a)
   d.3 - Mejor f1, ya que es admisible y está más informada que f4

e) e.1- No, no es posible conocer el coste real de la solución porque en dicho caso se conocería la solución. h(n) es una estimación de h*(n).
   e.2- Se puede hacer un análisis general del problema y comprobar si la heurística es admisible. Además si se encuentra un contraejemplo en el que para un nodo h(n) > h*(n), entonces ya se puede decir que no es admisible. También sería posible si se conociera una cota inferior del coste de la solución óptima y h(n) ser inferior a dicha cota.
   e.3- En algunos casos sí, si se puede comprobar que h(n1) <= h(n2) + coste (n1,n2)

Ejercicio 7

Para la resolución de un problema mediante un proceso de búsqueda en un espacio de estados se han empleado tres operadores O1, O2 y O3 de coste 10, 20 y 30 respectivamente. El programa solicita al usuario un nivel máximo de profundidad de expansión del árbol, y devuelve el nivel dónde ha encontrado la solución, el número de nodos

generados y el coste de la solución. Se sabe que para un nivel máximo de profundidad D=5 y una estrategia de búsqueda en anchura, la solución se ha encontrado en el nivel 2 y el coste de la solución es de 50 (en el nivel 2 sólo existe una solución). De acuerdo a esta información

a) ¿Es esta solución encontrada la solución óptima de menor coste? Razona la respuesta. En caso negativo, ¿qué estrategia utilizarías para encontrar la solución óptima?
b) Si el usuario selecciona D=8 y estrategia de búsqueda en anchura, ¿cómo será la solución encontrada? Compárala con la solución que se presenta en el enunciado.
c) Si el usuario selecciona D=8 y estrategia de búsqueda por profundización iterativa, ¿cómo será la solución encontrada? Compárala con la solución que se presenta en el enunciado.

Solución:

a) No, no se puede garantizar que sea la solución de menor coste ya que la estrategia de búsqueda en anchura sólo es admisible si todos los operadores tienen el mismo coste. En este caso, se puede decir que la solución es la de menor longitud pero no necesariamente la de menor coste.
   En el caso de utilizar una estrategia no informada utilizaría BCU, ya que es admisible; en caso de una estrategia con información, utilizaría una búsqueda A*.

b) Será la misma solución ya que la estrategia de búsqueda en anchura encuentra la solución de menor profundidad en el árbol y por tanto se parará cuando encuentre la solución en el nivel 2.

c) También será la misma porque la estrategia de búsqueda por profundización iterativa parará cuando, desarrollando el árbol de nivel 2, encuentre la solución que está en dicho nivel.

Ejercicio 8

Para un determinado problema de búsqueda, se ejecutan las siguientes estrategias: anchura, profundidad, un algoritmo A* con h1(n), otro algoritmo A* con h2(n) > h1(n) y un algoritmo no A*. El usuario limita el nivel de profundidad en la generación del árbol, y se obtienen los resultados que se muestran en la tabla. Explica razonadamente a qué estrategia de búsqueda corresponde cada línea de la tabla.

Nivel usuario	Nivel solución	Num. nodos
15	13	264
18	13	860
18	18	565
20	13	325
25	20	205

Respuesta:

Claramente la solución óptima está en el nivel 13. Hay por tanto tres estrategias que devuelven la sol. óptima y que deberán corresponderse con anchura y los dos algoritmos A*. Atendiendo al número de nodos tenemos:

Fila 2: anchura (mayor número de nodos, información heurística nula)
Fila 4: algoritmo A* con h1(n) (325 nodos, menos informada que h2(n))
Fila 1: algoritmo A* con h2(n) (264 nodos, más informada que h1(n))

Fila 3: podría corresponderse con profundidad o el algoritmo no A*. Atendiendo al número de nodos, es más probable que sea la estrategia profundidad ya que se trata de un número elevado de nodos.

Fila 5: podría corresponderse con profundidad o el algoritmo no A*. Atendiendo al número de nodos, es más probable que sea el algoritmo no A* ya que no encuentra la solución óptima y el número de nodos generados es inferior al del mejor algoritmo A*, lo que indica que la función heurística está sobre informada.

Ejercicio 9

Tenemos un problema de búsqueda cuyos operadores tienen distinto coste. Si aplicamos la estrategia BPI (Búsqueda por Profundización Iterativa) para resolver el problema, ¿qué características generales tendrá la solución encontrada?. Determina si BPI encontrará solución y en caso que así sea define la solución en términos de optimalidad, coste temporal, espacial, etc.

Respuesta:

Sí, BPI encuentra solución ya que se trata de una estrategia completa. BPI es una estrategia que obtiene las mismas soluciones que anchura pero con un coste espacial equivalente al de la estrategia de profundidad. Por tanto no se puede garantizar que la solución encontrada sea la óptima ya que los operadores tienen diferente coste; al igual que la estrategia de anchura, se puede garantizar que la solución encontrada será la más corta pero no necesariamente la óptima. El coste temporal será igual que el de la estrategia de anchura (b^d) y el espacial igual que el de la estrategia de profundidad (b.d) donde *b* es el factor de ramificación y *d* el nivel máximo de profundidad alcanzado.

Ejercicio 10

La siguiente configuración del 8-puzzle:

2	3	4
1	6	5
8	7	

la resolvemos con un algoritmo de búsqueda que emplea la función de evaluación f(n)=g(n)+Distancias-Mahattan (n), y con otro que emplea la función de evaluación f(n)=g(n)+Piezas descolocadas(n). ¿Qué proceso de búsqueda generará menos nodos para dicha configuración? ¿Porqué?

Respuesta:

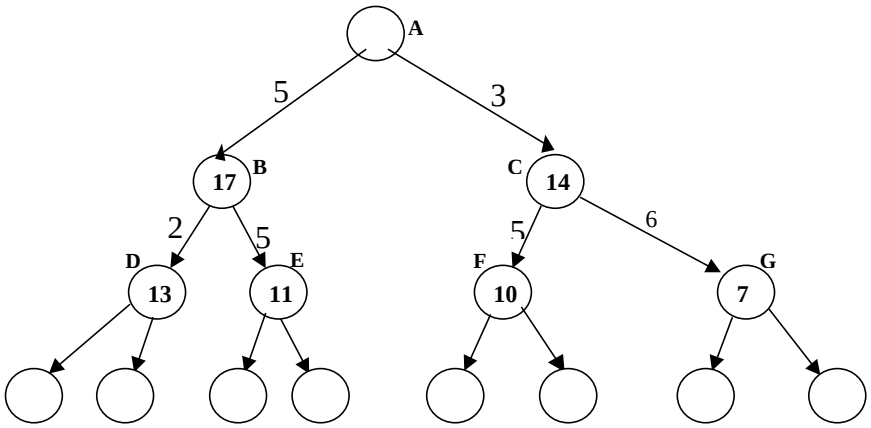
Ambas estrategias generarán el mismo número de nodos ya que ambas devuelven el mismo valor heurístico. Concretamente, los resultados que se obtendrán con ambas estrategias de búsqueda son IDÉNTICOS.

La razón es que para esta configuración en particular el número de piezas descolocadas coincide con el valor que devuelve la función distancias, ya que todas las piezas están a una distancia de 1 de su posición objetivo; y al desarrollar los árboles de búsqueda en ambas estrategias se puede observar que el nodo que se expande siempre es el mismo, un nodo para el cual el valor de piezas-descolocadas es el mismo que distancias. En otras palabras, el valor que miniminiza h(n) a lo largo del desarrollo de los dos árboles de búsqueda coincide exactamente en el mismo nodo.

Ejercicio 11

Dado el espacio de búsqueda de la figura, ***listar el orden*** en que serán expandidos los nodos (A, B, C, D, E, F y G), los nodos terminales no se expanden, siguiendo las siguientes estrategias de búsqueda (eligiendo siempre el nodo de más a la izquierda). Los valores en las ramas es g(n). En los nodos es h(n).

- **a)** Primero en profundidad
- **b)** Primero en anchura
- **c)** Profundización iterativa (aumento en 1 la profundidad límite de cada iteración)
- **d)** Coste uniforme
- **e)** Búsqueda voraz
- **f)** Algoritmo A



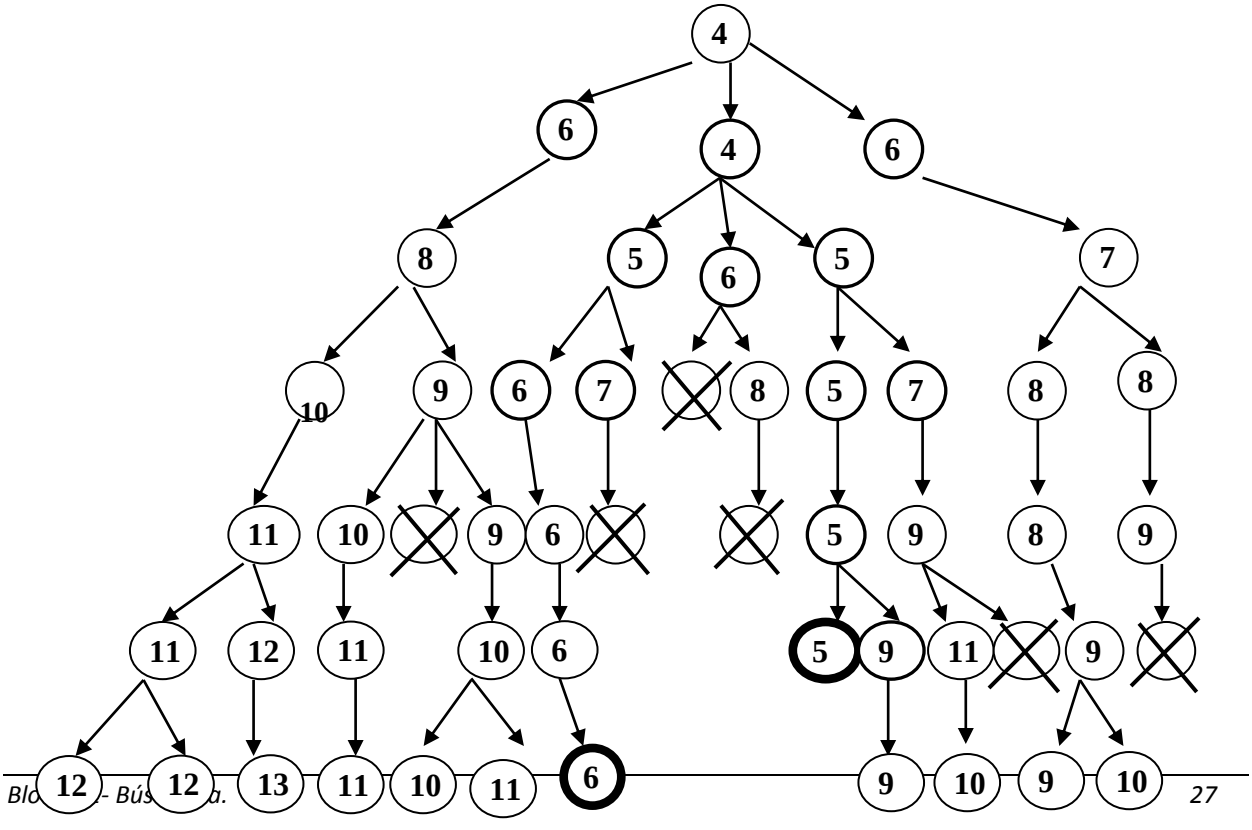
Primero en profundidad {A, B, D, E, C, F, G}
Primero en anchura {A, B, C, D, E, F, G}
Profundización iterativa (aumento en 1 la profundidad límite de cada iteración) {A, A, B, C, A, B, D, E, C, F, G}
Coste uniforme {A, C, B, D, F, G, E}
Búsqueda voraz {A, C, G, F, B, E, D}
Algoritmo A {A, C, G, F, B, D, E}

Ejercicio 12

Sea el siguiente árbol que representa un proceso de búsqueda en un espacio de estados. Los valores de cada nodo son el resultado de aplicar una función f(n)=g(n)+h(n), h(n) <= h*(n) (heurística admisible ó A*), y el coste de aplicación de cada operador es 1. Los nodos que aparecen en negrita son dos posibles soluciones al problema, y los nodos que aparecen pintados con una aspa indican que no se generan nodos sucesores. Contesta a las siguientes preguntas:

a) si se realiza una búsqueda heurística aplicando la función f(n) mencionada anteriormente, ¿qué solución encontraría el proceso de búsqueda? ¿cuál es el coste de dicha solución? ¿cuántos nodos se generarían para encontrar dicha solución?

b) a la vista de los valores f(n) de cada nodo, ¿es el algoritmo aplicado un A* monótono? ¿Porqué? Justifica brevemente la respuesta.



Respuesta:

a) La solución que encontraría el proceso de búsqueda es la del nodo 5. Dado que el coste de cada operador es 1, el coste de dicha solución sería 5. El número de nodos generados sería 13 como máximo (incluyendo nodo meta pero no el inicial; los nodos son los señalados en negrita en el árbol). Como el nodo 4 del nivel 1 tiene dos sucesores con el mismo valor de f(n)=5, podría suceder que se expandiera antes el nodo sucesor de la derecha, en cuyo caso el número de nodos totales generados sería 11.

b) Todos los nodos del árbol satisfacen la condición h(n1) <= h(n2) + c(n1,n2) excepto el nodo 11 de nivel 6 que tiene un nodo sucesor con un valor de 10. En este caso, nodo11= g(n)+h(n)=5+6, y el nodo sucesor, nodo10=g(n)+h(n)=6+4, por tanto no se satisface 6 <= 4+1. Esto también puede observarse en que el valor f(n) en estos dos nodos no es monotónicamente no decreciente, ya que el valor de f(n) decrece. Por tanto, no se puede afirmar que sea A* monótono.

Ejercicio 13

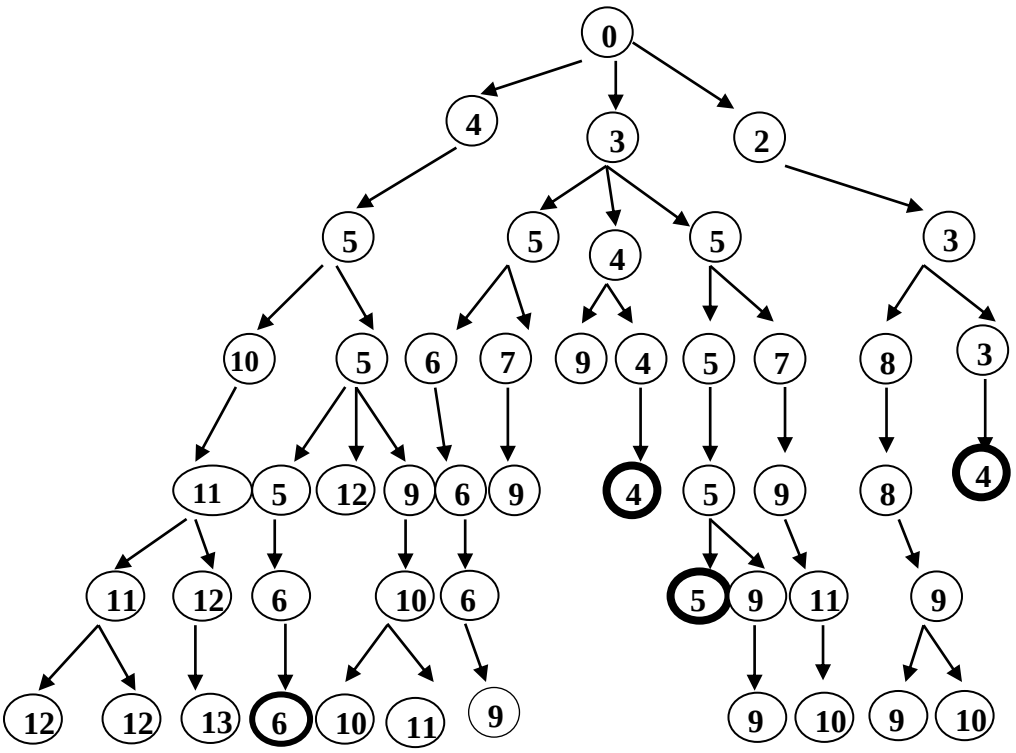
Sea el siguiente árbol que representa el espacio de soluciones de un problema determinado. Los valores de cada nodo son el resultado de aplicar una función f(n)=g(n)+h(n), donde h(n) es una función heurística que da lugar a una búsqueda de tipo A* ($h(n) \leq h^*(n)$). El coste de aplicación de cada operador es uniforme (1). Los nodos que aparecen en negrita son estados solución del problema. Contesta a las siguientes preguntas:

a) si se realiza una búsqueda heurística aplicando la función f(n) anterior, ¿qué solución encontraría el proceso de búsqueda?, ¿cuántos nodos se generarían para encontrar dicha solución?

b) Si sobre dicho espacio de soluciones se realiza una búsqueda por profundización iterativa ¿qué solución encontraría el proceso de búsqueda?, ¿son iguales las dos soluciones, en caso contrario cual es mejor?

c) ¿Qué solución se encontraría si se realiza una búsqueda en profundidad?

d) Utilizando una función de evaluación f1(n)= g(n)+h1(n), tal que $h^*(n) \geq h1(n) > h(n)$, ¿la senda solución que se obtenga con f1(n), será mejor/peor/igual que la que se obtenía con f(n)?, ¿qué función de evaluación realizará más/menos/igual búsqueda?

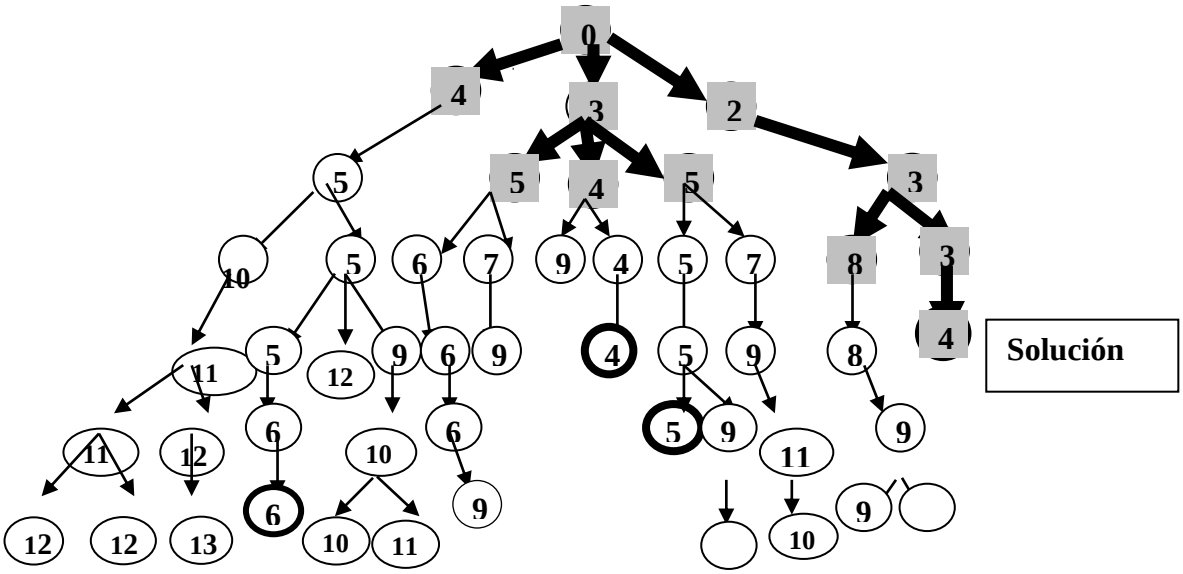


Respuesta:

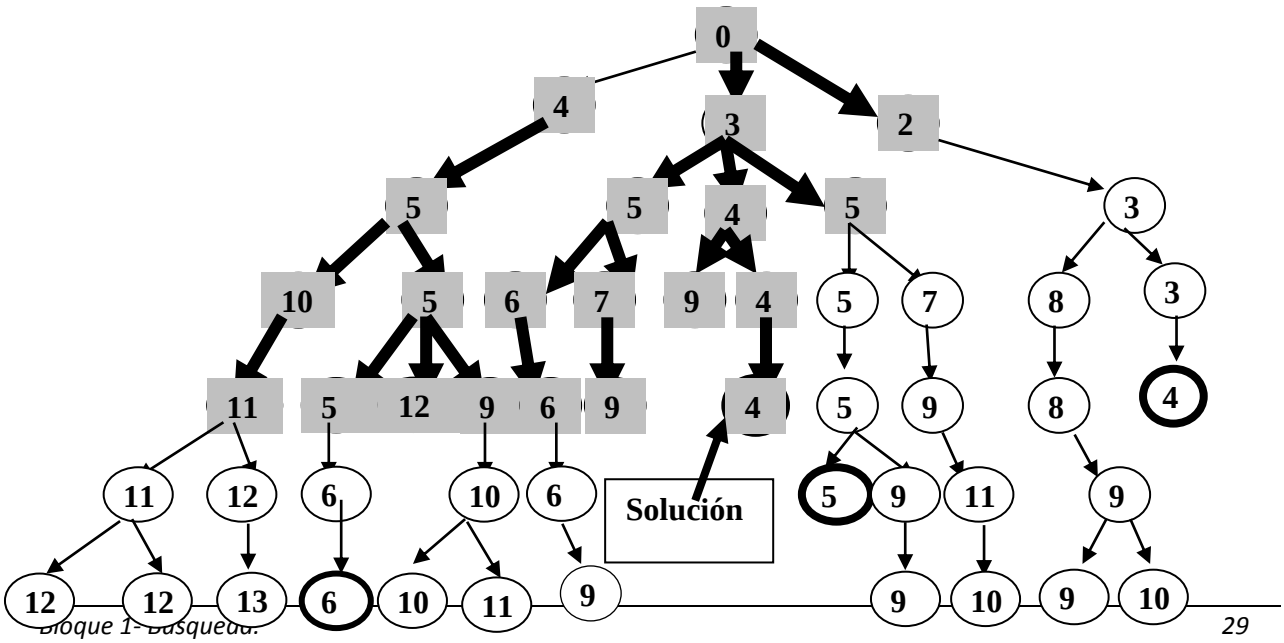
a) En trazo grueso se indica el espacio de búsqueda generado. En caso de un mismo valor de la función de evaluación, asumiremos que se expande en primer lugar el nodo más reciente, o sea el nodo más profundo.

OPEN={2,3,4} se saca el nodo de valor 2 de la lista; no es solución y se expande
OPEN={3 (nivel2),3 (nivel1),4} se saca el nodo de valor 3 de nivel 2 de la lista; no es solución y se expande
OPEN={3 (nivel3), 3 (nivel1), 4, 8} se saca el nodo de valor 3 de nivel 3 de la lista; no es solución y se expande
OPEN ={3 (nivel1), 4 (nivel4), 4 (nivel1), 8} se saca el nodo el nodo de valor 3 de nivel 1 de la lista; no es solución y se expande.
OPEN= {4 (nivel4), 4 (nivel2), 5 (nivel2), 5 (nivel2), 8} se saca el nodo de valor 4 de nivel 4 de la lista; es solución y el algoritmo termina.

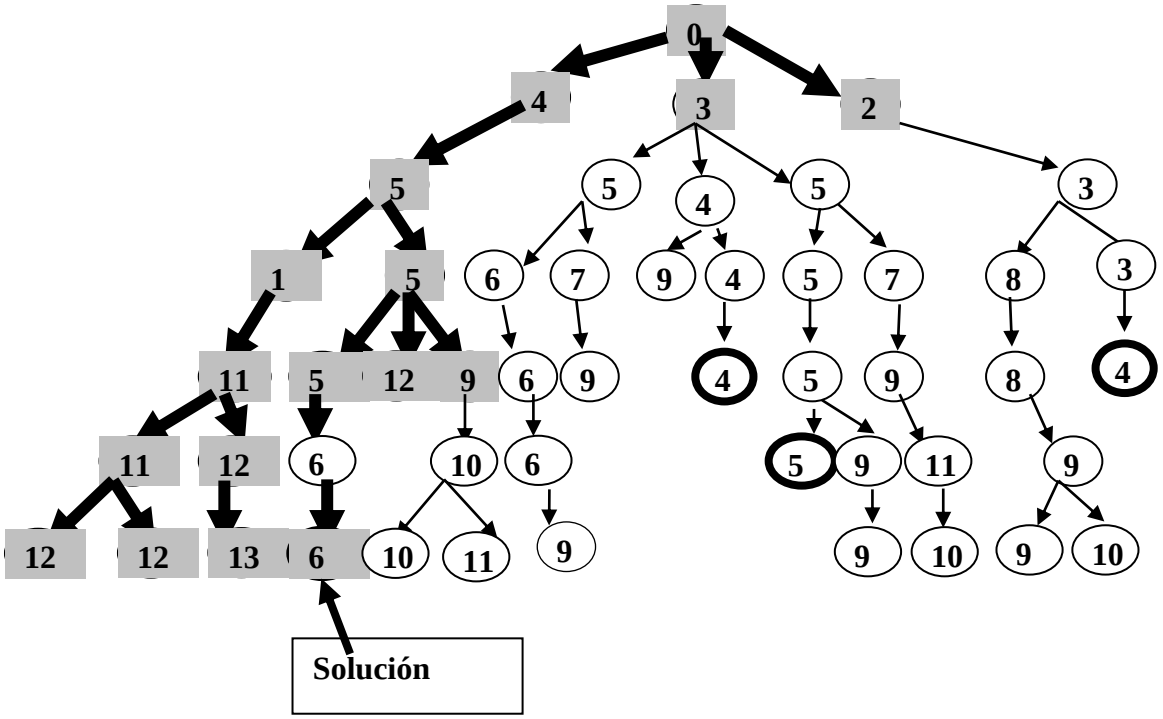
En total se han generado 10 nodos (incluyendo nodo meta pero no el inicial) que equivale a 10 nodos expandidos.



b) En el caso de realizar una búsqueda por profundización iterativa, se generarían diversos árboles de búsqueda, para profundidad 1, 2, y 3 que irán explorando exhaustivamente el espacio de soluciones, a dichas profundidades, ya que en estas profundidades no se encuentra ninguna solución. Será en la siguiente iteración, 4, donde se encuentre la solución que se indica en la siguiente figura donde se representa, mediante trazo grueso y marcando los nodos, el espacio de búsqueda en profundidad a dicha profundidad.



c) En la figura está representado el espacio de búsqueda que se generará en este caso. Como se puede observar el coste de la solución es mayor que en los casos anteriores, como ya sabemos esto puede suceder ya que, a diferencia de las estrategias de los apartados anteriores, la búsqueda primero en profundidad no es admisible.

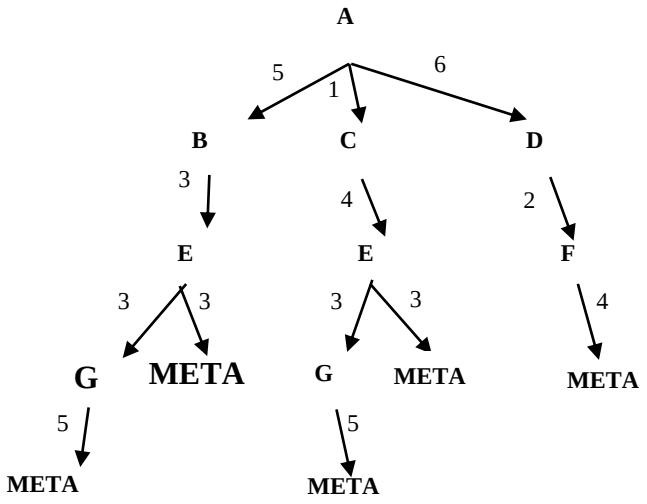


d) Como en ambos casos tendremos una búsqueda A*, que como ya sabemos es admisible, la solución encontrada será la mejor solución. Lo que si sucederá es que la función más informada, h1(n), producirá potencialmente más poda por lo tanto el espacio de búsqueda generado por f1(n) será menor.

Ejercicio 14

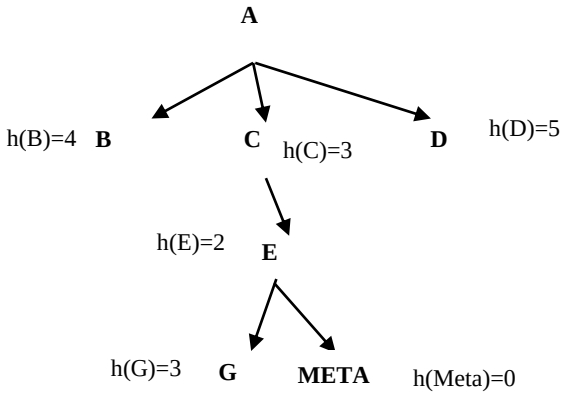
Dado el siguiente árbol de búsqueda (donde el valor del coste de cada paso se indica en las ramas) y donde los valores h(n) de cada nodo son: h(A)= 5, h(B)=4, h(C)=3, h(D)=5, h(E)=2, h(F)=1, h(G)=3, establecer la búsqueda que se realizaría mediante:

1) Una búsqueda voraz (guiada por h(n)
2) una búsqueda A
3) en base a los valores indicados, ¿es una búsqueda A*?

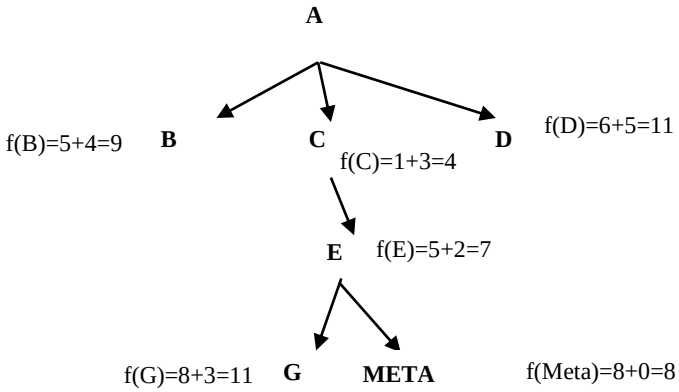


La búsqueda voraz se guía únicamente por el valor heurístico. f(n)=h(n). De acuerdo a esta función de evaluación, la expansión sería del siguiente modo:

OPEN={C,B,D} por orden creciente de h(n). Se saca C de la lista; no es solución y se expande
OPEN={E,B,D} Se saca E de la lista; no es solución y se expande
OPEN={Meta, G, B, D} se ordena primero Meta porque h(Meta)=0



1) Aplicando una búsqueda A, f(n)=g(n)+h(n)



2) En primer lugar, la búsqueda anterior devuelve la solución óptima (hay varios caminos solución pero el camino de menor coste es el encontrado, coste = 8). En caso de no devolver la solución óptima, podríamos afirmar que no sería un algoritmo A*. Sin embargo, el hecho que devuelva la solución óptima no se garantía de que sea A*; para afirmarlo, hay que mirar los valores h(n) de cada uno de los nodos.

   Se puede observar que los valores h(n) de cada uno de los nodos son siempre menores o iguales que el coste real hasta la solución. Por ejemplo, h(B)=4 y h*(B)=6, h(C)=3 y h*(C)=7, h(D)=5 y h*(D)=6, etc.

Ejercicio 15

Responder brevemente a las siguientes cuestiones:

a) Considerar un espacio de estados donde el estado inicial es el número 1 y la función sucesor (operador) para cualquier estado 'n' devuelve dos estados 2n y (2n+1).

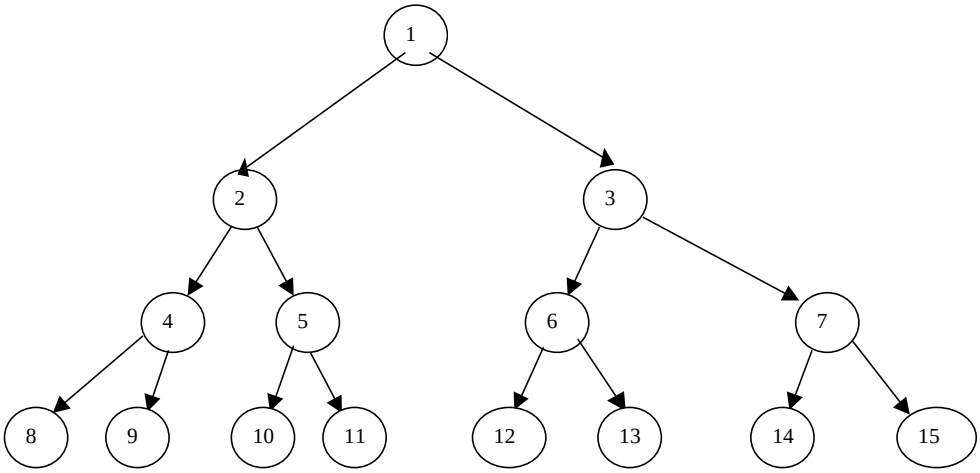
1. Dibujar el espacio de búsqueda, solo para los estados del 1 al 15.
2. Suponiendo que el estado objetivo es el 11. Representar el espacio de estados, y enumerar el orden en que se expanden los nodos, en el caso en que se desarrolle una búsqueda primero en anchura y la búsqueda de profundización iterativa.

b) Dada la siguiente función de evaluación f(n)= (2-a).g(n)+h(n) tal que h(n) ≤ h*,

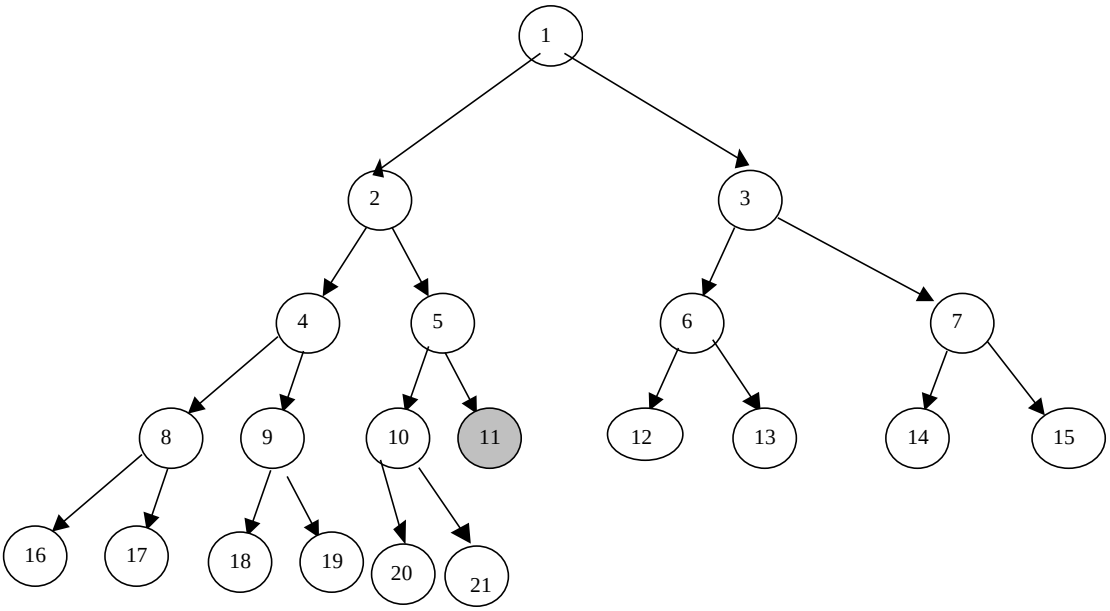
- ¿Para que valores de ‘a’ está garantizado que se encuentra la solución óptima?.
- ¿Para a=2 se encontrará la solución óptima?.

Solución:

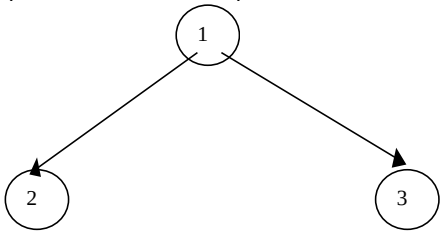
1. El espacio de estados, para los estados del 1 al 15, sería el siguiente:



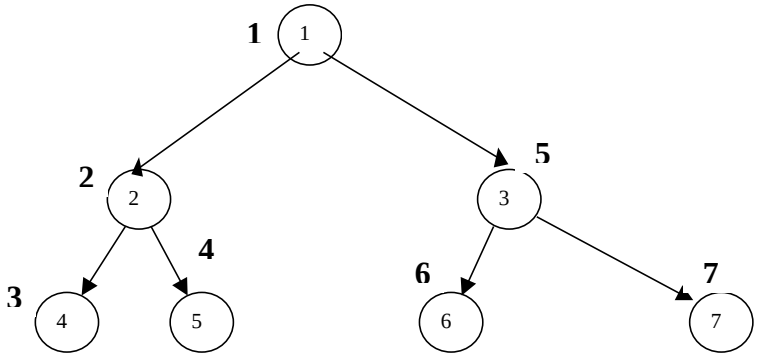
2. En el caso de desarrollar una búsqueda primero en anchura el espacio de búsqueda será:



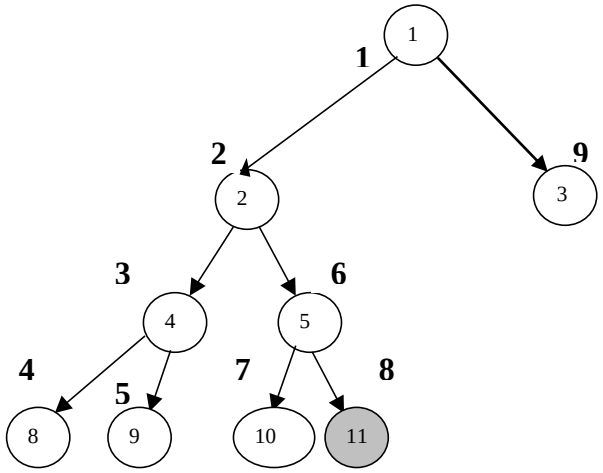
El orden de expansión de los nodos coincide con el etiquetado de los mismos. Concretamente, antes de expandir el nodo 11 y comprobar si es el nodo meta, se habrán expandido previamente los nodos 8, 9 y 10.
En el caso de desarrollar una búsqueda iterativa, en la primera iteración se generará el siguiente árbol de búsqueda:



En la segunda iteración el espacio de búsqueda será:



La solución se encuentra al desarrollar la tercera iteración del árbol de BPI. Nótese que, a diferencia de anchura, no se expanden los nodos del nivel 3 del árbol.



b) Podremos garantizar que se encuentra la solución óptima para valores de 'a' que cumplan a<2, ya que en este caso tendremos una búsqueda de tipo A* que es admisible, lo que hacemos es multiplicar el coste real g(n) en cualquier nodo por el mismo factor, por lo que al final encontraremos un camino óptimo aunque el coste que obtengamos al aplicar f(n) al nodo hoja de ese camino esté multiplicado por (2-a).

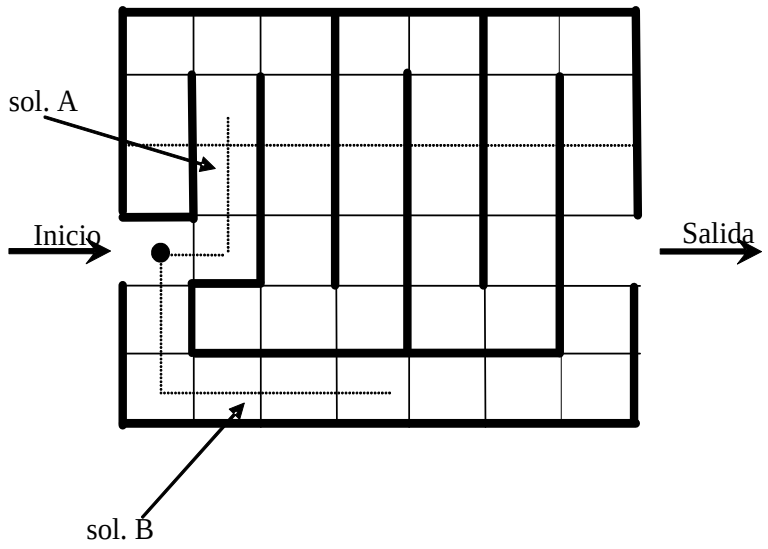
Para valores negativos de (2-a) no encontraríamos la solución óptima ya que al aplicar la función de evaluación al nodo hoja del camino encontrado obtendríamos el valor más pequeño posible al aplicar la función que, en realidad se correspondería con el mayor valor de g(n) al estar multiplicado por un factor negativo.

Para a=2 no se encontrará la solución óptima. Al no incluir en la función de evaluación el coste real del camino desde el nodo raíz al nodo valuado (g(n)), tendremos una búsqueda voraz que no es admisible.

Ejercicio 16

Sea el laberinto que se muestra en la siguiente figura donde las líneas de trazo grueso indican obstáculos; el objetivo consiste en encontrar la Salida empezando desde la posición marcada como Inicio. Los posibles operadores son ARRIBA, ABAJO, DERECHA e IZQUIERDA, los cuales solo se pueden aplicar si no se atraviesa una línea de trazo grueso (obstáculo) entre dos casillas. Como se puede observar en la figura, existen dos posibles soluciones para este problema.

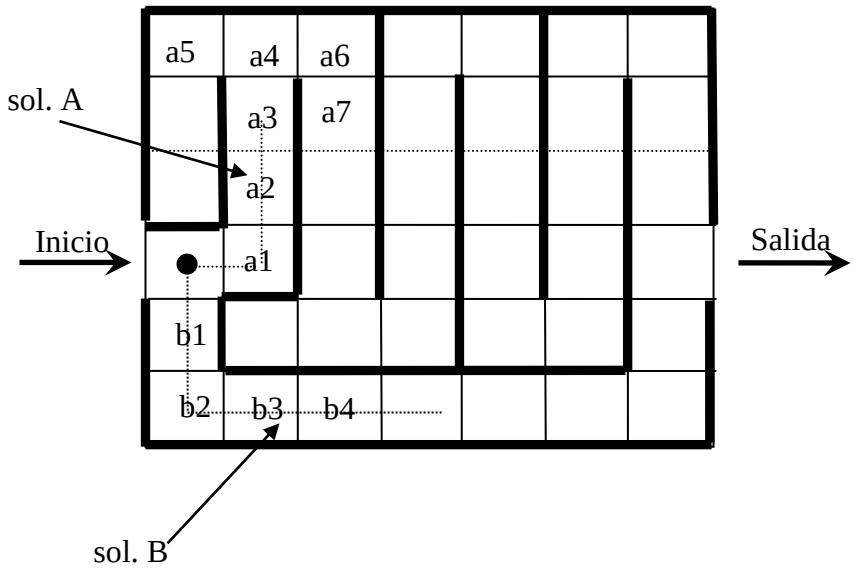
Considera la heurística Distancias de Manhattan para este problema (igual aplicación que para el 8-puzzle desde Inicio hasta Salida). Asumimos que todos los operadores tienen un coste de una unidad. Asumimos que en el proceso de búsqueda se controlan los ciclos directos. Contesta a las siguientes preguntas:



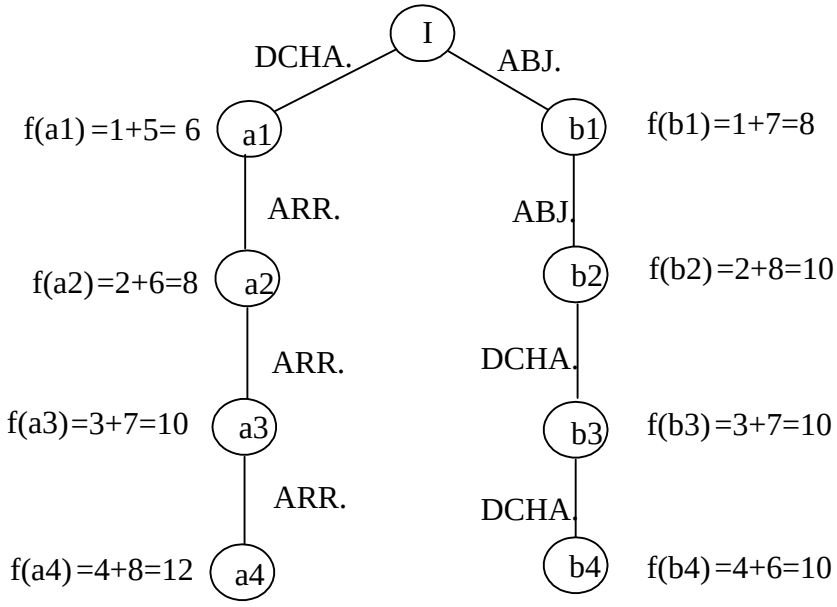
1) Genera el árbol que resulta de aplicar un algoritmo de tipo A (f(n)=g(n)+h(n)) mostrando los 7 primeros nodos que se expanden. ¿Qué solución encontrará este algoritmo? ¿Es una búsqueda A*? Razona las respuestas.
2) Si el coste del operador DERECHA fuera 2 en lugar de 1, la solución encontrada por el algoritmo A en el apartado 1) ¿sería la misma (misma calidad y mismo coste computacional)? ¿Por qué?
3) Si aplicamos una búsqueda primero-el-mejor, ¿qué solución se encontrará? Justifica tu respuesta.
4) Si aplicamos una búsqueda en anchura, ¿qué solución se encontrará? Justifica tu respuesta.
5) Supongamos que queremos aplicar una búsqueda en profundidad. Especifica qué parámetros sería necesario establecer garantizar que se encuentra una solución e indica cuál sería dicha solución.

Solución:

1) Asumimos que los nodos que constituyen la sol. A se numeran consecutivamente como a1, a2, a3, a4, …. etc. y los nodos que conforman la sol. B se numeran consecutivamente como b1, b2, b3 … etc.



Como se puede observar a partir del desarrollo del árbol, la solución que encontrará es la B. porque dicha rama se seguirá expandiendo con un valor f=10 para el resto de nodos (un punto más de g(n) y un punto menos de h(n)) hasta llegar a la Salida. Y justamente el valor de la sol. B es 10.



Efectivamente encuentra la sol. óptima (coste=10). Es un algoritmo A* porque la estimación de las distancias es siempre <= que el coste real ya que se ignoran los posibles obstáculos que se podrían atravesar en el camino. Concretamente:

a) la estimación para las casillas de la sol. A estará muy por debajo del coste real y se irá acercando a h* a medida que las casillas están más cerca de la Salida
b) la estimación para las casillas de la sol. B será h(n)=h*(n) excepto para la casilla b1 donde h(b1) < h*(b1)

2) Si el coste del operador DERECHA vale 2 cambiarán los valores de g(n) y h*(n) pero el valor de h(n) sigue siendo el mismo. Por tanto, la solución que encuentra sería la misma (sol. B) pero el coste de la búsqueda de dicha solución sería distinto, concretamente mayor que en el apartado 1. La razón es la siguiente:

- la sol. B tendría un coste total de 16.
- como la heurística es A* (y lo sigue siendo con un coste del operador DERECHA=2) el máximo valor de f(n) que tomarían las casillas de la ruta B es por tanto 16
- las casillas de la sol. A se irán expandiendo mientras su valor sea < 16, cosa que sucede hasta la casilla a11 (f(a11)=13+4=17). A partir de la casilla a11, los valores de f(n) serán superiores a 16 y nunca se expandirán, haciendo por tanto que se expanden las casillas b_i y encontrando la sol. B

Se generan por tanto 6 nodos más que en el apartado 1). Este incremento se justifica también porque al aumentar el coste del operador DERECHA, se incrementa el coste real de la solución (h*(n)) y por tanto h(n) << h*(n), es decir, h(n) proporcionará menos información en este caso que en el apartado 1) (función heurística menos informada) y por tanto se expanden más nodos.

3) Aquí ocurre un fenómeno curioso. Como se puede observar en el árbol del apartado 1) y fijándonos solo en el valor de h(n), llega un punto en el que tendríamos en la lista OPEN dos nodos con valor h(n)=8 (b2 y a4). A partir de este punto pueden ocurrir dos cosas:

- si se decide expandir b2 entonces la solución que encontrará será la sol. B porque la heurística a partir del nodo b2 siempre decrece
- si se decide expandir a4 sucede lo mismo pero en sentido contrario, que la solución que encontrará será la sol. A, justamente por la misma razón, porque a partir de la casilla a4 los valores heurísticos de las siguientes casillas siempre serán < 8.

4) Búsqueda anchura y con costes uniformes encontrará la solución más corta, es decir la B.

5) El parámetro más importante para una estrategia en profundidad es el nivel de profundidad. Por tanto si:

- nivel-prof $\in$ [10,27] encontrará la solución B

- nivel-prof >= 28 entonces podría encontrar cualquiera de las dos soluciones. En este caso depende de qué rama expande primero: si empieza por una casilla a1 entonces encontrará la sol. A, si empieza por la casilla b1 entonces encontrará la sol. B

Ejercicio 17

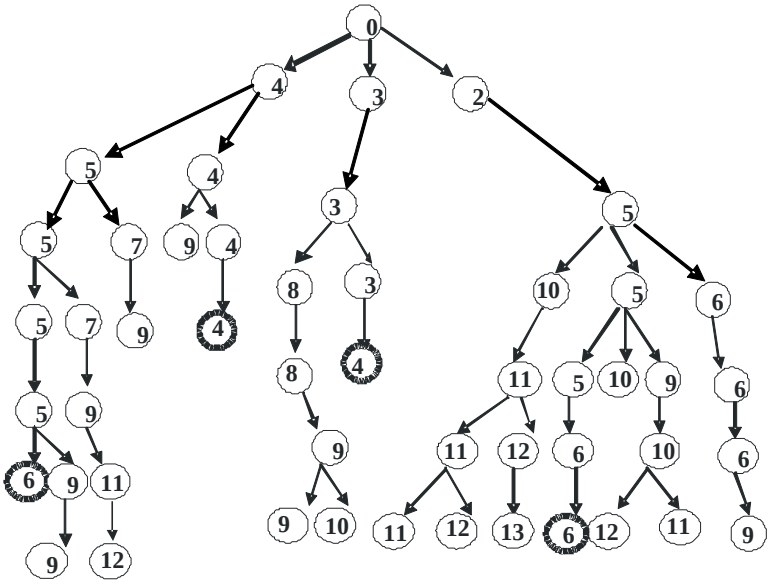
Sea el siguiente árbol que representa el espacio de soluciones de un problema determinado. Los valores de cada nodo son el resultado de aplicar una función f(n)=g(n)+h(n), donde h(n) es una función heurística que da lugar a una búsqueda de tipo A* (h(n) ≤ h*(n)). El coste de aplicación de cada operador es uniforme (1). Los nodos que aparecen en negrita son estados solución del problema. Contesta a las siguientes preguntas:

a) Si se realiza una búsqueda heurística aplicando la función f(n) anterior, ¿qué solución encontraría el proceso de búsqueda?, ¿cuántos nodos se generarían para encontrar dicha solución?

b) Si sobre dicho espacio de soluciones se realiza una búsqueda por profundización iterativa ¿qué solución encontraría el proceso de búsqueda?, ¿son iguales las dos soluciones, en caso contrario cual es mejor?¿Por qué?

c) Utilizando una función de evaluación f1(n)= g(n)+h1(n), tal que h(n) < h1(n) ≤h*(n), ¿la senda solución que se obtenga con f1(n), será mejor/peor/igual que la que se obtenía con f(n)?, ¿qué función de evaluación realizará más/menos/igual búsqueda?

d) Si se realiza una búsqueda por coste uniforme, ¿qué solución se encontraría?, ¿Qué nodos se generarían para encontrar la solución?. ¿Será mejor o peor que la encontrada en el apartado a)?¿Por qué?



Solución:

a) La solución que se encuentra es el nodo solución de valor 4 (nivel 4 del árbol) de la rama central del árbol. Se generan un total de 9 nodos, incluido el nodo raíz.
b) En este caso, asumiendo una expansión de izquierda a derecha, la solución que se encontraría es el nodo solución de valor 4 (nivel 4 del árbol) de la rama izquierda del árbol. Son soluciones distintas pero ambas son óptimas.
c) Ambas son admisibles por lo que encontrarán una solución óptima. Presumiblemente, la función heurística que está más informada (h1(n)) realizará más poda y por tanto generará menos nodos.
d) En este caso en que el coste de aplicación de cada operador es uniforme, la búsqueda por coste uniforme coincide con una búsqueda en anchura, que a su vez coincidirá con la solución encontrada por la estrategia de profundización iterativa. Por tanto, asumiendo de nuevo un orden de expansión de izquierda a derecha, la solución encontrada es la misma que la del apartado b). Para encontrar la solución se generarán todos los nodos de los cuatro primeros niveles del árbol y los dos primeros nodos del nivel 5 hasta que se extrae de la

lista de nodos abiertos el nodo 4 de nivel 4 para ser expandido (total: 30 nodos generados incluido el nodo raíz). Respecto a la comparativa con la solución del apartado a), la solución es distinta pero ambas son óptimas ya que las dos estrategias son admisibles.

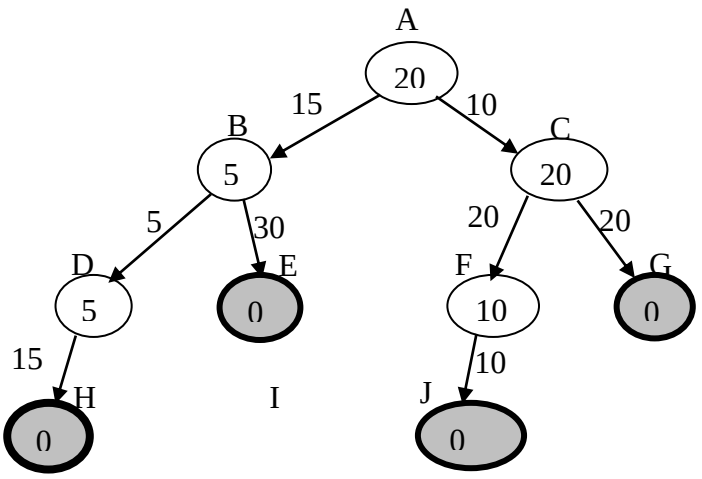
Ejercicio 18

Dado el siguiente árbol de búsqueda, que representa el espacio de búsqueda completo a partir de nodo inicial A, y en el que se indica:

- en cada una de las ramas, el coste de aplicación de la regla, $g(n_i \rightarrow n_j)$,
- en cada uno de los nodos, la estimación del coste a meta, h(n),
- los nodos meta, como nodos sombreados

Efectuando una búsqueda de tipo-A, a partir del nodo inicial A, responder:

a) ¿Es una búsqueda tipo A*?
b) Indicar *claramente* el orden en el que se expandirían los nodos, por ejemplo:
"Paso-i: Se expande el nodo X, y genera los nodos Y y Z"
c) Indicar el estado que devolvería el algoritmo como estado-meta.
d) Devuelve el proceso de búsqueda la solución óptima. ¿Por qué?
e) Es monótona la función f(n)=g(n)+h(n) utilizada?
f) Indicar el orden de expansión de los nodos, si se hiciera una búsqueda 'voraz' y el estado meta que devolvería

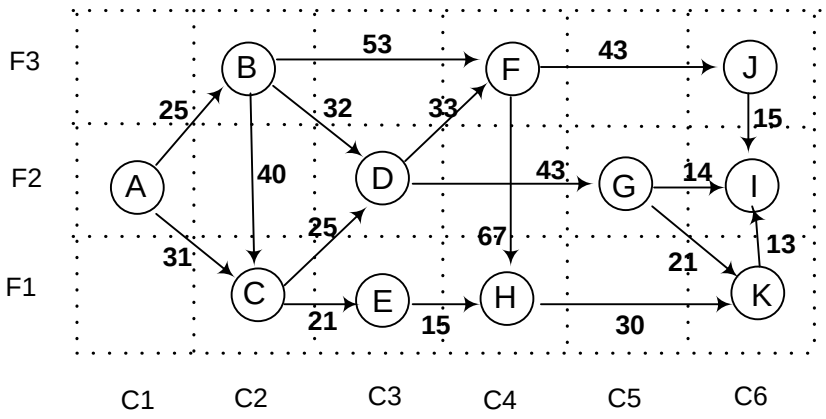


Solución:

a) Es una búsqueda de tipo A*, ya que se cumple que $\forall n, h(n) \leq h^*(n)$.
b) Paso-1 Se expande A, se generan B, con f(B)=20, y C, con f(C)=30
Paso-2 Se expande B, se generan D, con f(D)=25, y E, con f(E)=45
Paso-3 Se expande D, se genera H, con f(H)=35
Paso-4 Se expande C, se generan F, con f(F)=40, y G, con f(G)=30
Paso-5 Se expande G. Se detecta que nodo meta. FIN.
c) El proceso devuelve G como nodo meta.
d) Claramente, devuelve la senda óptima, como cabría esperar de una búsqueda tipo A*
e) Claramente, es una función f(n) monótona, al cumplirse en todos los casos:
$\forall n, h(n2) \geq h(n1) - Coste(n1 \rightarrow n2)$
g) Con búsqueda 'voraz', el orden de expansión sería: "A, B, E". El estado meta que devolvería sería E.

Ejercicio 19

Sea el grafo que se muestra en la figura, donde los nodos representan ciudades y los números de los arcos representan la distancia real en km. entre pares de ciudades. El grafo aparece representado en un mapa con los diferentes cuadrantes donde se sitúa cada nodo ciudad.



Considera la heurística Distancias de Manhattan para este problema, cuya aplicación es exactamente igual que para el problema del 8-puzzle desde un nodo inicio a un nodo final a través de los cuadrantes del mapa (por ejemplo, la Distancia Manhattan entre la ciudad A y E es 3; la distancia Manhattan entre la ciudad E y la I es 4).

Sea h(n)=Distancias_Manhattan(n)*10; el coste de los operadores es la distancia real en km. con la que aparecen etiquetados los arcos entre ciudades.

Contesta a las siguientes preguntas justificando las respuestas. En cada una de las preguntas, indica y detalla claramente la lista de nodos ABIERTOS que se genera en cada iteración de los procesos de búsqueda.

1) Se quiere encontrar una solución para ir de la ciudad A a la ciudad I. Muestra el árbol que resulta de aplicar un algoritmo de tipo A (f(n)=g(n)+h(n)). ¿Qué solución encontrará este algoritmo? ¿Es una búsqueda A*?

2) Sea el mismo problema de ir de la ciudad A la ciudad I. Muestra el árbol de búsqueda que resulta de aplicar una búsqueda primero en profundidad con un límite máximo de profundidad = 4. Asume que se expanden primero los nodos alfabéticamente anteriores. ¿Qué solución encontrará este proceso? ¿cuántos nodos se generan y cuántos se expanden en esta búsqueda?

3) Sea el problema de ir de la ciudad A a cualquier ciudad de la columna C6. Si aplicamos una búsqueda voraz, ¿qué ciudad es la que se alcanza primero?¿qué coste tiene esta solución? Muestra el árbol que resulta de aplicar esta búsqueda.

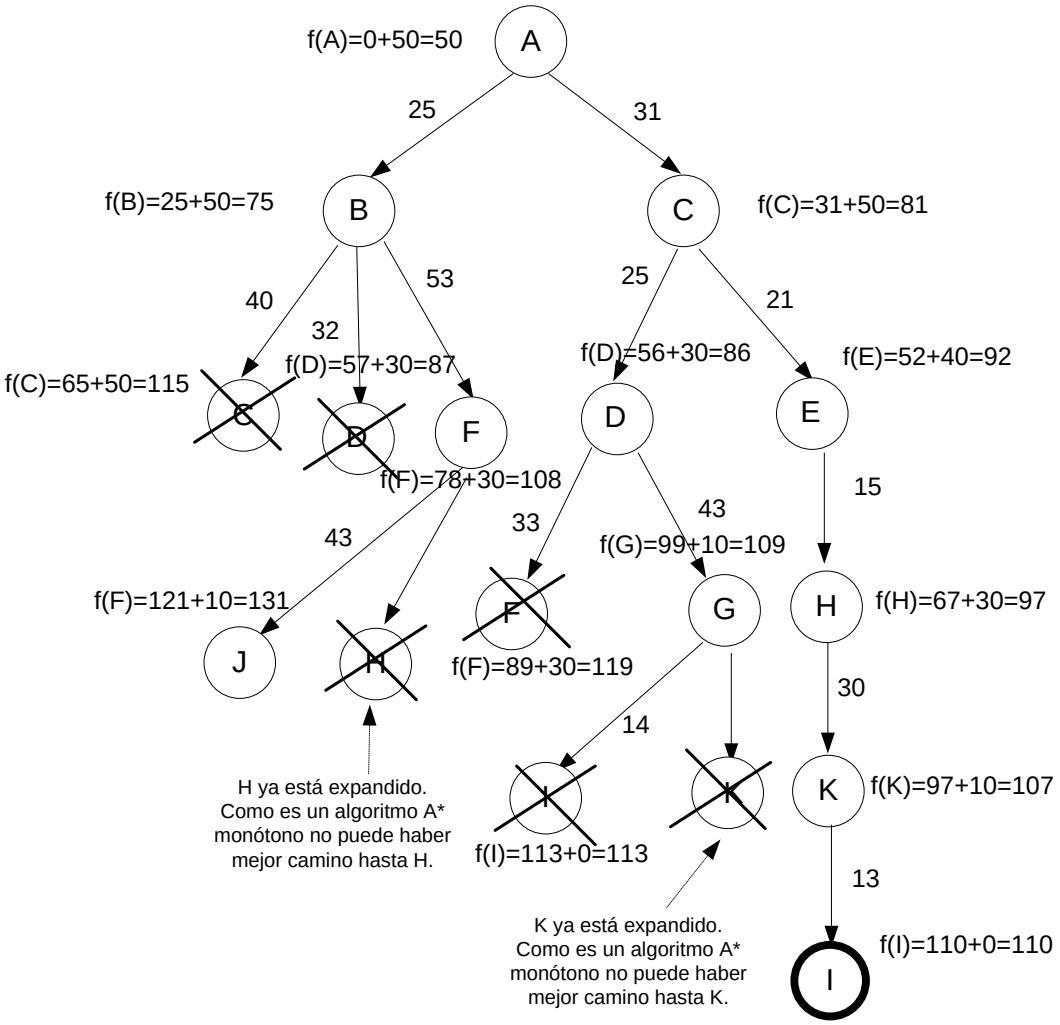
Solución:

1) Iteraciones del algoritmo. Entre paréntesis se muestra el valor f(n) del nodo.

OPEN ={A(50}) se extrae y expande el nodo A
OPEN={B(75), C(81)} se extrae y se expande B
OPEN={C(81), D(87), F(108)}

El nodo C(115) no se introduce en la lista porque ya existe un camino mejor hasta dicho nodo y solo nos interesa encontrar una solución. Además, al tratarse de un algoritmo A* monótono cuando se expanda C(81) sabemos que no puede existir camino mejor desde el nodo raíz hasta el nodo C.
Se extrae y se expande el nodo C(81).

OPEN={D(86), E(92), F(108)}
En esta iteración del algoritmo se genera un nuevo camino hasta el nodo D(86) con un coste menor que el que existe en la lista OPEN; eliminamos el nodo D(87) e insertamos el nodo D(86) ya que solo nos interesa encontrar una solución.
Se extrae y expande el nodo D(86).



OPEN={E(92), F(108), G(109)}
Al expandir D se genera un nuevo camino hasta F que no es mejor que el ya existente. No se introduce este nuevo camino encontrado por las razones expuestas anteriormente.
Se extrae y expande el nodo E(92).

OPEN={H(97), F(108), G(109)} Se extrae y expande el nodo H(97).

OPEN={K(107), F(108), G(109)} Se extrae y expande el nodo K(107).

OPEN={F(108), G(109), I(110)} Se extrae y expande el nodo F(108).

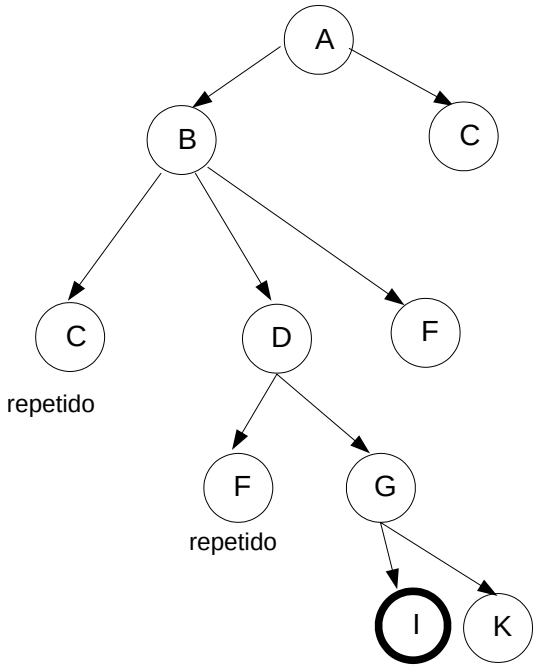
OPEN={G(109), I(110), J(131)}
El nodo H ya está expandido por tanto no es necesario considerar su evaluación ya que se trata de un algoritmo A* monótono.
Se extrae y expande el nodo G(109)

OPEN={I(110), J(131)}
El nodo K ya está expandido por tanto no es necesario considerar su evaluación ya que se trata de un algoritmo A* monótono. Se encuentra un nuevo camino hasta el nodo I(113) que no es mejor que el que existe en la lista OPEN y, por tanto, no se introduce en la lista. Se extrae y expande el nodo I.

SOLUCIÓN: A-C-E-H-K-I
Coste: 110

Se trata de un algoritmo A* porque todas las estimaciones entre pares de ciudades son menores que el coste real entre ellas. También es un A* monótono porque se cumple que h(padre) <= h(hijo)+ coste(padre, hijo). Por esta razón, no es necesario considerar la expansión del nodo D(87) ni la evaluación del nodo K como hijo del nodo G.

2) CON CONTROL DE NODOS REPETIDOS



OPEN={A} se extrae y se expande el nodo A

OPEN={B, C} se extrae y se expande el nodo B. La expansión de B genera de nuevo el nodo C, que ya está en la lista OPEN. Como g(C)=1 para el nodo que está en la lista OPEN, y g(C)=2 para el nodo C que es hijo de B, nos quedamos con el nodo C de coste 1.

OPEN={D,F,C} se extrae y se expande el nodo D y se generan F y G. Como ya existe un nodo F en OPEN de menor coste, nos quedamos con éste nodo y no insertamos el hijo F de D.

OPEN={G,F,C} se extrae y se expande el nodo G hijo de D

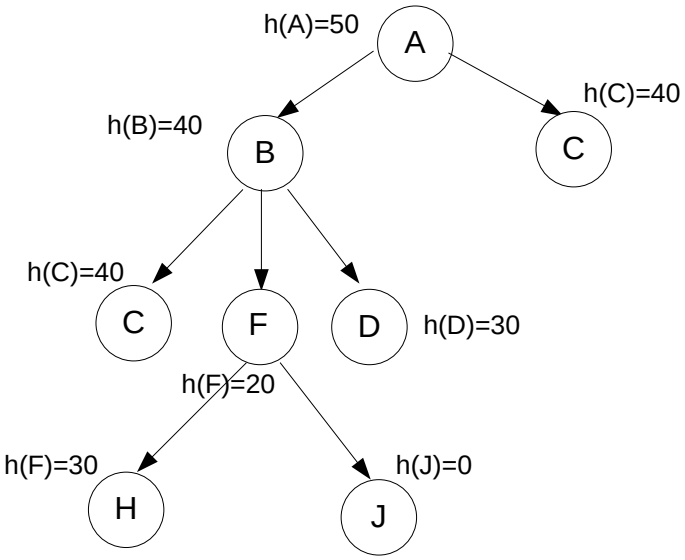
OPEN={I,K,F,C}
Se extrae el nodo I, se expande y se comprueba que es SOLUCION.

SOLUCIÓN: A-B-D-G-I
Coste: 114
Se generan 10 nodos y se expanden 5 nodos.

3)

OPEN={A(50)} Se extrae y expande el nodo A
OPEN={B(40), C(40)} Podríamos expandir cualquiera de los dos. Extraemos y expandimos B.
OPEN={F(20), D(30), C(40)}
Ya existe un nodo C con valor 40, así que no se vuelve a insertar. Se extrae y expande el nodo F.
OPEN={J(0), H(30), D(30),C(40)} Se extrae y expande el nodo J. SOLUCIÓN.

SOLUCIÓN: A-B-F-J Coste: 121



Si se expande antes el nodo C las soluciones podrían ser:

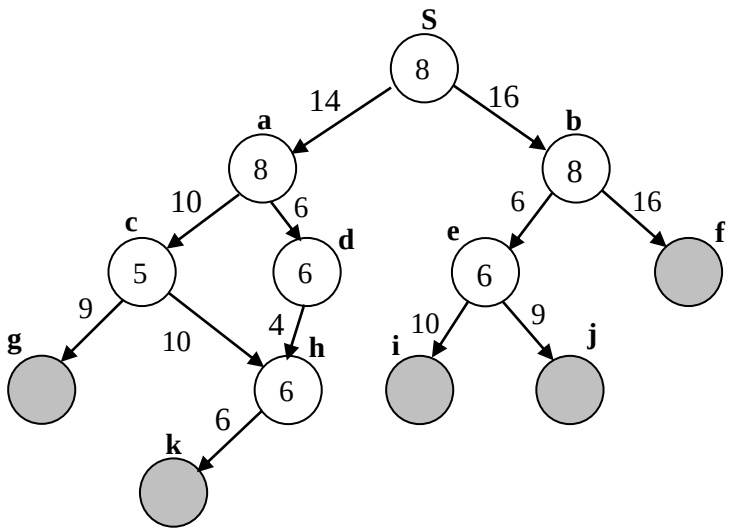
1) A-C-D-G-I Coste: 113
2) A-C-E-H-K Coste: 97

Ejercicio 20

Dado el siguiente espacio de búsqueda, con los valores de g(n) indicados en las ramas, de h(n) en los nodos, y los nodos meta sombreados, realizar un proceso de búsqueda A y un proceso de búsqueda de coste uniforme. En cada caso, indicar:

- El orden en el que se expanden los nodos y el nodo meta obtenido.
- ¿Devuelve la senda de coste óptimo? ¿Por qué?

Nota: A igualdad de la evaluación para la expansión, se expande siempre el de menor nivel.



Solución

Búsqueda Coste Uniforme:

Expande: S;	Genera: A, B;	Lista Abiertos: (A-14, B-16)
Expande: A;	Genera: C, D;	Lista Abiertos: (B-16, C-24, D-20)
Expande: B;	Genera: E, F;	Lista Abiertos: (C-24, D-20, E-22, F-32)
Expande: D;	Genera: H;	Lista Abiertos: (C-24, E-22, F-32, H-24)
Expande: E;	Genera: I, J;	Lista Abiertos: (C-24, F-32, H-24, I-32, J-31)
Expande: C;	Genera: G, H;	Lista Abiertos: (F-32, H-24, I-32, J-31, G-33)

Expande: H;	Genera: K;	Lista Abiertos: (F-32, I-32, J-31, G-33, K-30)
Expande: K;	NODO META	

Búsqueda A:

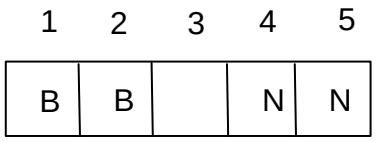
Expande: S;	Genera: A, B;	Lista Abiertos: (A-22, B-24)
Expande: A;	Genera: C, D;	Lista Abiertos: (B-24, C-29, D-26)
Expande: B;	Genera: E, F;	Lista Abiertos: (C-29, D-26, E-28, F-32)
Expande: D;	Genera: H;	Lista Abiertos: (C-29, E-28, F-32, H-30)
Expande: E;	Genera: I, J;	Lista Abiertos: (C-29, F-32, H-30, I-32, J-31)
Expande: C;	Genera: G, H;	Lista Abiertos: (F-32, H-30, I-32, J-31, G-33)
Expande: H;	Genera: K;	Lista Abiertos: (F-32, I-32, J-31, G-33, K=30)
Expande: K;	NODO META	

Al ser h(n)<h*(n), es por tanto A*.

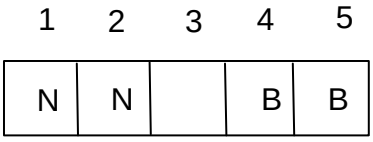
En ambos casos, obtienen la senda óptima, Tanto A* como coste uniforme son admisibles.

Ejercicio 21

La figura (a) muestra el estado inicial de un puzzle lineal que consta de 5 casillas: las dos de la izquierda contienen 2 fichas blancas (B), la del medio es un espacio vacío, y las dos de la derecha contienen 2 piezas negras (N). La figura (b) muestra el estado final al que se desea llegar.



(a) Estado inicial



(b) Estado final

Las acciones que se pueden aplicar en este problema son:

A1. Una ficha se puede mover al espacio vacío si está al lado del espacio vacío (coste = 1)
A2. Una ficha se puede mover al espacio vacío saltando sobre otra ficha de cualquier color (coste =1)
A3. Una ficha se puede mover al espacio vacío saltando sobre dos fichas de cualquier color (coste =2)

Además, hay que tener en cuenta que las fichas B solo se pueden mover hacia la DERECHA, y las fichas N solo se pueden mover hacia la IZQUIERDA.

Sea la siguiente heurística basada en la distancia de una ficha a su zona objetivo. Se define "zona objetivo" de una ficha a las dos casillas de destino posibles de dicha ficha. Así, la zona objetivo de las fichas B son las casillas 4 y 5, y la zona objetivo de las fichas N son las casillas 1 y 2. La función 'h' aplicada sobre una ficha que está en una posición determinada (1, 2, 3, 4 ó 5) devuelve una estimación del coste de mover esa ficha a su zona objetivo. Los valores de 'h' sobre cada ficha y cada posición son:

POSICIONES	1	2	3	4	5
funcion $h(B_i)$:	2	1	1	0	0
funcion $h(N_i)$:	0	0	1	1	2

Por ejemplo, la estimación de una ficha B que está en la posición 1 tiene un coste de 2 (h(B1)=2), o la estimación de una ficha N que está en la posición 1 es cero porque ya está en su zona objetivo (h(N1)=0). La función heurística h(n) aplicada sobre un estado del problema es la suma de las heurísticas para cada ficha. Por ejemplo, dado el estado BB_NN, el cálculo de h(n) sería:

$$h(BB\_NN)=h(B1)+h(B2)+h(N4)+h(N5)=2+1+1+2=6$$

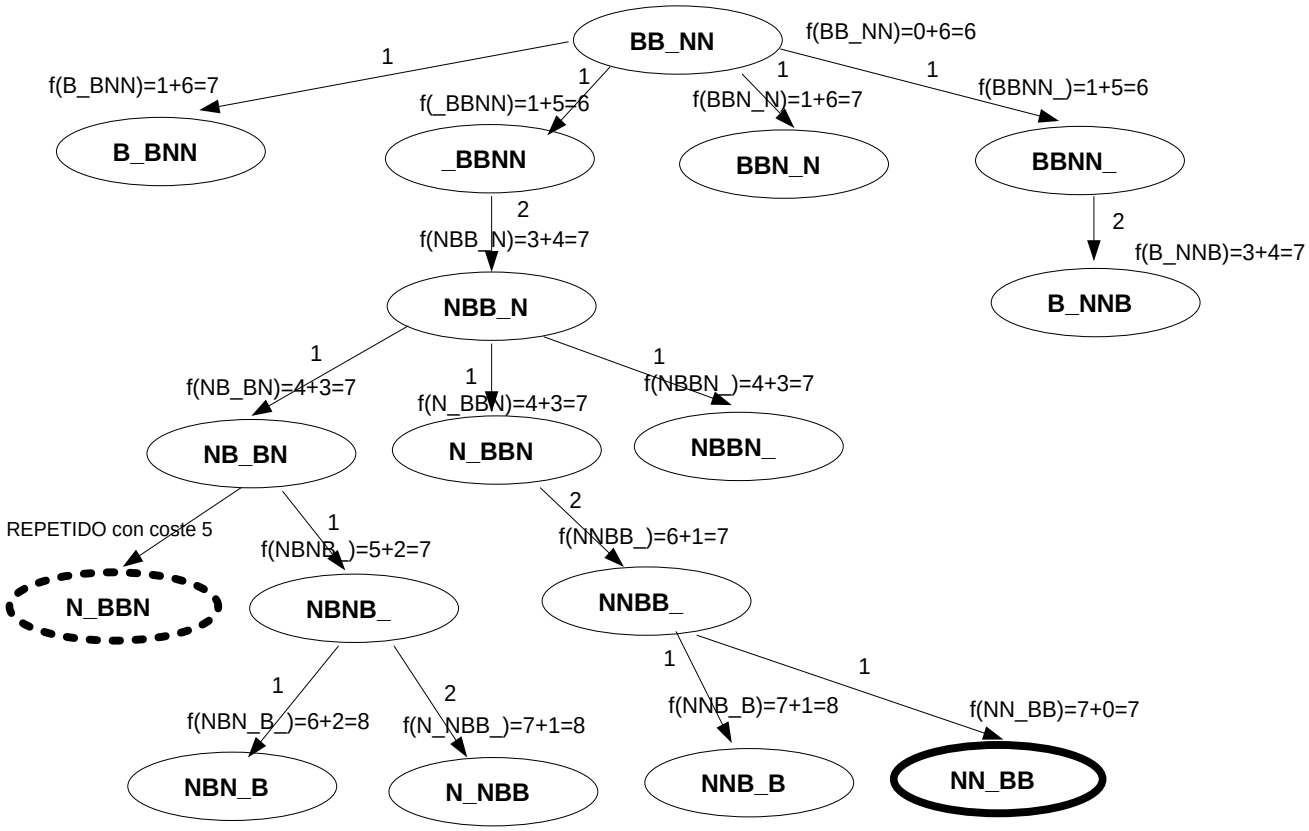
Contesta a las siguientes preguntas justificando las respuestas:

1) Genera y dibuja el árbol que resulta de aplicar un algoritmo de tipo A (f(n)=g(n)+h(n)). Muestra claramente el valor f(n) de cada nodo, el orden en que se expanden y el estado de la lista OPEN en cada iteración. ¿Qué solución encontrará este algoritmo y cuál es el coste de dicha solución? ¿Es una búsqueda A*? Razona las respuestas.
    a. *Nota: Para generar el árbol aplicar este orden de movimientos: primero un movimiento A1 sobre fichas B, luego A2 sobre B, A3 sobre B, A1 sobre N, A2 sobre N, y finalmente A3 sobre N.*
    b. *Nota: Si dos nodos tienen mismo valor f(n) expandir antes el nodo **<u>más profundo</u>**. Ante igualdad en el mismo nivel de profundidad, expandir primero **<u>el más antiguo</u>** creado.*
    c. *Nota: Evitar nodos repetidos.*
2) La aplicación de un algoritmo búsqueda voraz, ¿qué solución encontraría? ¿sería una solución óptima? Razona las respuestas.
3) Si se quiere aplicar una búsqueda en profundidad, ¿sería necesario limitar el nivel de profundidad máximo del árbol para garantizar que se encuentra una solución?¿Por qué?. En caso afirmativo determina cuál sería el valor de este límite. Justifica las respuestas.

<u>Solución</u>

<u>Pregunta 1</u>

1) OPEN= {BB_NN (6)}
2) OPEN= {_BBNN (6), BBNN_ (6), B_BNN (7), BBN_N (7)}
3) OPEN= { BBNN_ (6), NBB_N (7), B_BNN (7), BBN_N (7)}
4) OPEN= { NBB_N (7), B_NNB (7), B_BNN (7), BBN_N (7)}
5) OPEN= { NB_BN (7), N_BBN (7), NBBN_ (7), B_NNB (7), B_BNN (7), BBN_N (7)}
6) OPEN= { NB_NB (7), N_BBN (7), NBBN_ (7), B_NNB (7), B_BNN (7), BBN_N (7)}
7) OPEN= { N_BBN (7), NBBN_ (7), B_NNB (7), B_BNN (7), BBN_N (7), NBN_B (8), N_NBB (8)}
8) OPEN= { NNBB_ (7), NBBN_ (7), B_NNB (7), B_BNN (7), BBN_N (7), NBN_B (8), N_NBB (8)}
9) OPEN= { NN_BB (7), NNB_B (8), NBBN_ (7), B_NNB (7), B_BNN (7), BBN_N (7), NBN_B (8), N_NBB (8)}
10) Se extrae NN_BB => Solución (5 movimientos, coste =7)



Es un algoritmo A* porque con los valores h de la tabla se satisface siempre la relación h(n)<= h*(n) para todo nodo del árbol dado que:

- el coste mínimo de mover B1 o N5 a su zona objetivo es 2 (salto sobre 2 fichas)
- el coste mínimo de mover B2 o N4 a su zona objetivo es 1 (salto sobre 1 ficha)
- el coste mínimo de mover B3 o N3 a su zona objetivo es 1 (desplazar al espacio blanco)

Por ejemplo, h(BB_NN)=6 y h*(BB_NN)=7.

Pregunta 2

Fijándonos SOLO en los valores h(n) del árbol de la pregunta 1, podemos ver que el orden de expansión sería:

1. nodo inicial (h=6)
2. _BBNN (h=5)
3. NBB_N (h=4)
4. NB_BN (h=3)
5. NBNB_ (h=2)
6. N_NBB (h=1)
7. Ahora expandiría el nodo N_NBB llegando al nodo solución NN_BB (h=0)

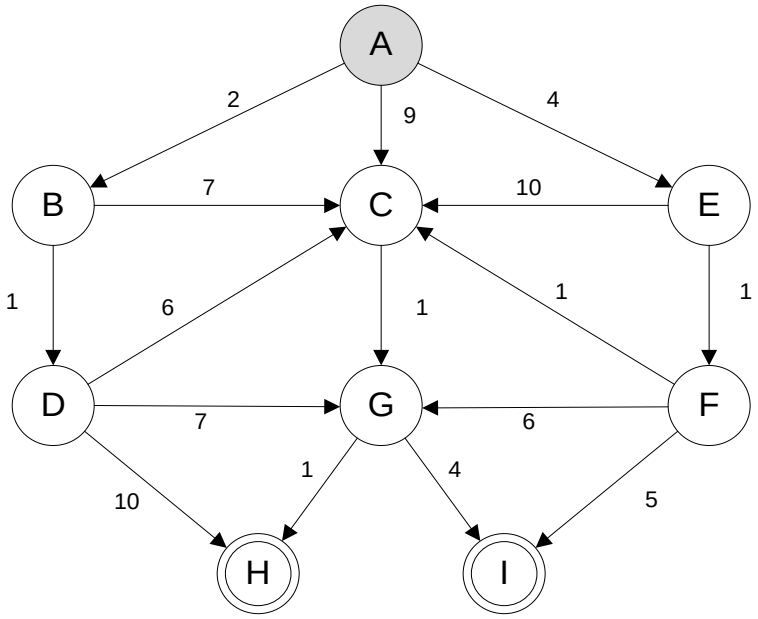
La solución que encontraría este algoritmo se compone de 6 movimientos y tiene un coste=8. Por tanto, no es la óptima.

Pregunta 3

No, en este problema no es necesario limitar el árbol porque no hay operadores reversibles ya que las fichas B solo se pueden mover hacia la derecha y las fichas N solo se pueden mover a la izquierda. Por tanto, no hay riesgo de quedarse estancado en un ciclo. Una búsqueda en profundidad expandiría a partir del hijo B_BNN y eventualmente encontraría una solución.

Ejercicio 22

El siguiente grafo representa un problema de espacio de estados. Los nodos del grafo son los estados del problema, las aristas conectan cada estado con sus sucesores, y el valor numérico de cada arista representa el coste de pasar de un estado al sucesor correspondiente. El estado inicial del problema es el nodo A y los estados finales son H e I. La función heurística h(n) se indica en la tabla:



Nodo	h(n)
A	7
B	7
C	2
D	7
E	4
F	9
G	1
H	0
I	0

Realiza una búsqueda por coste uniforme y con un algoritmo A, indicando en cada caso, las listas de nodos abiertos y cerrados, la solución obtenida y su coste. Se recomienda controlar los nodos repetidos. ¿Se obtiene la solución óptima en ambos casos? ¿La función h(n) es admisible? ¿Por qué?

***Nota**:* Si dos nodos tienen mismo valor f(n) expandir antes el nodo **más profundo**

Solución

Coste uniforme: f(n)=g(n)
OPEN={ A (0), B (2), E (4), C (9), CLOSED={A, B,

Al expandir B, se generan D con coste 3 y C con coste 9 (no se añade a OPEN por tener el mismo coste que el nodo que ya se había añadido al expandir A – esto sucede en otras ocasiones, pero ya no se comentará, simplemente no se añadirá el nodo correspondiente a la lista OPEN).
OPEN={ E (4), C (9), D (3), G (10), H (13), F (5), CLOSED={A, B, D, E, F,

Al expandir F, se genera C con coste 6, que sí se añade a OPEN al tener un coste menor que el que ya figura en la lista, que se elimina; se genera G con coste 11, que no se añade e I con coste 10:
OPEN={ G (10), H (13), C (6), I (10) CLOSED={A, B, D, E, F, C,

Al expandir C, se genera G con coste 7, que sustituye al nodo que está ahora en OPEN.
OPEN={ ~~H (13),~~ I (10), G (7), H (8), CLOSED={A, B, D, E, F, C, G, H }

Al extraer H de la lista OPEN, se comprueba que es un nodo solución -> FIN
Solución: A -> E -> F -> C -> G -> H Coste: 8

Algoritmo A: f(n)=g(n)+h(n)

OPEN={ A (0+7), B (2+7), C (9+2), E (4+4), F (5+9), D (3+7), G (10+1), H (13+0)
CLOSED={ A, E, B, D, G,

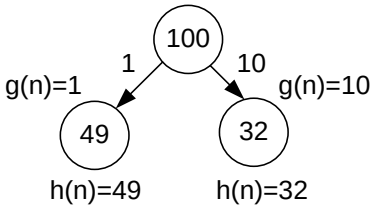
Al expandir G, se genera H con coste 11, por lo que sustituye al nodo que figura ahora en OPEN:
OPEN={ C (9+2), F (5+9), H (11+0), I (14+0) CLOSED={ A, E, B, D, G, H }

Solución: A -> B -> D -> G -> H Coste: 11

Se obtiene la solución óptima solamente en el caso de coste uniforme. h(n) no es admisible porque h(F)=9 > h*(F)=3 (F-C-G-H).

Ejercicio 23

Considerar un espacio de soluciones donde el estado inicial es el número 100 y la función sucesor (operador) para cualquier estado 'n' devuelve dos estados int(n/2)-1 *(el coste de este operador es 1)* y int(n/3)-1 *(el coste de este operador es 10)*. Supongamos que se aplica una función de evaluación h(n) para cada nodo, tal que h(n) es igual al valor del nodo n. Véase el ejemplo para el primer nivel:

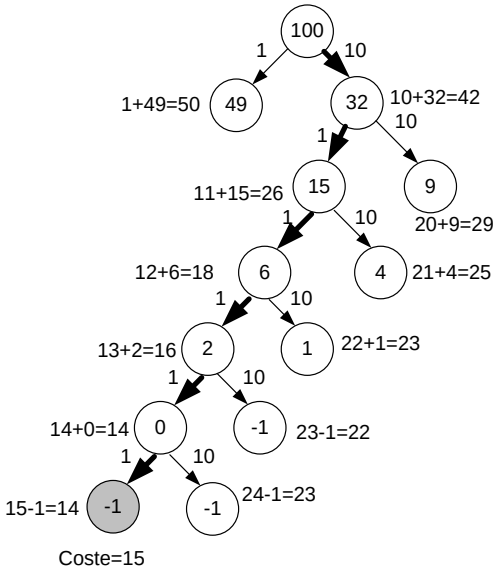


Obtener la senda solución hasta un estado meta (***la condición de nodo meta es que su valor sea negativo***) y el ***coste de la senda solución obtenida***, aplicando:

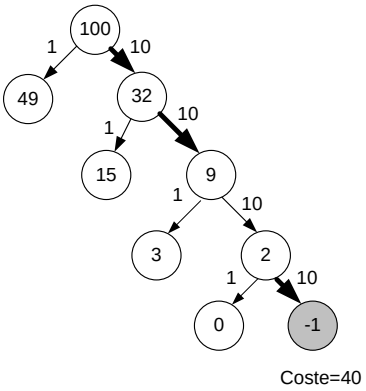
a) Una búsqueda de tipo A, indicando los valores g(n) y h(n) de cada nodo generado.
b) Una búsqueda 'voraz'.
c) Responder justificadamente si la función h(n) aplicada es admisible.

Solución:

Búsqueda A:



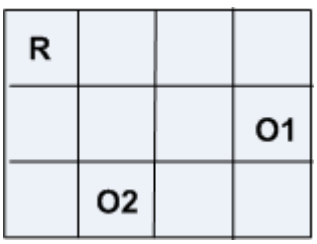
Búsqueda 'voraz':



Claramente, la función h(n) utilizada **NO es admisible**. Por ejemplo, estima para el nodo inicial un coste h(n)=100, cuando se ha obtenido una senda de coste menor.

Ejercicio 24

Queremos encontrar la ruta que tiene que seguir un robot (R) para recoger los objetos de una habitación (O1 y O2). La situación es la que se muestra en la figura. El robot solo puede moverse horizontal y verticalmente, y en cada movimiento solo puede avanzar una casilla. El robot recoge un objeto cuando llega al cuadrado donde está dicho objeto.



Sea la siguiente función heurística: $h(n)=\Sigma d_j$ donde d_j es la distancia (número de movimientos) que tiene que realizar el robot para alcanzar el objeto 'j'. Es decir, la función heurística es el sumatorio de la distancia de Manhattan del robot a cada objeto pendiente de recoger en la habitación. Cuando el robot recoge un objeto, asumiremos que la distancia a dicho objeto es 0, y que el objeto se mueve con el robot a partir de ese estado. El coste de cada movimiento es 1.

1) Genera y dibuja el árbol que resulta de aplicar un algoritmo de tipo A (f(n)=g(n)+h(n)) para encontrar el camino que tiene que seguir el robot para recoger los dos objetos. Muestra claramente el valor f(n) de cada nodo, el orden en que se expanden y el estado de la lista OPEN en cada iteración. ¿Qué solución encontrará este algoritmo y cuál es el coste de dicha solución? La función '**h**', ¿es una heurística A*? ¿Por qué?. Razona las respuestas.
2) Partiendo del árbol desarrollado en el apartado 1, la aplicación de un algoritmo búsqueda voraz, ¿qué solución encontraría? ¿sería una solución óptima? ¿Por qué? Justifica las respuestas.

NOTAS para realizar la expansión de los árboles:

1. *Utilizar el siguiente orden de aplicación de movimientos: primero un movimiento ARRIBA, segundo ABAJO, tercero DERECHA, cuarto IZQUIERDA.*
2. *Si dos nodos tienen mismo valor f(n) expandir antes el nodo **más profundo**. Ante igualdad en el mismo nivel de profundidad, expandir primero **el más antiguo** creado.*
3. *Evitar nodos repetidos.*

Solución:

A) Se muestra el estado de la lista OPEN en cada iteración de acuerdo a la figura. Los valores entre paréntesis muestran el valor f(n) del nodo correspondiente.

1) OPEN={S0(7)}
2) OPEN={S1(6), S2(6)}
3) OPEN={S4(5), S2(6), S3(7)}
4) OPEN={S5(6), S6(6), S2(6), S3(7)}
5) OPEN={S7(6), S8(6), S6(6), S2(6), S3(7), S9(8)}
6) OPEN={S11(6), S8(6), S6(6), S2(6), S3(7), S10(8), S12(8), S9(8)}
7) OPEN={S14(6), S8(6), S6(6), S2(6), S3(7), S13(8), S10(8), S12(8), S9(8)}
8) Se extrae de la lista S14 que es un nodo solución

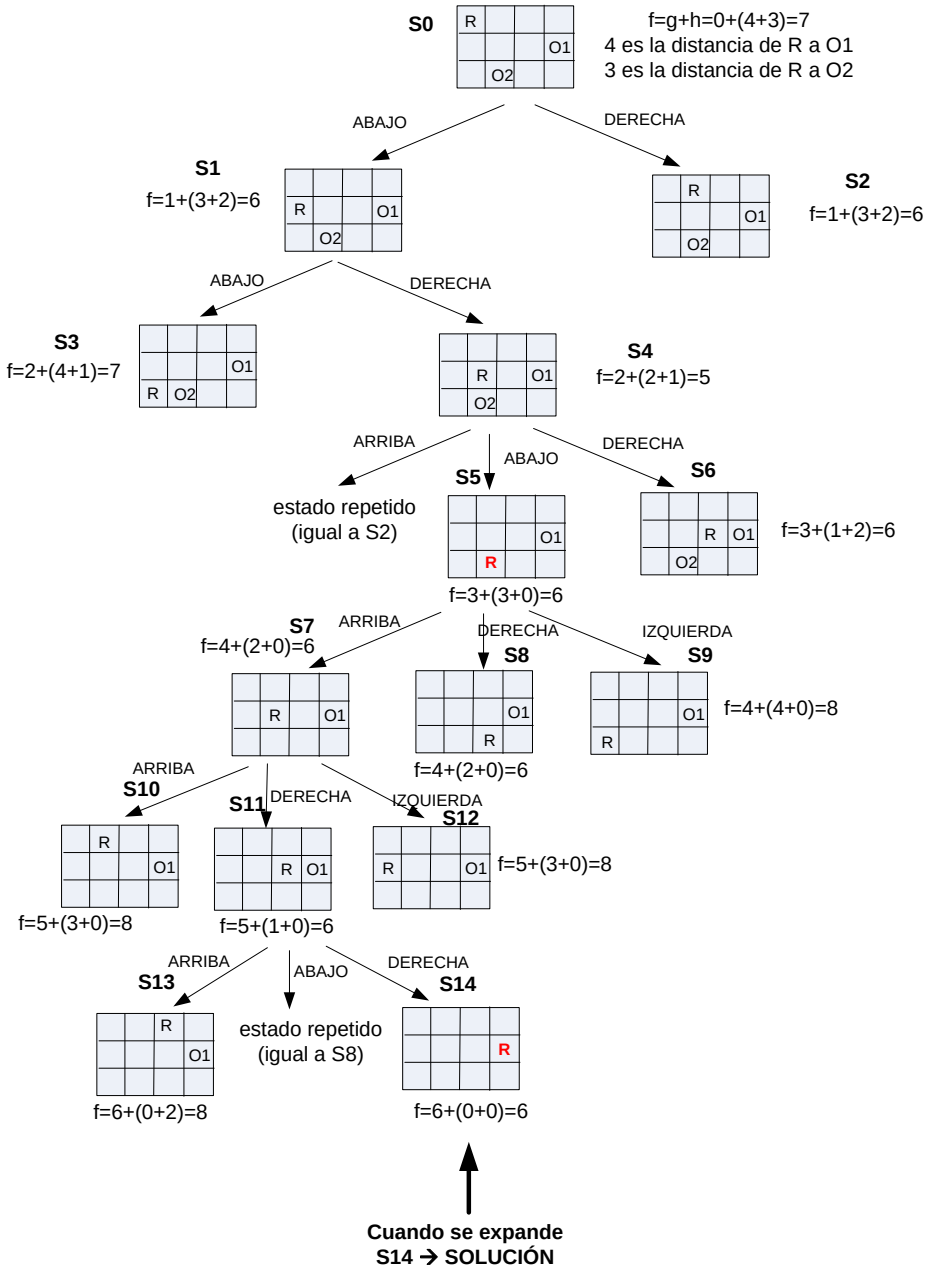
La solución que encuentra el algoritmo A es: ABAJO-DERECHA-ABAJO-ARRIBA-DERECHA-DERECHA

que tiene un coste de 6 pasos.
Como se puede observar, el algoritmo ha encontrado la solución óptima (6 pasos). Sin embargo, la ***heurística no es admisible*** porque se puede encontrar algún caso en el que la función 'h' sobreestima el coste real. Véase por ejemplo el estado inicial S0, donde la heurística estima una distancia de 7 siendo el coste óptimo 6. Otro ejemplo se puede ver en S3. La heurística estima una distancia total de 5 cuando el coste óptimo a partir del estado S3 es 4 (un paso para recoger O2 y tres pasos más luego para recoger O1).

La razón de que la heurística no sea A* es que asume que recoger un objeto es independiente de recoger el otro objeto, es decir que el plan de acciones para recoger un objeto es independiente del plan de acciones para recoger el otro objeto, razón por la cual suma la distancia del robot R a cada uno de los objetos. Sin embargo, esto no es así, y la heurística no está teniendo en cuenta que R irá a recoger un segundo objeto a partir de la posición donde se haya quedado tras recoger el primer objeto.

NOTAS:

a. No se han dibujado los estados repetidos que implican un ciclo en la misma rama. Se han indicado algunos estados repetidos con nodos de otras ramas, aunque no es necesario pintarlos.
b. Nótese que el estado S7 no es igual que el S4, y por tanto no es repetido. Aunque la posición del robot es la misma, los estados son distintos porque en S4 el robot no tiene el objeto O2, y en el estado S7 sí.



B**)** Un algoritmo de búsqueda voraz se guía únicamente por el valor h(n), y encontrará exactamente la misma solución que en el apartado A). Por tanto, misma solución y mismo coste. Concretamente, el proceso de búsqueda sería del siguiente modo:

a) Se expande S0. Se generan S1 y S2. Ambos tienen el mismo valor heurístico. Se escoge, por tanto, S1.
b) Se expande S1. Se generan S3 y S4. S4 es el nodo que tiene menor valor heurístico de todos los nodos abiertos (h(S4)=3).
c) Se expande S4. Se generan S5, S6. h(S5)=h(S6)=3 es el menor valor heurístico entre todos los nodos abiertos. Se escoge S5.
d) Se expande S5. Se genera S7, S8 y S9. h(S7)=h(S8)=2. Se escoge S7.
e) Se expande S7. Se genera S10, S11 y S12. h(S11)=1. Se escoge S11.
f) Se expande S11. Se genera S13 y S14. h(S14)=0. Se escoge S14.
g) Se expande S14 → SOLUCIÓN

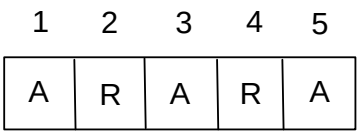
Ejercicio 25

Tenemos cinco monedas dispuestas en una fila tal como indica la figura (a), donde **A** representa el anverso de la moneda, y **R** el reverso. La única operación que se puede realizar en este problema es dar la vuelta (de **A** a **R** ó de **R** a

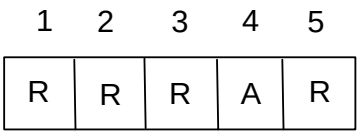
A) a cualquier **par de monedas contiguas** (moneda1-moneda2, moneda2-moneda3, moneda3-moneda4 ó moneda4-moneda5), y dicha operación tiene un coste de 1. Deseamos obtener la situación que se muestra en la figura (b).

Función heurística. Para calcular la función heurística **h(n)** se analizan las monedas en dos fases, en la primera fase se analizan de izquierda a derecha y en la segunda fase de derecha a izquierda. En cada fase, se analizan los siguientes pares de monedas contiguas:

- de izda. a dcha.: se mira moneda 1 con moneda 2 y moneda 3 con moneda 4
- de dcha. a izda.: se mira moneda 5 con moneda 4 y moneda 3 con moneda 2



(a) Estado inicial



(b) Estado final

Los valores de 'h' para cada pareja de monedas son:

h(moneda X colocada, moneda Y colocada) = 0 puntos
h(moneda X descolocada, moneda Y descolocada= 1 punto
h(moneda X colocada, moneda Y descolocada) = 2 puntos
h(moneda X descolocada, moneda Y colocada) = 2 puntos

En resumen, si las dos monedas contiguas están colocadas entonces sumamos 0 puntos. Si las dos monedas contiguas están descolocadas entonces sumamos 1 punto. Si de las dos monedas contiguas una está colocada y la otra no entonces sumamos 2 puntos.

La función h(n) para un estado 'n' del problema se calcula del siguiente modo: **h(n)=min(h_izq_der(n),h_der_izq(n))**(mínimo valor entre recorrer las fichas de izquierda a derecha y recorrer las fichas de derecha a izquierda) donde:

h_izq_der(n)= h(moneda1,moneda 2)+ h(moneda3,moneda4)
h_der_izq(n)= h(moneda5,moneda 4)+ h(moneda3,moneda2)

Por ejemplo, para la configuración de la figura (a), el valor de h(n) sería:

h(ARARA)=min(2+1,1+2)=3

Contesta a las siguientes preguntas justificando las respuestas:

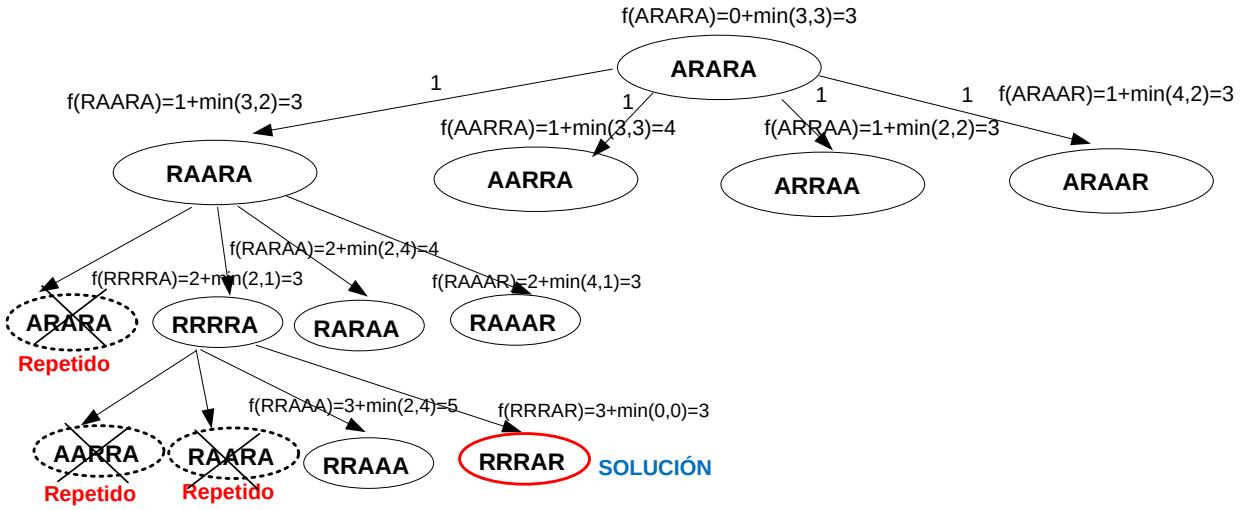
1) Genera y dibuja el árbol que resulta de aplicar un algoritmo de tipo A (f(n)=g(n)+h(n)) al estado inicial de la figura ¿a). Muestra claramente el valor f(n) de cada nodo, el orden de expansión de los nodos y el estado de la lista OPEN en cada iteración. ¿Qué solución encontrará este algoritmo y cuál es el coste de dicha solución? ¿La solución obtenida es de coste óptimo?
2) La heurística utilizada, ¿es admisible? Es decir, ¿se trata de una búsqueda A*? ¿por qué? Razona la respuesta.
3) Asumiendo que un usuario selecciona m=5 como máximo nivel de profundidad del árbol, ¿qué solución devolverá la estrategia de profundidad y cuál es el coste de dicha solución?.

NOTAS

a) Para generar los árboles aplicar este orden de movimientos: primero moneda 1 con moneda 2, luego moneda 2 con 3, luego moneda 3 con 4, y finalmente moneda 4 con 5.
*b) Si dos nodos tienen mismo valor f(n) expandir antes el nodo **más profundo**. Ante igualdad en el mismo nivel de profundidad, expandir primero **el más antiguo** creado.*
c) Aplicar SIEMPRE control de nodos repetidos.

Solución:

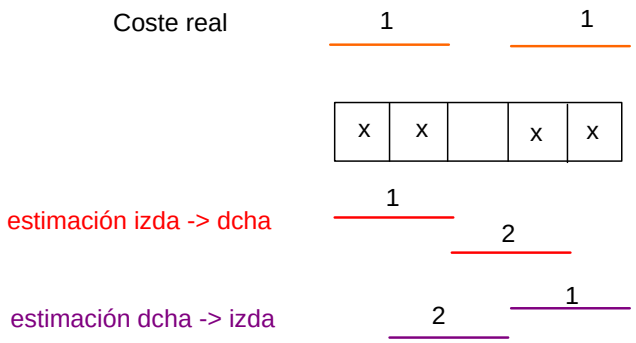
1)



1) OPEN= { ARARA (5)}
2) OPEN= { RAARA (3), ARRAA (3), ARAAR (3), AARRA (4)}
3) OPEN= { RRRRA(3), RAAAR (3), ARRAA (3), ARAAR (3), RARAA (4), AARRA (4)}
4) OPEN= { RRRAR (3), RAAAR (3), ARRAA (3), ARAAR (3), RARAA (4), AARRA (4), RRAAA (5)}
5) Se extrae RRRAR => Solución 3 movimientos. Coste = 3.

La solución que obtiene es cambiar **moneda1-moneda2, moneda2-moneda3, moneda4-moneda5**. Esta solución es de coste óptimo (3 movimientos).

2) En general la heurística estima bastante bien pero dependiendo de cómo estén las parejas contiguas de piezas descolocadas el análisis podría sobreestimar el coste. Dado que los análisis de izquierda a derecha y de derecha a izquierda siempre toman parejas fijas y no hacen todas las posibles combinaciones de parejas, podría darse el caso que las parejas estudiadas tuvieran siempre en cuenta una ficha que no está descolocada. Por ejemplo, supongamos que todas las piezas están mal colocadas excepto la del medio.

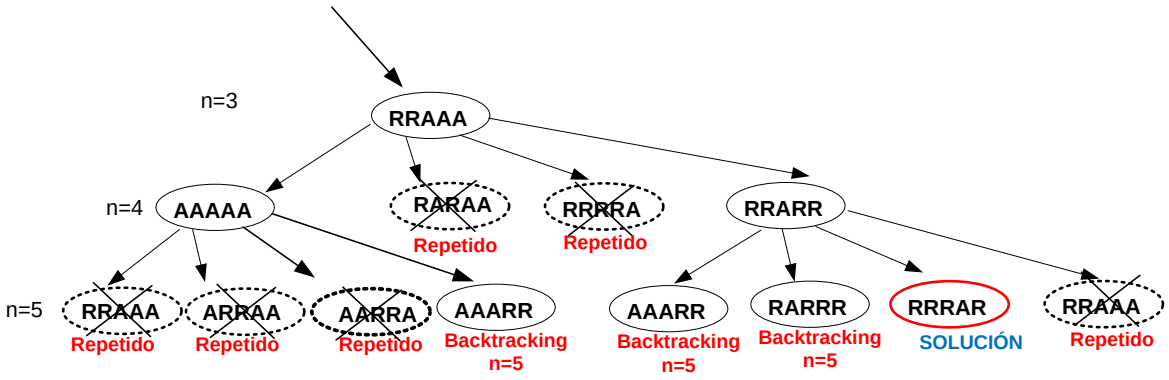


El mínimo de la estimaciones devolverá 3 porque siempre tienen en cuenta la moneda del medio en las estimaciones pero el coste real son 2 porque la moneda del medio no hace falta moverla. Un caso claro de esto se puede observar en el segundo hijo (empezando por la izquierda) del nodo raíz.

Por tanto, la heurística no es A*.

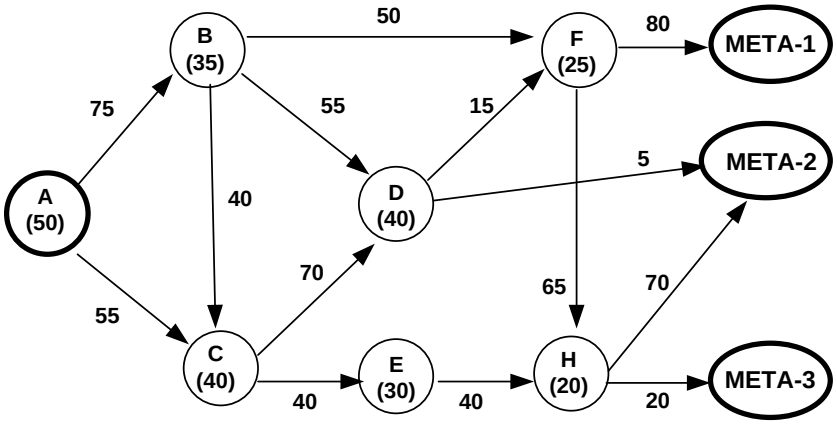
3) El árbol de la figura 1 sería equivalente al de una búsqueda en profundidad hasta m=3 por la primera rama del nodo raíz. Para continuar la búsqueda en profundidad hasta m=5 tendríamos que seguir por el nodo RRAAA, manteniendo los nodos OPEN del árbol de la figura 1.

La solución que encuentra está en el nivel n=5 (moneda1-moneda2, moneda2-moneda3, moneda3-moneda4, moneda4-moneda5, moneda3-moneda4)



Ejercicio 26

Sea el siguiente espacio de estados, donde se representa el estado inicial (A), los estados meta (Meta-1, Meta-2 y Meta-3), y los estados accesibles desde cada estado. En los arcos se etiqueta el coste del arco y en los nodos el valor de una función h(n).



a) Obtener la senda solución desde el estado A hasta un estado meta, aplicando los siguientes métodos de búsqueda, indicando claramente, para cada uno de ellos:

- El árbol de búsqueda que genera, remarcando la senda solución que obtiene
- Indicar, de forma clara, el orden en el que se han expandido los nodos.
- Indicar, de forma clara, la meta alcanzada y coste de la senda solución encontrada.

Métodos a aplicar (a igualdad de criterio, se expanden los nodos en orden alfabético):

a.1) Búsqueda voraz
a.2) Algoritmo A
a.3) Coste uniforme

b) Es posible conocer, con alguno de los métodos aplicados, la senda óptima? La función h(n) es admisible?

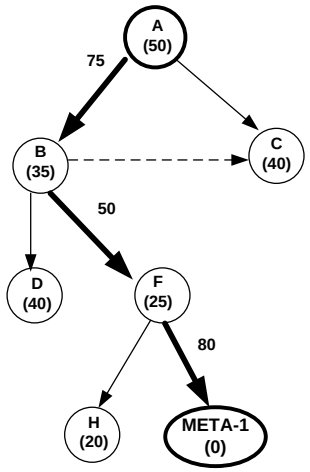
Solución

a.1) Voraz.

Orden de expansión de nodos: A, B, F, Meta-1.

Estado Objetivo alcanzado: Meta-1,

Coste Senda solución = 205

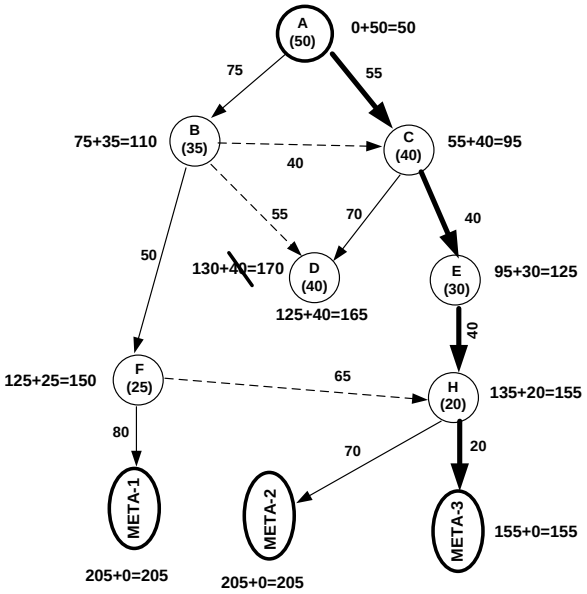


a.2) Algoritmo A

Orden de expansión de nodos: A, C, B, E, F, H, Meta3

Estado Objetivo alcanzado: Meta-3,

Coste Senda solución = 155

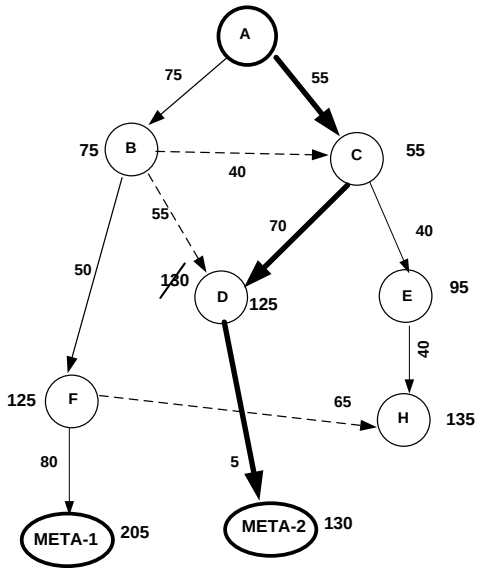


a.3) Coste Uniforme

Orden de expansión de nodos: A, C, B, E, D, F, Meta2

Estado Objetivo alcanzado: Meta-2,

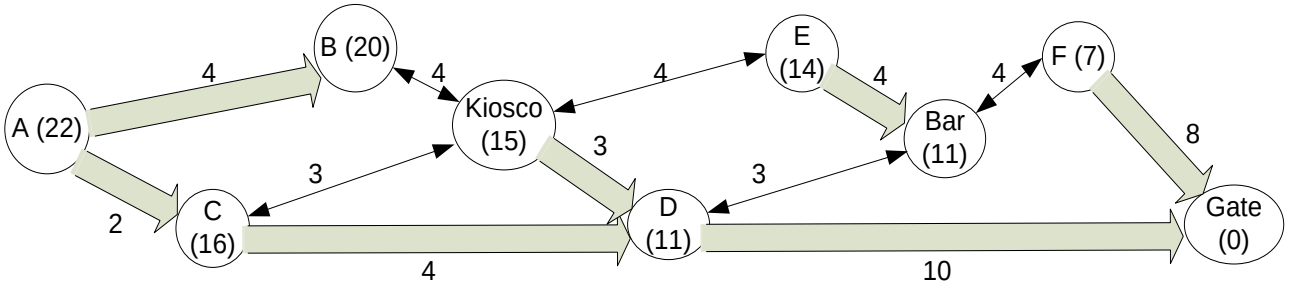
Coste Senda solución = 130



b) La senda óptima es la encontrada por la búsqueda de coste uniforme (coste 130). Claramente, la función h(n) no es admisible, ya que el algoritmo A no ha encontrado la solución óptima.

Ejercicio 27

Un viajero se encuentra en un aeropuerto (cuyo mapa simplificado se muestra en la figura) y debe llegar por el camino más corto desde el punto A a la puerta de embarque (Gate), pasando antes por el kiosco para comprar el periódico y por el bar para tomar un café. Las líneas gruesas representan cintas transportadoras de pasajeros (en un solo sentido) mientras que las líneas finas representan pasillos para ir de un punto a otro (en ambos sentidos). Los números sobre cada línea representan el coste de tomar esta cinta o pasillo para ir de un punto X a un punto Y. Los valores dentro de los nodos indican el coste estimado para ir desde el punto X a la puerta de embarque.



Se pide:

a) Obtener la senda solución desde el punto A hasta la puerta de embarque, aplicando un algoritmo A con control de nodos repetidos. Un estado de este problema consta de la siguiente información: el punto donde se encuentra el pasajero, si éste ya ha pasado por el kiosco o no, y si el pasajero ha pasado ya por el bar o no. Por tanto, dos estados se consideran repetidos si contienen exactamente la misma información respecto a estos tres elementos, independientemente de la senda utilizada para llegar a este estado. Es decir, el estado alcanzado tras el recorrido A→B→Kiosco→D y el estado alcanzado tras el recorrido A→C→Kiosco→D, se consideran estados repetidos ya que en ambos estados el viajero está en el punto D y ha visitado el kiosco, pero no el bar. Por tanto, una senda solución no lo será únicamente por alcanzar la puerta de embarque, sino que se deberá tener en cuenta que, en este camino, se ha visitado el kiosco y el bar. Se debe indicar claramente:

- El árbol de búsqueda que genera el algoritmo, remarcando la senda solución que obtiene.
- El orden en el que se han expandido los nodos.
- El coste de la senda solución encontrada.
- El número de nodos generados y expandidos.

Nota: <u>En el caso del mismo valor de f(n), se debe expandir en primer lugar el nodo más profundo</u>.

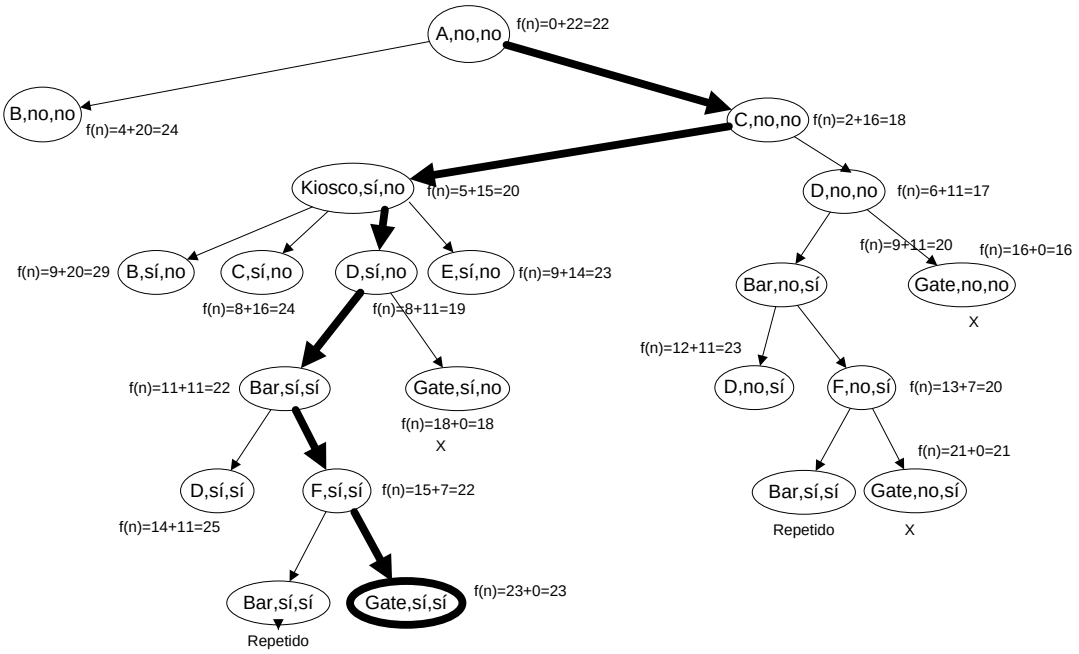
b) ¿Es admisible la heurística utilizada? Justificar la respuesta.

c) Aplicar de nuevo el algoritmo A con la siguiente heurística: dado un estado 'e', donde la posición actual del pasajero es *n*, el coste estimado para alcanzar la puerta de embarque será el valor indicado en el nodo *n* (h(n)) más 10, si no se ha visitado el kiosco, más 10, si no se ha visitado el bar. Por ejemplo, dado un estado 'e' con el recorrido A→B→Kiosco→D, h(e)= 11 (por estar en D) + 10 (por no haber visitado el bar) = 21.

- Se debe indicar claramente el árbol de búsqueda que genera el algoritmo, remarcando la senda solución que obtiene; el orden en el que se han expandido los nodos; el coste de la senda solución encontrada y el número de nodos generados y expandidos.
- Compara esta heurística con la utilizada en el apartado (a). ¿Es admisible esta nueva heurística? Justifica la repuesta.

Solución:

a) Árbol de búsqueda:
Representamos cada estado con la tripleta (posición del pasajero, kiosco visitado (sí/no), bar visitado (sí/no)).

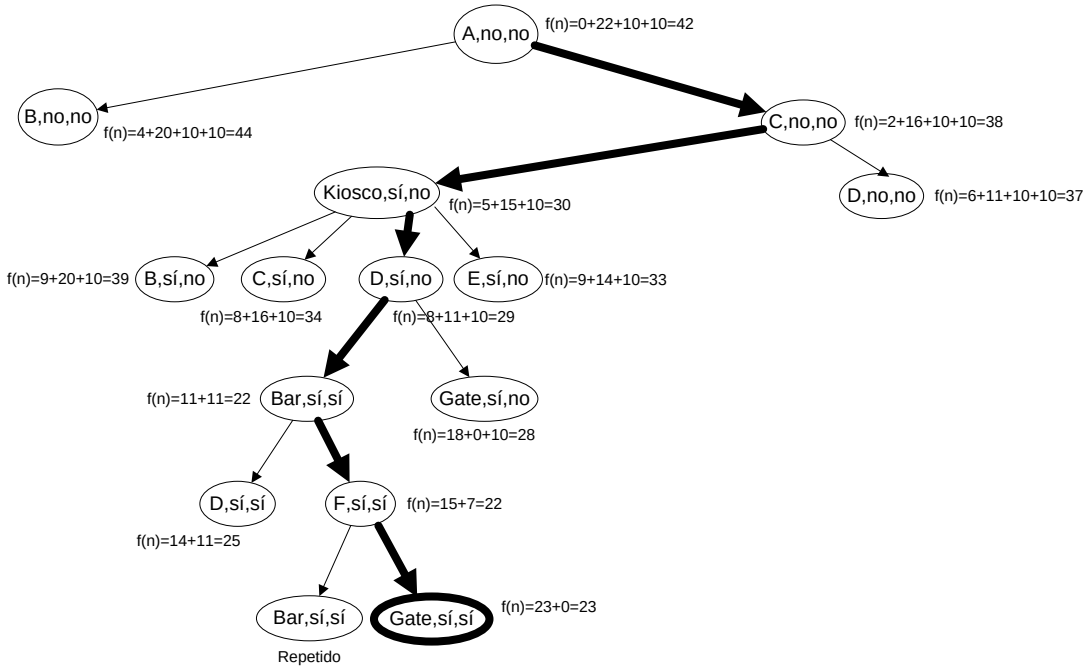


1. OPEN={(A,no,no,22)}. Se extrae el nodo A y se expande. Se añaden B y C.
2. OPEN={(C,no,no,18), (B,no,no,24)}. Se extrae el nodo C y se añaden Kiosco y D.
3. OPEN={(D,no,no,17), (Kiosco,sí,no,20), (B,no,no,24)}. Se extrae D y se generan Bar y Gate, que se añaden a OPEN).
4. OPEN={(Gate,no,no,16), (Bar,no,sí,20), (Kiosco,sí,no,20), (B,no,no,24)}. Se extrae Gate. No es meta, ya que no se han visitado ni el kiosco ni el bar. Desde éste, no se puede generar ningún otro nodo. Se extrae Bar y se generan D y F, que se añaden a OPEN. F se ordena antes que Kiosco, por estar en un nivel de profundidad mayor en el árbol.
5. OPEN={(F,no,sí,20), (Kiosco,sí,no,20), (D,no,sí,23), (B,no,no,24)}. Se extrae F y se genera Gate.
6. OPEN={(Kiosco,sí,no,20), (Gate,no,sí,21), (D,no,sí,23), (B,no,no,24)}. Se extrae Kiosco, y se generan B, C, D y E (ninguno es un nodo repetido).
7. OPEN={(D,sí,no,19), (Gate,no,sí,21), (D,no,sí,23), (E,sí,no,23), (C,sí,no,24), (B,no,no,24),(B,sí,no,29)}. Se extrae D y se generan Bar y Gate.
8. OPEN={(Gate,sí,no,18), (Gate,no,sí,21), (Bar,sí,sí,22), (D,no,sí,23), (E,sí,no,23), (C,sí,no,24), (B,no,no,24),(B,sí,no,29)}. Se extrae Gate y no es un estado meta porque no se ha visitado el bar. Se extrae el siguiente nodo Gate, que tampoco es un estado meta porque no se ha visitado el kiosco. Se extrae Bar y se generan D y F.
9. OPEN={(F,sí,sí,22), (D,no,sí,23), (E,sí,no,23), (C,sí,no,24), (B,no,no,24), (D,sí,sí,25), (B,sí,no,29)}. Se extrae F y se genera Bar (repetido) y Gate.
10. OPEN={(Gate,sí,sí,23), (D,no,sí,23), (E,sí,no,23), (C,sí,no,24), (B,no,no,24), (D,sí,sí,25), (B,sí,no,29)}. Se extrae Gate y SÍ es un nodo meta, ya que se han visitado tanto el kiosco como el bar.

El coste de esta solución es 23. Se han generado 21 nodos y se han expandido 13.

b) La heurística NO es admisible, ya que el valor indicado en cada nodo (estimación del coste desde ese nodo hasta Gate) NO es siempre menor o igual que el coste real de ir desde ese nodo hasta Gate. Por ejemplo, en el caso del punto D, la estimación es 11 cuando se podría llegar hasta Gate en 10.

c) Árbol de búsqueda con la nueva heurística:



1. OPEN={(A,no,no,42)}. Se extrae A y se generan B y C.
2. OPEN={(C,no,no,38), (B,no,no,44)}. Se extrae C y se generan Kiosco y D.
3. OPEN={(Kiosco,sí,no,30), (D,no,no,37), (B,no,no,44)}. Se extrae Kiosco y se generan B, C, D y E.
4. OPEN={(D,sí,no,29), (E,sí,no,33), (C,sí,no,34), (D,no,no,37), (B,sí,no,39), (B,no,no,44)}. Se extrae D y se generan Bar y Gate.
5. OPEN={(Bar,sí,sí,22), (Gate,sí,no,28), (E,sí,no,33), (C,sí,no,34), (D,no,no,37), (B,sí,no,39), (B,no,no,44)}. Se extrae Bar y se generan D y F.
6. OPEN={(F,sí,sí,22), (D,sí,sí,25), (Gate,sí,no,28), (E,sí,no,33), (C,sí,no,34), (D,no,no,37), (B,sí,no,39), (B,no,no,44)}. Se extrae F y se genera Bar (repetido) y Gate.
7. OPEN={(Gate,sí,sí,23), (D,sí,sí,25), (Gate,sí,no,28), (E,sí,no,33), (C,sí,no,34), (D,no,no,37), (B,sí,no,39), (B,no,no,44)}. Se extrae Gate y es un nodo meta.

La solución encontrada es la misma que aplicando la primera heurística. Se generan 15 nodos y se expanden 7. Esta heurística NO es admisible, ya que, por ejemplo, el valor de h(n) del nodo A es 42, cuando el coste de la solución es 23. Sin embargo, esta heurística permite enfocar la búsqueda claramente hacia la solución, porque asigna un menor valor a aquellos nodos que ya han visitado el kiosco y/o el bar y que, por tanto, están más próximos a la solución.

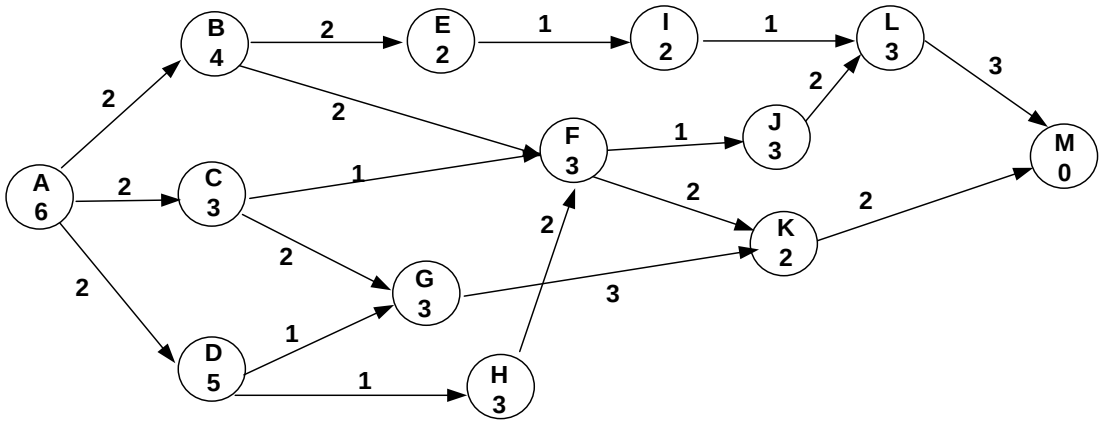
Ejercicio 28

Sea el grafo que se muestra en la figura, donde el estado inicial es A y el estado final que se quiere alcanzar es M. Los valores dentro de los nodos indican el valor h(n) de cada nodo. Los valores de las aristas representan el coste para ir de un nodo a otro.

a) Aplicar un algoritmo A con control de nodos repetidos. Mostrar el árbol de búsqueda que genera el algoritmo, indicar claramente el valor f(n) de cada nodo y el orden en el que se expanden los nodos. Señala cuál es el coste del camino encontrado, el número de nodos generados y expandidos. ***NOTA**: Si dos nodos tienen el mismo valor f(n), expandir antes el nodo alfabéticamente anterior.*

b) La solución encontrada por el algoritmo A, ¿es la solución óptima? ¿La función h(n) es admisible? Justifica las respuestas.

c) La función h(n) es no monótona. Muestra los valores del grafo y/o árbol que indican que h(n) es no monótona y explica las consecuencias de la ausencia de esta propiedad en el árbol desarrollado.



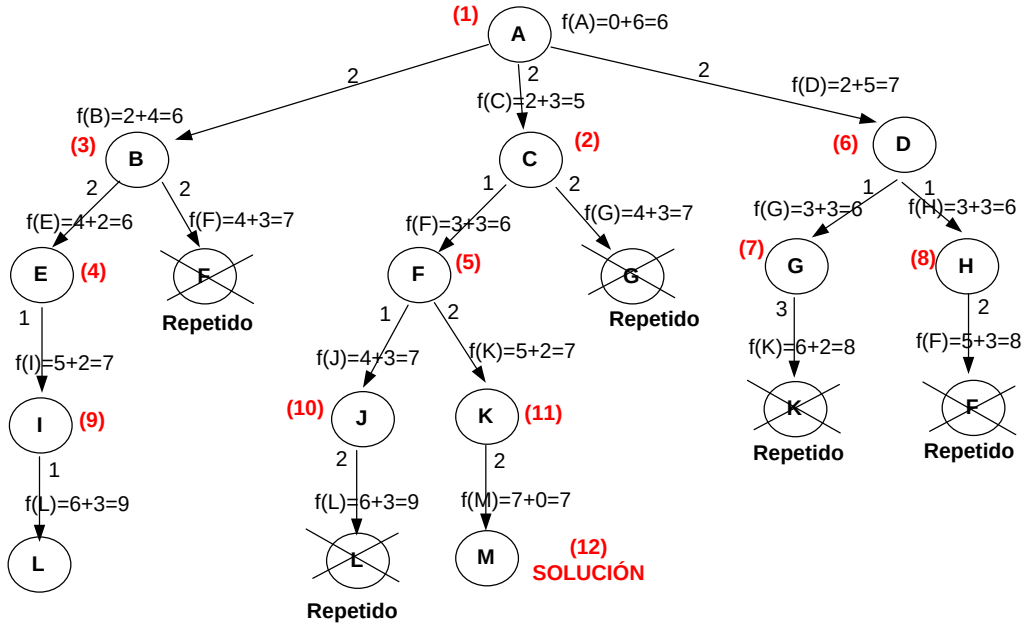
d) Muestra el árbol de búsqueda que resulta de aplicar una búsqueda primero en profundidad con un límite máximo de profundidad m=4 y con control de nodos repetidos. Muestra el estado de las listas OPEN y CLOSED en cada iteración. ¿Qué solución devolverá la estrategia de profundidad y cuál es el coste de dicha solución?. ***NOTA****: Si dos nodos están en el mismo nivel de profundidad, expandir antes el nodo alfabéticamente anterior.*

Solución

Se genera un total de 18 nodos. Se expande un total de 12 nodos.
La solución es A-C-F-K-M. El coste de la solución es 7.

Observaciones: se producen dos tipos de casos de **nodos repetidos**, aquellos nodos que cuando se generan no mejoran el coste del nodo existente, y aquellos nodos que cuando se generan, mejoran el coste del nodo existente. Por ejemplo:

a) el nodo repetido F sucesor de B se genera cuando ya existe en la lista OPEN el nodo F sucesor de C. Por tanto, el nuevo nodo F generado no se inserta en OPEN
b) el nodo repetido G sucesor de C se detecta cuando se genera el nodo G sucesor de D, el cual tiene un coste mejor. Por tanto, en este caso, se elimina de OPEN el nodo G sucesor de C y se inserta el nuevo nodo G.



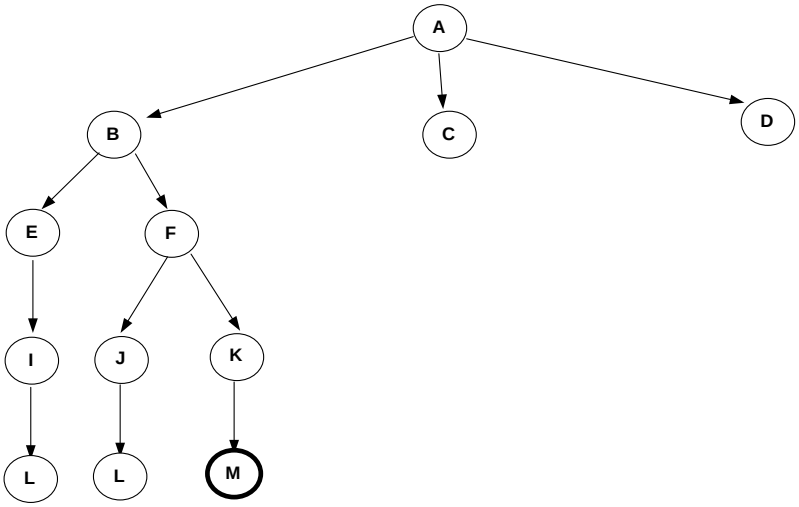
b) Sí, es la solución óptima. La función h(n) es admisible, todas las estimaciones de los nodos son menores ó iguales que el coste real. Por ejemplo, h(L)=3, h*(L)=3, h(K)=2, h*(K)=2, h(J)=3, h*(J)=5.

c) En el grafo. Se puede observar que la función h(n) no es consistente con los costes en 2 puntos: A –> C y D → G. Por ejemplo, de A a C, la heurística desciende en tres unidades, mientras que el coste de la arista es solo 2.

En el árbol. Se puede observar que la función f(n) no es monotónicamente no decreciente, es decir, los valores de f(n) pueden ahora decrecer a lo largo del árbol. Esto se observa en tres puntos: de f(A) a f(C) (el valor pasa de 6 a 5), y de f(D) a f(G) y f(H) (el valor pasa de 7 a 6).

Una de las consecuencias de la no monotonía es que podría expandirse un nodo y posteriormente encontrar otro camino de menor coste a dicho nodo. En el árbol ocurre que se expande el nodo F en el orden (5), y por consiguiente se introduce en la lista CLOSED, y posteriormente se vuelve a generar el nodo F como sucesor de H. En este caso, el camino a F a través de H (coste=5) no es mejor que el que ya se tiene (el coste de F a través de C es 3) y por tanto se descarta el nuevo nodo generado. Pero si el nuevo camino de A a F a través de D y H fuera mejor que el del nodo ya expandido e introducido en CLOSED, habría que re-expandir otra vez el subárbol a partir del nodo F con su nuevo valor de g(n).

d) PROFUNDIDAD m=4



1. OPEN={A} se extrae el nodo A, no es solución y el nivel no es 4. Se inserta en CLOSED={A} y se expande A.
2. OPEN={B C D} se extrae el nodo B, no es solución y el nivel no es 4. Se inserta en CLOSED={A B} y se expande B.
3. OPEN={E F C D} se extrae el nodo E, no es solución y el nivel no es 4. Se inserta en CLOSED={A B E} y se expande E.
4. OPEN={I F C D} se extrae el nodo I, no es solución y el nivel no es 4. Se se inserta en CLOSED={A B E I} y se expande I.
5. OPEN={L F C D} se extrae el nodo L, no es solución y hemos llegado al nivel 4. Backtracking.
6. OPEN={F C D} Se extrae el nodo F, no es solución, y el nivel no es 4. Se inserta en CLOSED y se expande. Se actualiza la lista CLOSED de modo que el nodo anterior en CLOSED sea el nodo padre de F; CLOSED={A B F}
7. OPEN={J K C D} se extrae el nodo J, no es solución y el nivel no es 4. Se inserta en CLOSED={A B F J} y se expande.
8. OPEN={L K C D} se extrae el nodo L, no es solución y hemos llegado al nivel 4. Backtracking.
9. OPEN={K C D} se extrae el nodo K, no es solución, y el nivel no es 4. Se inserta en CLOSED y se expande. Se actualiza la lista CLOSED de modo que el nodo anterior en CLOSED sea el nodo padre de K; CLOSED={A B F K}
10. OPEN={M C D} se extrae M, es solución y lo insertamos en **CLOSED={A B F K M}**

La solución es el camino indicado en CLOSED. El coste de la solución es 8.

Ejercicio 29

Dada la siguiente configuración inicial de un tablero lineal, donde hay dos piezas negras (N), dos piezas blancas (B) y una casilla vacía:

1	2	3	4	5
N	N	B	B	

Estado inicial

Considera las siguientes acciones:

a) Una pieza se puede mover a la casilla vacía adyacente con un coste igual a 1
b) Una pieza puede saltar sobre otra (solo sobre una) para colocarse en la casilla vacía con un coste igual a 1. Si la pieza que salta y la pieza sobre la que se salta son de diferente color entonces la pieza sobre la que se salta cambia de color.

El objetivo es que todas las piezas del tablero sean de color negro pudiendo estar en cualquier posición. Es decir, la posición final de la casilla vacía es indiferente.

1) Considera la siguiente función heurística: h(n)= $\sum_{i=1}^{5} b(i) * c(i)$ donde:

- b(i)=1, si la casilla i contiene una pieza B. b(i)=0, en cualquier otro caso
- c(i)=1, si una de las casillas adyacentes (i-1) o (i+1) es la casilla vacía. c(i)=2, en cualquier otro caso

En otras palabras, la heurística solo considera las piezas B, anotando un punto de coste si la casilla vacía es adyacente a la pieza B ó 2 puntos de coste en caso contrario.

a) Dibuja el árbol de búsqueda que resulta de aplicar un algoritmo A con f(n)=g(n)+h(n).
b) Indica sobre el árbol el *orden de expansión* de los nodos.
c) Muestra la solución encontrada y el coste de dicha solución.

NOTAS: **(a)** Evitar nodos repetidos. **(b)** Si dos nodos tienen el mismo valor de f(n) expandir antes el nodo más profundo.

2) La función h(n) descrita en 1), ¿es admisible? Razona la respuesta.

3) Sin necesidad de desarrollar un árbol, ni realizar ninguna búsqueda adicional, contesta a las siguientes preguntas razonadamente:

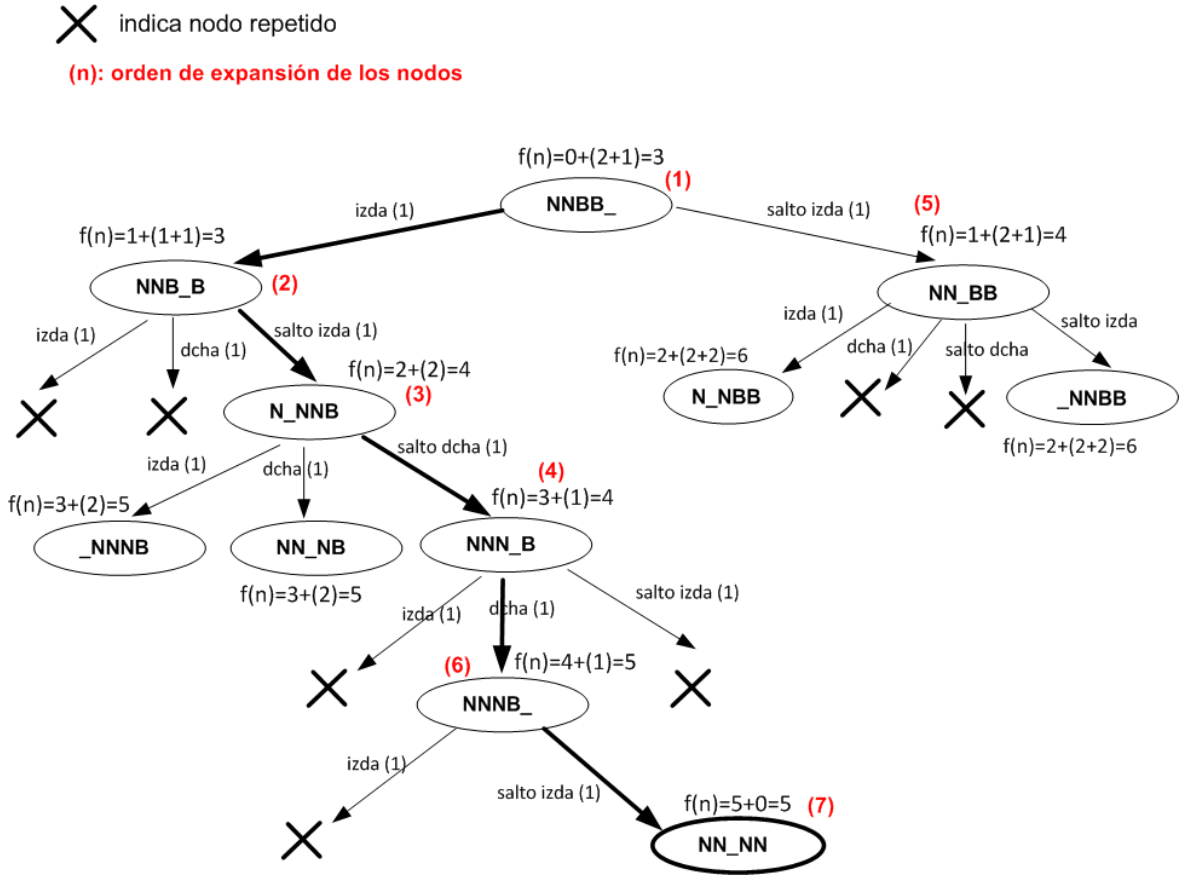
a) ¿Cuántas iteraciones (árboles de búsqueda) tendría que realizar la estrategia Búsqueda por Profundización Iterativa para encontrar una solución?
b) Si el coste de la operación de salto fuera 2 en lugar de 1, ¿encontraría el algoritmo A del apartado 1 la misma solución?

Solución

Cuestión 1

Planteamos los movimientos en términos de la casilla vacía (mover vacío a la izda, mover vacío a la dcha, saltar vacío a la izda, saltar vacío a la dcha).

La solución encontrada es la que se muestra en negrita en la figura. El coste de la solución es 5.



Cuestión 2

Como se puede observar en el árbol, las estimaciones de todos los nodos son menores que el coste real y la solución que ha encontrado tiene un coste de 5 pasos. En este caso, se puede comprobar que se trata de la solución óptima para el estado inicial dado pero no se puede garantizar que el algoritmo encuentre siempre la óptima porque la heurística no es admisible. Particularmente, se puede encontrar al menos un caso en el que la estimación supera el coste real: por ejemplo, para el nodo n= NBNB_ donde h(n)=3 y h*(n)=2.

Cuestión 3

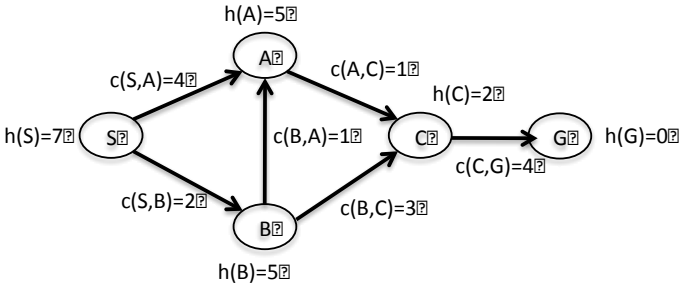
a) Asumiendo que la solución requerirá un mínimo de 5 pasos para el estado inicial dado, una búsqueda por profundización iterativa tendrá que realizar iteraciones para nivel=0,1,..,5.

b) Como la heurística no cambia, para todos los nodos del árbol para los que se cumple h(n)<h*(n) se seguirá cumpliendo ahora h(n)<<h*(n). Particularmente, la sobreestimación que se ha comentado en la cuestión 2 ahora no se produciría con los nuevos costes (por ejemplo, para el nodo n= NBNB_ donde h(n)=3, ahora h*(n)=4), por lo que la heurística sería admisible en este caso. Podría por tanto encontrar la misma solución (ahora con coste=8) o bien otra solución distinta con el mismo coste que la óptima.

Ejercicio 30

Dado el espacio de estados de la figura desde un nodo-origen (S) a nodo-meta (G), donde h(x) denota el valor de aplicar la función heurística 'h' al estado 'x', c(x,y) denota el coste de la acción que permite ir del estado 'x' al estado 'y'. Se pide:

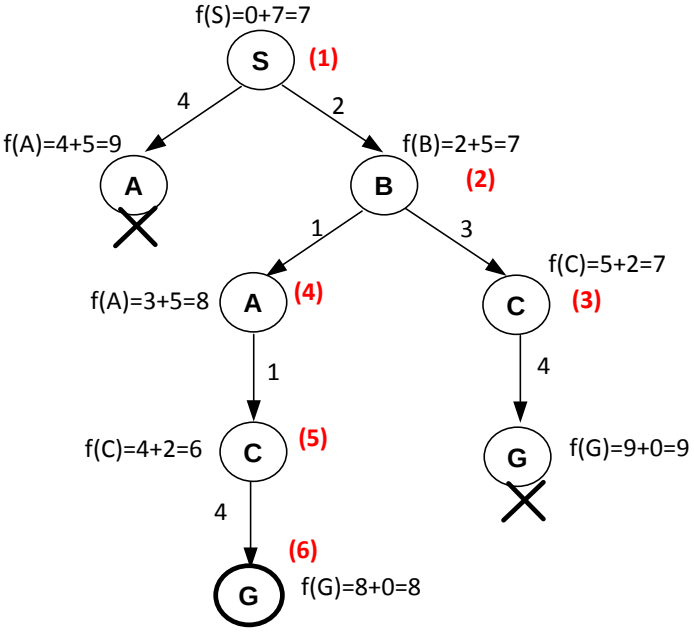
a) Realizar una búsqueda tipo A. Obtiene la senda óptima de (S) a (G)?
b) La heurística h(n) es admisible? Y consistente?



Solución

a) Búsqueda en árbol (TREE-SEARCH)

(n): orden de expansión de los nodos



Si se hace una búsqueda en grafo (GRAPH-SEARCH) manteniendo la lista de nodos cerrados CLOSED entonces al encontrar el nodo repetido C con f(C)=6 podrían suceder dos cosas:

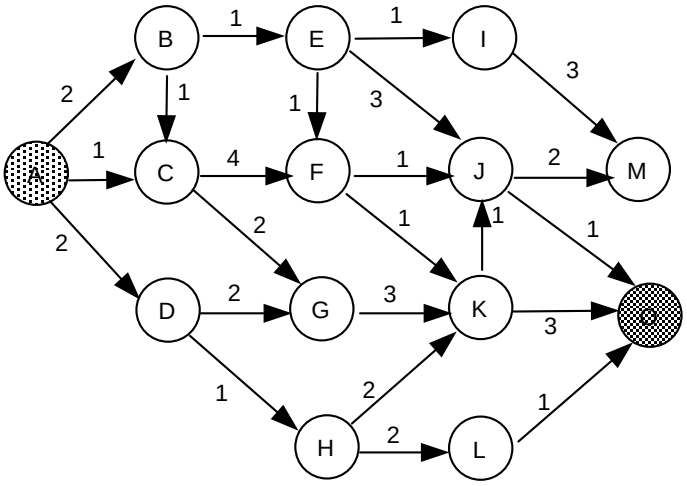
1. Se ignora este nodo por estar repetido en la lista CLOSED. En este caso expandiríamos a continuación el nodo G con valor f(G)=9 y no se encontraría la solución óptima.
2. Se comprueba que el coste de la senda del nodo S a C por el nodo A es mejor que la senda previa por B. Se rehacen los punteros de (C) desde (B) hacia el nodo (A). El nodo G que está en la lista OPEN actualiza su valor a f(G)=8+0=8, expandiéndolo a continuación y encontrando así la solución óptima.

b) la heurística es admisible porque se cumple para todos los nodos que h(n)<=h*(n). Por ejemplo: h(S)=7, h*(S)=8, h(A)=5, h*(A)=5, etc. Pero no es consistente. Esto puede observarse de dos modos distintos:

- bien en el árbol desarrollado donde se ve que el valor de 'f' es decreciente del nodo A al nodo C
- bien comprobando que la propiedad h(n)<=h(n')+c(n,n') no se cumple entre A y C porque h(A)>=h(C)+c(A,C);

Ejercicio 31 (Examen 2013)

Sea el siguiente grafo donde cada arco indica su coste y la tabla indica la estimación del coste 'h' hasta la solución. El nodo 'A' es el estado inicial y el nodo 'O' es el estado final.



n	A	B	C	D	E	F	G	H	I	J	K	L	M	O
h(n)	6	5	6	6	3	5	5	4	8	2	2	1	5	0

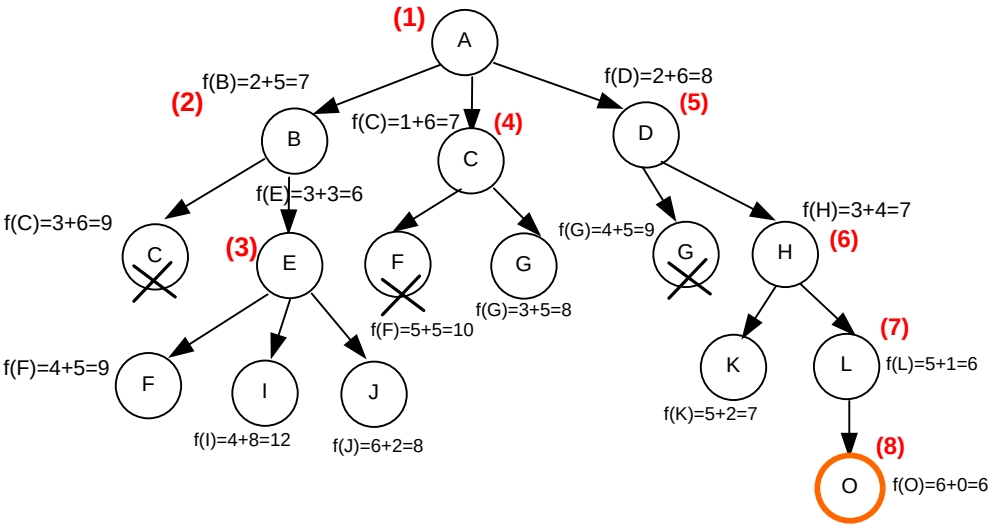
1. Muestra el árbol de búsqueda que resultaría de la aplicación de un algoritmo de tipo A (f(n)=g(n)+h(n)). Aplicar la versión grafo del algoritmo evitando nodos repetidos. Indica al final el número de nodos generados y expandidos. Indica claramente el valor de la función de evaluación (f(n)) en cada nodo y el orden de expansión de los nodos. Si dos nodos tienen el mismo valor de f(n), expandir antes el nodo alfabéticamente anterior.

2. De acuerdo a los datos del problema y el árbol desarrollado en el apartado anterior: ¿Devuelve el algoritmo la solución óptima? ¿La función heurística es admisible? ¿Y consistente (monótona)? Justifica todas las respuestas.

3. Sin desarrollar un árbol de búsqueda, contesta a las siguientes preguntas y justifica la respuesta:
   a. ¿Qué estrategia utilizarías si queremos encontrar la solución que atraviese el menor número de nodos? Indica una solución que encontraría esta estrategia.
   b. La aplicación de un algoritmo de profundización iterativa, ¿en qué nivel del árbol encontraría la solución?¿por qué?
   c. Si aplicamos un algoritmo en profundidad y no establecemos un límite máximo de profundidad, ¿encontraría el algoritmo una solución? ¿por qué?

Solución

1) Se generan 16 nodos y se expanden 8.



2)
Sí, el algoritmo devuelve la solución óptima. Se puede ver en el grafo que no hay solución de menor coste.

No, la función no es admisible. Esto se puede observar en algunos casos, por ejemplo: h(H)=4, h*(H)=3; h(J)=2, h*(J)=1.

No, si la función no es admisible no puede ser consistente ya que la consistencia es una propiedad más fuerte que la admisibilidad. Esto se puede observar, por ejemplo, en: h(H)=4, h(L)=1, c(H,L)=2, por tanto no se cumple que h(H)<= h(L)+c(H,L). Esto también se puede observar entre el nodo B y E, D y H ó H y L, prueba de lo cual es que la función f(n) es decreciente en el árbol desarrollado.

3)

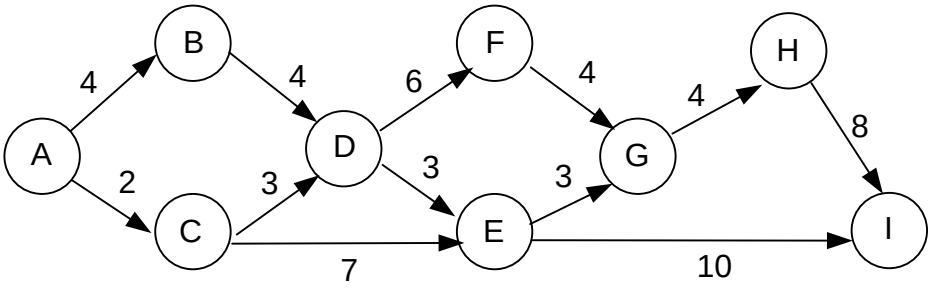
a) Para encontrar la solución más corta aplicaríamos la estrategia de anchura. Una solución que encontraría anchura es A-B-E-J-0, que es la solución más corta posible para este problema (4 pasos).

b) El algoritmo Profundización Iterativa encontraría la solución en el nivel 4 porque PI encuentra una solución de la misma calidad que anchura.

b) Sí, el algoritmo encontraría una solución porque el espacio de estados es finito y no contiene ciclos por tanto la búsqueda en profundidad no se quedaría estancada en un espacio de búsqueda infinito.

Ejercicio 32 (Examen 2013)

Sea el siguiente grafo donde cada arco indica su coste y la tabla indica la estimación del coste 'h' hasta la solución. El nodo 'A' es el estado inicial y el nodo 'I' es el estado final.



n	A	B	C	D	E	F	G	H
h(n)	15	14	13	12	11	10	9	8

1) Asumiendo que se aplica un algoritmo en anchura, que ante el mismo valor de la función 'f(n)' se expande antes el nodo alfabéticamente anterior y que se realiza control de nodos repetidos (descartar los nodos más profundos o nodos expandidos con anterioridad en caso del mismo nivel de profundidad), contesta a las siguientes preguntas:

   a) Escribe los nodos del camino solución desde el nodo A hasta el nodo I.
   A C E I

   b) ¿Cuántos nodos se han generado en total y cuántos nodos se han expandido en el árbol?
   12 nodos generados (3 de ellos repetidos), y 8 nodos expandidos (incluido el nodo I)

2) Asumiendo que se aplica la versión grafo de un algoritmo A con control de nodos repetidos, contesta a las siguientes preguntas:

   a) Escribe los nodos del camino solución desde el nodo A hasta el nodo I y el coste de dicho camino solución; el camino encontrado, ¿es la solución óptima?
   A C D E I; el coste del camino solución es 18. Sí, es la solución óptima porque no existe un camino de menor coste entre el nodo A y el nodo I

   b) ¿Cuántos nodos se han generado en total y cuántos nodos se han expandido en el árbol?
   10 nodos generados (2 de ellos repetidos), 6 nodos expandidos (incluido el nodo I)

   c) Indica los nodos expandidos y su orden de expansión.
   A C D B E I

3) Responde brevemente a las siguientes preguntas justificando las respuestas:

   a) La función heurística de este problema, ¿es admisible?¿por qué?
   No, porque h(E)=11 y h*(E)=10

   b) La función heurística de este problema, ¿es consistente?¿por qué?
   No, porque no se cumple h(E) <= h(I)+c(E,I), o sea, no se cumple 11 <= 0 + 10

   c) Si aplicamos un algoritmo de profundización iterativa en este problema, ¿qué solución encontraría? Indica los nodos del camino solución así como el número total de nodos generados.

   Encontraría la misma solución que anchura, A C E I.

   Se generan 4 árboles en profundidad, desde el nivel 0 al nivel 3. El número de nodos generados sería, contabilizando los nodos por árbol generado: 1+3+6+12=22; o también, contabilizando el número de veces que se generan los nodos de cada nivel: 4 veces * 1 (el nodo de nivel 0) + 3 veces * 2 (dos nodos en nivel 1) + 2 veces * 3 (tres nodos en nivel 2) + 1 vez *6 (cuatro nodos de nivel 3) = 22.