Ejercicios Resueltos sobre Problemas de Búsqueda en Espacios de Estados

Severino Fernández Galán
Departamento de Inteligencia Artificial
Escuela Técnica Superior de Ingeniería Informática
Universidad Nacional de Educación a Distancia (UNED)
Correo-e: seve@dia.uned.es

Primera versión: Abril 2012
Última versión: Abril 2017

1. INTRODUCCIÓN

El presente texto es un compendio de ejercicios resueltos sobre diferentes problemas de búsqueda en espacios de estados. Ha sido escrito a partir de ejercicios diseñados por el autor para alumnos de la asignatura:

"Fundamentos de Inteligencia Artificial"
Grado en Ingeniería Informática y Grado en Ingeniería en Tecnologías de la Información
Universidad Nacional de Educación a Distancia

El texto está organizado por años: se parte del año 2012 y cada año corresponde a la realización de seis ejercicios. La estructura de ejercicios se repite para cada año, aunque con enunciados diferentes.

Cualquier persona interesada en comunicar erratas o en utilizar parte del contenido del presente texto puede ponerse en contacto con el autor en la dirección de correo electrónico arriba indicada.

AÑO 2012

EJERCICIO 1:

Dibuje mediante un grafo dirigido o describa detalladamente mediante una tabla el espacio de estados (o espacio de búsqueda) completo para el *problema del barquero*. Para ello especifique: el conjunto de todos los estados posibles, el estado inicial, el o los estados meta, los operadores aplicables a cada estado y el coste asociado a cada operador. En el problema del barquero, un barquero se encuentra en la orilla de un río con un puma, una cabra y una lechuga. Su intención es trasladar los tres elementos anteriores a la otra orilla por medio de un bote con capacidad para dos (el propio barquero y uno cualquiera de los elementos mencionados). La dificultad reside en que si el puma se quedara solo con la cabra entonces la devoraría y lo mismo sucedería si la cabra se quedara sola con la lechuga. ¿Cuál es la solución menos costosa para el problema del barquero?

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 1:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- Los estados se especifican correctamente: 2.5 puntos
- Los operadores se especifican correctamente: 2.5 puntos
- Los costes de los operadores se especifican correctamente: 1 punto
- El espacio de búsqueda se dibuja (mediante un grafo dirigido) o se describe (mediante una tabla) correctamente: 3 puntos
- La solución de menor coste se especifica correctamente: 1 punto

SOLUCIÓN DEL EJERCICIO 1 (por Severino Fernández Galán):

Para representar los **estados** posibles utilizaremos la siguiente notación: “B” se corresponde con el barquero, “C” con la cabra, “L” con la lechuga y “P” con el puma. Además, el estado del problema lo representaremos como (X - Y), donde X contiene los elementos de {B, C, L, P} que están en la orilla izquierda e Y contiene el resto de elementos, que están en la orilla derecha. Por ejemplo, (B - CLP) representa el estado en que el barquero está en la orilla izquierda y el resto de elementos están en la orilla derecha.

Supongamos que inicialmente todos los elementos están en la orilla izquierda y queremos pasarlos a la orilla derecha. Por tanto, el **estado inicial** será (BCLP -), mientras que el único **estado meta** será (- BCLP). Obsérvese que los elementos de una misma orilla los ordenamos alfabéticamente por sencillez en la exposición.

Dado un estado cualquiera, el número de **operadores** aplicables al mismo coincide con el número de elementos situados en la orilla del barquero, incluido el propio barquero. Por ejemplo, los operadores aplicables al estado (CL - BP) son dos: que el barquero se traslade a la otra orilla o que el barquero se traslade con el puma a la otra orilla. Denominaremos “B”, “BC”, “BL”, “BP” a los cuatro operadores posibles: que el barquero se traslade a la otra orilla sólo, con la cabra, con la lechuga o con el puma, respectivamente. Se puede considerar que cada operador tiene **coste** unidad, ya que buscamos la solución que realice menos viajes de orilla a orilla con el bote.

En la tabla 1.1 figuran los estados posibles del sistema, los operadores que se pueden aplicar a cada estado posible y los estados resultantes de aplicar cada operador.

ESTADOS	OPERADORES			
	B	BC	BL	BP
(BCLP -)	(CLP - B)	(LP - BC)	(CP - BL)	(CL - BP)
(CLP - B)	Estado no permitido			
(BLP - C)	(LP - BC)	No aplicable	(P - BCL)	(L - BCP)
(BCP - L)	(CP - BL)	(P - BCL)	No aplicable	(C - BLP)
(BCL - P)	(CL - BP)	(L - BCP)	(C - BLP)	No aplicable
(LP - BC)	(BLP - C)	(BCLP -)	No aplicable	No aplicable
(CP - BL)	Estado no permitido			
(CL - BP)	Estado no permitido			
(BP - CL)	Estado no permitido			
(BL - CP)	Estado no permitido			
(BC - LP)	(C - BLP)	(- BCLP)	No aplicable	No aplicable
(P - BCL)	(BP - CL)	(BCP - L)	(BLP - C)	No aplicable
(L - BCP)	(BL - CP)	(BCL - P)	No aplicable	(BLP - C)
(C - BLP)	(BC - LP)	No aplicable	(BCL - P)	(BCP - L)
(B - CLP)	Estado no permitido			
(- BCLP)	Estado META			

Tabla 1.1: Estados posibles, operadores aplicables a cada estado y estados resultantes de aplicar cada operador para el problema del barquero.

De forma alternativa, la información incluida en la tabla 1.1 se puede representar mediante un grafo dirigido tal como se muestra en la figura 1.1. En dicha figura, los estados no permitidos se han marcado con líneas discontinuas y se ha utilizado línea doble para marcar el estado inicial y el estado meta. Por otra parte, obsérvese que el estado (B - CLP) queda aislado en el espacio de búsqueda.

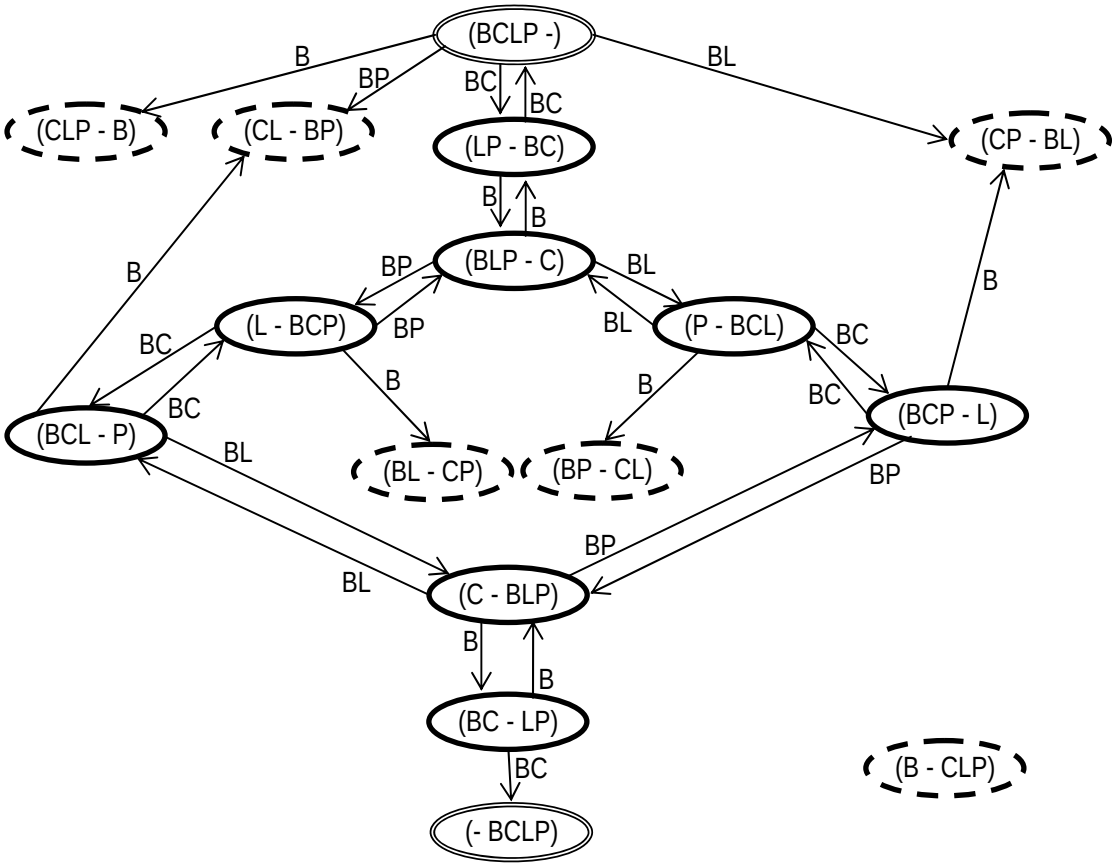


Figura 1.1: Grafo dirigido que representa el espacio de búsqueda para el problema del barquero.

La solución menos costosa para el problema del barquero (o, en este caso, las dos soluciones menos costosas) consiste en: llevar la cabra a la otra orilla, regresar solo, llevar la lechuga o el puma a la otra orilla, regresar con la cabra, llevar el elemento que estaba solo en la orilla original a la otra orilla, regresar solo y llevar la cabra a la otra orilla.

EJERCICIO 2:

Considere el espacio de búsqueda de la figura 2.1, que tiene forma de árbol, donde el nodo raíz del árbol es el nodo inicial, existe un único nodo meta y cada operador tiene asociado un coste. Explique razonadamente en qué orden se expandirían los nodos de dicho árbol de búsqueda a partir de cada uno de los métodos siguientes de búsqueda sin información del dominio:

1. Búsqueda Primero en Anchura (de izquierda a derecha)
2. Búsqueda Primero en Profundidad (de derecha a izquierda)
3. Búsqueda de Coste Uniforme
4. Búsqueda en Anchura Iterativa (de derecha a izquierda)
5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

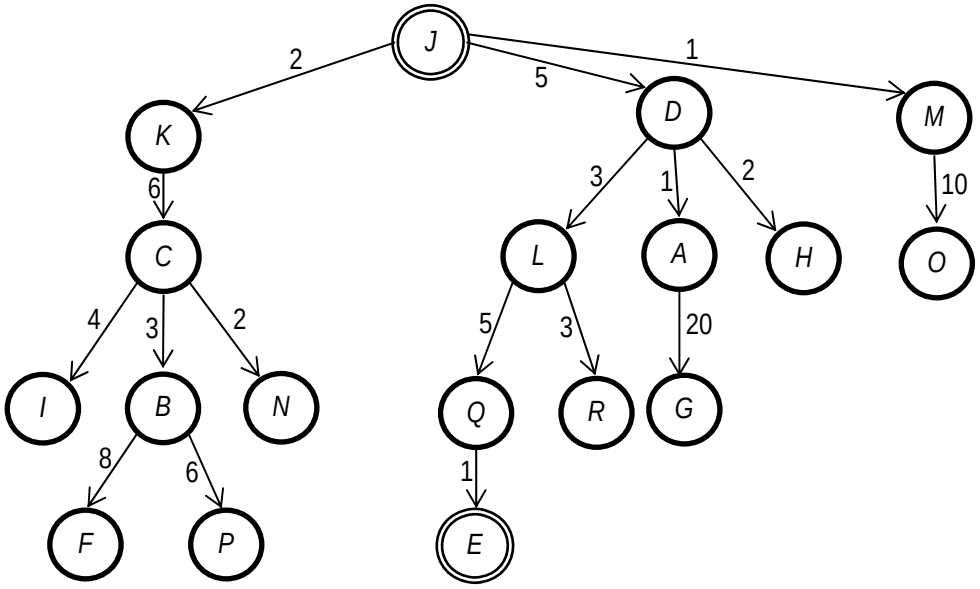


Figura 2.1: Árbol de búsqueda en el que el nodo inicial es *J*, el nodo meta es *E* y el coste de cada operador aparece al lado del arco que lo representa.

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 2:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- Cada uno de los cinco apartados, correspondiente a un método de búsqueda concreto, se puntúa sobre 2 puntos.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no expande los nodos en el orden debido: la puntuación del apartado bajaría 1.6 puntos si el orden dado como respuesta varía significativamente del correcto, mientras que si el orden dado como respuesta varía del correcto como consecuencia de un despiste no conceptual entonces la puntuación del apartado bajaría sólo 0.4 puntos en vez de los 1.6 puntos mencionados.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no finaliza cuando es debido, la puntuación del apartado bajaría 0.4 puntos. (Esto es independiente del orden de expansión de nodos dado como respuesta, que ya ha sido valorado anteriormente con un máximo de 1.6 puntos.)

SOLUCIÓN DEL EJERCICIO 2 (por Severino Fernández Galán):

1. Búsqueda Primero en Anchura (de izquierda a derecha)

Este algoritmo explora el árbol de búsqueda por niveles de profundidad, así que el orden de expansión de los nodos de izquierda a derecha sería el reflejado en la figura 2.2.

2. Búsqueda Primero en Profundidad (de derecha a izquierda)

Este algoritmo explora el árbol de búsqueda bajando de nivel siempre que sea posible. Si no es posible, se sube al nodo más cercano al nodo actual desde el que poder seguir bajando de nivel. El orden de expansión de los nodos de derecha a izquierda según este algoritmo se dibuja en la figura 2.3.

3. Búsqueda de Coste Uniforme

Este algoritmo explora el árbol de búsqueda expandiendo aquel nodo disponible cuyo coste al nodo inicial sea el menor. El orden de expansión de nodos según este criterio se indica en la figura 2.4. Obsérvese que después de la sexta expansión (nodo *H*), hay un empate entre los nodos *C* y *L*, cuyos costes al nodo inicial son iguales a 8 en ambos casos. Nosotros hemos elegido el nodo *C* para realizar la séptima expansión, pero también se podría haber elegido el nodo *L*. Los empates posteriores también los deshacemos de esta forma arbitraria.

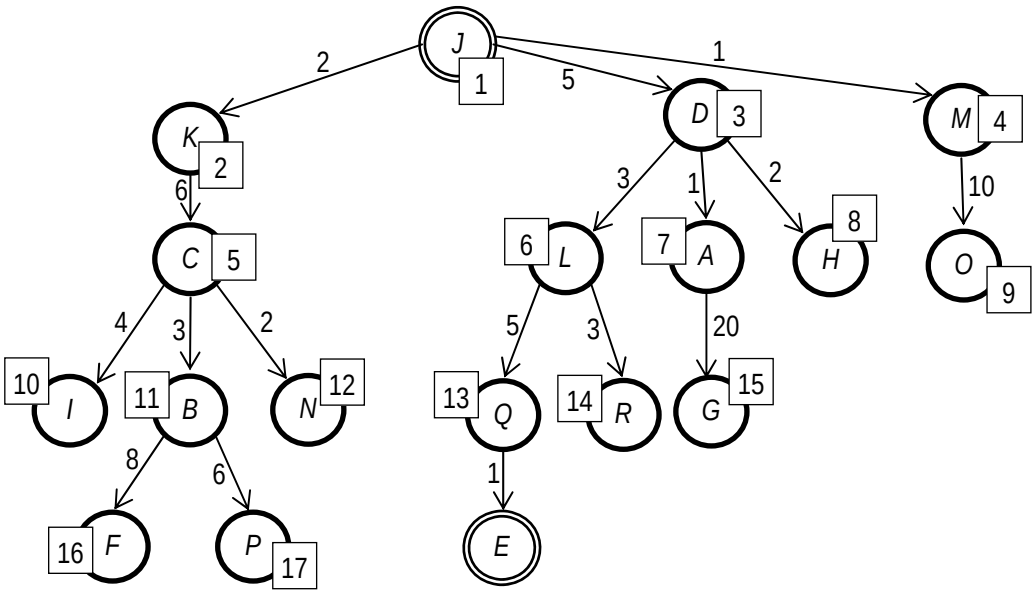


Figura 2.2: Orden de expansión de nodos para el árbol de la figura 2.1 según la búsqueda primero en anchura (de izquierda a derecha). Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto, el algoritmo termina en ese momento sin que se lleguen a generar sus hijos.

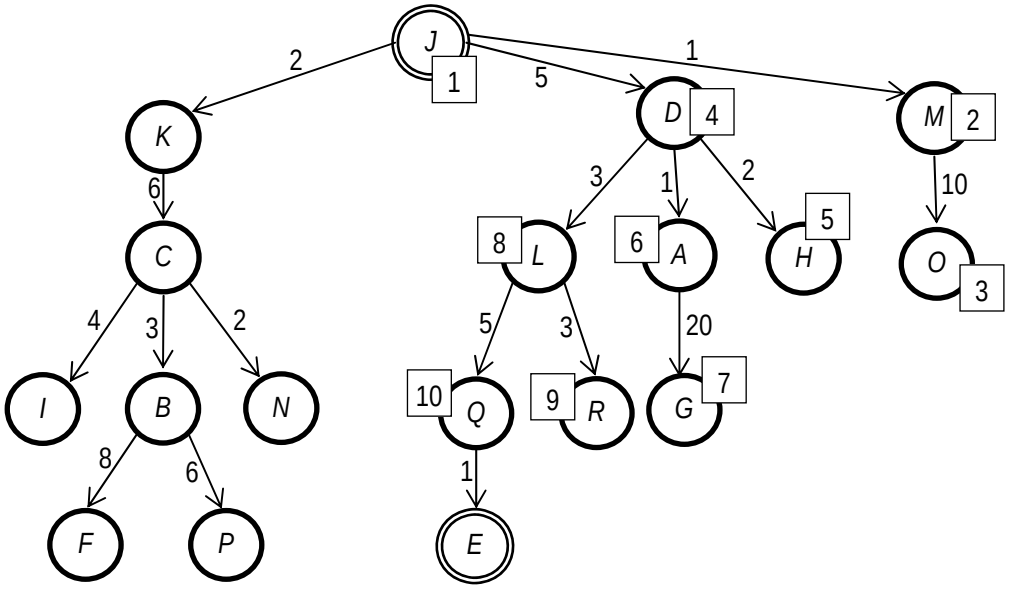


Figura 2.3: Orden de expansión de nodos para el árbol de la figura 2.1 según la búsqueda primero en profundidad (de derecha a izquierda). Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto, el algoritmo termina en ese momento sin que se lleguen a generar sus hijos.

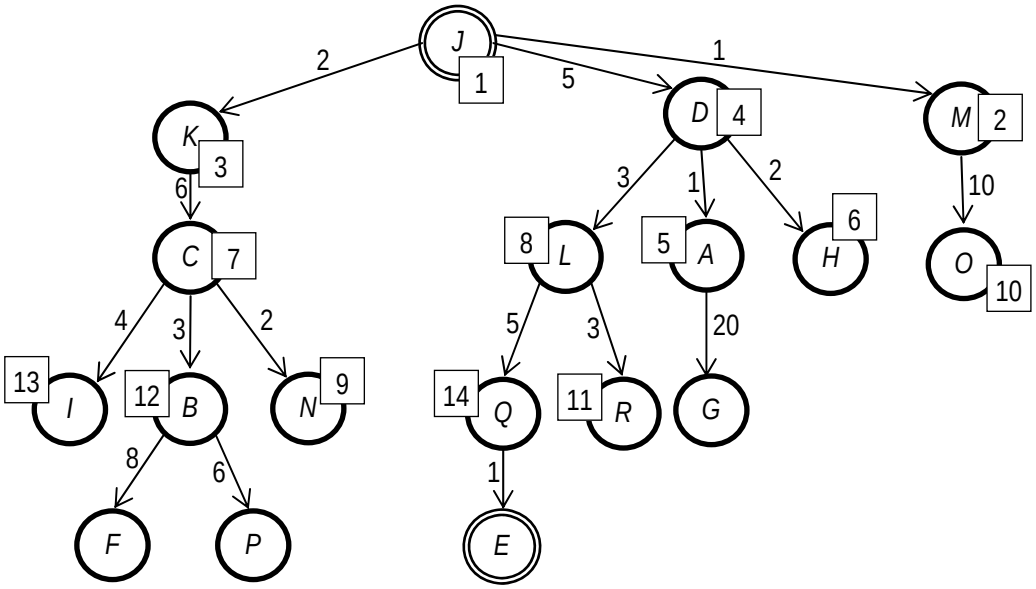


Figura 2.4: Orden de expansión de nodos para el árbol de la figura 2.1 según la búsqueda de coste uniforme. Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto, el algoritmo termina en ese momento sin que se lleguen a generar sus hijos.

4. Búsqueda en Anchura Iterativa (de derecha a izquierda)

Este algoritmo ejecuta iterativamente varias búsquedas primero en anchura, de manera que entre iteración e iteración se incrementa en una unidad el número máximo de hijos que se generan en cada expansión de un nodo padre. Al principio (en la primera iteración), únicamente un hijo es generado en cada expansión. En las figuras 2.5a, 2.5b y 2.5c se muestran los órdenes de expansión de nodos para cada una de las iteraciones de búsqueda primero en anchura que son necesarias para este ejemplo de búsqueda en anchura iterativa.

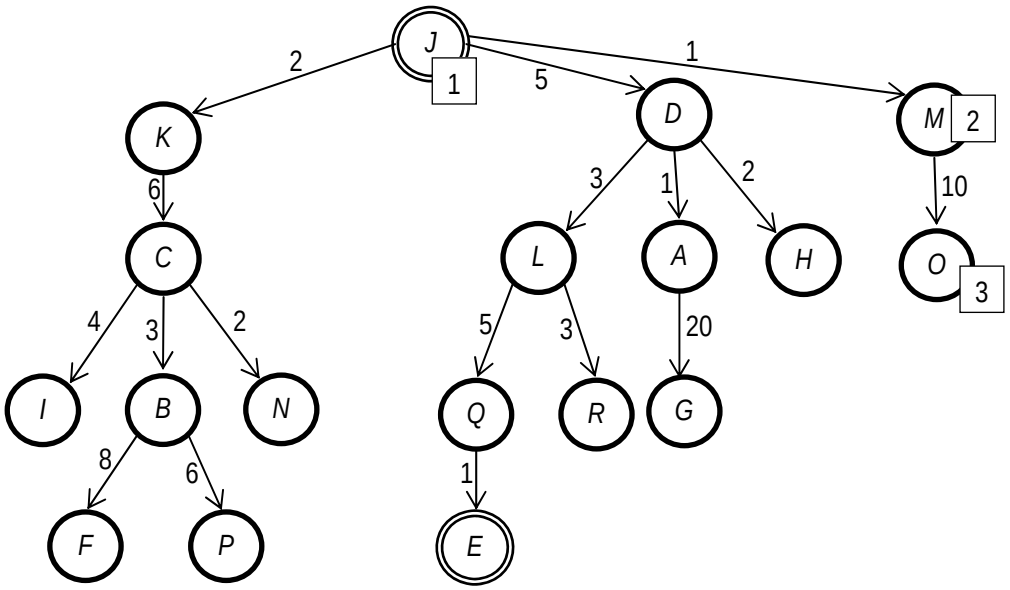


Figura 2.5a: Primera iteración de la búsqueda en anchura iterativa de derecha a izquierda, en la que como máximo un hijo es generado en cada expansión de un nodo padre.

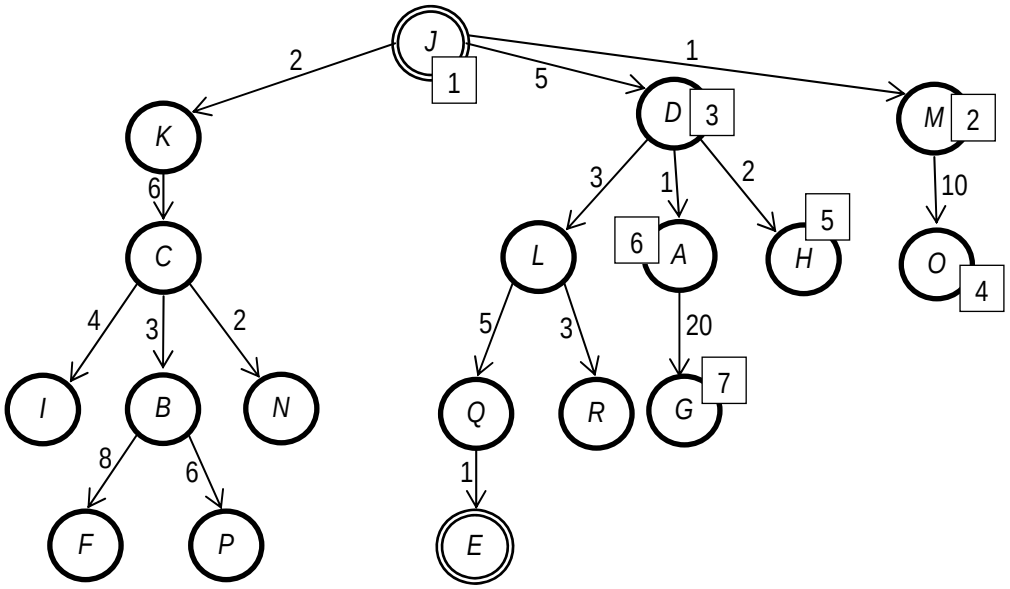


Figura 2.5b: Segunda iteración de la búsqueda en anchura iterativa de derecha a izquierda, en la que como máximo dos hijos son generados en cada expansión de un nodo padre.

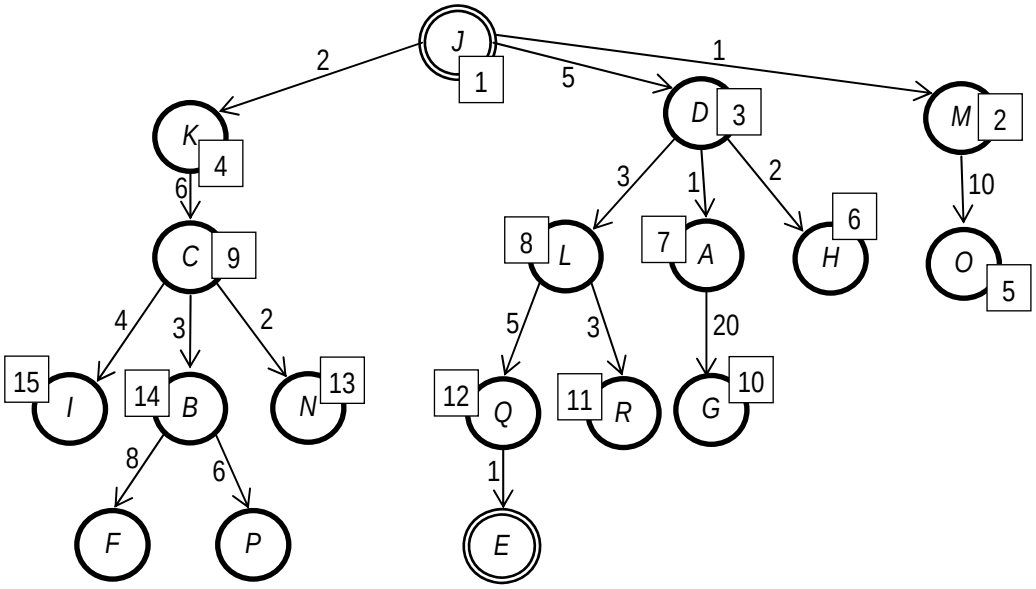


Figura 2.5c: Tercera y última iteración de la búsqueda en anchura iterativa de derecha a izquierda, en la que como máximo tres hijos son generados en cada expansión de un nodo padre.

Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto, el algoritmo termina en ese momento sin que se lleguen a generar sus hijos.

5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

Este algoritmo ejecuta iterativamente varias búsquedas primero en profundidad, de manera que entre iteración e iteración se incrementa en una unidad la profundidad límite. Al principio (en la primera iteración), la profundidad límite es igual a 1, así que sólo se podrá expandir el nodo inicial, cuya profundidad es igual a 0. En las figuras 2.6a, 2.6b, 2.6c y 2.6d se muestran los órdenes de expansión de nodos para cada una de las iteraciones de búsqueda primero en profundidad que son necesarias para este ejemplo de búsqueda en profundidad iterativa.

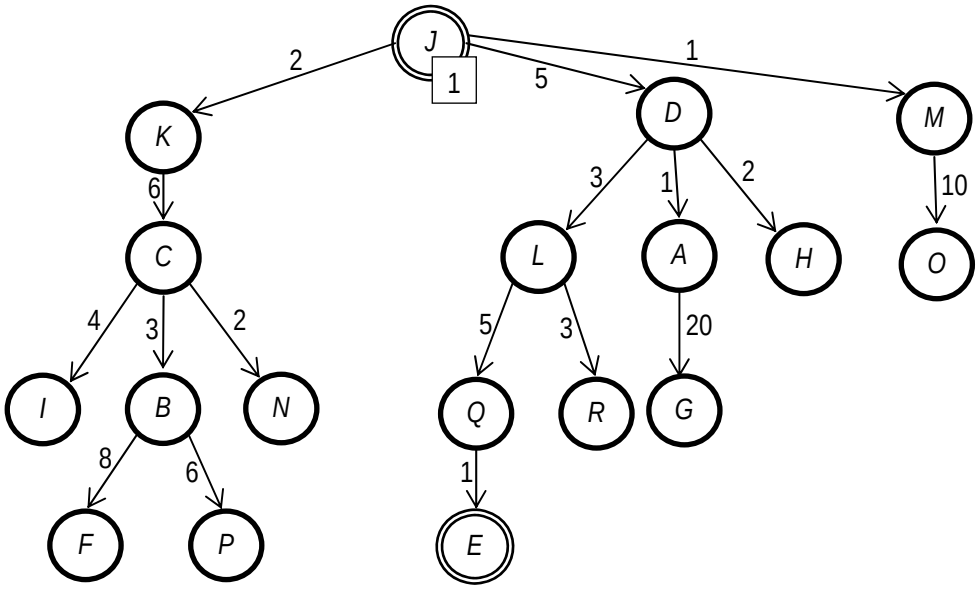


Figura 2.6a: Primera iteración de la búsqueda en profundidad iterativa de izquierda a derecha, en la que la profundidad límite es igual a 1 y únicamente son expandidos nodos con una profundidad máxima de 0.

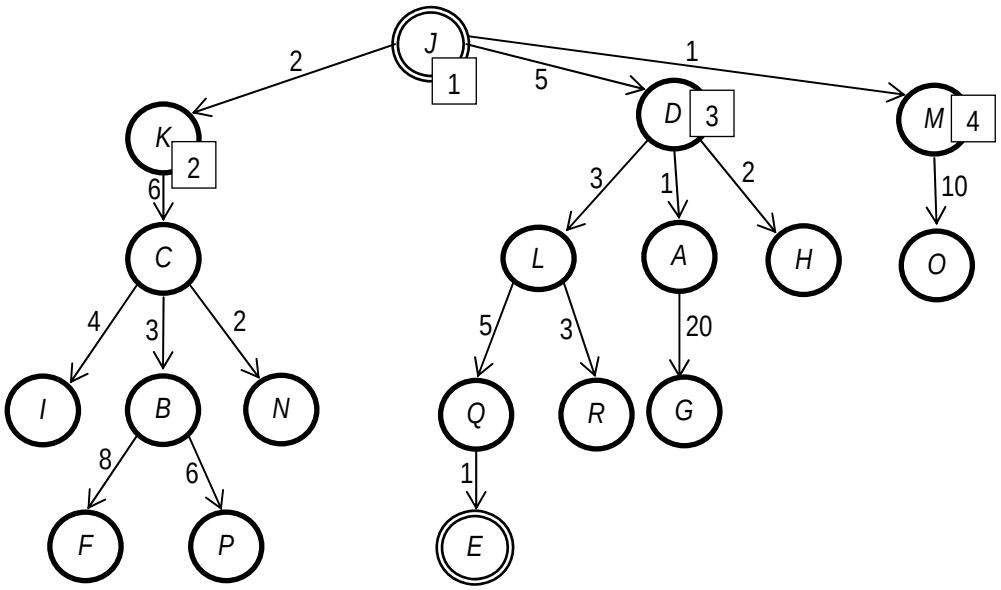


Figura 2.6b: Segunda iteración de la búsqueda en profundidad iterativa de izquierda a derecha, en la que la profundidad límite es igual a 2 y únicamente son expandidos nodos con una profundidad máxima de 1.

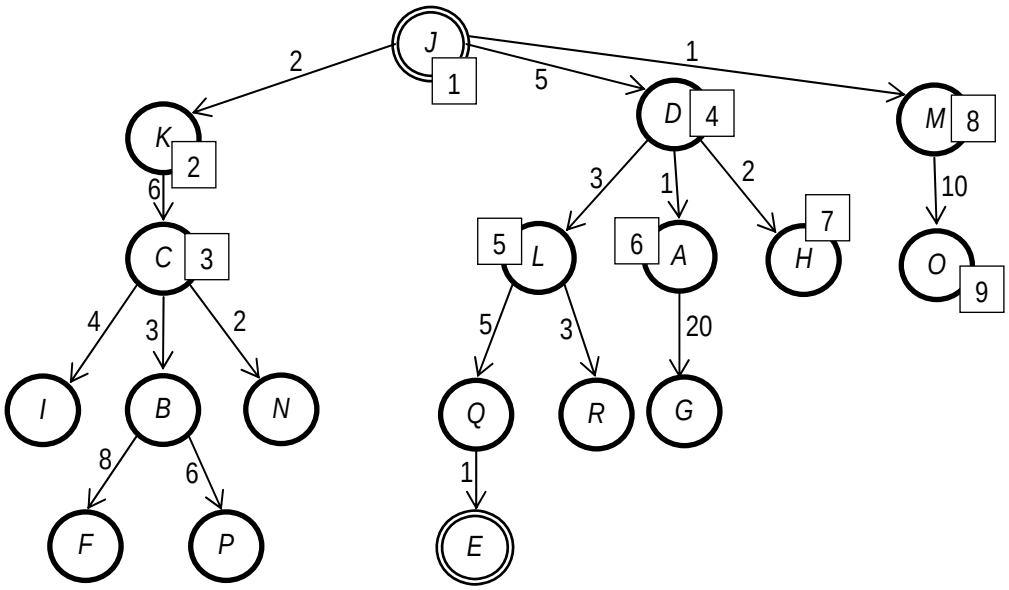


Figura 2.6c: Tercera iteración de la búsqueda en profundidad iterativa de izquierda a derecha, en la que la profundidad límite es igual a 3 y únicamente son expandidos nodos con una profundidad máxima de 2.

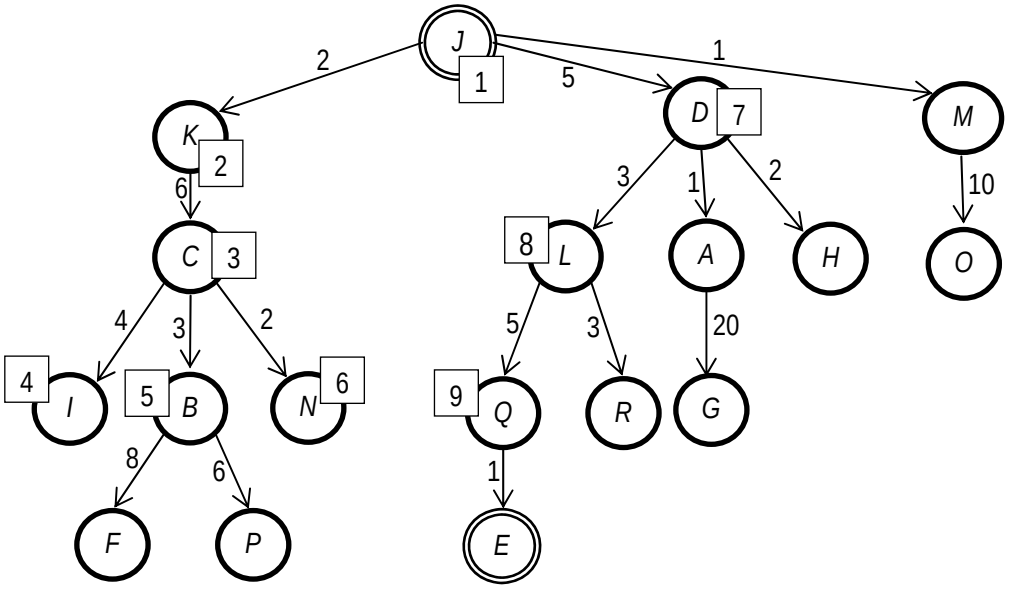


Figura 2.6d: Cuarta iteración de la búsqueda en profundidad iterativa de izquierda a derecha, en la que la profundidad límite es igual a 4 y únicamente son expandidos nodos con una profundidad máxima de 3.

Observe que realmente el nodo meta no es expandido en esta última iteración porque se encuentre a la profundidad límite. Esta condición no se llega a comprobar, ya que justo antes se averigua que el nodo es meta y, por tanto, el algoritmo termina.

EJERCICIO 3:

Considere el espacio de búsqueda de la figura 3.1, que tiene forma de árbol, donde el nodo raíz del árbol es el nodo inicial, existe un único nodo meta y cada operador tiene asociado un coste. Describa cuál es el contenido de ABIERTA, previamente a cada extracción de un nodo de la misma, a partir de cada uno de los métodos siguientes de búsqueda sin información del dominio:

1. Búsqueda Primero en Anchura (de izquierda a derecha)
2. Búsqueda Primero en Profundidad (de derecha a izquierda)
3. Búsqueda de Coste Uniforme
4. Búsqueda en Anchura Iterativa (de derecha a izquierda)
5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

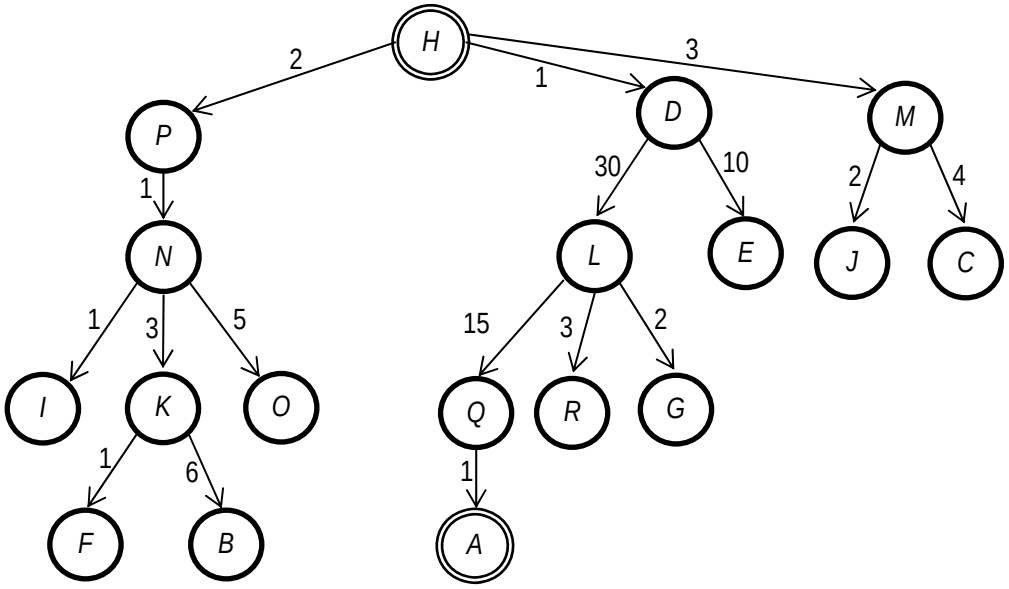


Figura 3.1: Árbol de búsqueda en el que el nodo inicial es *H*, el nodo meta es *A* y el coste de cada operador aparece al lado del arco que lo representa.

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 3:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- Cada uno de los cinco apartados, correspondiente a un método de búsqueda concreto, se puntúa sobre 2 puntos.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no gestiona ABIERTA en la forma debida: la puntuación del apartado bajaría 1.6 puntos si la respuesta dada varía significativamente de la correcta, mientras que si la respuesta dada varía de la correcta como consecuencia de un despiste no conceptual entonces la puntuación del apartado bajaría sólo 0.4 puntos en vez de los 1.6 puntos mencionados.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no finaliza cuando es debido, la puntuación del apartado bajaría 0.4 puntos. (Esto es independiente de la gestión de ABIERTA dada como respuesta, que ya ha sido valorada anteriormente con un máximo de 1.6 puntos.)

SOLUCIÓN DEL EJERCICIO 3 (por Severino Fernández Galán):

1. Búsqueda Primero en Anchura (de izquierda a derecha)

Este algoritmo usa ABIERTA como una cola, de manera que siempre se saca el primer nodo de la cola y se introducen sus hijos al final de la misma. El contenido de ABIERTA previamente a cada extracción de un nodo es el siguiente:

Antes de sacar *H*: {*H*}
Antes de sacar *P*: {*P*, *D*, *M*}
Antes de sacar *D*: {*D*, *M*, *N*}
Antes de sacar *M*: {*M*, *N*, *L*, *E*}
Antes de sacar *N*: {*N*, *L*, *E*, *J*, *C*}
Antes de sacar *L*: {*L*, *E*, *J*, *C*, *I*, *K*, *O*}
Antes de sacar *E*: {*E*, *J*, *C*, *I*, *K*, *O*, *Q*, *R*, *G*}
Antes de sacar *J*: {*J*, *C*, *I*, *K*, *O*, *Q*, *R*, *G*}
Antes de sacar *C*: {*C*, *I*, *K*, *O*, *Q*, *R*, *G*}
Antes de sacar *I*: {*I*, *K*, *O*, *Q*, *R*, *G*}
Antes de sacar *K*: {*K*, *O*, *Q*, *R*, *G*}
Antes de sacar *O*: {*O*, *Q*, *R*, *G*, *F*, *B*}
Antes de sacar *Q*: {*Q*, *R*, *G*, *F*, *B*}
Antes de sacar *R*: {*R*, *G*, *F*, *B*, *A*}
Antes de sacar *G*: {*G*, *F*, *B*, *A*}
Antes de sacar *F*: {*F*, *B*, *A*}
Antes de sacar *B*: {*B*, *A*}
Antes de sacar *A*: {*A*}
META ENCONTRADA

Observe que, tras cada expansión del nodo sacado de ABIERTA, sus nodos hijos más a la izquierda se introducen en ABIERTA antes que los situados más a la derecha.

2. Búsqueda Primero en Profundidad (de derecha a izquierda)

Este algoritmo usa ABIERTA como una pila, de manera que siempre se saca el primer nodo de la pila y se introducen sus hijos al principio de la misma. El contenido de ABIERTA previamente a cada extracción de un nodo es el siguiente:

Antes de sacar *H*: {*H*}
Antes de sacar *M*: {*M*, *D*, *P*}
Antes de sacar *C*: {*C*, *J*, *D*, *P*}
Antes de sacar *J*: {*J*, *D*, *P*}
Antes de sacar *D*: {*D*, *P*}
Antes de sacar *E*: {*E*, *L*, *P*}
Antes de sacar *L*: {*L*, *P*}
Antes de sacar *G*: {*G*, *R*, *Q*, *P*}
Antes de sacar *R*: {*R*, *Q*, *P*}
Antes de sacar *Q*: {*Q*, *P*}
Antes de sacar *A*: {*A*, *P*}
META ENCONTRADA

Observe que, tras cada expansión del nodo sacado de ABIERTA, sus nodos hijos más a la derecha se introducen en ABIERTA después que los situados más a la izquierda.

3. Búsqueda de Coste Uniforme

Este algoritmo mantiene ABIERTA ordenada según el coste del camino desde cada nodo al nodo inicial. En este ejercicio desharemos de forma arbitraria los posibles empates que surjan al determinar qué nodo

se debe sacar de ABIERTA. El contenido de ABIERTA previamente a cada extracción de un nodo es el siguiente:

Antes de sacar *H*: {*H*(0)}
Antes de sacar *D*: {*D*(1), *P*(2), *M*(3)}
Antes de sacar *P*: {*P*(2), *M*(3), *E*(11), *L*(31)}
Antes de sacar *M*: {*M*(3), *N*(3), *E*(11), *L*(31)}
Antes de sacar *N*: {*N*(3), *J*(5), *C*(7), *E*(11), *L*(31)}
Antes de sacar *I*: {*I*(4), *J*(5), *K*(6), *C*(7), *O*(8), *E*(11), *L*(31)}
Antes de sacar *J*: {*J*(5), *K*(6), *C*(7), *O*(8), *E*(11), *L*(31)}
Antes de sacar *K*: {*K*(6), *C*(7), *O*(8), *E*(11), *L*(31)}
Antes de sacar *C*: {*C*(7), *F*(7), *O*(8), *E*(11), *B*(12), *L*(31)}
Antes de sacar *F*: {*F*(7), *O*(8), *E*(11), *B*(12), *L*(31)}
Antes de sacar *O*: {*O*(8), *E*(11), *B*(12), *L*(31)}
Antes de sacar *E*: {*E*(11), *B*(12), *L*(31)}
Antes de sacar *B*: {*B*(12), *L*(31)}
Antes de sacar *L*: {*L*(31)}
Antes de sacar *G*: {*G*(33), *R*(34), *Q*(46)}
Antes de sacar *R*: {*R*(34), *Q*(46)}
Antes de sacar *Q*: {*Q*(46)}
Antes de sacar *A*: {*A*(47)}
META ENCONTRADA

Observe que, tras cada expansión de un nodo, sus nodos hijos son introducidos en ABIERTA, donde todos los nodos quedan siempre ordenados por coste creciente. Para facilitar el seguimiento del algoritmo, entre paréntesis y al lado de cada nodo de ABIERTA se especifica el coste del camino para ir desde dicho nodo hasta el nodo inicial.

4. Búsqueda en Anchura Iterativa (de derecha a izquierda)

Debido a que este algoritmo es una iteración de la búsqueda primero en anchura, los dos gestionan ABIERTA del mismo modo, es decir, como una cola. La diferencia reside en que en la búsqueda en anchura iterativa en cada iteración se fija un número máximo de hijos que pueden ser generados al expandir un nodo padre. Este número empieza valiendo 1 y es incrementado en una unidad al principio de cada nueva iteración.

Iteración 1 (cada expansión de un nodo padre genera como máximo un hijo):
Antes de sacar *H*: {*H*}
Antes de sacar *M*: {*M*}
Antes de sacar *C*: {*C*}

Iteración 2 (cada expansión de un nodo padre genera como máximo dos hijos):
Antes de sacar *H*: {*H*}
Antes de sacar *M*: {*M*, *D*}
Antes de sacar *D*: {*D*, *C*, *J*}
Antes de sacar *C*: {*C*, *J*, *E*, *L*}
Antes de sacar *J*: {*J*, *E*, *L*}
Antes de sacar *E*: {*E*, *L*}
Antes de sacar *L*: {*L*}
Antes de sacar *G*: {*G*, *R*}
Antes de sacar *R*: {*R*}

Iteración 3 (cada expansión de un nodo padre genera como máximo tres hijos):
Antes de sacar *H*: {*H*}
Antes de sacar *M*: {*M*, *D*, *P*}
Antes de sacar *D*: {*D*, *P*, *C*, *J*}
Antes de sacar *P*: {*P*, *C*, *J*, *E*, *L*}
Antes de sacar *C*: {*C*, *J*, *E*, *L*, *N*}

Antes de sacar *J*: {*J*, *E*, *L*, *N*}
Antes de sacar *E*: {*E*, *L*, *N*}
Antes de sacar *L*: {*L*, *N*}
Antes de sacar *N*: {*N*, *G*, *R*, *Q*}
Antes de sacar *G*: {*G*, *R*, *Q*, *O*, *K*, *I*}
Antes de sacar *R*: {*R*, *Q*, *O*, *K*, *I*}
Antes de sacar *Q*: {*Q*, *O*, *K*, *I*}
Antes de sacar *O*: {*O*, *K*, *I*, *A*}
Antes de sacar *K*: {*K*, *I*, *A*}
Antes de sacar *I*: {*I*, *A*, *B*, *F*}
Antes de sacar *A*: {*A*, *B*, *F*}
META ENCONTRADA

5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

Debido a que este algoritmo es una iteración de la búsqueda primero en profundidad, los dos gestionan ABIERTA del mismo modo, es decir, como una pila. La diferencia reside en que en la búsqueda en profundidad iterativa en cada iteración se fija una profundidad límite propia. Este número empieza valiendo 1, lo cual permite expandir únicamente el nodo inicial, y es incrementado en una unidad al principio de cada nueva iteración.

Iteración 1 (profundidad límite igual a 1):
Antes de sacar *H*: {*H*}
Antes de sacar *P*: {*P*, *D*, *M*}
Nótese que *P* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *D*: {*D*, *M*}
Nótese que *D* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *M*: {*M*}
Nótese que *M* no es expandido por coincidir su profundidad con la profundidad límite.

Iteración 2 (profundidad límite igual a 2):
Antes de sacar *H*: {*H*}
Antes de sacar *P*: {*P*, *D*, *M*}
Antes de sacar *N*: {*N*, *D*, *M*}
Nótese que *N* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *D*: {*D*, *M*}
Antes de sacar *L*: {*L*, *E*, *M*}
Nótese que *L* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *E*: {*E*, *M*}
Nótese que *E* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *M*: {*M*}
Antes de sacar *J*: {*J*, *C*}
Nótese que *J* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *C*: {*C*}
Nótese que *C* no es expandido por coincidir su profundidad con la profundidad límite.

Iteración 3 (profundidad límite igual a 3):
Antes de sacar *H*: {*H*}
Antes de sacar *P*: {*P*, *D*, *M*}
Antes de sacar *N*: {*N*, *D*, *M*}
Antes de sacar *I*: {*I*, *K*, *O*, *D*, *M*}
Nótese que *I* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *K*: {*K*, *O*, *D*, *M*}
Nótese que *K* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *O*: {*O*, *D*, *M*}

Nótese que *O* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *D*: {*D*, *M*}
Antes de sacar *L*: {*L*, *E*, *M*}
Antes de sacar *Q*: {*Q*, *R*, *G*, *E*, *M*}
Nótese que *Q* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *R*: {*R*, *G*, *E*, *M*}
Nótese que *R* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *G*: {*G*, *E*, *M*}
Nótese que *G* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *E*: {*E*, *M*}
Antes de sacar *M*: {*M*}
Antes de sacar *J*: {*J*, *C*}
Antes de sacar *C*: {*C*}

Iteración 4 (profundidad límite igual a 4):
Antes de sacar *H*: {*H*}
Antes de sacar *P*: {*P*, *D*, *M*}
Antes de sacar *N*: {*N*, *D*, *M*}
Antes de sacar *I*: {*I*, *K*, *O*, *D*, *M*}
Antes de sacar *K*: {*K*, *O*, *D*, *M*}
Antes de sacar *F*: {*F*, *B*, *O*, *D*, *M*}
Nótese que *F* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *B*: {*B*, *O*, *D*, *M*}
Nótese que *B* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *O*: {*O*, *D*, *M*}
Antes de sacar *D*: {*D*, *M*}
Antes de sacar *L*: {*L*, *E*, *M*}
Antes de sacar *Q*: {*Q*, *R*, *G*, *E*, *M*}
Antes de sacar *A*: {*A*, *R*, *G*, *E*, *M*}
META ENCONTRADA

EJERCICIO 4:

Considere el espacio de búsqueda de la figura 4.1, que tiene forma de árbol, donde el nodo raíz del árbol es el nodo inicial, existe un único nodo meta y cada operador tiene asociado un coste. Describa cuál es el contenido de TABLA_A, posteriormente a cada expansión de un nodo, a partir de cada uno de los métodos siguientes de búsqueda sin información del dominio:

1. Búsqueda Primero en Anchura (de izquierda a derecha)
2. Búsqueda Primero en Profundidad (de derecha a izquierda)
3. Búsqueda de Coste Uniforme
4. Búsqueda en Anchura Iterativa (de derecha a izquierda)
5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

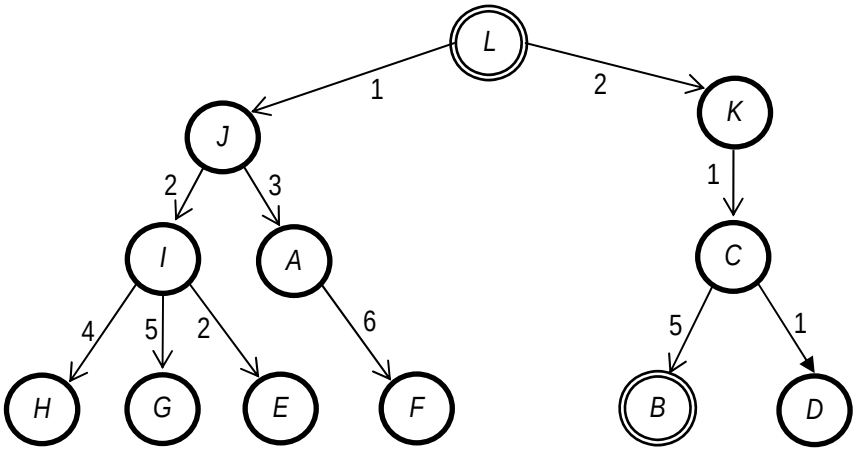


Figura 4.1: Árbol de búsqueda en el que el nodo inicial es *L*, el nodo meta es *B* y el coste de cada operador aparece al lado del arco que lo representa.

Para cada nodo de TABLA_A incluya la siguiente información: su nodo padre y el coste al nodo inicial. (En este ejercicio no es necesario incluir en TABLA_A información sobre los hijos de cada nodo expandido, ya que sólo existe un camino desde cada nodo al nodo inicial y, por tanto, el mejor camino desde cada nodo al nodo inicial no cambia a lo largo del proceso de búsqueda.)

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 4:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

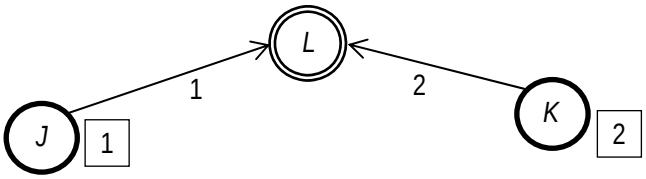
- Cada uno de los cinco apartados, correspondiente a un método de búsqueda concreto, se puntúa sobre 2 puntos.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no gestiona TABLA_A en la forma debida: la puntuación del apartado bajaría 1.6 puntos si la respuesta dada varía significativamente de la correcta, mientras que si la respuesta dada varía de la correcta como consecuencia de un despiste no conceptual entonces la puntuación del apartado bajaría sólo 0.4 puntos en vez de los 1.6 puntos mencionados.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no finaliza cuando es debido, la puntuación del apartado bajaría 0.4 puntos. (Esto es independiente de la gestión de TABLA_A dada como respuesta, que ya ha sido valorada anteriormente con un máximo de 1.6 puntos.)

SOLUCIÓN DEL EJERCICIO 4 (por Severino Fernández Galán):

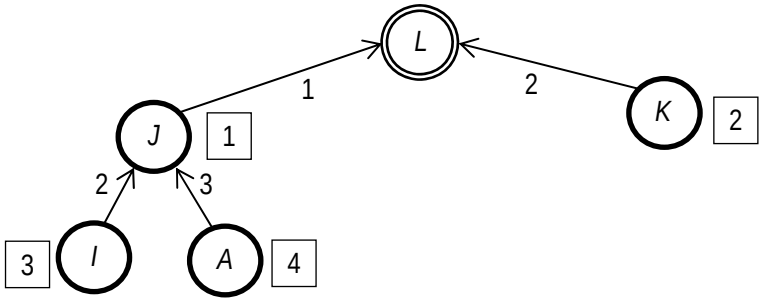
De cara a ilustrar la respuesta convenientemente, incluimos la información de TABLA_A gráficamente: por un lado, para cada nodo generado en una expansión trazamos un nodo ascendente a su padre y, por otro lado, indicamos el coste al nodo inicial al lado de cada nodo generado.

1. Búsqueda Primero en Anchura (de izquierda a derecha)

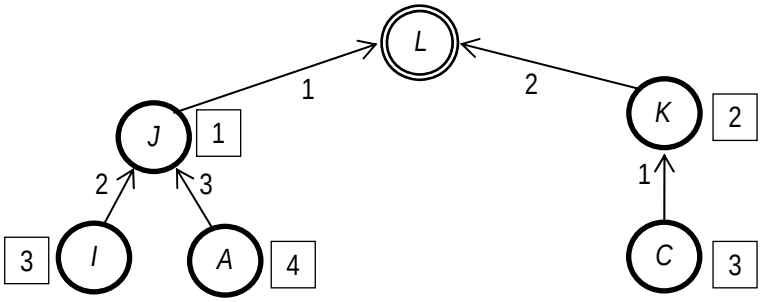
- En primer lugar expandimos el nodo inicial, generando sus nodos hijos. Desde cada nodo hijo trazamos un nodo ascendente a su nodo padre. Al lado de cada nodo hijo indicamos el coste desde dicho nodo al nodo inicial.



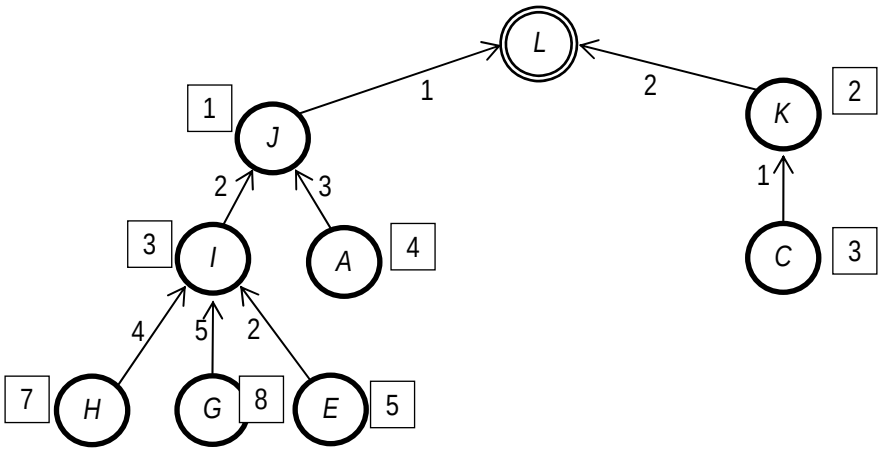
- Expandimos *J*:



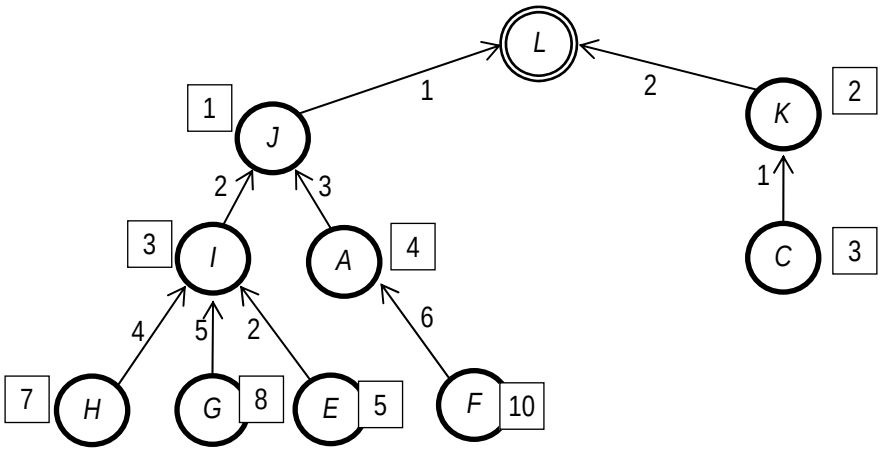
- Expandimos *K*:



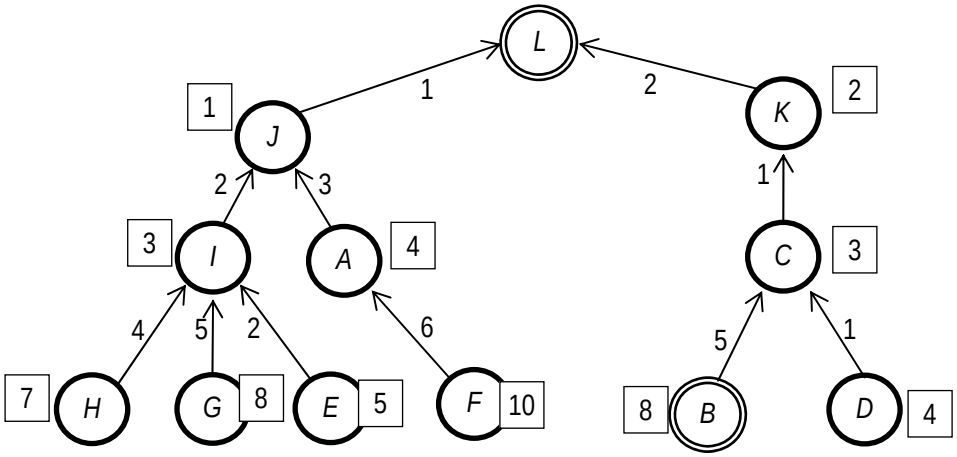
- Expandimos *I*:



- Expandimos *A*:



- Expandimos *C*:

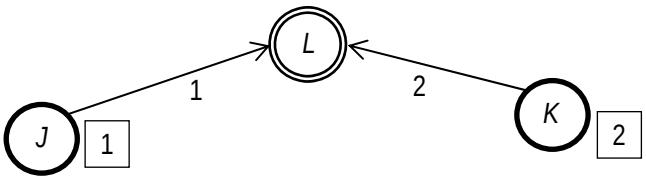


- TABLA_A no cambiaría tras las expansiones sucesivas de *H*, *G*, *E*, *F* y *B*. Como *B* es meta, el algoritmo terminaría al intentar expandir *B*.

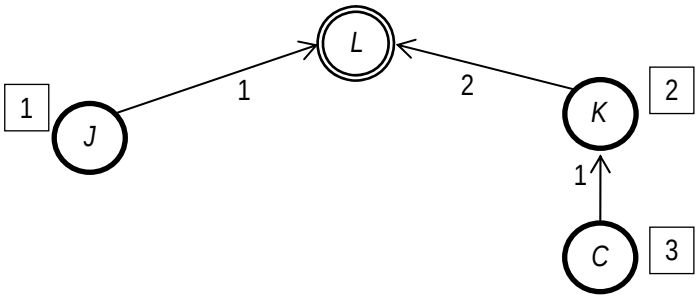
2. Búsqueda Primero en Profundidad (de derecha a izquierda)

Cuando sea necesario, en este algoritmo aplicaremos la función LimpiarTABLA_A (véase texto base, página 318). Tras la extracción de un nodo de ABIERTA y su posible expansión, dicha función elimina aquellos nodos que ya no son necesarios en el proceso de búsqueda de la meta. Esto permite preservar la linealidad de la complejidad espacial de este algoritmo con respecto a la profundidad de la solución.

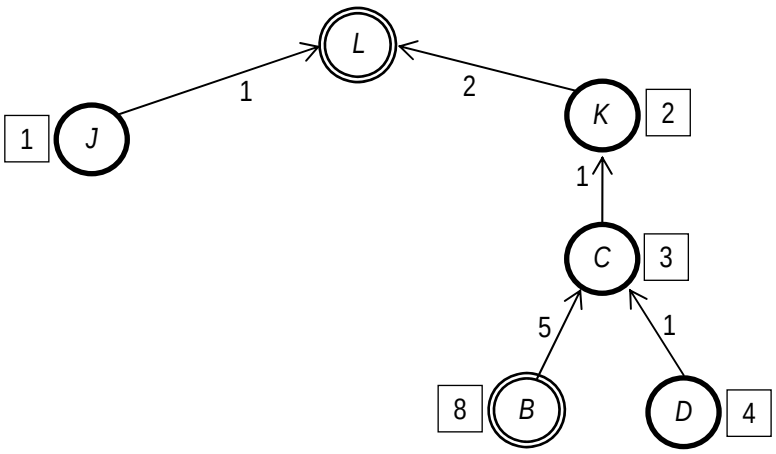
- En primer lugar expandimos el nodo inicial.



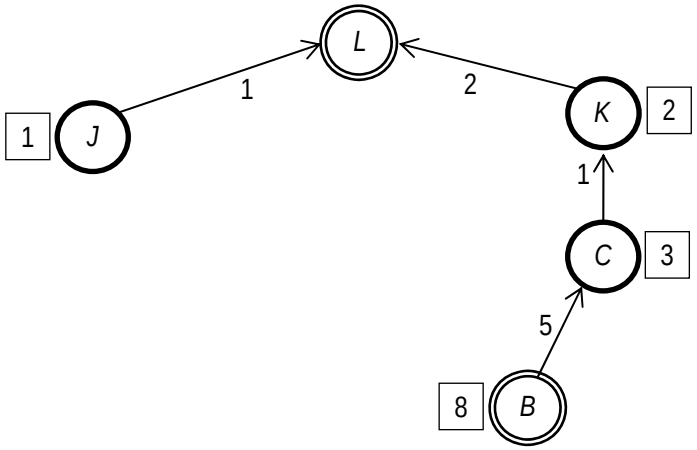
- Expandimos *K*:



- Expandimos *C*:



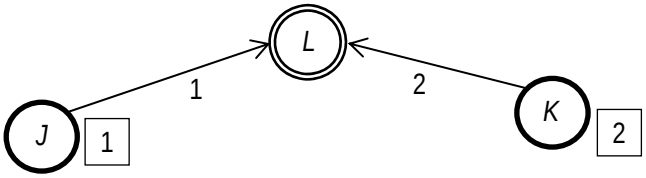
- Expandimos *D*. Seguidamente aplicamos LimpiarTABLA_A a *D*, es decir, sacamos *D* de TABLA_A por no tener hijos en ABIERTA:



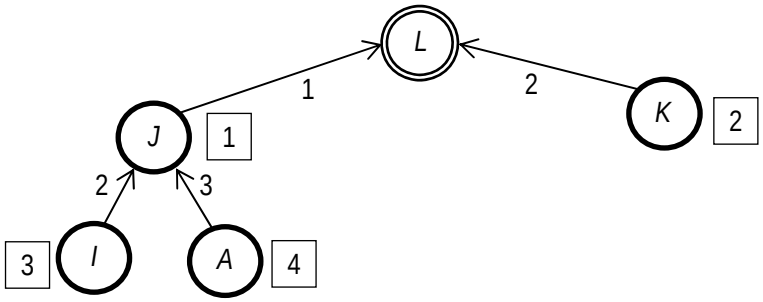
- A continuación llegaríamos a la meta al elegir *B* para su expansión.

3. Búsqueda de Coste Uniforme

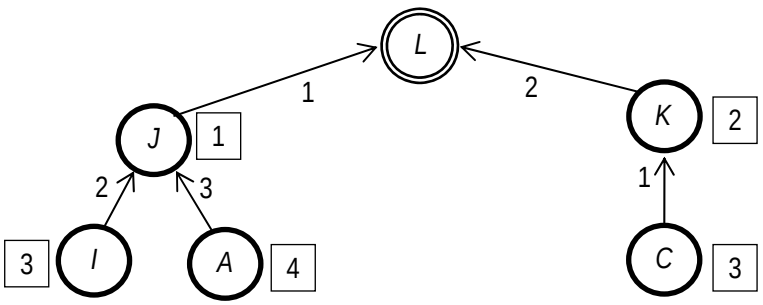
- En primer lugar expandimos el nodo inicial.



- Expandimos *J*:

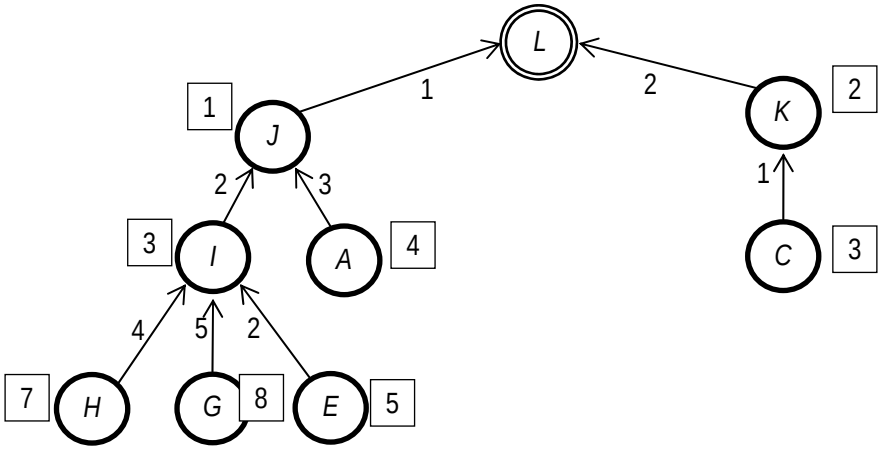


- Expandimos *K*:

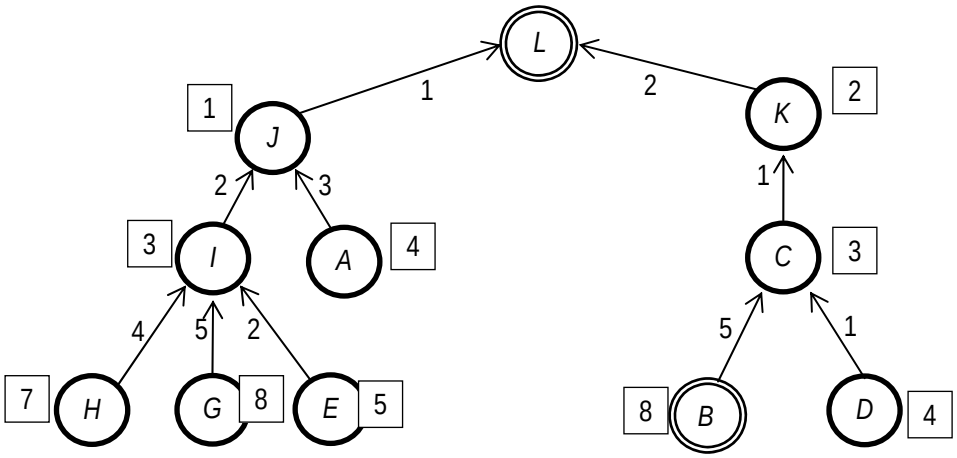


Seguidamente podríamos expandir *I* o *C*, ya que el coste desde ambos nodos al inicial es igual a tres. Elegimos *I* de forma arbitraria.

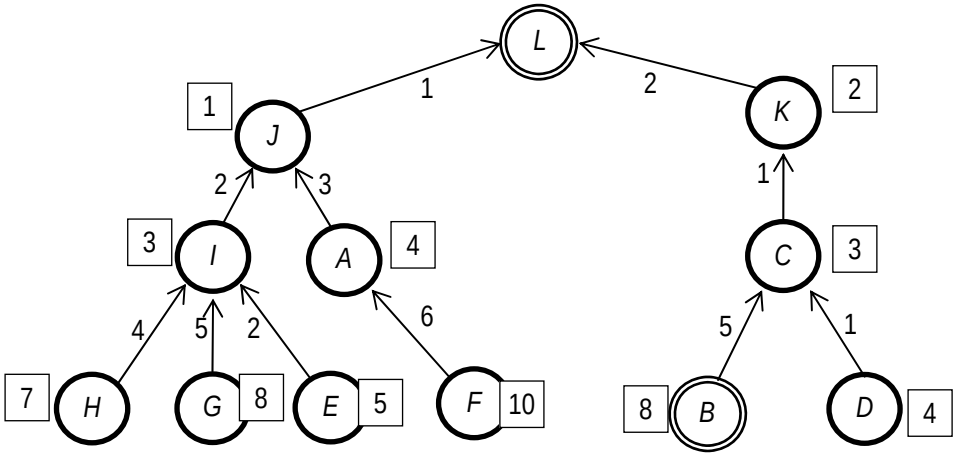
- Expandimos *I*:



- Expandimos C:



- Tras expandir *D* arbitrariamente (frente a *A*), TABLA_A no cambia. Nótese que *D* no podría volver a ser expandido, ya que ha sido extraído de ABIERTA. Seguidamente expandimos *A*:

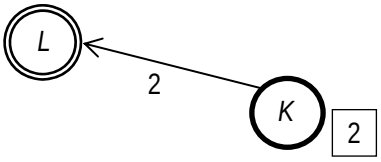


- Tras expandir *E* y *H*, TABLA_A no cambia. A continuación elegimos expandir *B* de forma arbitraria (frente a *G*) y alcanzamos la meta.

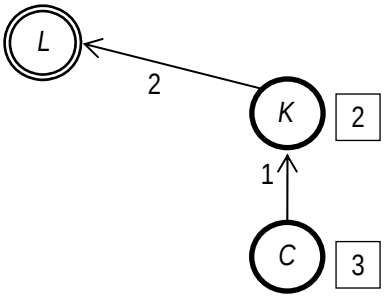
4. Búsqueda en Anchura Iterativa (de derecha a izquierda)

Iteración 1 (se genera como máximo 1 nodo hijo en cada expansión de un nodo padre):

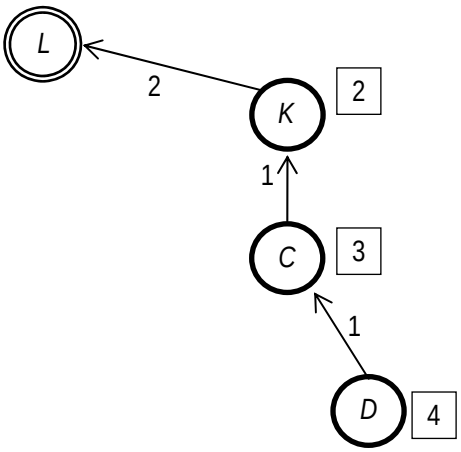
- Expandimos el nodo inicial:



- Expandimos *K*:



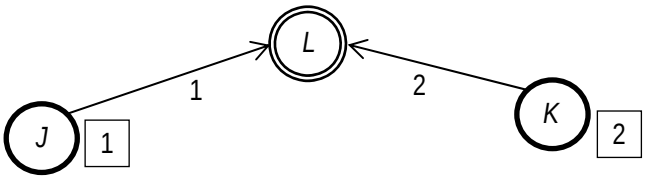
- Expandimos *C*:



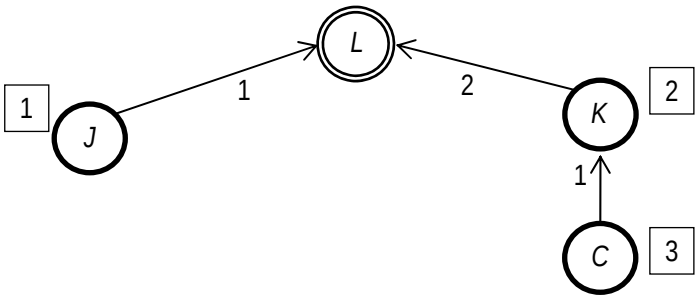
- Al expandir *D* finaliza la iteración actual.

Iteración 2 (se generan como máximo 2 nodos hijos en cada expansión de un nodo padre):

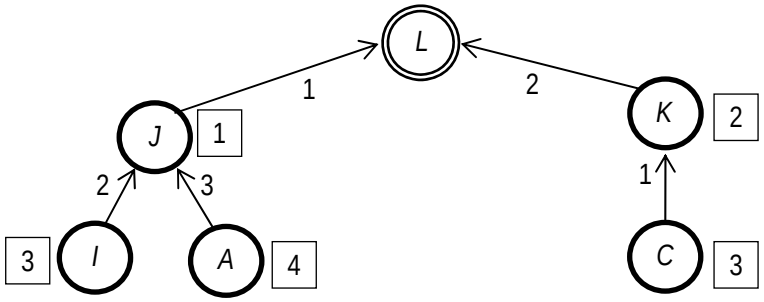
- Expandimos el nodo inicial:



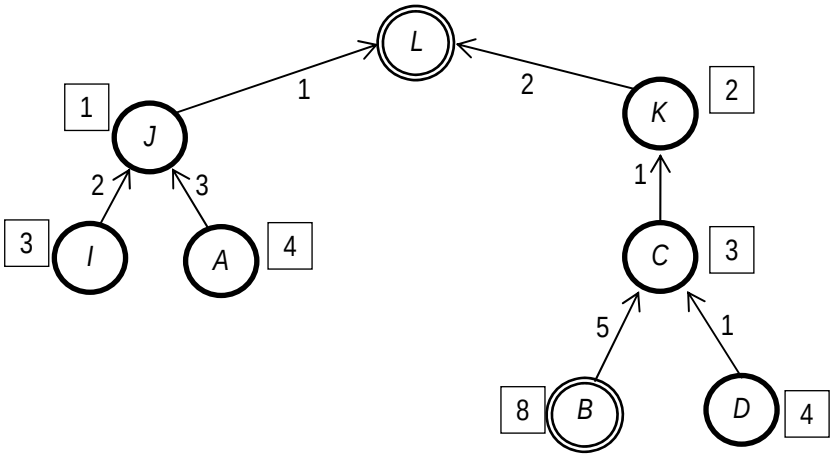
- Expandimos K:



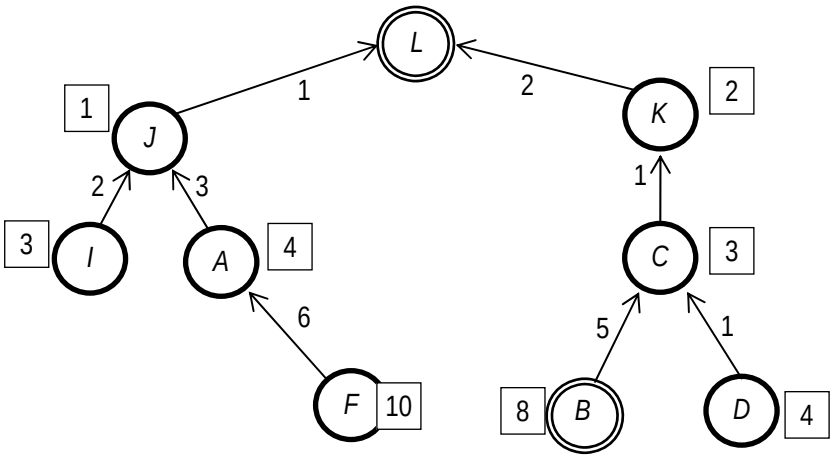
- Expandimos *J*:



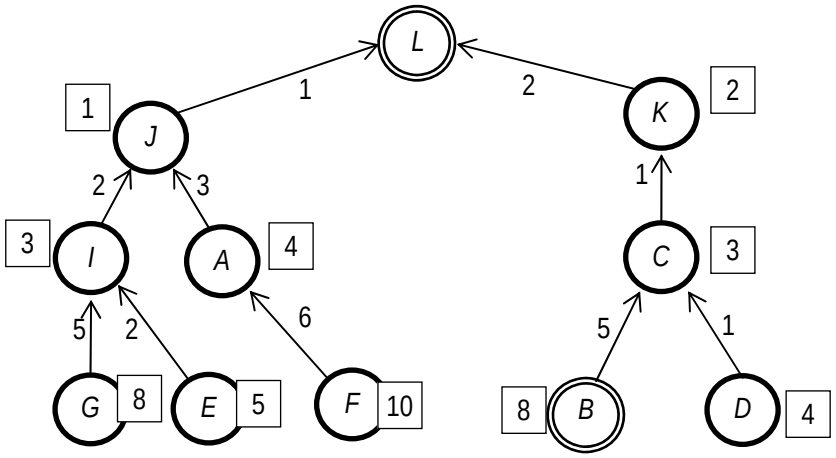
- Expandimos *C*:



- Expandimos *A*:



- Expandimos *I*:

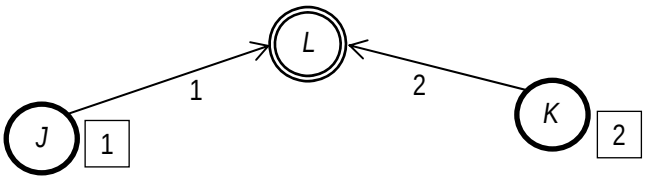


- Tras expandir *D*, TABLA_A no cambia. Finalmente, intentamos expandir *B* y alcanzamos la meta.

5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

<u>Iteración 1 (profundidad límite igual a 1)</u>:

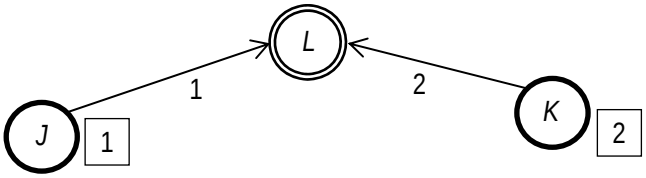
- Expandimos el nodo inicial:



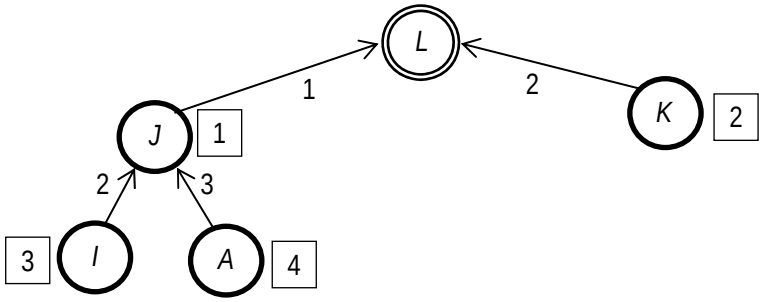
Tras comprobar que *J* no es nodo meta, se le aplica la función LimpiarTABLA_A por estar en la profundidad límite y es sacado de TABLA_A. De igual manera, tras comprobar que *K* no es nodo meta, se le aplica la función LimpiarTABLA_A por estar en la profundidad límite y es sacado de TABLA_A. Seguidamente, tras aplicarle la función LimpiarTABLA_A, *L* es sacado de TABLA_A por no tener hijos en ABIERTA. A continuación pasamos a la siguiente iteración.

Iteración 2 (profundidad límite igual a 2):

- Expandimos el nodo inicial:

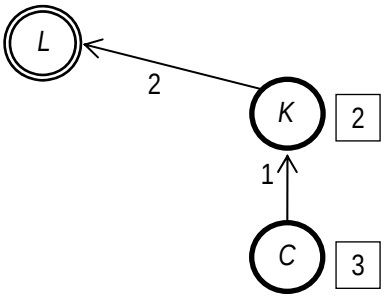


- Expandimos *J*:



Tras comprobar que *I* no es nodo meta, se le aplica la función LimpiarTABLA_A por encontrarse en la profundidad límite y es sacado de TABLA_A. Del mismo modo, tras comprobar que *A* no es nodo meta, se le aplica la función LimpiarTABLA_A por estar en la profundidad límite y es sacado de TABLA_A. A continuación, tras aplicarle la función LimpiarTABLA_A, *J* es sacado de TABLA_A por no tener hijos en ABIERTA.

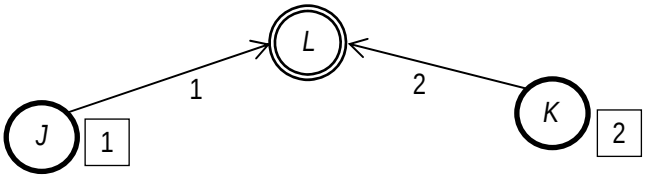
- Expandimos *K*:



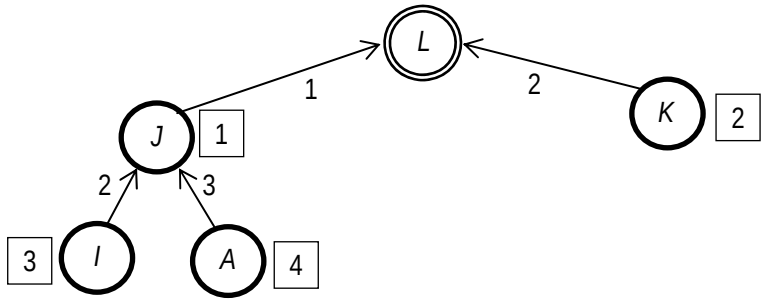
Tras comprobar que *C* no es nodo meta, se le aplica la función LimpiarTABLA_A por estar en la profundidad límite y es sacado de TABLA_A. Tras aplicarle la función LimpiarTABLA_A, *K* es sacado de TABLA_A por no tener hijos en ABIERTA. Lo mismo ocurre posteriormente con *L*, por lo que pasamos a la siguiente iteración.

Iteración 3 (profundidad límite igual a 3):

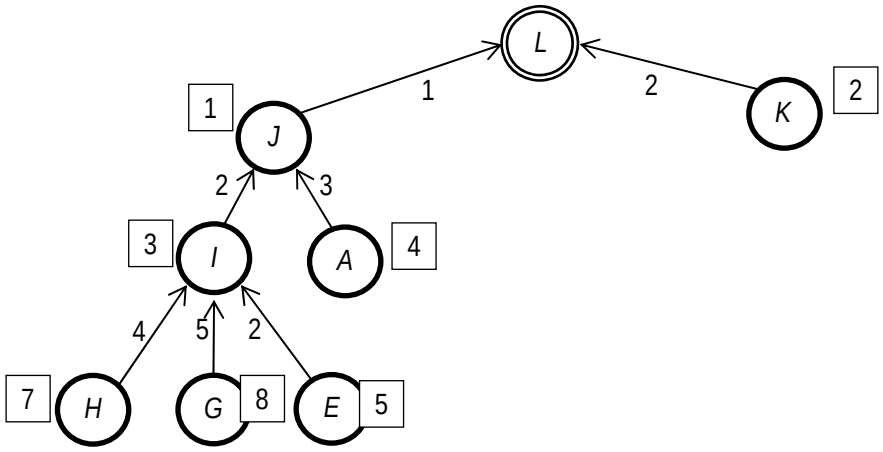
- Expandimos el nodo inicial:



- Expandimos *J*:

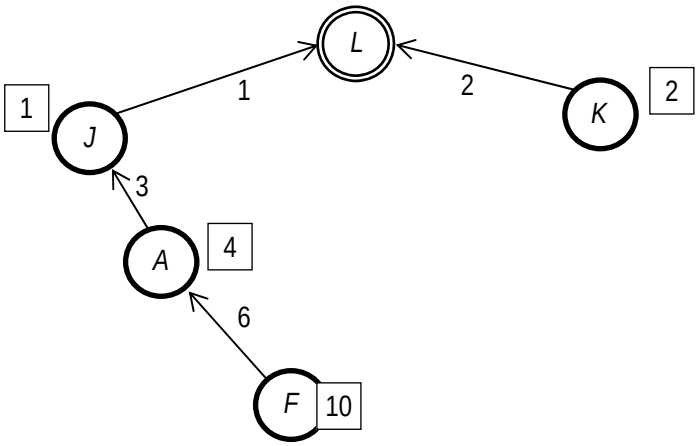


- Expandimos *I*:



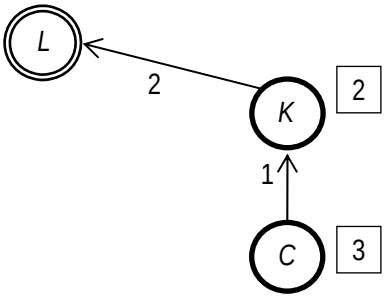
Tras comprobar que *H* no es nodo meta, se le aplica la función LimpiarTABLA_A por encontrarse en la profundidad límite y es sacado de TABLA_A. Del mismo modo, tras comprobar que *G* no es nodo meta, se le aplica la función LimpiarTABLA_A por encontrarse en la profundidad límite y es sacado de TABLA_A. De nuevo, tras comprobar que *E* no es nodo meta, se le aplica la función TABLA_A por encontrarse en la profundidad límite y es sacado de TABLA_A. Seguidamente, tras aplicarle la función LimpiarTABLA_A, *I* es sacado de TABLA_A por no tener más hijos en ABIERTA.

- Expandimos *A*:

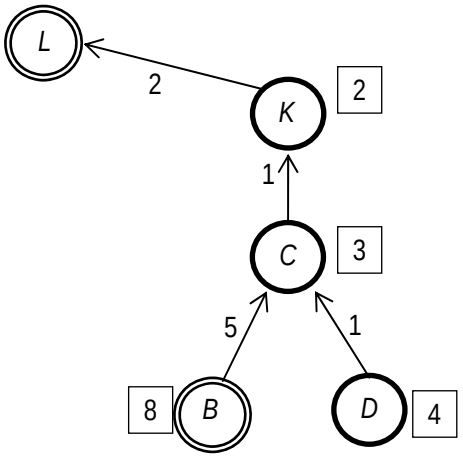


Tras comprobar que *F* no es nodo meta, se le aplica la función LimpiarTABLA_A por encontrarse en la profundidad límite y es sacado de TABLA_A. Seguidamente, tras aplicarle la función LimpiarTABLA_A, *A* es sacado de TABLA_A por no tener hijos en ABIERTA. Lo mismo ocurre posteriormente con *J*.

- Expandimos *K*:



- Expandimos *C*:



- Al elegir *B* para su expansión, habríamos llegado a la meta.

EJERCICIO 5:

Considere el grafo de la figura 5.1, donde el nodo inicial es *B* y donde los nodos meta son *H* e *I*. Cada arco u operador lleva asociado su coste y en cada nodo aparece la estimación de la menor distancia desde ese nodo a una meta. Aplique paso a paso el algoritmo A* al grafo dado, indicando de forma razonada la siguiente información en cada paso del algoritmo:

1. Qué nodo es expandido.
2. Cuál es el contenido de ABIERTA tras la expansión del nodo, indicando el valor de la función de evaluación heurística para cada nodo de ABIERTA.
3. Cuál es el contenido de TABLA_A tras la expansión del nodo. Para cada nodo de TABLA_A incluya la siguiente información:
    a) Su nodo padre que indique el camino de menor coste hasta el nodo inicial encontrado hasta el momento
    b) El coste del camino de menor coste hasta el nodo inicial encontrado hasta el momento
    c) Sus nodos hijos (si el nodo de TABLA_A actual ya ha sido expandido)

Por último, ¿cuál es el camino solución hallado y su coste?

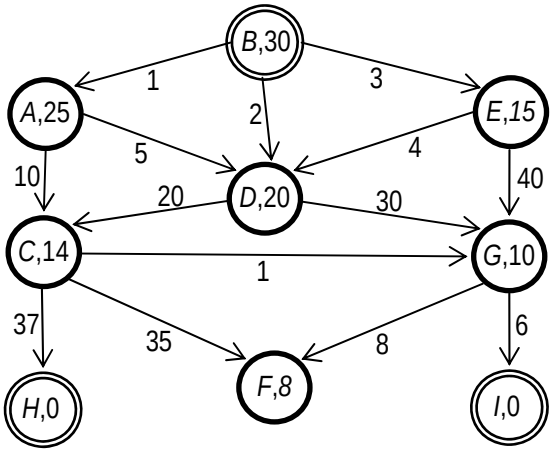


Figura 5.1: Grafo de búsqueda en el que el nodo inicial es *B*, los nodos meta son *H* e *I,* el coste de cada operador aparece al lado del arco que lo representa y al lado de cada nodo aparece la estimación de la menor distancia desde ese nodo a una meta.

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 5:

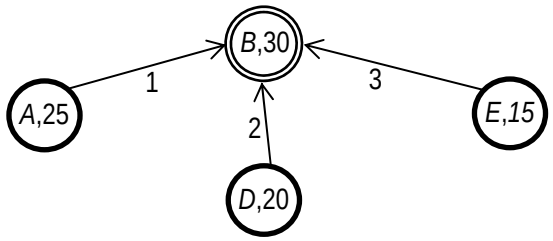
La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- El orden seguido en la expansión de los nodos se puntúa sobre 1.5 puntos.
- La forma en que se gestiona ABIERTA se puntúa sobre 3.75 puntos. Se hará especial énfasis en comprobar qué nodos hay en ABIERTA en cada paso del algoritmo y qué valores de la función de evaluación heurística se les asignan.
- La forma en que se gestiona TABLA_A se puntúa sobre 3.75 puntos. Se hará especial énfasis en comprobar qué nodos hay en TABLA_A en cada paso del algoritmo y qué padre mejor se les asigna (teniendo en cuenta las posibles reorientaciones o rectificaciones de enlaces)
- La correcta terminación del algoritmo se puntúa sobre 1 punto. Se hará especial énfasis en comprobar cuándo termina el algoritmo y qué camino solución devuelve.

SOLUCIÓN DEL EJERCICIO 5 (por Severino Fernández Galán):

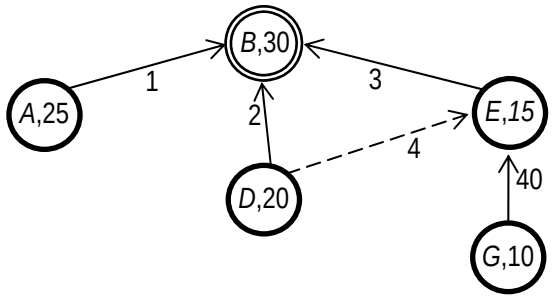
De cara a ilustrar la respuesta convenientemente, incluimos la información pedida sobre TABLA_A gráficamente. Para ello es necesario trazar, para cada nodo generado en una expansión, un nodo ascendente a su padre expandido; además, hay que anotar para cada nodo su mejor padre encontrado hasta el momento (punto 3a del enunciado). De esta manera, siguiendo los enlaces al mejor padre, se puede saber cuál es el mejor camino encontrado hasta el momento desde cada nodo al nodo inicial (punto 3b del enunciado). Además, los arcos ascendentes que llegan a un nodo ya expandido lo enlazan a sus nodos hijos (punto 3c del enunciado).

- PASO 1. El nodo inicial *B* es introducido en ABIERTA y posteriormente expandido. Tras la expansión, la situación es la siguiente:



ABIERTA = {*E*(3+15), *D*(2+20), *A*(1+25)}

- PASO 2. Expandimos *E* por ser el nodo de ABIERTA con menor valor de la función de evaluación heurística, *f*=*g*+*h* (al ser *g*=3 y *h*=15).

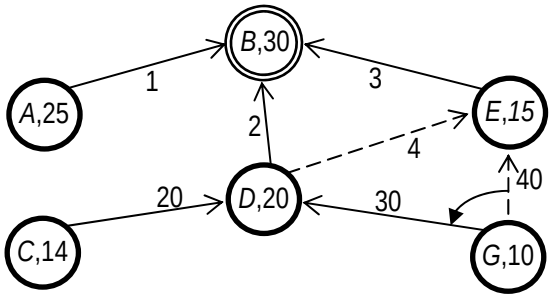


ABIERTA = {*D*(2+20), *A*(1+25), *G*(43+10)}

Observe que el mejor camino desde *D* al nodo inicial lo marca su padre *B* (coste 2) y no su padre *E* (coste 4+3=7). Por ello, el arco ascendente de *D* a *B* se marca con trazo continuo y el arco ascendente de *D* a *E* se marca con trazo discontinuo.

Es importante darse cuenta que el conjunto de arcos con trazo continuo formará siempre un árbol en el grafo parcial de búsqueda desarrollado hasta el momento.

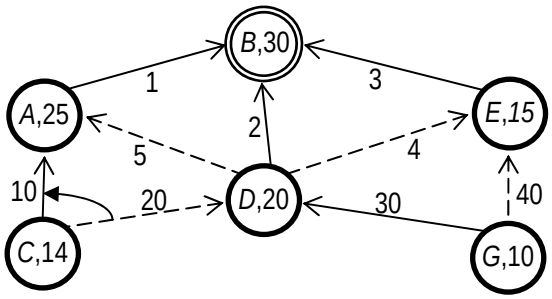
- PASO 3. Expandimos *D*.



ABIERTA = {*A*(1+25), *C*(22+14), *G*(32+10)}

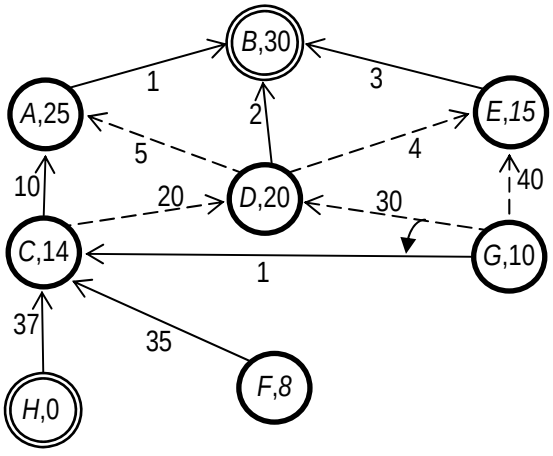
Observe que ha habido una reorientación del mejor padre de *G*, que antes era *E* y ahora pasa a ser *D*. El nuevo coste de *G* al nodo inicial es 30+2=32, que se puede calcular a partir de los costes de los mejores arcos ascendentes hallados hasta el momento: $G \rightarrow D$ y $D \rightarrow B$.

- PASO 4. Expandimos *A*.



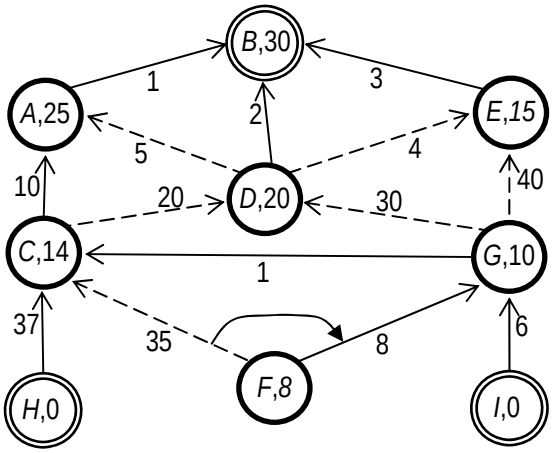
ABIERTA = {*C*(11+14), *G*(32+10)}

- PASO 5. Expandimos *C*.



ABIERTA = {*G*(12+10), *H*(48+0), *F*(46+8)}

- PASO 6. Expandimos *G*.



ABIERTA = {*I*(18+0), *F*(20+8) , *H*(48+0)}

- PASO 7. Intentamos expandir *I* y alcanzamos una meta, con lo que el algoritmo termina. El camino solución es: $B \rightarrow A \rightarrow C \rightarrow G \rightarrow I$, cuyo coste es 18.

EJERCICIO 6:

Considere el grafo de la figura 6.1, donde el nodo inicial es *D* y donde los nodos meta son desconocidos. Cada arco u operador lleva asociado su coste y en cada nodo aparece su valor de la función de evaluación heurística (que hay que minimizar). Aplique paso a paso el algoritmo de escalada o máximo gradiente al grafo dado. Para ello indique de forma razonada qué nodo se expande en cada paso y cuál es el nodo final devuelto por el algoritmo. Utilice como criterio de selección el de mejor vecino. Utilice como criterio de terminación el que no se hayan producido mejoras durante los cinco últimos pasos del algoritmo.

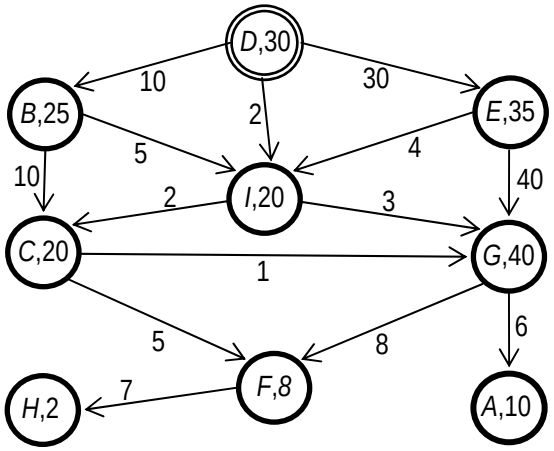


Figura 6.1: Grafo de búsqueda en el que el nodo inicial es *D*, los nodos meta son desconocidos, el coste de cada operador aparece al lado del arco que lo representa y al lado de cada nodo aparece el valor de su función de evaluación heurística (que hay que minimizar).

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 6:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- La correcta aplicación en cada paso del algoritmo del *criterio de selección* del vecino o hijo del nodo actual (qué vecino o hijo del nodo actual es considerado como candidato para sustituirlo) se puntúa sobre 4.5 puntos.
- La correcta aplicación en cada paso del algoritmo del *criterio de aceptación* del vecino o hijo seleccionado (si el vecino o hijo candidato sustituye o no finalmente al nodo actual) se puntúa sobre 4.5 puntos.
- La correcta aplicación del *criterio de finalización* del algoritmo se puntúa sobre 1 punto.

SOLUCIÓN DEL EJERCICIO 6 (por Severino Fernández Galán):

- PASO 1: Al principio expandimos el nodo inicial *D*, generando sus nodos hijos {*B*(25), *I*(20), *E*(35)}. Seleccionamos el nodo *I* por ser el mejor de los nodos hijos. A continuación aceptamos el nodo *I* como nuevo nodo actual en sustitución de *D*, debido a que el valor de la función de evaluación heurística de *I* es mejor o igual que el de *D* ($20 \leq 30$).

- PASO 2: Expandimos el nodo actual *I* y generamos sus hijos: {*C*(20), *G*(40)}. Seleccionamos el nodo *C* por ser el mejor de los nodos hijos. Seguidamente aceptamos el nodo *C* como nuevo nodo actual en sustitución de *I*, debido a que el valor de la función de evaluación heurística de *C* es mejor o igual que el de *I* ($20 \leq 20$).

- PASO 3: Expandimos el nodo actual *C* y generamos sus hijos: {*F*(8), *G*(40)}. Seleccionamos el nodo *F* por ser el mejor de los nodos hijos. A continuación aceptamos el nodo *F* como nuevo nodo actual en sustitución de *C*, debido a que el valor de la función de evaluación heurística de *F* es mejor o igual que el de *C* ($8 \leq 20$).

- PASO 4: Expandimos el nodo actual *F* y generamos sus hijos: {*H*(2)}. Aceptamos el nodo *H* como nuevo nodo actual en sustitución de *F*, debido a que el valor de la función de evaluación heurística de *H* es mejor o igual que el de *F* ($2 \leq 8$).

Dado que *H* no tiene sucesores, la búsqueda terminaría. El algoritmo devolvería el nodo *H*(2) como mejor nodo encontrado.

AÑO 2013

EJERCICIO 1:

Dibuje mediante un grafo dirigido o describa detalladamente mediante una tabla el espacio de estados (o espacio de búsqueda) completo para el *problema de los misioneros y los caníbales*. Para ello especifique: el conjunto de todos los estados posibles, el estado inicial, el o los estados meta, los operadores aplicables a cada estado y el coste asociado a cada operador. En el problema de los misioneros y los caníbales, tres misioneros están junto a tres caníbales en una de las orillas de un río. Existe un bote disponible en el río con capacidad máxima para dos personas. Se pide encontrar la manera de trasladar a los misioneros y a los caníbales a la otra orilla del río, con la condición de que en ningún momento quede en contacto un número de misioneros con un número mayor de caníbales. La dificultad reside en que si la condición mencionada no se cumple, alguno de los misioneros es devorado y no se puede alcanzar el objetivo del juego. ¿Cuál es la solución menos costosa para este problema?

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 1:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- Los estados se especifican correctamente: 2.5 puntos
- Los operadores se especifican correctamente: 2.5 puntos
- Los costes de los operadores se especifican correctamente: 1 punto
- El espacio de búsqueda se dibuja (mediante un grafo dirigido) o se describe (mediante una tabla) correctamente: 3 puntos
- La solución de menor coste se especifica correctamente: 1 punto

SOLUCIÓN DEL EJERCICIO 1 (por Severino Fernández Galán):

Para representar los estados posibles utilizaremos la siguiente notación: “M” se corresponde con un misionero, “C” se corresponde con un caníbal y “B” se corresponde con el bote. Además, el estado del problema lo representamos como (X-Y), donde X contiene los elementos que están en la orilla izquierda y donde Y contiene el resto de elementos, que están en la orilla derecha. Por ejemplo, (MMCB-MCC) representa el estado en el que en la orilla izquierda hay dos misioneros junto con un caníbal y el bote, mientras que en la orilla derecha hay un misionero junto con dos caníbales.

Supongamos que inicialmente todos los elementos están en la orilla izquierda y queremos pasarlos a la orilla derecha. Por tanto, el estado inicial será (BCCCMMM-), mientras que el único estado meta será (-BCCCMMM). Obsérvese que los elementos de una misma orilla los ordenamos alfabéticamente por sencillez en la exposición.

Dado un estado cualquiera, el número de operadores aplicables al mismo depende del número de personas situadas en la orilla del bote. Además, hay que tener en cuenta que la capacidad máxima del bote es de dos personas. La notación que empleamos para cada operador es una lista Z tal que sus elementos denotan las personas que viajan en el bote, independientemente de la dirección del viaje. De este modo, existen cinco operadores posibles: C, M, CC, MM y CM. Se puede considerar que cada operador tiene coste unidad, ya que buscamos la solución que realice menos viajes de orilla a orilla con el bote.

En la tabla 1.1 figuran los estados posibles del sistema, los operadores que se pueden aplicar a cada estado posible y los estados resultantes de aplicar cada operador.

De forma alternativa, la información incluida en la tabla 1.1 se puede representar mediante un grafo dirigido tal como se muestra en la figura 1.1. En dicha figura, los estados no permitidos se han marcado con líneas discontinuas y se ha utilizado línea doble para marcar el estado inicial y el estado meta. Por otra parte, obsérvese que los estados {BMMM-CCC, BMM-CCCM, BCCM-CMM, BM-CCCMM, BC-CCMMM, B-CCCMMM, CCCMMM-B, CCC-BMMM} quedan aislados del espacio de búsqueda, al no ser alcanzables desde el estado inicial.

La solución menos costosa para el problema de los misioneros y los caníbales (o, en este caso, las soluciones menos costosas) consisten en: desplazar un caníbal junto con otra persona (misionero o caníbal) a la orilla derecha, hacer regresar a la persona mencionada, desplazar dos caníbales a la orilla derecha, hacer regresar a uno de ellos, desplazar dos misioneros a la orilla derecha, hacer que regresen un caníbal junto con un misionero, desplazar dos misioneros a la orilla derecha, hacer que regrese un caníbal, desplazar dos caníbales a la orilla derecha, hacer que regrese únicamente un misionero o un caníbal y, finalmente, hacer que las dos personas restantes en la orilla izquierda se desplacen a la orilla derecha. Las soluciones menos costosas correspondientes aparecen marcadas en negrita en la figura 1.1.

ESTADOS	OPERADORES				
	C	M	CC	MM	CM
BCCCMMM-	CCMMM-BC	CCCMM-BM	CMMM-BCC	CCCM-BMM	CCMM-BCM
BCCMMM-C	CMMM-BCC	CCMM-BCM	MMM-BCCC	CCM-BCMM	CMM-BCCM
BCMMM-CC	MMM-BCCC	CMM-BMCC	No aplicable	CM-BCCMM	MM-BCCCM
BMMM-CCC	No aplicable	MM-BCCCM	No aplicable	M-BCCCMM	No aplicable
BCCCMM-M	Estado no permitido				
BCCMM-CM	CMM-BCCM	CCM-BCMM	MM-BCCCM	CC-BCMMM	CM-BCCMM
BCMM-CCM	Estado no permitido				
BMM-CCCM	Estado no permitido				
BCCCM-MM	Estado no permitido				
BCCM-CMM	Estado no permitido				
BCM-CCMM	M-BCCCMM	C-BCCMMM	No aplicable	No aplicable	-BCCCMMM
BM-CCCMM	Estado no permitido				
BCCC-MMM	CC-BCMMM	No aplicable	C-BCCMMM	No aplicable	No aplicable
BCC-CMMM	C-BCCMMM	No aplicable	-BCCCMMM	No aplicable	No aplicable
BC-CCMMM	-BCCCMMM	No aplicable	No aplicable	No aplicable	No aplicable
B-CCCMMM	Estado no permitido				
CCCMMM-B	Estado no permitido				
CCMMM-BC	BCCCMMM-	No aplicable	No aplicable	No aplicable	No aplicable
CMMM-BCC	BCCMMM-C	No aplicable	BCCCMMM-	No aplicable	No aplicable
MMM-BCCC	BCMMM-CC	No aplicable	BCCMMM-C	No aplicable	No aplicable
CCCMM-BM	Estado no permitido				
CCMM-BCM	BCCCMM-M	BCCMMM-C	No aplicable	No aplicable	BCCCMMM-
CMM-BCCM	Estado no permitido				
MM-BCCCM	Estado no permitido				
CCCM-BMM	Estado no permitido				
CCM-BCMM	Estado no permitido				
CM-BCCMM	BCCM-CMM	BCMM-CCM	BCCCM-MM	BCMMM-CC	BCCMM-CM
M-BCCCMM	Estado no permitido				
CCC-BMMM	No aplicable	BCCCM-MM	No aplicable	BCCCMM-M	No aplicable
CC-BCMMM	BCCC-MMM	BCCM-CMM	No aplicable	BCCMM-CM	BCCCM-MM
C-BCCMMM	BCC-CMMM	BCM-CCMM	BCCC-MMM	BCMM-CCM	BCCM-CMM
-BCCCMMM	Estado META				

Tabla 1.1: Estados posibles, operadores aplicables a cada estado y estados resultantes de aplicar cada operador para el problema de los misioneros y los caníbales.

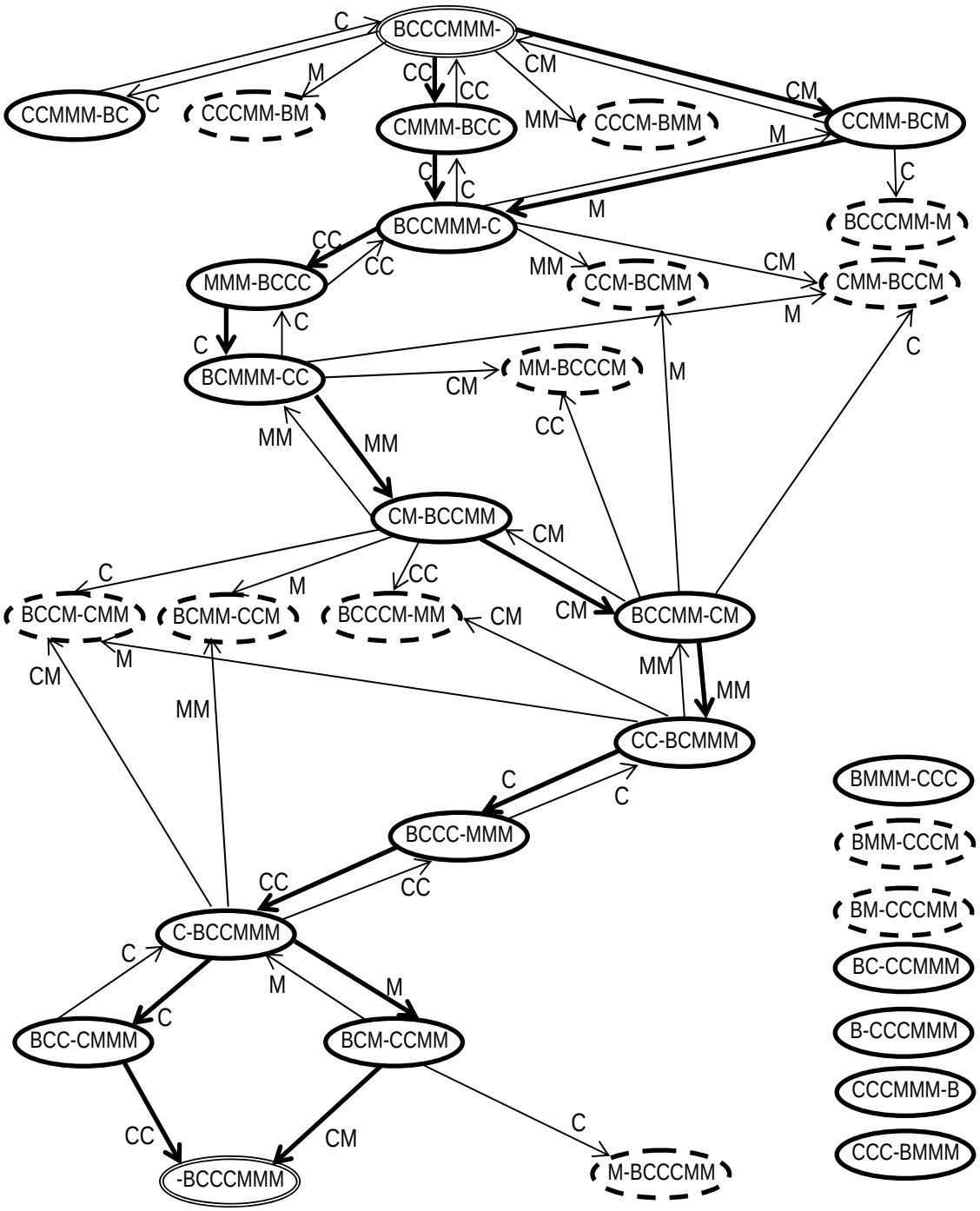


Figura 1.1: Grafo dirigido que representa el espacio de búsqueda para el problema de los misioneros y los caníbales. Se han incluido también los estados no alcanzables desde el estado inicial, que quedan aislados del espacio de búsqueda.

EJERCICIO 2:

Considere el espacio de búsqueda de la figura 2.1, que tiene forma de árbol, donde el nodo raíz del árbol es el nodo inicial, existe un único nodo meta y cada operador tiene asociado un coste. Explique razonadamente en qué orden se expandirían los nodos de dicho árbol de búsqueda a partir de cada uno de los métodos siguientes de búsqueda sin información del dominio:

1. Búsqueda Primero en Anchura (de izquierda a derecha)
2. Búsqueda Primero en Profundidad (de derecha a izquierda)
3. Búsqueda de Coste Uniforme
4. Búsqueda en Anchura Iterativa (de derecha a izquierda)
5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

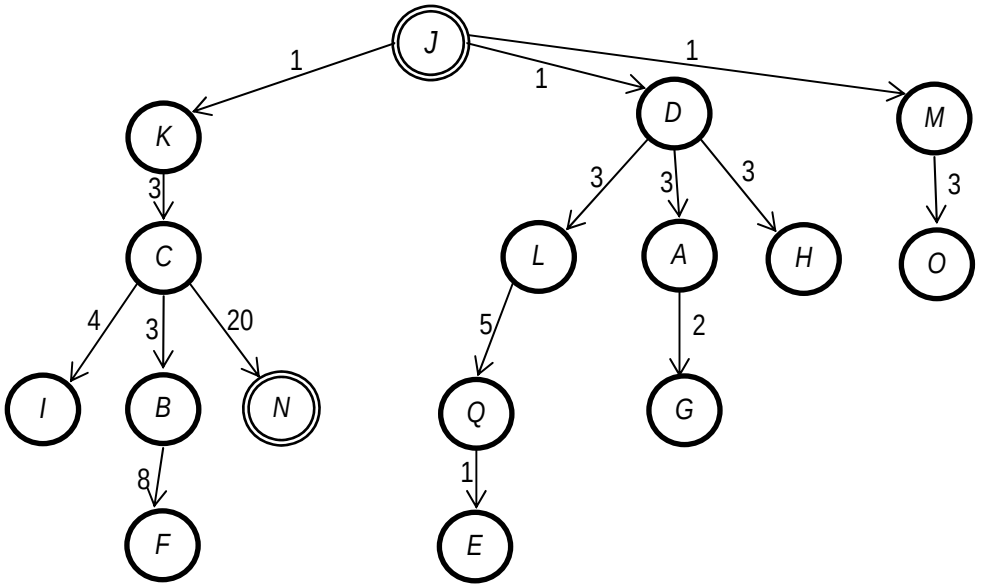


Figura 2.1: Árbol de búsqueda en el que el nodo inicial es *J*, el nodo meta es *N* y el coste de cada operador aparece al lado del arco que lo representa.

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 2:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- Cada uno de los cinco apartados, correspondiente a un método de búsqueda concreto, se puntúa sobre 2 puntos.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no expande los nodos en el orden debido: la puntuación del apartado bajaría 1.6 puntos si el orden dado como respuesta varía significativamente del correcto, mientras que si el orden dado como respuesta varía del correcto como consecuencia de un despiste no conceptual entonces la puntuación del apartado bajaría sólo 0.4 puntos en vez de los 1.6 puntos mencionados.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no finaliza cuando es debido, la puntuación del apartado bajaría 0.4 puntos. (Esto es independiente del orden de expansión de nodos dado como respuesta, que ya ha sido valorado anteriormente con un máximo de 1.6 puntos.)

SOLUCIÓN DEL EJERCICIO 2 (por Severino Fernández Galán):

1. Búsqueda Primero en Anchura (de izquierda a derecha)

Este algoritmo explora el árbol de búsqueda por niveles de profundidad, así que el orden de expansión de los nodos de izquierda a derecha sería el reflejado en la figura 2.2.

2. Búsqueda Primero en Profundidad (de derecha a izquierda)

Este algoritmo explora el árbol de búsqueda bajando de nivel siempre que sea posible. Si no es posible, se sube al nodo más cercano al nodo actual desde el que poder seguir bajando de nivel. El orden de expansión de los nodos de derecha a izquierda según este algoritmo se dibuja en la figura 2.3.

3. Búsqueda de Coste Uniforme

Este algoritmo explora el árbol de búsqueda expandiendo aquel nodo disponible cuyo coste al nodo inicial sea el menor. El orden de expansión de nodos según este criterio se indica en la figura 2.4. Obsérvese que después de la primera expansión (nodo *J*), hay un empate entre los nodos *K*, *D* y *M*, cuyos costes al nodo inicial son iguales a 1 en los tres casos. Nosotros hemos elegido el nodo *K* para realizar la segunda expansión, pero también se podría haber elegido cualquiera de los otros dos nodos, *D* o *M*. Los empates posteriores también los deshacemos de esta forma arbitraria.

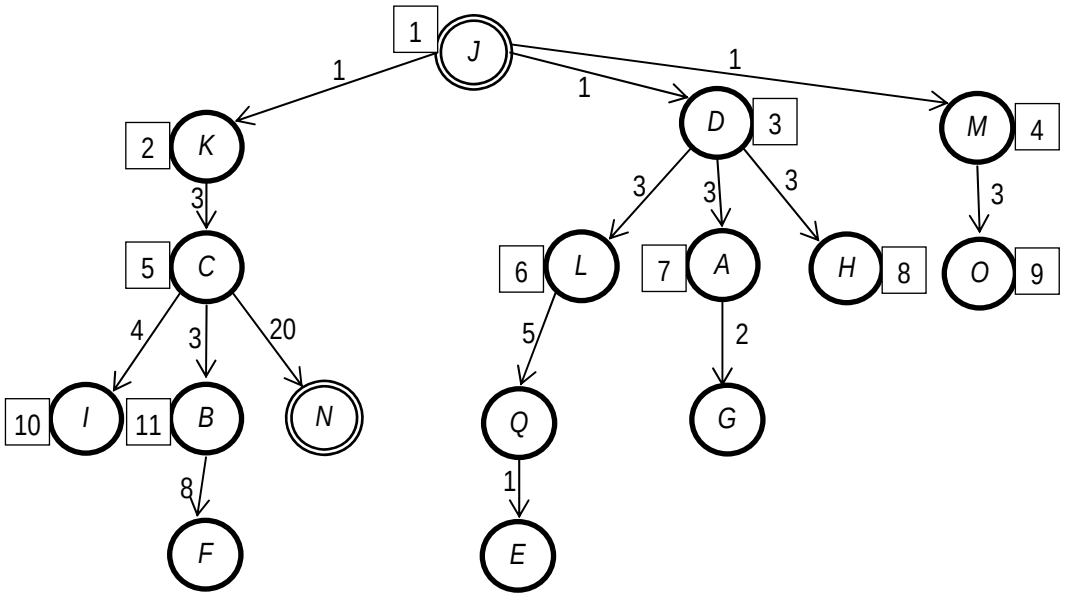


Figura 2.2: Orden de expansión de nodos para el árbol de la figura 2.1 según la búsqueda primero en anchura (de izquierda a derecha). Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto, el algoritmo termina en ese momento sin que se lleguen a generar sus hijos.

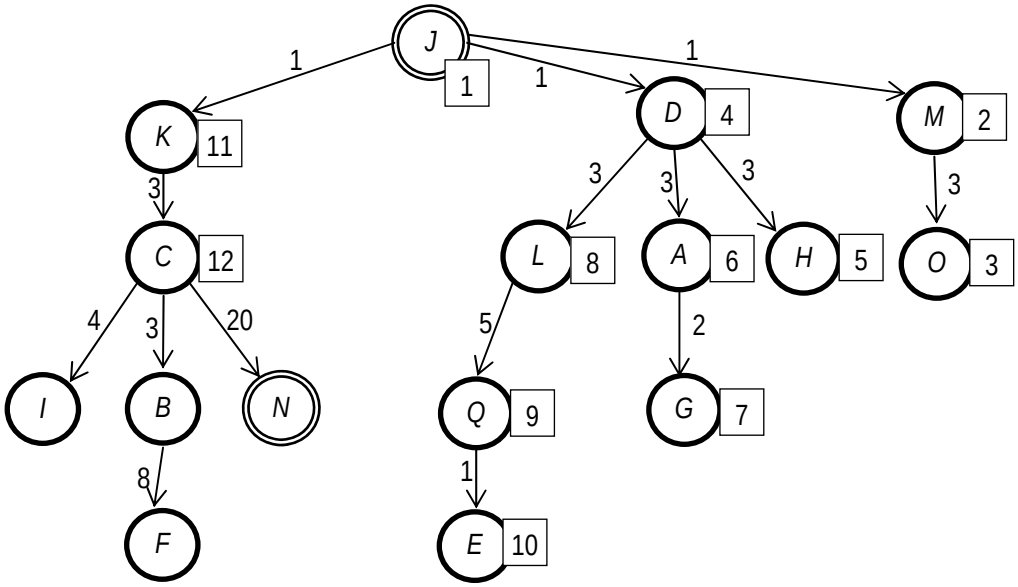


Figura 2.3: Orden de expansión de nodos para el árbol de la figura 2.1 según la búsqueda primero en profundidad (de derecha a izquierda). Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto, el algoritmo termina en ese momento sin que se lleguen a generar sus hijos.

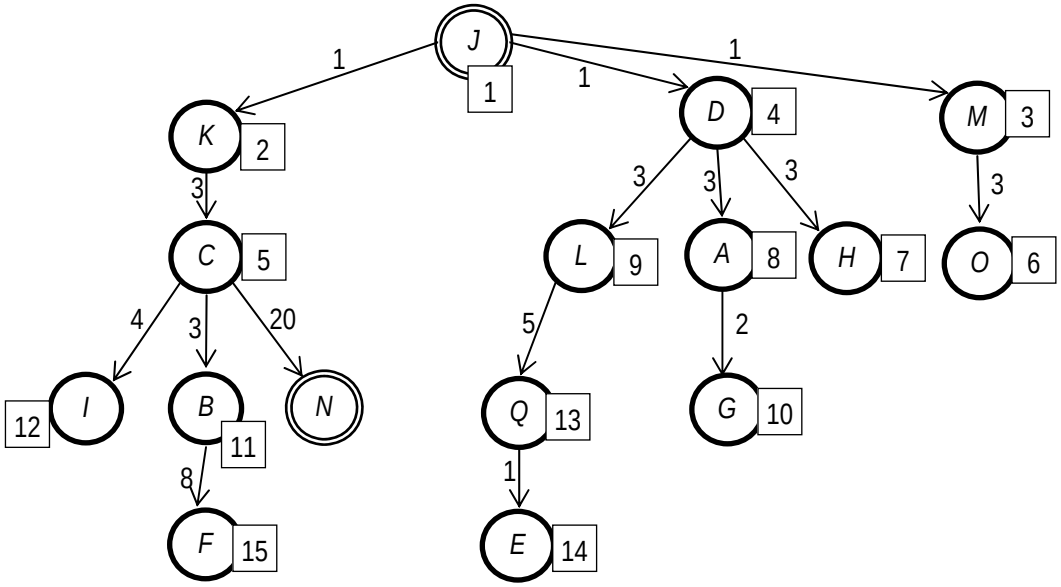


Figura 2.4: Orden de expansión de nodos para el árbol de la figura 2.1 según la búsqueda de coste uniforme. Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto, el algoritmo termina en ese momento sin que se lleguen a generar sus hijos.

4. Búsqueda en Anchura Iterativa (de derecha a izquierda)

Este algoritmo ejecuta iterativamente varias búsquedas primero en anchura, de manera que entre iteración e iteración se incrementa en una unidad el número máximo de hijos que se generan en cada expansión de un nodo padre. Al principio (en la primera iteración), únicamente un hijo es generado en cada expansión. En las figuras 2.5a, 2.5b y 2.5c se muestran los órdenes de expansión de nodos para cada una de las iteraciones de búsqueda primero en anchura que son necesarias para este ejemplo de búsqueda en anchura iterativa.

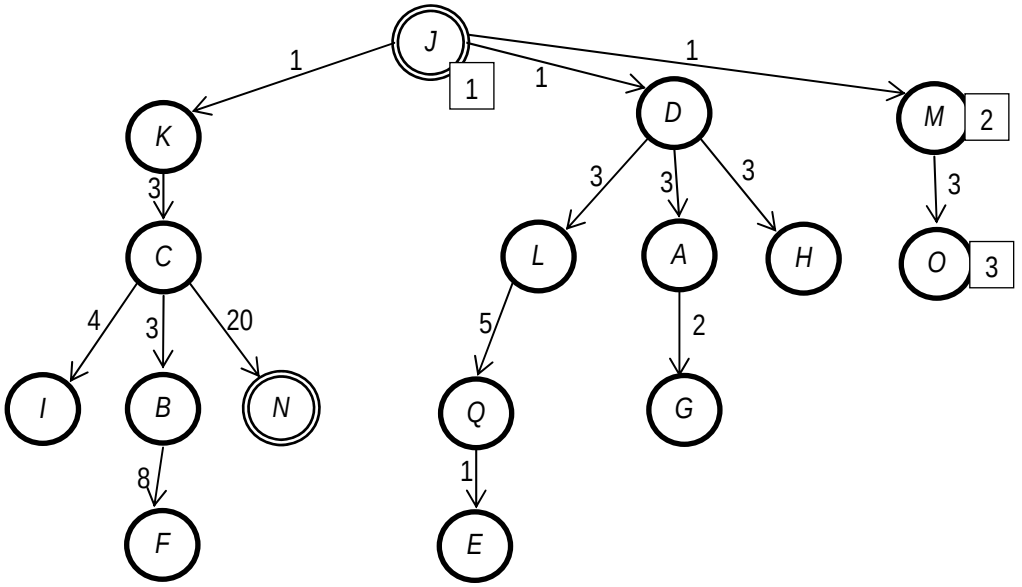


Figura 2.5a: Primera iteración de la búsqueda en anchura iterativa de derecha a izquierda, en la que como máximo un hijo es generado en cada expansión de un nodo padre.

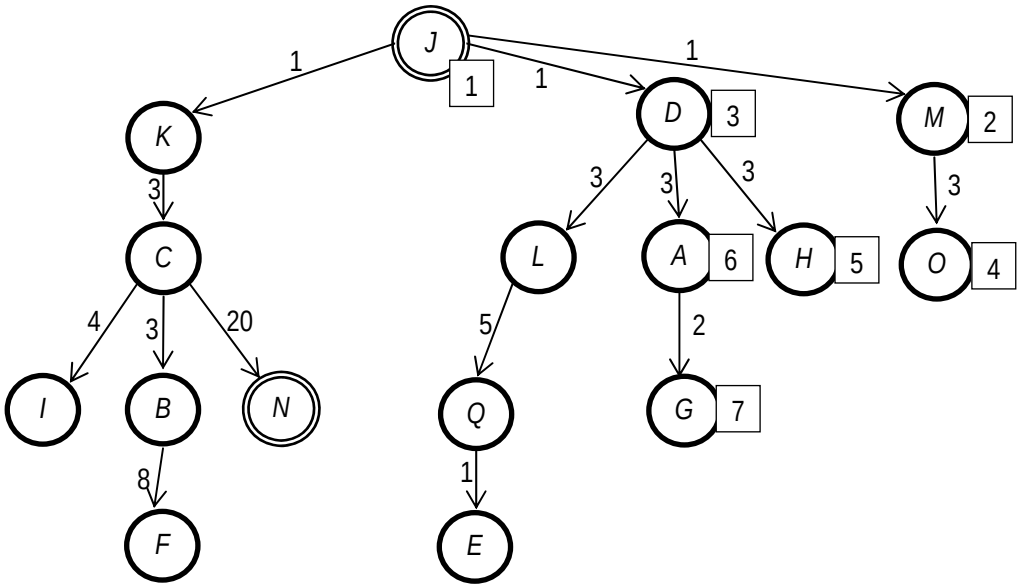


Figura 2.5b: Segunda iteración de la búsqueda en anchura iterativa de derecha a izquierda, en la que como máximo dos hijos son generados en cada expansión de un nodo padre.

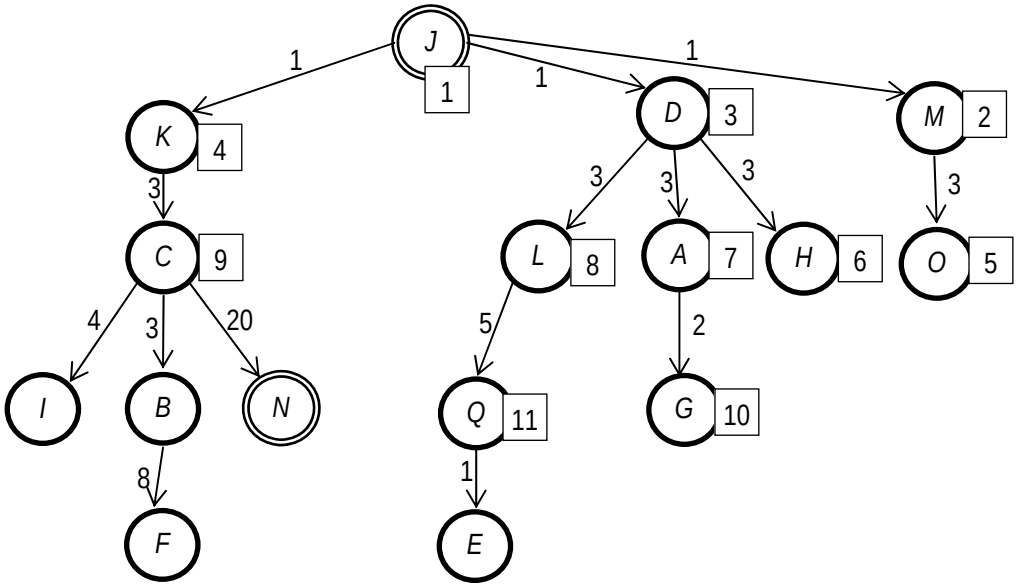


Figura 2.5c: Tercera y última iteración de la búsqueda en anchura iterativa de derecha a izquierda, en la que como máximo tres hijos son generados en cada expansión de un nodo padre.

Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto, el algoritmo termina en ese momento sin que se lleguen a generar sus hijos.

5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

Este algoritmo ejecuta iterativamente varias búsquedas primero en profundidad, de manera que entre iteración e iteración se incrementa en una unidad la profundidad límite. Al principio (en la primera iteración), la profundidad límite es igual a 1, así que sólo se podrá expandir el nodo inicial, cuya profundidad es igual a 0. En las figuras 2.6a, 2.6b y 2.6c se muestran los órdenes de expansión de nodos para cada una de las iteraciones de búsqueda primero en profundidad que son necesarias para este ejemplo de búsqueda en profundidad iterativa.

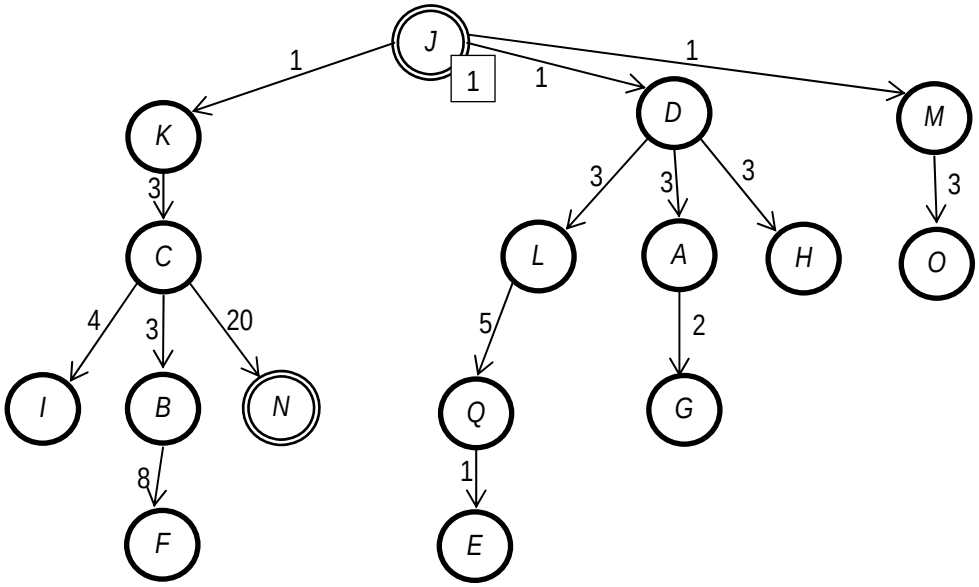


Figura 2.6a: Primera iteración de la búsqueda en profundidad iterativa de izquierda a derecha, en la que la profundidad límite es igual a 1 y únicamente son expandidos nodos con una profundidad máxima de 0.

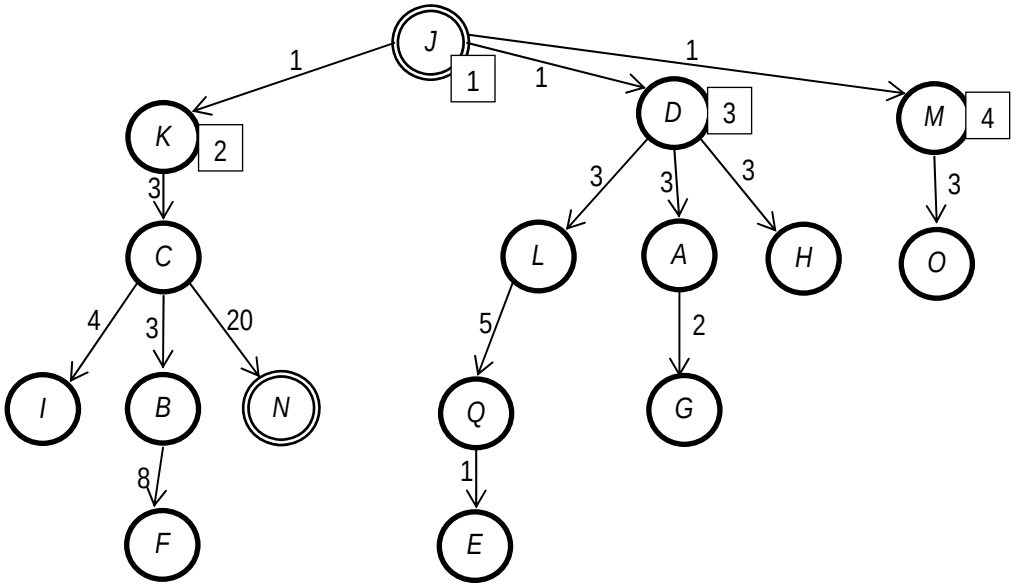


Figura 2.6b: Segunda iteración de la búsqueda en profundidad iterativa de izquierda a derecha, en la que la profundidad límite es igual a 2 y únicamente son expandidos nodos con una profundidad máxima de 1.

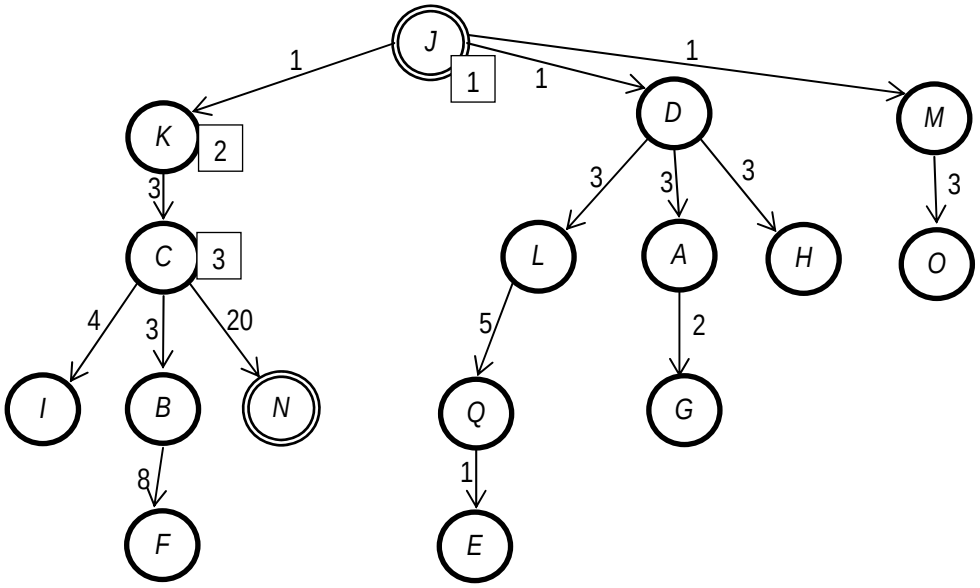


Figura 2.6c: Tercera iteración de la búsqueda en profundidad iterativa de izquierda a derecha, en la que la profundidad límite es igual a 3 y únicamente son expandidos nodos con una profundidad máxima de 2.

Observe que realmente el nodo meta no es expandido en esta última iteración porque se encuentre a la profundidad límite. Esta condición no se llega a comprobar, ya que justo antes se averigua que el nodo es meta y, por tanto, el algoritmo termina.

EJERCICIO 3:

Considere el espacio de búsqueda de la figura 3.1, que tiene forma de árbol, donde el nodo raíz del árbol es el nodo inicial, existe un único nodo meta y cada operador tiene asociado un coste. Describa cuál es el contenido de ABIERTA, previamente a cada extracción de un nodo de la misma, a partir de cada uno de los métodos siguientes de búsqueda sin información del dominio:

1. Búsqueda Primero en Anchura (de izquierda a derecha)
2. Búsqueda Primero en Profundidad (de derecha a izquierda)
3. Búsqueda de Coste Uniforme
4. Búsqueda en Anchura Iterativa (de derecha a izquierda)
5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

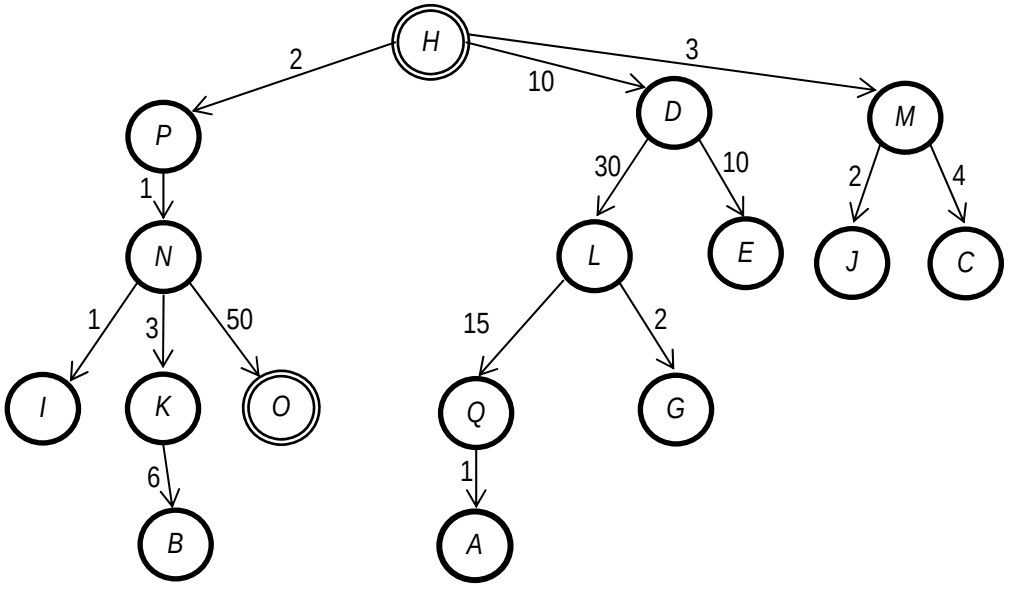


Figura 3.1: Árbol de búsqueda en el que el nodo inicial es *H*, el nodo meta es *O* y el coste de cada operador aparece al lado del arco que lo representa.

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 3:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- Cada uno de los cinco apartados, correspondiente a un método de búsqueda concreto, se puntúa sobre 2 puntos.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no gestiona ABIERTA en la forma debida: la puntuación del apartado bajaría 1.6 puntos si la respuesta dada varía significativamente de la correcta, mientras que si la respuesta dada varía de la correcta como consecuencia de un despiste no conceptual entonces la puntuación del apartado bajaría sólo 0.4 puntos en vez de los 1.6 puntos mencionados.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no finaliza cuando es debido, la puntuación del apartado bajaría 0.4 puntos. (Esto es independiente de la gestión de ABIERTA dada como respuesta, que ya ha sido valorada anteriormente con un máximo de 1.6 puntos.)

SOLUCIÓN DEL EJERCICIO 3 (por Severino Fernández Galán):

1. Búsqueda Primero en Anchura (de izquierda a derecha)

Este algoritmo usa ABIERTA como una cola, de manera que siempre se saca el primer nodo de la cola y se introducen sus hijos al final de la misma. El contenido de ABIERTA previamente a cada extracción de un nodo es el siguiente:

Antes de sacar *H*: {*H*}
Antes de sacar *P*: {*P*, *D*, *M*}
Antes de sacar *D*: {*D*, *M*, *N*}
Antes de sacar *M*: {*M*, *N*, *L*, *E*}
Antes de sacar *N*: {*N*, *L*, *E*, *J*, *C*}
Antes de sacar *L*: {*L*, *E*, *J*, *C*, *I*, *K*, *O*}
Antes de sacar *E*: {*E*, *J*, *C*, *I*, *K*, *O*, *Q*, *G*}
Antes de sacar *J*: {*J*, *C*, *I*, *K*, *O*, *Q*, *G*}
Antes de sacar *C*: {*C*, *I*, *K*, *O*, *Q*, *G*}
Antes de sacar *I*: {*I*, *K*, *O*, *Q*, *G*}
Antes de sacar *K*: {*K*, *O*, *Q*, *G*}
Antes de sacar *O*: {*O*, *Q*, *G*, *B*}
META ENCONTRADA

Observe que, tras cada expansión del nodo sacado de ABIERTA, sus nodos hijos más a la izquierda se introducen en ABIERTA antes que los situados más a la derecha.

2. Búsqueda Primero en Profundidad (de derecha a izquierda)

Este algoritmo usa ABIERTA como una pila, de manera que siempre se saca el primer nodo de la pila y se introducen sus hijos al principio de la misma. El contenido de ABIERTA previamente a cada extracción de un nodo es el siguiente:

Antes de sacar *H*: {*H*}
Antes de sacar *M*: {*M*, *D*, *P*}
Antes de sacar *C*: {*C*, *J*, *D*, *P*}
Antes de sacar *J*: {*J*, *D*, *P*}
Antes de sacar *D*: {*D*, *P*}
Antes de sacar *E*: {*E*, *L*, *P*}
Antes de sacar *L*: {*L*, *P*}
Antes de sacar *G*: {*G*, *Q*, *P*}
Antes de sacar *Q*: {*Q*, *P*}
Antes de sacar *A*: {*A*, *P*}
Antes de sacar *P*: {*P*}
Antes de sacar *N*: {*N*}
Antes de sacar *O*: {*O*, *K*, *I*}
META ENCONTRADA

Observe que, tras cada expansión del nodo sacado de ABIERTA, sus nodos hijos más a la derecha se introducen en ABIERTA después que los situados más a la izquierda.

3. Búsqueda de Coste Uniforme

Este algoritmo mantiene ABIERTA ordenada según el coste del camino desde cada nodo al nodo inicial. En este ejercicio desharemos de forma arbitraria los posibles empates que surjan al determinar qué nodo se debe sacar de ABIERTA. El contenido de ABIERTA previamente a cada extracción de un nodo es el siguiente:

Antes de sacar *H*: {*H*(0)}
Antes de sacar *P*: {*P*(2), *M*(3), *D*(10)}

Antes de sacar *N*: {*N*(3), *M*(3), *D*(10)}
Antes de sacar *M*: {*M*(3), *I*(4), *K*(6), *D*(10), *O*(53)}
Antes de sacar *I*: {*I*(4), *J*(5), *K*(6), *C*(7), *D*(10), *O*(53)}
Antes de sacar *J*: {*J*(5), *K*(6), *C*(7), *D*(10), *O*(53)}
Antes de sacar *K*: {*K*(6), *C*(7), *D*(10), *O*(53)}
Antes de sacar *C*: {*C*(7), *D*(10), *B*(12), *O*(53)}
Antes de sacar *D*: {*D*(10), *B*(12), *O*(53)}
Antes de sacar *B*: {*B*(12), *E*(20), *L*(40), *O*(53)}
Antes de sacar *E*: {*E*(20), *L*(40), *O*(53)}
Antes de sacar *L*: {*L*(40), *O*(53)}
Antes de sacar *G*: {*G*(42), *O*(53), *Q*(55)}
Antes de sacar *O*: {*O*(53), *Q*(55)}
META ENCONTRADA

Observe que, tras cada expansión de un nodo, sus nodos hijos son introducidos en ABIERTA, donde todos los nodos quedan siempre ordenados por coste creciente. Para facilitar el seguimiento del algoritmo, entre paréntesis y al lado de cada nodo de ABIERTA se especifica el coste del camino para ir desde dicho nodo hasta el nodo inicial.

4. Búsqueda en Anchura Iterativa (de derecha a izquierda)

Debido a que este algoritmo es una iteración de la búsqueda primero en anchura, los dos gestionan ABIERTA del mismo modo, es decir, como una cola. La diferencia reside en que en la búsqueda en anchura iterativa en cada iteración se fija un número máximo de hijos que pueden ser generados al expandir un nodo padre. Este número empieza valiendo 1 y es incrementado en una unidad al principio de cada nueva iteración.

Iteración 1 (cada expansión de un nodo padre genera como máximo un hijo):
Antes de sacar *H*: {*H*}
Antes de sacar *M*: {*M*}
Antes de sacar *C*: {*C*}

Iteración 2 (cada expansión de un nodo padre genera como máximo dos hijos):
Antes de sacar *H*: {*H*}
Antes de sacar *M*: {*M*, *D*}
Antes de sacar *D*: {*D*, *C*, *J*}
Antes de sacar *C*: {*C*, *J*, *E*, *L*}
Antes de sacar *J*: {*J*, *E*, *L*}
Antes de sacar *E*: {*E*, *L*}
Antes de sacar *L*: {*L*}
Antes de sacar *G*: {*G*, *Q*}
Antes de sacar *Q*: {*Q*}
Antes de sacar *A*: {*A*}

Iteración 3 (cada expansión de un nodo padre genera como máximo tres hijos):
Antes de sacar *H*: {*H*}
Antes de sacar *M*: {*M*, *D*, *P*}
Antes de sacar *D*: {*D*, *P*, *C*, *J*}
Antes de sacar *P*: {*P*, *C*, *J*, *E*, *L*}
Antes de sacar *C*: {*C*, *J*, *E*, *L*, *N*}
Antes de sacar *J*: {*J*, *E*, *L*, *N*}
Antes de sacar *E*: {*E*, *L*, *N*}
Antes de sacar *L*: {*L*, *N*}
Antes de sacar *N*: {*N*, *G*, *Q*}
Antes de sacar *G*: {*G*, *Q*, *O*, *K*, *I*}
Antes de sacar *Q*: {*Q*, *O*, *K*, *I*}
Antes de sacar *O*: {*O*, *K*, *I*, *A*}
META ENCONTRADA

5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

Debido a que este algoritmo es una iteración de la búsqueda primero en profundidad, los dos gestionan ABIERTA del mismo modo, es decir, como una pila. La diferencia reside en que en la búsqueda en profundidad iterativa en cada iteración se fija una profundidad límite propia. Este número empieza valiendo 1, lo cual permite expandir únicamente el nodo inicial, y es incrementado en una unidad al principio de cada nueva iteración.

Iteración 1 (profundidad límite igual a 1):

Antes de sacar *H*: {*H*}
Antes de sacar *P*: {*P*, *D*, *M*}
Nótese que *P* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *D*: {*D*, *M*}
Nótese que *D* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *M*: {*M*}
Nótese que *M* no es expandido por coincidir su profundidad con la profundidad límite.

Iteración 2 (profundidad límite igual a 2):

Antes de sacar *H*: {*H*}
Antes de sacar *P*: {*P*, *D*, *M*}
Antes de sacar *N*: {*N*, *D*, *M*}
Nótese que *N* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *D*: {*D*, *M*}
Antes de sacar *L*: {*L*, *E*, *M*}
Nótese que *L* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *E*: {*E*, *M*}
Nótese que *E* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *M*: {*M*}
Antes de sacar *J*: {*J*, *C*}
Nótese que *J* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *C*: {*C*}
Nótese que *C* no es expandido por coincidir su profundidad con la profundidad límite.

Iteración 3 (profundidad límite igual a 3):

Antes de sacar *H*: {*H*}
Antes de sacar *P*: {*P*, *D*, *M*}
Antes de sacar *N*: {*N*, *D*, *M*}
Antes de sacar *I*: {*I*, *K*, *O*, *D*, *M*}
Nótese que *I* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *K*: {*K*, *O*, *D*, *M*}
Nótese que *K* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *O*: {*O*, *D*, *M*}
META ENCONTRADA

EJERCICIO 4:

Considere el espacio de búsqueda de la figura 4.1, que tiene forma de árbol, donde el nodo raíz del árbol es el nodo inicial, existe un único nodo meta y cada operador tiene asociado un coste. Describa cuál es el contenido de TABLA_A, posteriormente a cada expansión de un nodo, a partir de cada uno de los métodos siguientes de búsqueda sin información del dominio:

1. Búsqueda Primero en Anchura (de izquierda a derecha)
2. Búsqueda Primero en Profundidad (de derecha a izquierda)
3. Búsqueda de Coste Uniforme
4. Búsqueda en Anchura Iterativa (de derecha a izquierda)
5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

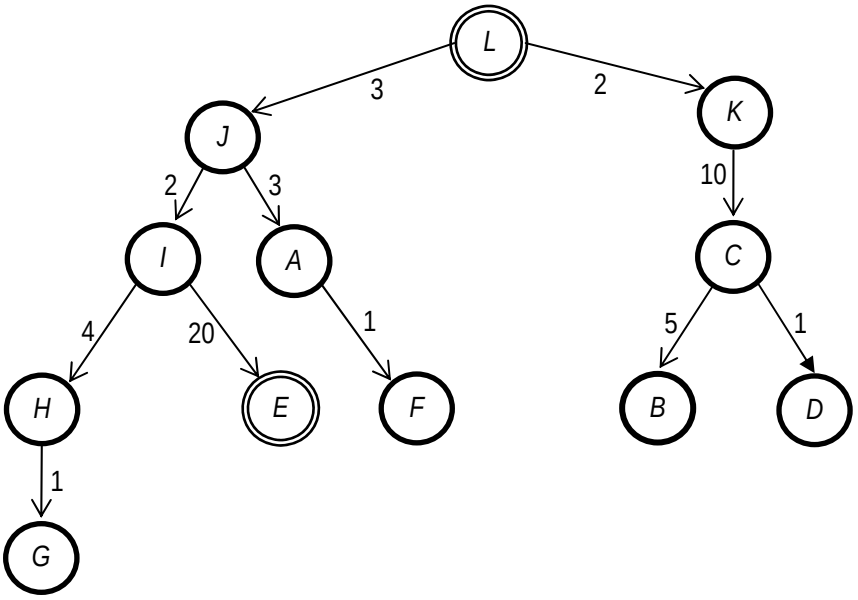


Figura 4.1: Árbol de búsqueda en el que el nodo inicial es *L*, el nodo meta es *E* y el coste de cada operador aparece al lado del arco que lo representa.

Para cada nodo de TABLA_A incluya la siguiente información: su nodo padre y el coste al nodo inicial. (En este ejercicio no es necesario incluir en TABLA_A información sobre los hijos de cada nodo expandido, ya que sólo existe un camino desde cada nodo al nodo inicial y, por tanto, el mejor camino desde cada nodo al nodo inicial no cambia a lo largo del proceso de búsqueda.)

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 4:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

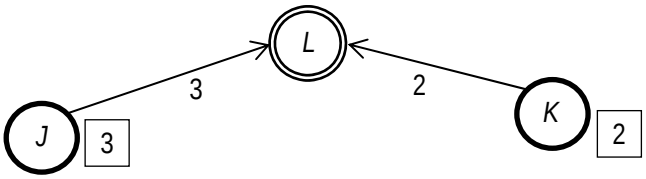
- Cada uno de los cinco apartados, correspondiente a un método de búsqueda concreto, se puntúa sobre 2 puntos.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no gestiona TABLA_A en la forma debida: la puntuación del apartado bajaría 1.6 puntos si la respuesta dada varía significativamente de la correcta, mientras que si la respuesta dada varía de la correcta como consecuencia de un despiste no conceptual entonces la puntuación del apartado bajaría sólo 0.4 puntos en vez de los 1.6 puntos mencionados.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no finaliza cuando es debido, la puntuación del apartado bajaría 0.4 puntos. (Esto es independiente de la gestión de TABLA_A dada como respuesta, que ya ha sido valorada anteriormente con un máximo de 1.6 puntos.)

SOLUCIÓN DEL EJERCICIO 4 (por Severino Fernández Galán):

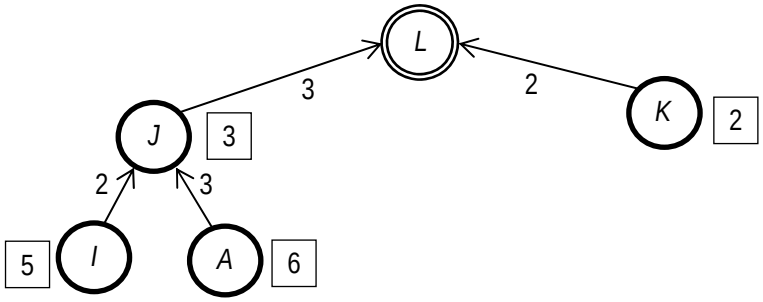
De cara a ilustrar la respuesta convenientemente, incluimos la información de TABLA_A gráficamente: por un lado, para cada nodo generado en una expansión trazamos un arco ascendente a su padre y, por otro lado, indicamos el coste al nodo inicial al lado de cada nodo generado.

1. Búsqueda Primero en Anchura (de izquierda a derecha)

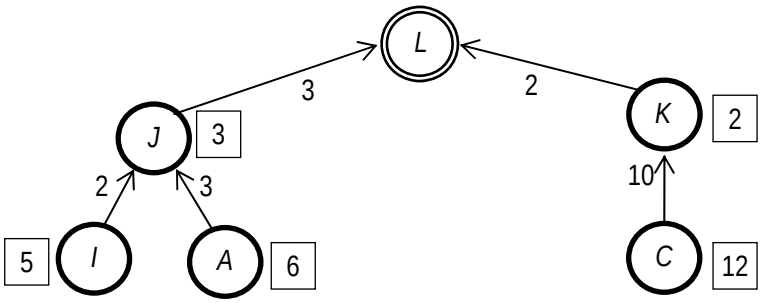
- Inicialmente, *L* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *L*. A continuación expandimos el nodo inicial, generando sus nodos hijos. Desde cada nodo hijo trazamos un nodo ascendente a su nodo padre. Al lado de cada nodo hijo indicamos el coste desde dicho nodo al nodo inicial.



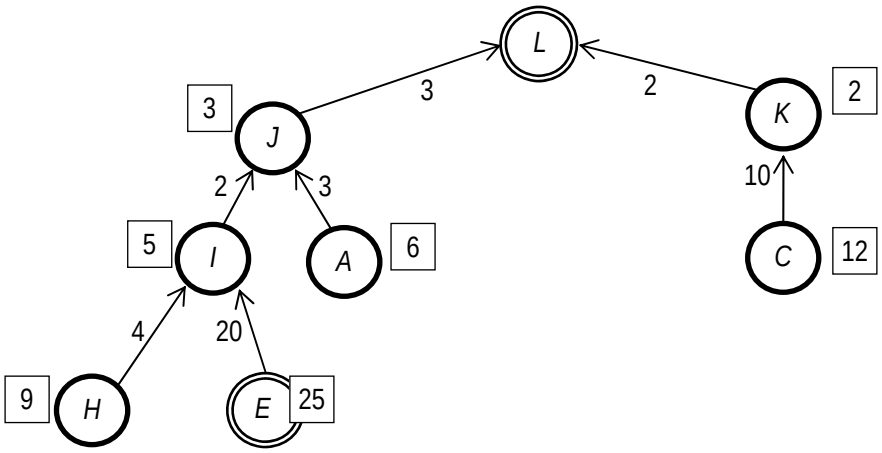
- Expandimos *J*:



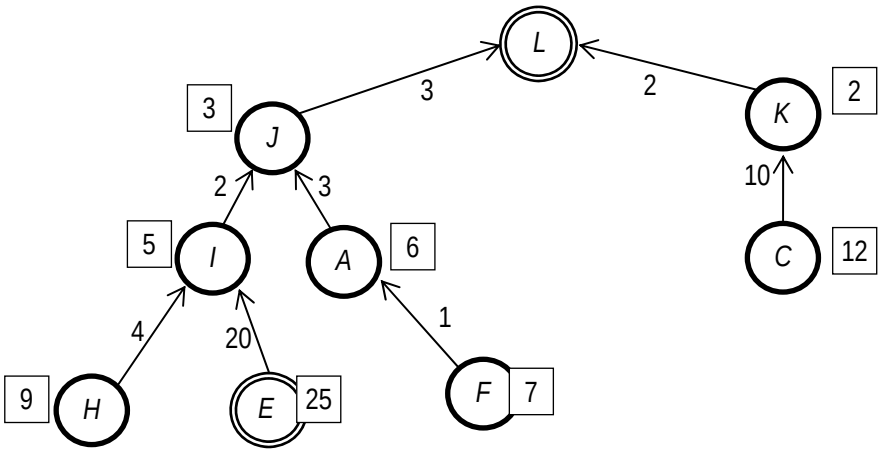
- Expandimos *K*:



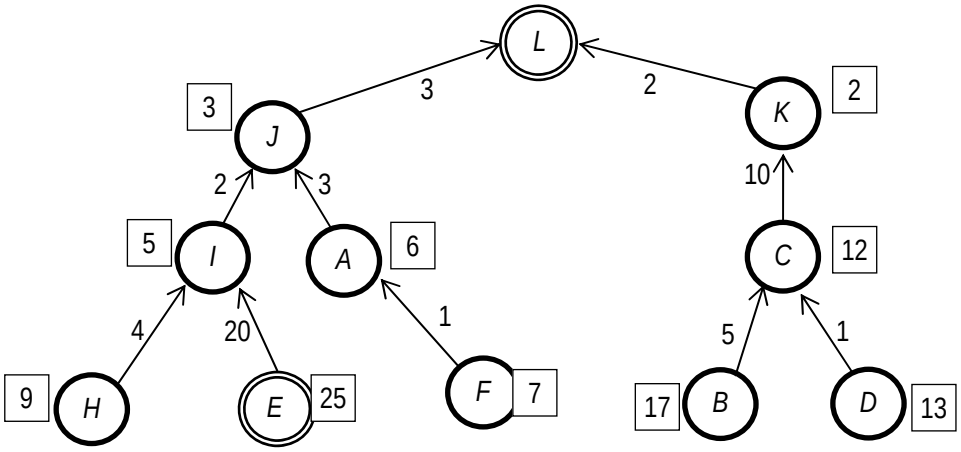
- Expandimos *I*:



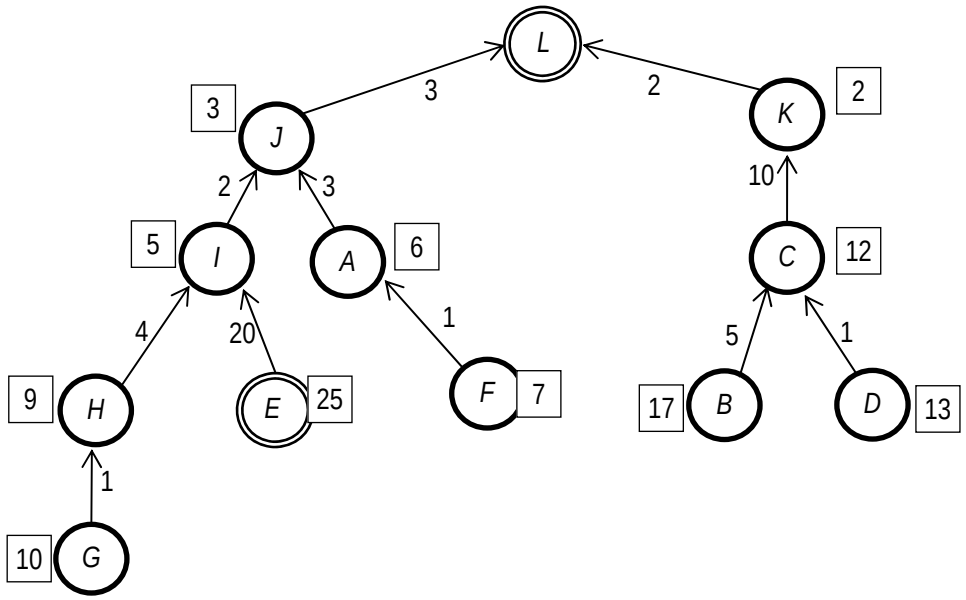
- Expandimos *A*:



- Expandimos *C*:



- Expandimos *H*:

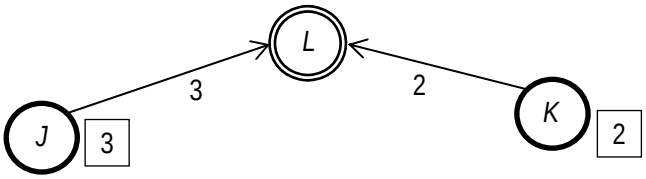


- A continuación elegiríamos *E* para su expansión y llegaríamos a la meta. El camino hallado hasta la meta tiene coste 25.

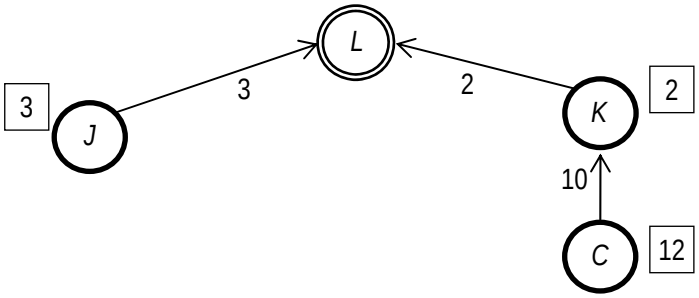
2. Búsqueda Primero en Profundidad (de derecha a izquierda)

Cuando sea necesario, en este algoritmo aplicaremos la función LimpiarTABLA_A (véase texto base, página 318). Tras la extracción de un nodo de ABIERTA y su posible expansión, dicha función elimina aquellos nodos que ya no son necesarios en el proceso de búsqueda de la meta. Esto permite preservar la linealidad de la complejidad espacial de este algoritmo con respecto a la profundidad de la solución.

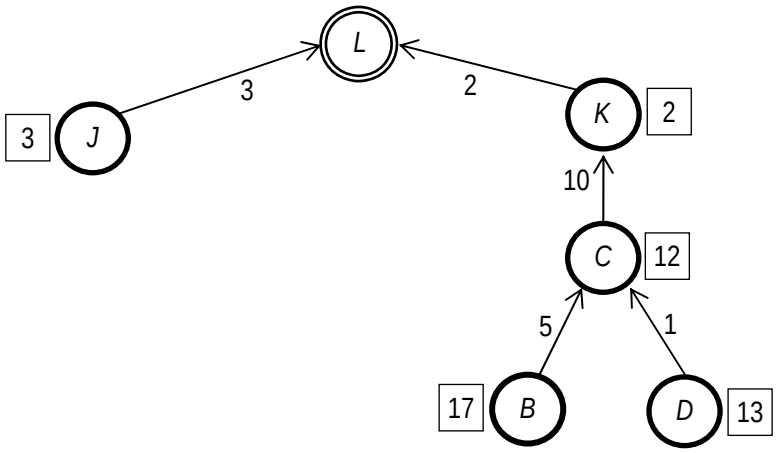
- Inicialmente, *L* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *L*. A continuación, expandimos el nodo inicial.



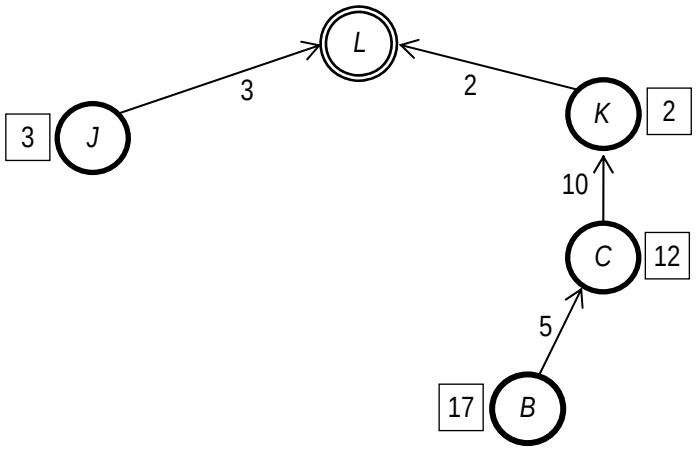
- Expandimos *K*:



- Expandimos *C*:

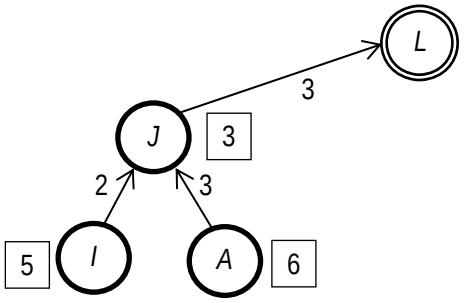


- Expandimos *D*. Seguidamente aplicamos LimpiarTABLA_A a *D*, es decir, sacamos *D* de TABLA_A por no tener hijos en ABIERTA:

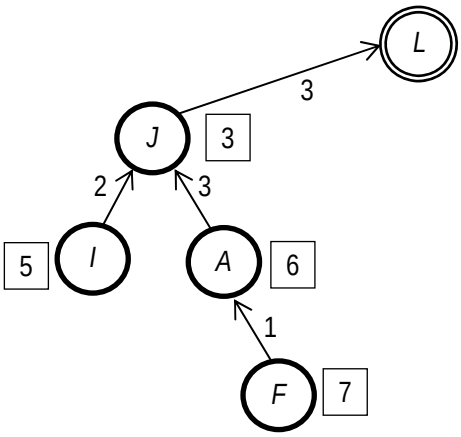


- Expandimos *B*. Seguidamente aplicamos LimpiarTABLA_A a *B*, es decir, sacamos *B* de TABLA_A por no tener hijos en ABIERTA. Por el mismo motivo sacamos de TABLA_A los nodos *C* y *K*.

- Expandimos *J*:

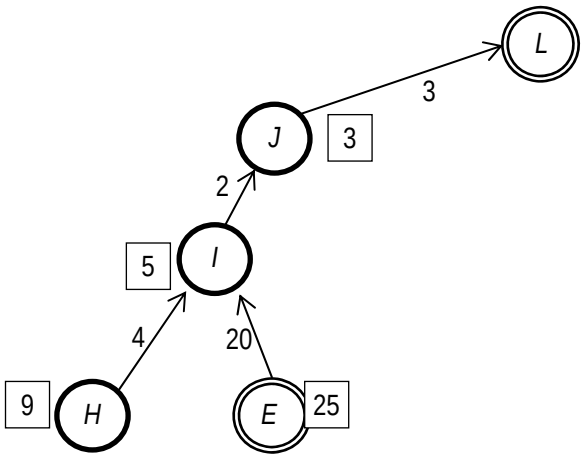


- Expandimos *A*:



- Expandimos *F*. Seguidamente aplicamos LimpiarTABLA_A a *F*, es decir, sacamos *F* de TABLA_A por no tener hijos en ABIERTA. Por el mismo motivo sacamos de TABLA_A el nodo *A*.

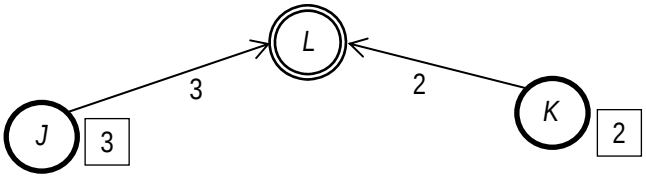
- Expandimos *I*:



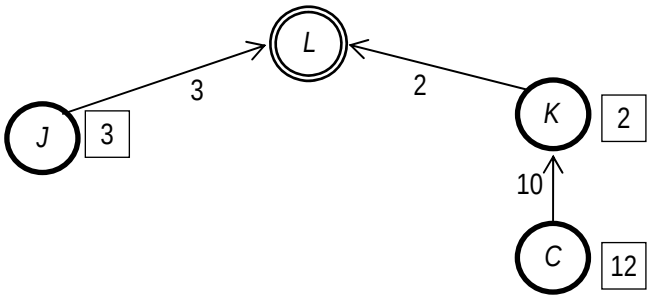
- A continuación elegiríamos *E* para su expansión y llegaríamos a la meta. El camino hallado hasta la meta tiene coste 25.

3. Búsqueda de Coste Uniforme

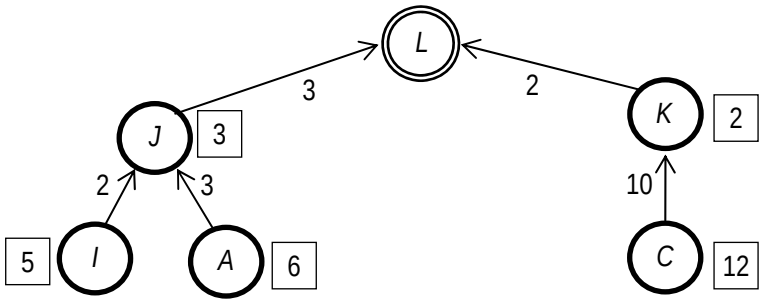
- Inicialmente, *L* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *L*. A continuación, expandimos el nodo inicial.



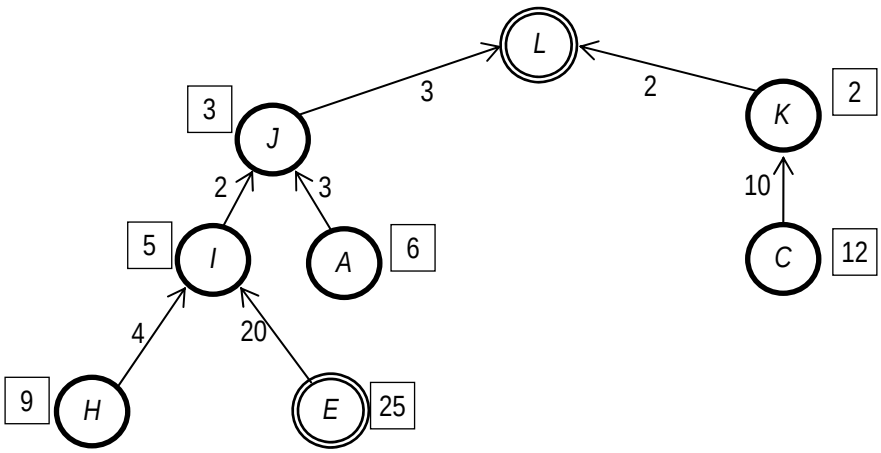
- Expandimos *K*:



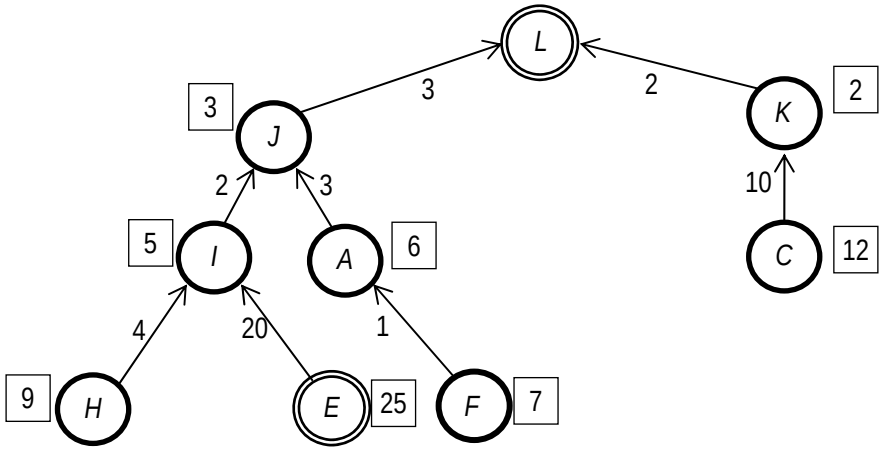
- Expandimos *J*:



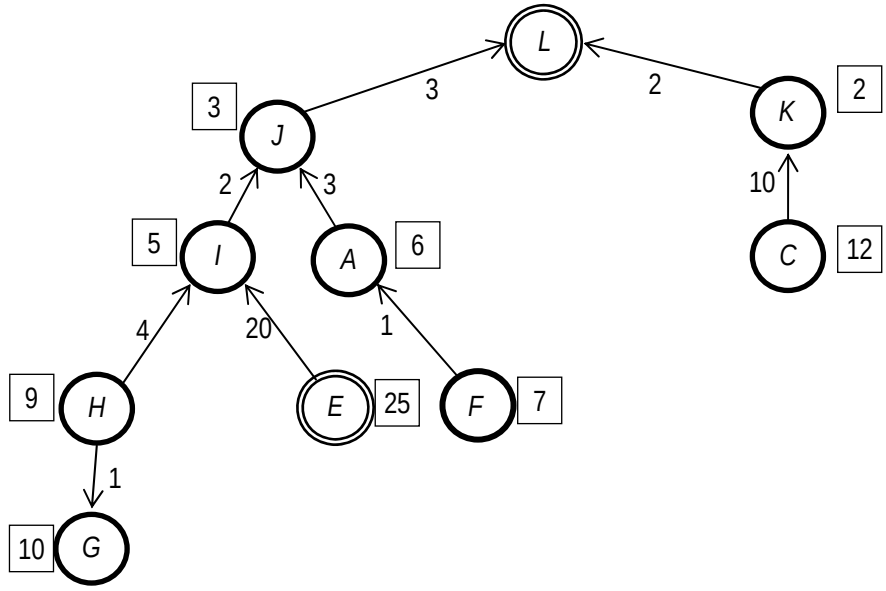
- Expandimos *I*:



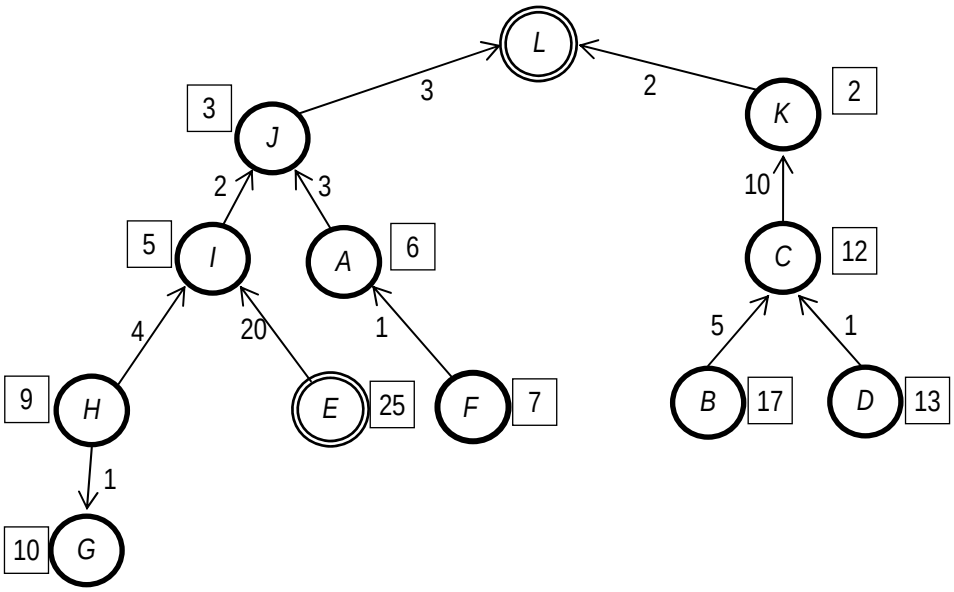
- Expandimos *A*:



- Expandimos *F*, con lo que TABLA_A no cambia. A continuación, expandimos *H*:



- Expandimos *G*, con lo que TABLA_A no cambia. A continuación, expandimos *C*:

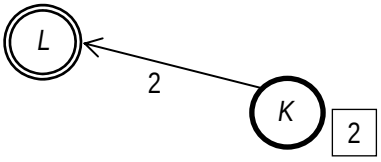


- Expandimos *D*, con lo que TABLA_A no cambia. Lo mismo ocurre tras expandir *B*. A continuación, intentamos expandir *E*, con lo que hemos alcanzado la meta. El coste del camino solución hallado es 25.

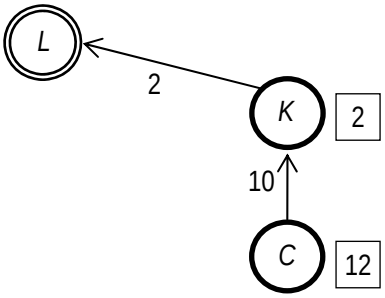
4. Búsqueda en Anchura Iterativa (de derecha a izquierda)

<u>Iteración 1 (se genera como máximo 1 nodo hijo en cada expansión de un nodo padre)</u>:

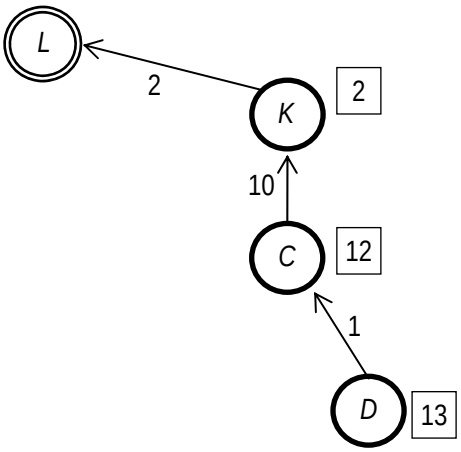
- Inicialmente, *L* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *L*. A continuación, expandimos el nodo inicial:



- Expandimos *K*:



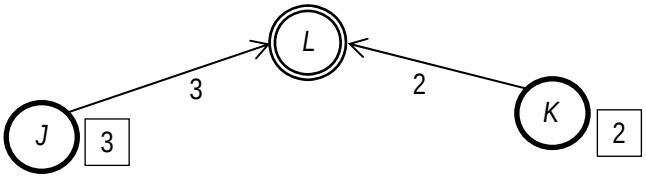
- Expandimos *C*:



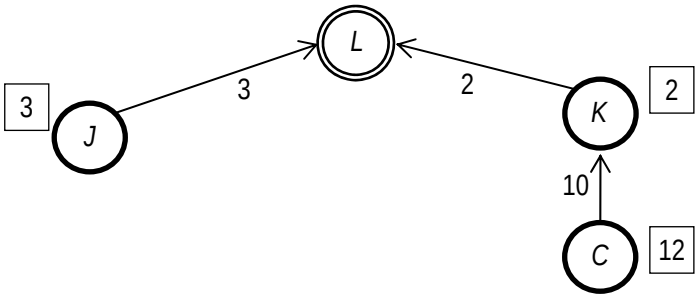
- Al expandir *D* finaliza la iteración actual.

<u>Iteración 2 (se generan como máximo 2 nodos hijos en cada expansión de un nodo padre)</u>:

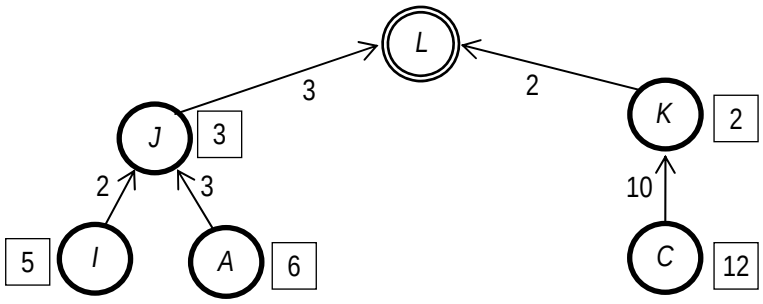
- Inicialmente, *L* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *L*. A continuación, expandimos el nodo inicial:



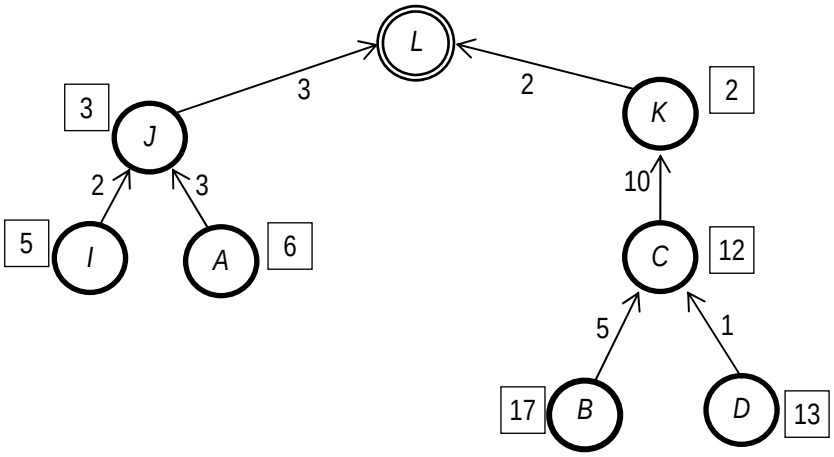
- Expandimos K:



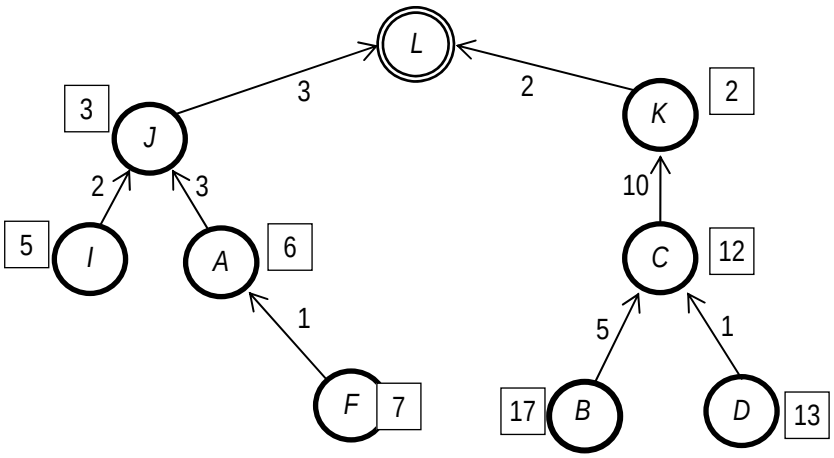
- Expandimos *J*:



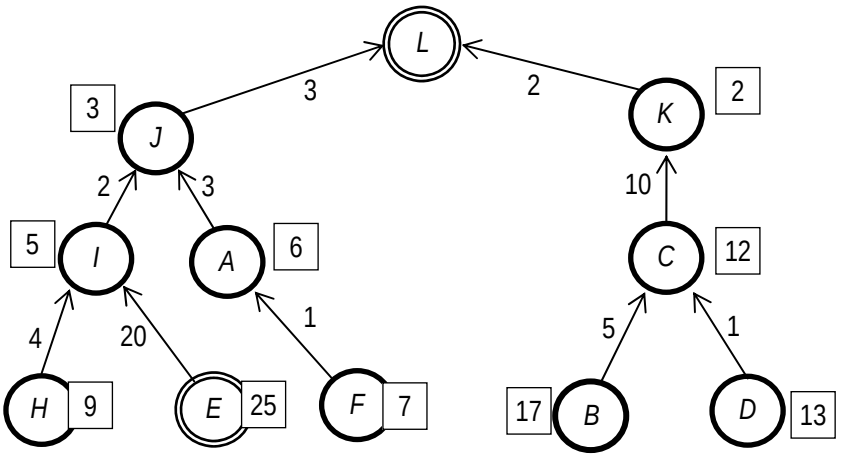
- Expandimos *C*:



- Expandimos *A*:



- Expandimos *I*:

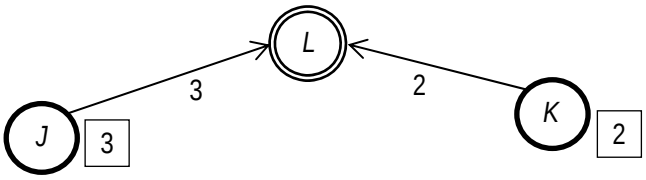


- Tras expandir *D*, *B* y *F*, TABLA_A no cambia. Finalmente, intentamos expandir *E* y alcanzamos la meta. El camino encontrado hasta el nodo inicial tiene coste 25.

5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

Iteración 1 (profundidad límite igual a 1):

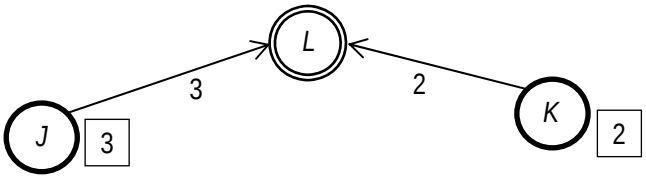
- Inicialmente, *L* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *L*. A continuación, expandimos el nodo inicial:



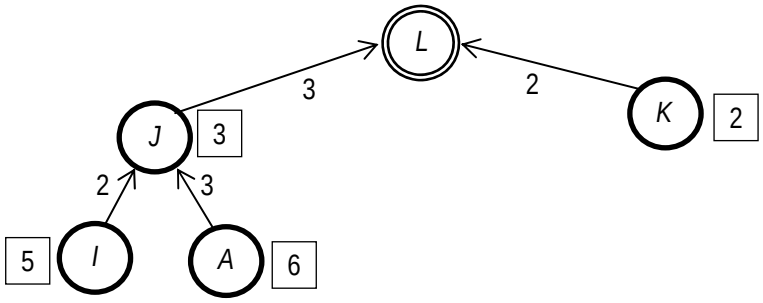
Tras comprobar que *J* no es nodo meta, se le aplica la función LimpiarTABLA_A por estar en la profundidad límite y es sacado de TABLA_A. De igual manera, tras comprobar que *K* no es nodo meta, se le aplica la función LimpiarTABLA_A por estar en la profundidad límite y es sacado de TABLA_A. Seguidamente, tras aplicarle la función LimpiarTABLA_A, *L* es sacado de TABLA_A por no tener hijos en ABIERTA. A continuación pasamos a la siguiente iteración.

Iteración 2 (profundidad límite igual a 2):

- Inicialmente, *L* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *L*. A continuación, expandimos el nodo inicial:

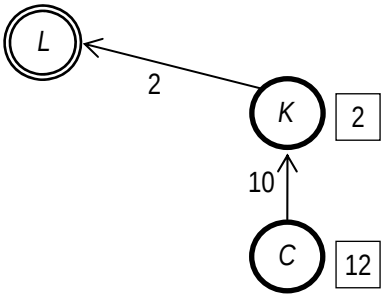


- Expandimos *J*:



Tras comprobar que *I* no es nodo meta, se le aplica la función LimpiarTABLA_A por encontrarse en la profundidad límite y es sacado de TABLA_A. Del mismo modo, tras comprobar que *A* no es nodo meta, se le aplica la función LimpiarTABLA_A por estar en la profundidad límite y es sacado de TABLA_A. A continuación, tras aplicarle la función LimpiarTABLA_A, *J* es sacado de TABLA_A por no tener hijos en ABIERTA.

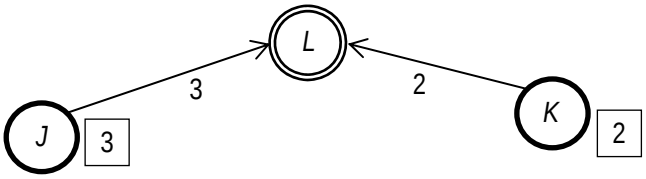
- Expandimos *K*:



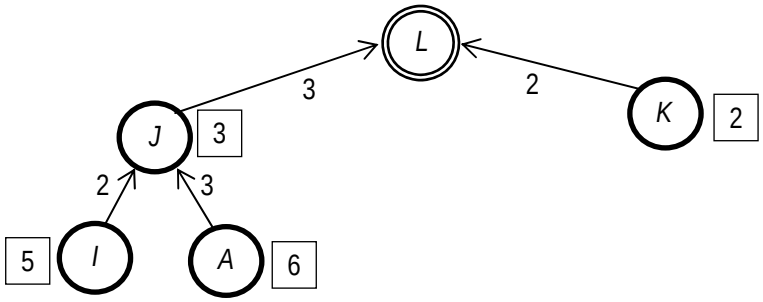
Tras comprobar que *C* no es nodo meta, se le aplica la función LimpiarTABLA_A por estar en la profundidad límite y es sacado de TABLA_A. Tras aplicarle la función LimpiarTABLA_A, *K* es sacado de TABLA_A por no tener hijos en ABIERTA. Lo mismo ocurre posteriormente con *L*, por lo que pasamos a la siguiente iteración.

Iteración 3 (profundidad límite igual a 3):

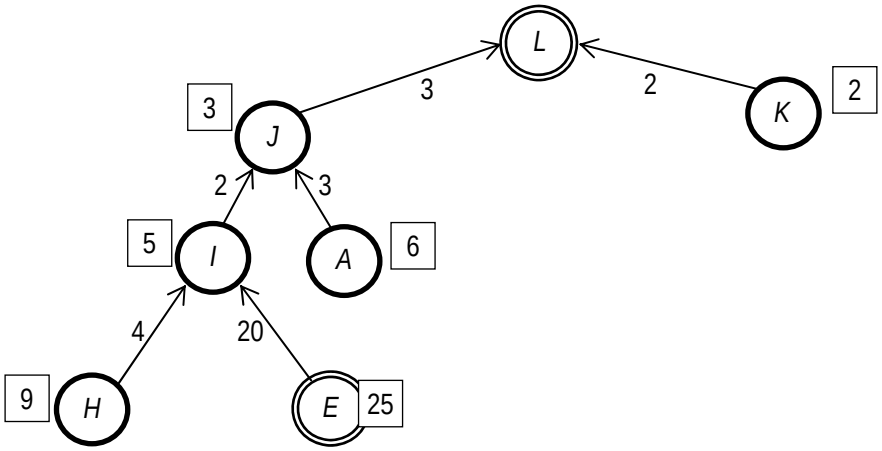
- Inicialmente, *L* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *L*. A continuación, expandimos el nodo inicial:



- Expandimos *J*:



- Expandimos *I*:



Tras comprobar que *H* no es nodo meta, se le aplica la función LimpiarTABLA_A por encontrarse en la profundidad límite y es sacado de TABLA_A. Seguidamente, tras comprobar que *E* es nodo meta, finaliza el algoritmo habiendo hallado un camino entre el nodo inicial y el nodo meta de coste 25.

EJERCICIO 5:

Considere el grafo de la figura 5.1, donde el nodo inicial es n_1 y donde el nodo meta es n_{10}. Cada arco u operador lleva asociado su coste y en cada nodo aparece la estimación de la menor distancia desde ese nodo a una meta. Aplique paso a paso el algoritmo A* al grafo dado, indicando de forma razonada la siguiente información en cada paso del algoritmo:

1. Qué nodo es expandido.
2. Cuál es el contenido de ABIERTA tras la expansión del nodo, indicando el valor de la función de evaluación heurística para cada nodo de ABIERTA.
3. Cuál es el contenido de TABLA_A tras la expansión del nodo. Para cada nodo de TABLA_A incluya la siguiente información:
    a) Su nodo padre que indique el camino de menor coste hasta el nodo inicial encontrado hasta el momento
    b) El coste del camino de menor coste hasta el nodo inicial encontrado hasta el momento
    c) Sus nodos hijos (si el nodo de TABLA_A actual ya ha sido expandido)

Por último, ¿cuál es el camino solución hallado y su coste?

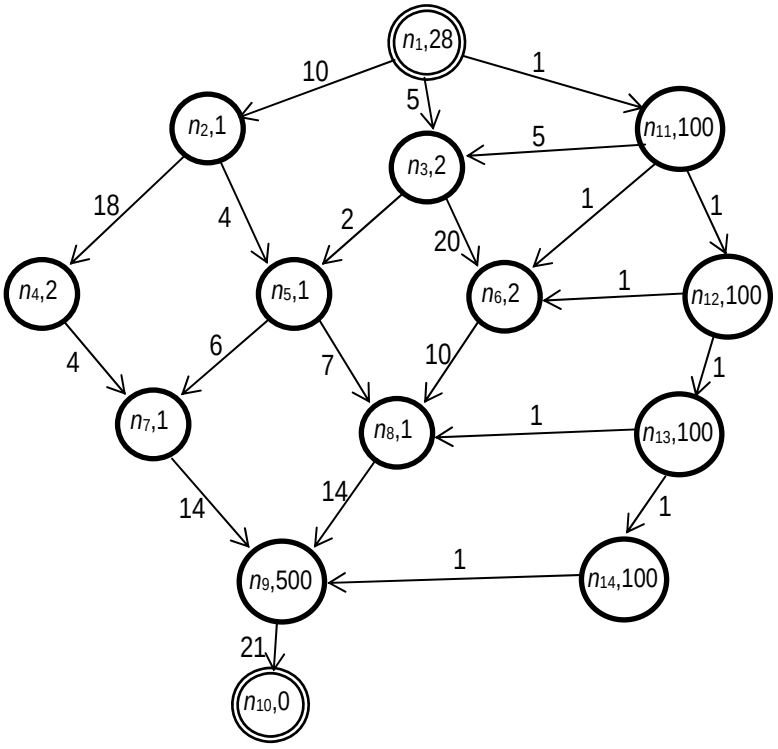


Figura 5.1: Grafo de búsqueda en el que el nodo inicial es n_1, el nodo meta es n_{10}, el coste de cada operador aparece al lado del arco que lo representa y al lado de cada nodo aparece la estimación de la menor distancia desde ese nodo a una meta.

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 5:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- El orden seguido en la expansión de los nodos se puntúa sobre 1.5 puntos.
- La forma en que se gestiona ABIERTA se puntúa sobre 3.75 puntos. Se hará especial énfasis en comprobar qué nodos hay en ABIERTA en cada paso del algoritmo y qué valores de la función de evaluación heurística se les asignan.

- La forma en que se gestiona TABLA_A se puntúa sobre 3.75 puntos. Se hará especial énfasis en comprobar qué nodos hay en TABLA_A en cada paso del algoritmo y qué padre mejor se les asigna (teniendo en cuenta las posibles reorientaciones o rectificaciones de enlaces)
- La correcta terminación del algoritmo se puntúa sobre 1 punto. Se hará especial énfasis en comprobar cuándo termina el algoritmo y qué camino solución devuelve.

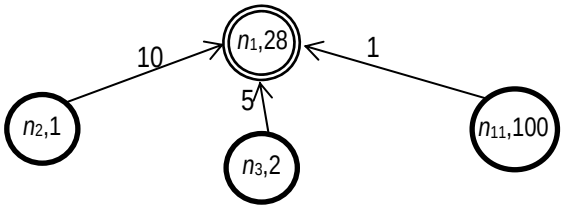
SOLUCIÓN DEL EJERCICIO 5 (por Severino Fernández Galán):

De cara a ilustrar la respuesta convenientemente, incluimos la información pedida sobre TABLA_A gráficamente. Para ello es necesario trazar, para cada nodo generado en una expansión, un arco ascendente a su padre expandido; además, hay que anotar para cada nodo su mejor padre encontrado hasta el momento (punto 3a del enunciado). De esta manera, siguiendo cada arco al mejor padre, se puede saber cuál es el mejor camino encontrado hasta el momento desde cada nodo al nodo inicial (punto 3b del enunciado). Además, los arcos ascendentes que llegan a un nodo ya expandido lo enlazan a sus nodos hijos (punto 3c del enunciado).

- PASO 0. El nodo inicial n_1 es introducido en ABIERTA y en TABLA_A, con lo que tenemos la siguiente situación:

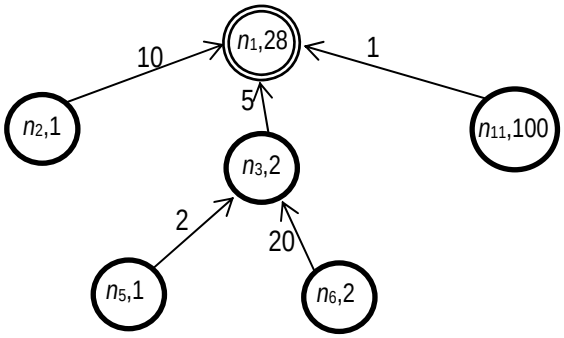
ABIERTA = {$\underline{n_1(0+28)}$}

- PASO 1. Expandimos el nodo n_1 de ABIERTA. Tras la expansión, la situación es la siguiente:



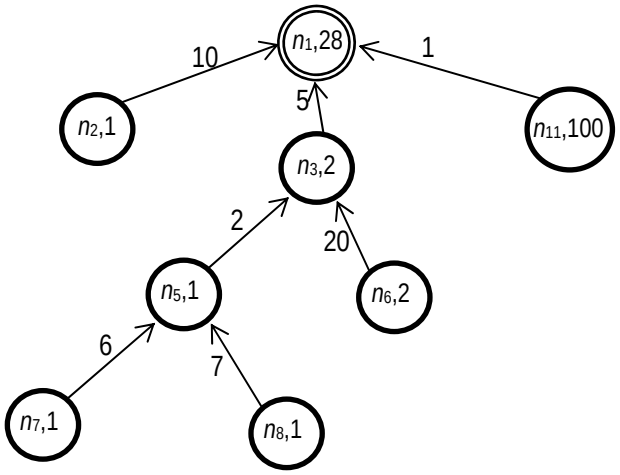
ABIERTA = {$\underline{n_3(5+2)}$, $n_2(10+1)$, $n_{11}(1+100)$}

- PASO 2. Expandimos n_3 por ser el nodo de ABIERTA con menor valor de la función de evaluación heurística, $f=g+h$ (al ser $g=5$ y $h=2$).



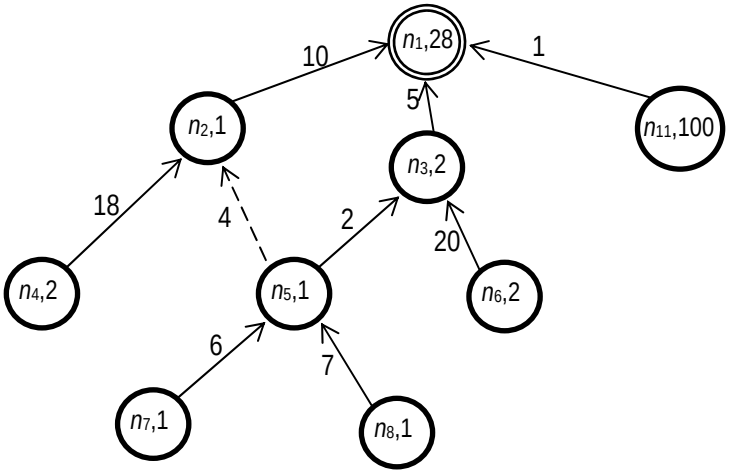
ABIERTA = {$\underline{n_5(7+1)}$, $n_2(10+1)$, $n_6(25+2)$, $n_{11}(1+100)$}

- PASO 3. Expandimos n_5.



ABIERTA = {<u>n_2(10+1)</u>, n_7(13+1), n_8(14+1), n_6(25+2), n_{11}(1+100)}

- PASO 4. Expandimos n_2.

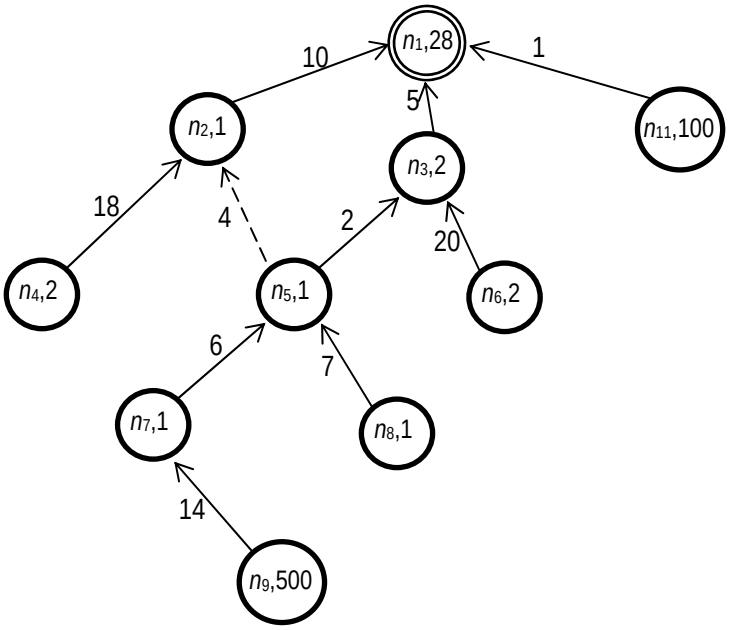


ABIERTA = {<u>n_7(13+1)</u>, n_8(14+1), n_6(25+2), n_4(28+2), n_{11}(1+100)}

Observe que el mejor camino desde n_5 al nodo inicial lo marca su padre n_3 (coste 7) y no su padre n_2 (coste 4+10=14). Por ello, el arco ascendente de n_5 a n_3 se marca con trazo continuo y el arco ascendente de n_5 a n_2 se marca con trazo discontinuo.

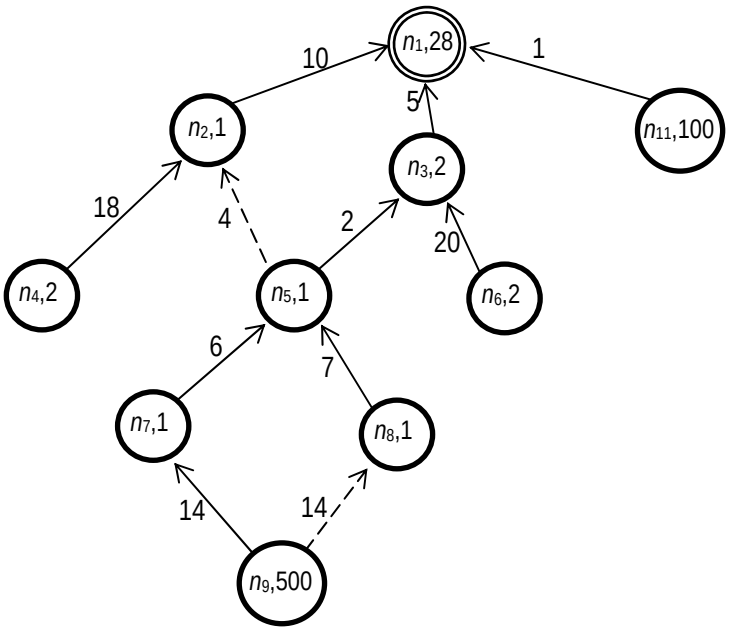
Es importante darse cuenta que el conjunto de arcos con trazo continuo formará siempre un árbol en el grafo parcial de búsqueda desarrollado hasta el momento.

- PASO 5. Expandimos n_7.



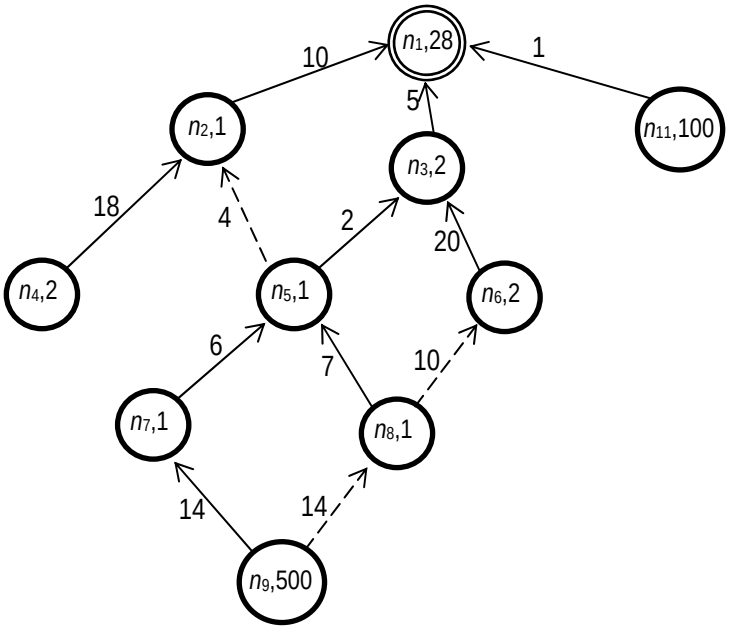
ABIERTA = {<u>n_8(14+1)</u>, n_6(25+2), n_4(28+2), n_{11}(1+100), n_9(27+500)}

- PASO 6. Expandimos n_8.



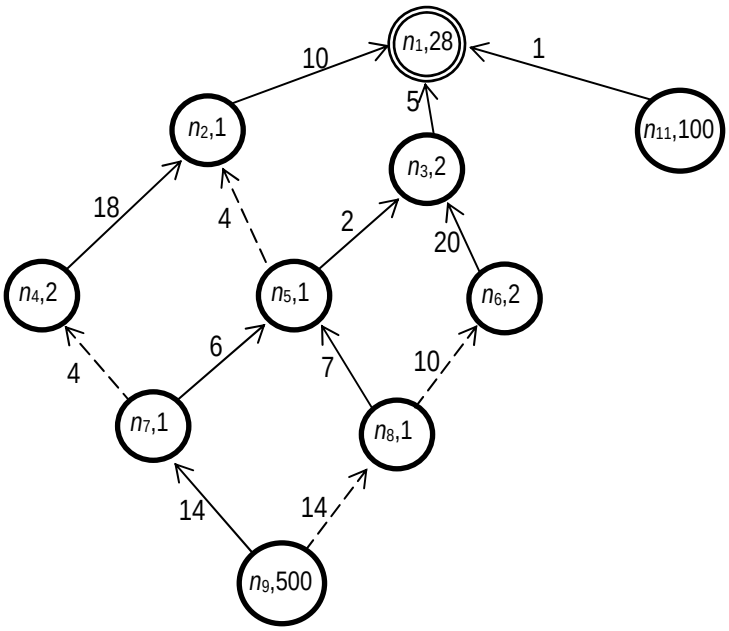
ABIERTA = {<u>n_6(25+2)</u>, n_4(28+2), n_{11}(1+100), n_9(27+500)}

- PASO 7. Expandimos n_6.



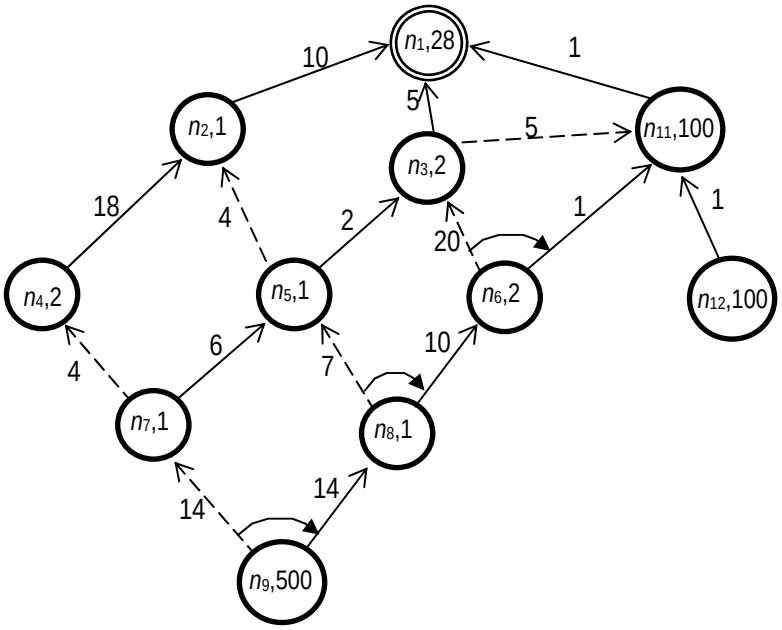
ABIERTA = {$\underline{n_4(28+2)}$, $n_{11}(1+100)$, $n_9(27+500)$}

- PASO 8. Expandimos n_4.



ABIERTA = {$\underline{n_{11}(1+100)}$, $n_9(27+500)$}

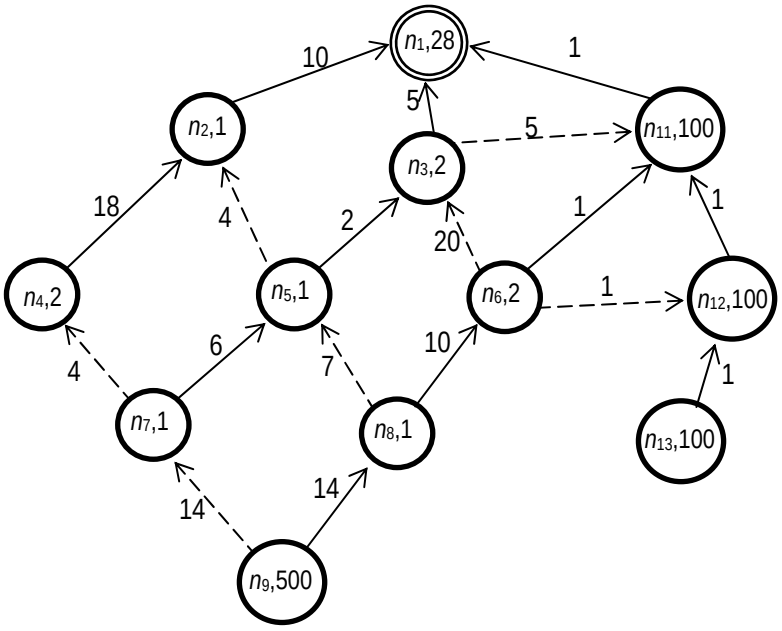
- PASO 9. Expandimos n_{11}.



ABIERTA = {$\underline{n_{12}(2+100)}$, $n_9(26+500)$}

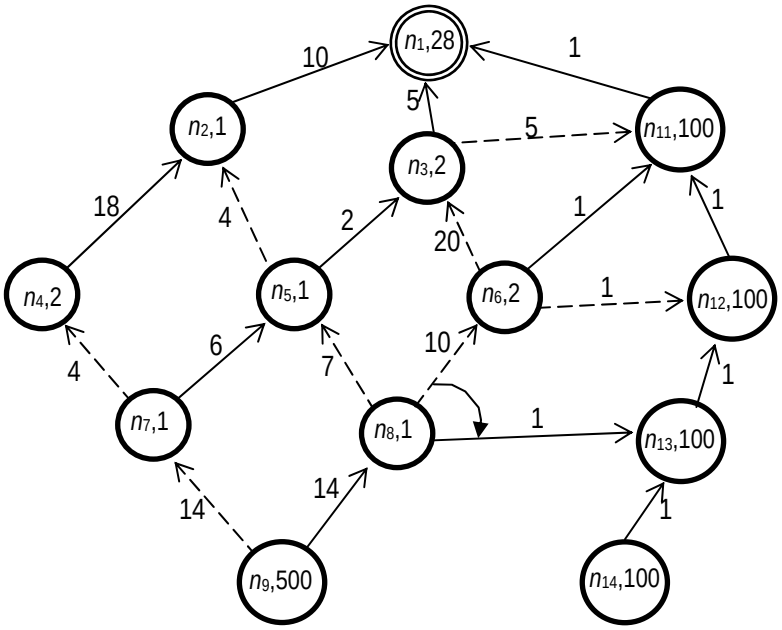
Observe que primeramente ha habido una reorientación del mejor padre de n_6, que antes era n_3 y ahora pasa a ser n_{11}. El nuevo mejor coste de n_6 al nodo inicial, $g(n_6)$, es ahora 1+1=2, que se puede calcular a partir de los costes de los mejores arcos ascendentes hallados hasta el momento: $n_6 \rightarrow n_{11}$ y $n_{11} \rightarrow n_1$. Como consecuencia de haber cambiado $g(n_6)$, se produce una reorientación del mejor padre de n_8. A su vez, como consecuencia de haber cambiado $g(n_8)$, se produce una reorientación del mejor padre de n_9.

- PASO 10. Expandimos n_{12}.



ABIERTA = {$\underline{n_{13}(3+100)}$, $n_9(26+500)$}

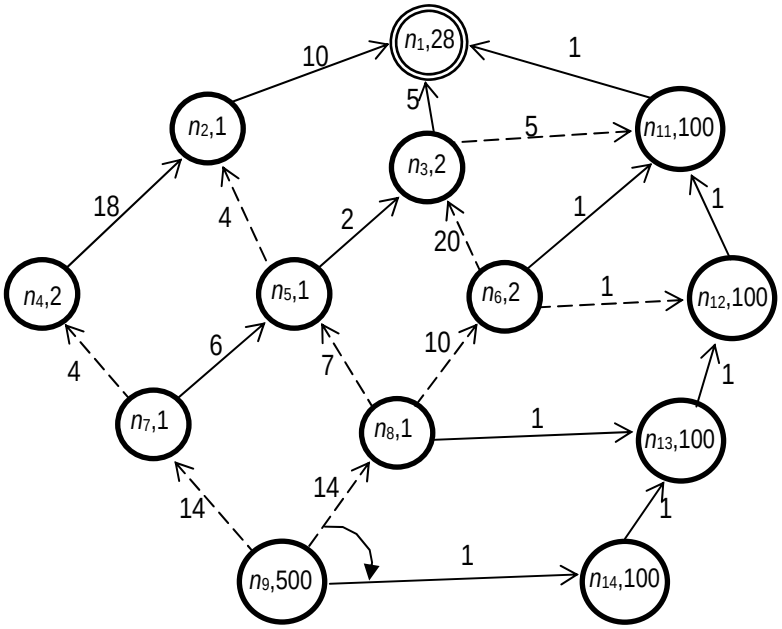
- PASO 11. Expandimos n_{13}.



ABIERTA = {$\underline{n_{14}(4+100)}$, $n_9(18+500)$}

Obsérvese que hay una reorientación del mejor padre de n_8.

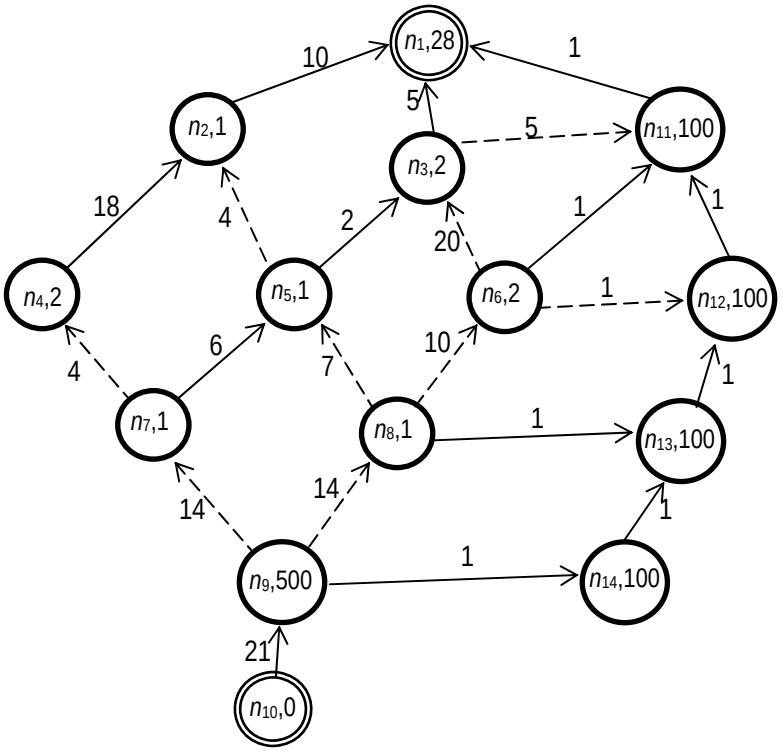
- PASO 12. Expandimos n_{14}.



ABIERTA = {$\underline{n_9(5+500)}$}

Obsérvese que hay una reorientación del mejor padre de n_9.

- PASO 13. Expandimos n_9.



ABIERTA = {$\underline{n_{10}(26+0)}$}

- PASO 14. Intentamos expandir n_{10} y alcanzamos una meta, con lo que el algoritmo termina. El camino solución encontrado es: $n_1 \rightarrow n_{11} \rightarrow n_{12} \rightarrow n_{13} \rightarrow n_{14} \rightarrow n_9 \rightarrow n_{10}$, cuyo coste es 26.

EJERCICIO 6:

Considere el grafo de la figura 6.1, donde el nodo inicial es *D* y donde los nodos meta son desconocidos. Cada arco u operador lleva asociado su coste y en cada nodo aparece su valor de la función de evaluación heurística (que hay que minimizar). Aplique paso a paso el algoritmo de escalada o máximo gradiente al grafo dado. Para ello indique de forma razonada qué nodo se expande en cada paso y cuál es el nodo final devuelto por el algoritmo. Utilice como *criterio de selección* el de mejor vecino. Utilice como *criterio de terminación* el que no se hayan producido mejoras durante los cinco últimos pasos del algoritmo.

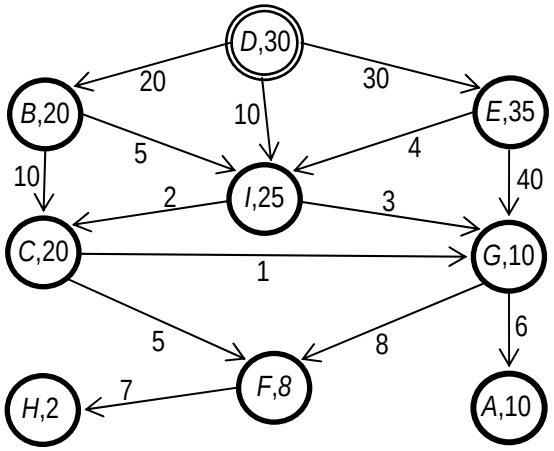


Figura 6.1: Grafo de búsqueda en el que el nodo inicial es *D*, los nodos meta son desconocidos, el coste de cada operador aparece al lado del arco que lo representa y al lado de cada nodo aparece el valor de su función de evaluación heurística (que hay que minimizar).

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 6:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- La correcta aplicación en cada paso del algoritmo del *criterio de selección* del vecino o hijo del nodo actual (qué vecino o hijo del nodo actual es considerado como candidato para sustituirlo) se puntúa sobre 4.5 puntos.
- La correcta aplicación en cada paso del algoritmo del *criterio de aceptación* del vecino o hijo seleccionado (si el vecino o hijo candidato sustituye o no finalmente al nodo actual) se puntúa sobre 4.5 puntos.
- La correcta aplicación del *criterio de finalización* del algoritmo se puntúa sobre 1 punto.

SOLUCIÓN DEL EJERCICIO 6 (por Severino Fernández Galán):

- PASO 1: Al principio expandimos el nodo inicial *D*, generando sus nodos hijos {*B*(20), *I*(25), *E*(35)}. Seleccionamos el nodo *B* por ser el mejor de los nodos hijos. A continuación aceptamos el nodo *B* como nuevo nodo actual en sustitución de *D*, debido a que el valor de la función de evaluación heurística de *B* es mejor o igual que el de *D* ($20 \leq 30$).

- PASO 2: Expandimos el nodo actual *B* y generamos sus hijos: {*C*(20), *I*(25)}. Seleccionamos el nodo *C* por ser el mejor de los nodos hijos. Seguidamente aceptamos el nodo *C* como nuevo nodo actual en sustitución de *B*, debido a que el valor de la función de evaluación heurística de *C* es mejor o igual que el de *B* ($20 \leq 20$).

- PASO 3: Expandimos el nodo actual *C* y generamos sus hijos: {*F*(8), *G*(10)}. Seleccionamos el nodo *F* por ser el mejor de los nodos hijos. A continuación aceptamos el nodo *F* como nuevo nodo actual en sustitución de *C*, debido a que el valor de la función de evaluación heurística de *F* es mejor o igual que el de *C* ($8 \leq 20$).

- PASO 4: Expandimos el nodo actual *F* y generamos sus hijos: {*H*(2)}. Aceptamos el nodo *H* como nuevo nodo actual en sustitución de *F*, debido a que el valor de la función de evaluación heurística de *H* es mejor o igual que el de *F* ($2 \leq 8$).

Dado que *H* no tiene sucesores, la búsqueda terminaría. El algoritmo devolvería el nodo *H*(2) como mejor nodo encontrado.

AÑO 2014

EJERCICIO 1:

Dibuje mediante un grafo dirigido o describa detalladamente mediante una tabla el espacio de estados (o espacio de búsqueda) completo para el *mundo de la aspiradora con tres localizaciones*. Para ello especifique: el conjunto de todos los estados posibles, el estado inicial, el o los estados meta, los operadores aplicables a cada estado y el coste asociado a cada operador. En el mundo de la aspiradora con tres localizaciones, existe una aspiradora que puede estar en una localización *A*, en una localización *B* o en una localización *C*. La localización *B* está situada a la derecha de la localización *A*, mientras que la localización *C* lo está a la derecha de la localización *B*. La aspiradora sólo puede percibir si hay o no suciedad en su localización actual; por otra parte, puede elegir si se mueve hacia la izquierda, si se mueve hacia la derecha o si aspira la suciedad presente en su localización actual. ¿Cuál es la solución menos costosa para este mundo si el estado inicial consta de las tres localizaciones sucias y la aspiradora está situada en la localización *A*? Un estado meta sería aquél en el que ninguna localización está sucia.

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 1:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- Los estados se especifican correctamente: 2.5 puntos
- Los operadores se especifican correctamente: 2.5 puntos
- Los costes de los operadores se especifican correctamente: 1 punto
- El espacio de búsqueda se dibuja (mediante un grafo dirigido) o se describe (mediante una tabla) correctamente: 3 puntos
- La solución de menor coste se especifica correctamente: 1 punto

SOLUCIÓN DEL EJERCICIO 1 (por Severino Fernández Galán):

Para representar los **estados** posibles utilizaremos la siguiente notación: el estado del problema lo representamos como (L_A,L_B,L_C), donde L_i se corresponde con la localización $i \in \{A,B,C\}$. Además, L_i admite los valores "AS", que se corresponde con la aspiradora, y "X", que se corresponde con una localización vacía. Por otra parte, cada L_i puede estar subrayada o no: una L_i subrayada indica que hay suciedad en la localización *i*, mientras que una L_i no subrayada indica que no hay suciedad en dicha localización. Por ejemplo, (X,AS,<u>X</u>) representa el estado en el que la localización *A* está vacía y sin suciedad, la localización *B* contiene la aspiradora y en dicha localización no hay suciedad y, por último, la localización *C* está vacía y con suciedad. Nótese que hay $3 \cdot 2^3 = 24$ posibles estados en este mundo.

Supongamos que el estado inicial consta de las tres localizaciones sucias y la aspiradora está situada en la localización *A*. Por tanto, el **estado inicial** es (<u>AS</u>,<u>X</u>,<u>X</u>) y existen tres **estados meta**: (AS,X,X), (X,AS,X) y (X,X,AS).

Dado un estado cualquiera, en principio hay tres **operadores** aplicables al mismo dependiendo de la acción que tome la aspiradora: moverse a la izquierda (I), moverse a la derecha (D) o aspirar la suciedad presente (P). Se puede considerar que cada operador tiene **coste** unidad, así que el coste de un camino es el número de arcos que lo componen.

En la tabla 1.1 figuran los estados posibles del sistema, los operadores que se pueden aplicar a cada estado posible y los estados resultantes de aplicar cada operador.

De forma alternativa, la información incluida en la tabla 1.1 se puede representar mediante un grafo dirigido tal como se muestra en la figura 1.1. En dicha figura, no existen estados no permitidos (que marcaríamos con líneas discontinuas) y se ha utilizado línea doble para marcar el estado inicial y los estados meta. Por otra parte, obsérvese que no existen estados aislados del espacio de búsqueda, que no sean alcanzables desde el estado inicial.

La **solución menos costosa** para el problema planteado en el enunciado consiste en: aspirar la suciedad en la localización *A*, mover la aspiradora a la derecha a la localización *B*, aspirar la suciedad en la localización *B*, mover la aspiradora a la derecha a la localización *C* y, finalmente, aspirar la suciedad en la localización *C*. La solución menos costosa correspondiente aparece marcada en negrita en la figura 1.1.

ESTADOS	OPERADORES		
	I	D	P
(AS,X,X)	(AS,X,X)	(X,AS,X)	(AS,X,X)
(X,AS,X)	(AS,X,X)	(X,X,AS)	(X,AS,X)
(X,X,AS)	(X,AS,X)	(X,X,AS)	(X,X,AS)
(AS,X,X)	(AS,X,X)	(X,AS,X)	No aplicable
(X,AS,X)	(AS,X,X)	(X,X,AS)	(X,AS,X)
(X,X,AS)	(X,AS,X)	(X,X,AS)	(X,X,AS)
(AS,X,X)	(AS,X,X)	(X,AS,X)	(AS,X,X)
(X,AS,X)	(AS,X,X)	(X,X,AS)	No aplicable
(X,X,AS)	(X,AS,X)	(X,X,AS)	(X,X,AS)
(AS,X,X)	(AS,X,X)	(X,AS,X)	(AS,X,X)
(X,AS,X)	(AS,X,X)	(X,X,AS)	(X,AS,X)
(X,X,AS)	(X,AS,X)	(X,X,AS)	No aplicable
(AS,X,X)	(AS,X,X)	(X,AS,X)	No aplicable
(X,AS,X)	(AS,X,X)	(X,X,AS)	No aplicable
(X,X,AS)	(X,AS,X)	(X,X,AS)	(X,X,AS)
(AS,X,X)	(AS,X,X)	(X,AS,X)	No aplicable
(X,AS,X)	(AS,X,X)	(X,X,AS)	(X,AS,X)
(X,X,AS)	(X,AS,X)	(X,X,AS)	No aplicable
(AS,X,X)	(AS,X,X)	(X,AS,X)	(AS,X,X)
(X,AS,X)	(AS,X,X)	(X,X,AS)	No aplicable
(X,X,AS)	(X,AS,X)	(X,X,AS)	No aplicable
(AS,X,X)	Estado META		
(X,AS,X)	Estado META		
(X,X,AS)	Estado META		

Tabla 1.1: Estados posibles, operadores aplicables a cada estado y estados resultantes de aplicar cada operador para el mundo de la aspiradora con tres localizaciones. El estado inicial y los estados meta se representan en negrita.

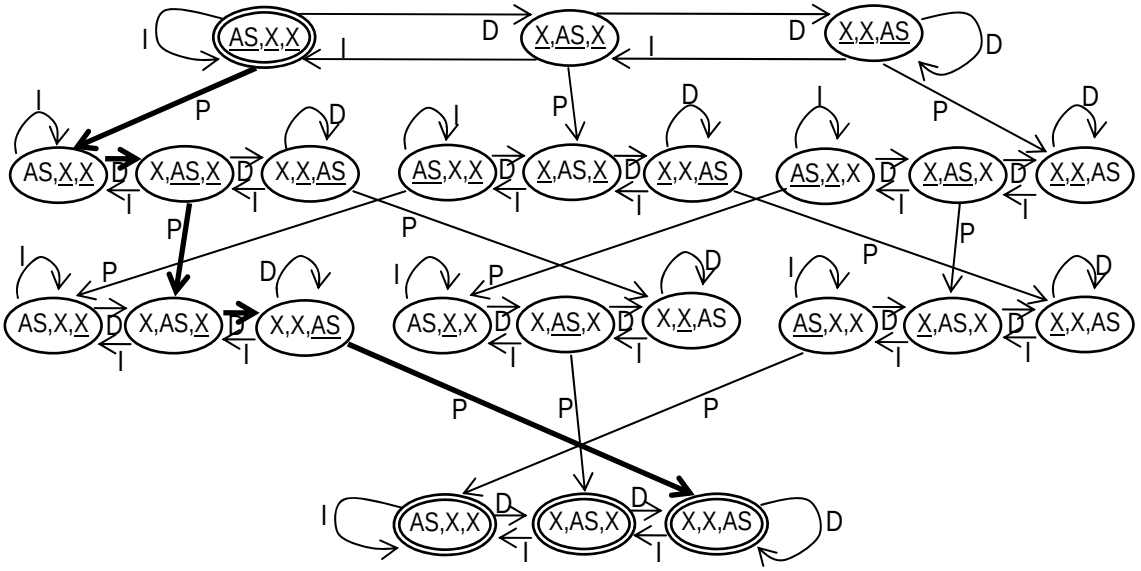


Figura 1.1: Grafo dirigido que representa el espacio de búsqueda para el mundo de la aspiradora con tres localizaciones. No existen estados no alcanzables desde el estado inicial, que queden aislados del espacio de búsqueda. El camino de menor coste se representa en negrita.

NOTA IMPORTANTE:

Un alumno sugirió en el foro de debate de la asignatura que si la aspiradora está situada en A (localización más a la izquierda), entonces no se podría aplicar el operador I de mover la aspiradora a la izquierda. (Lo mismo para la localización C y el operador D de mover la aspiradora a la derecha.) Esta otra opción comentada por el alumno también sería válida y conllevaría que, por ejemplo, en la figura 1.1 desaparecieran todos los arcos que unen un estado consigo mismo. Además, de forma análoga, en la tabla 1.1 los operadores I y D ya no serían aplicables en ciertos casos: por ejemplo, el operador I no sería aplicable al estado (AS,X,X).

EJERCICIO 2:

Considere el espacio de búsqueda de la figura 2.1, que tiene forma de árbol, donde el nodo raíz del árbol es el nodo inicial, existe un único nodo meta y cada operador tiene asociado un coste. Explique razonadamente en qué orden se expandirían los nodos de dicho árbol de búsqueda a partir de cada uno de los métodos siguientes de búsqueda sin información del dominio:

1. Búsqueda Primero en Anchura (de izquierda a derecha)
2. Búsqueda Primero en Profundidad (de derecha a izquierda)
3. Búsqueda de Coste Uniforme
4. Búsqueda en Anchura Iterativa (de derecha a izquierda)
5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

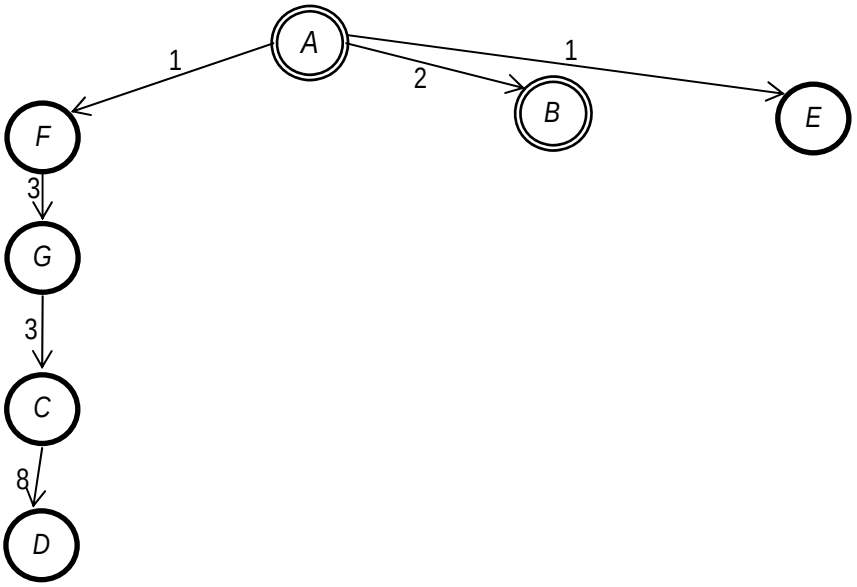


Figura 2.1: Árbol de búsqueda en el que el nodo inicial es *A*, el nodo meta es *B* y el coste de cada operador aparece al lado del arco que lo representa.

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 2:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- Cada uno de los cinco apartados, correspondiente a un método de búsqueda concreto, se puntúa sobre 2 puntos.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no expande los nodos en el orden debido: la puntuación del apartado bajaría 1.6 puntos si el orden dado como respuesta varía significativamente del correcto, mientras que si el orden dado como respuesta varía del correcto como consecuencia de un despiste no conceptual entonces la puntuación del apartado bajaría sólo 0.4 puntos en vez de los 1.6 puntos mencionados.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no finaliza cuando es debido, la puntuación del apartado bajaría 0.4 puntos. (Esto es independiente del orden de expansión de nodos dado como respuesta, que ya ha sido valorado anteriormente con un máximo de 1.6 puntos.)

SOLUCIÓN DEL EJERCICIO 2 (por Severino Fernández Galán):

1. Búsqueda Primero en Anchura (de izquierda a derecha)

Este algoritmo explora el árbol de búsqueda por niveles de profundidad, así que el orden de expansión de los nodos de izquierda a derecha sería el reflejado en la figura 2.2.

2. Búsqueda Primero en Profundidad (de derecha a izquierda)

Este algoritmo explora el árbol de búsqueda bajando de nivel siempre que sea posible. Si no es posible, se sube al nodo más cercano al nodo actual desde el que poder seguir bajando de nivel. El orden de expansión de los nodos de derecha a izquierda según este algoritmo se dibuja en la figura 2.3.

3. Búsqueda de Coste Uniforme

Este algoritmo explora el árbol de búsqueda expandiendo aquel nodo disponible cuyo coste al nodo inicial sea el menor. El orden de expansión de nodos según este criterio se indica en la figura 2.4. Obsérvese que después de la primera expansión (nodo *A*), hay un empate entre los nodos *F* y *E*, cuyos costes al nodo inicial son iguales a 1 en los dos casos. Nosotros hemos elegido el nodo *E* para realizar la segunda expansión, pero también se podría haber elegido el otro nodo, *F*. Los posibles empates posteriores también los desharíamos de esta forma arbitraria.

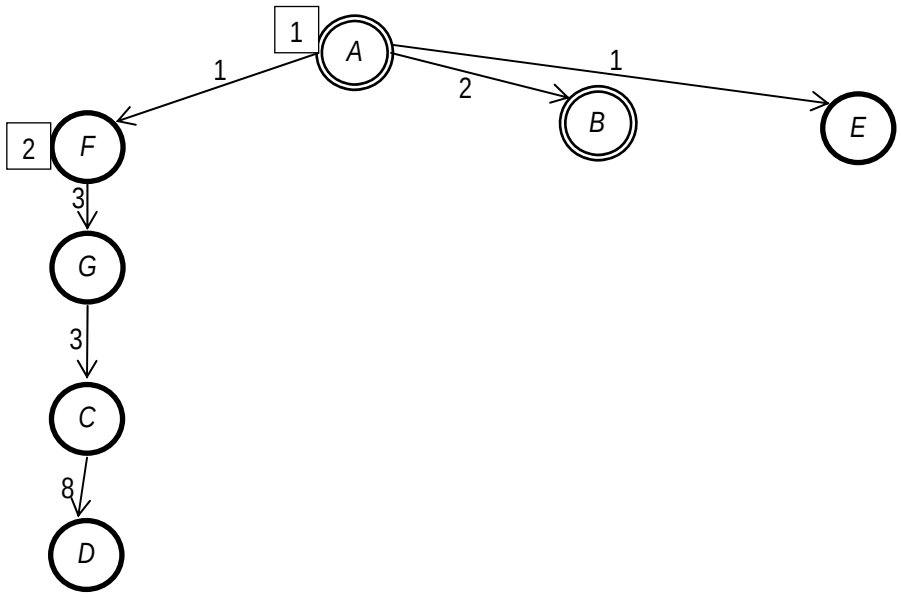


Figura 2.2: Orden de expansión de nodos para el árbol de la figura 2.1 según la búsqueda primero en anchura (de izquierda a derecha). Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto, el algoritmo termina en ese momento sin que se lleguen a generar sus hijos.

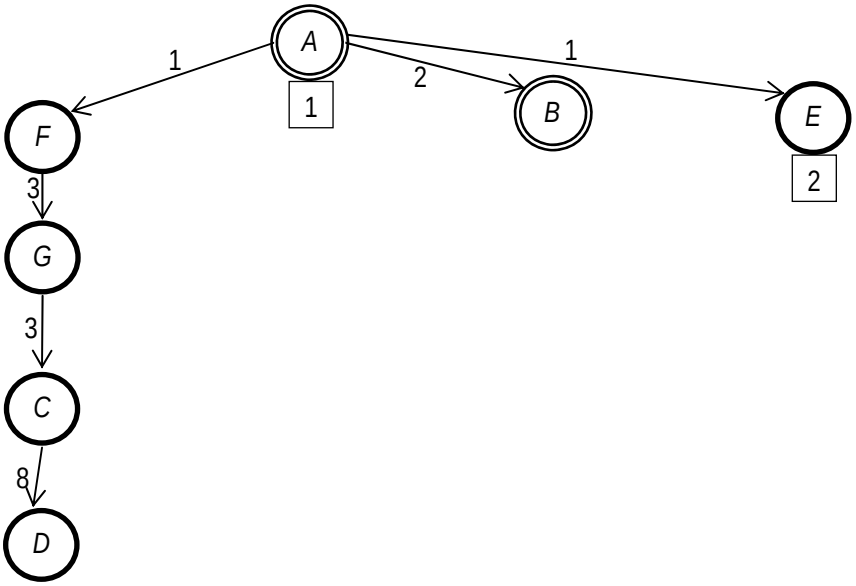


Figura 2.3: Orden de expansión de nodos para el árbol de la figura 2.1 según la búsqueda primero en profundidad (de derecha a izquierda). Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto, el algoritmo termina en ese momento sin que se lleguen a generar sus hijos.

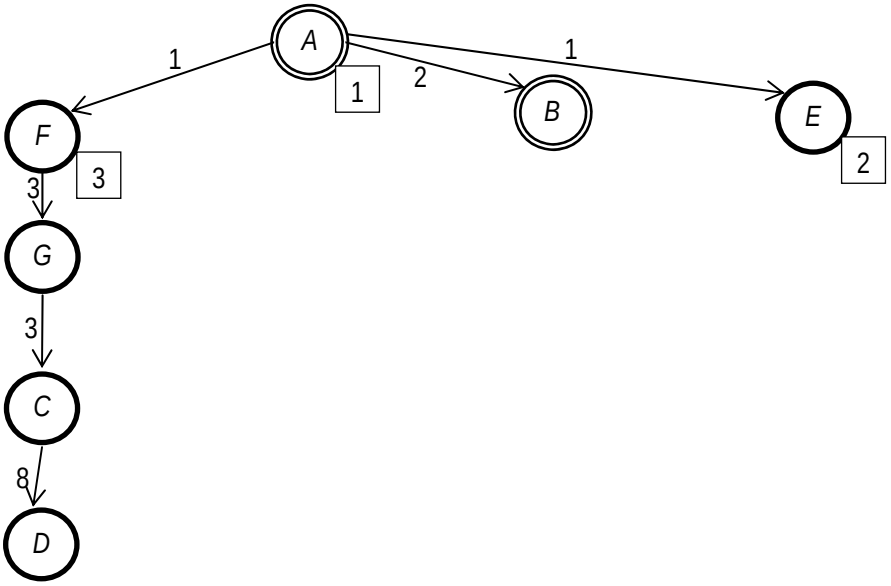


Figura 2.4: Orden de expansión de nodos para el árbol de la figura 2.1 según la búsqueda de coste uniforme. Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto, el algoritmo termina en ese momento sin que se lleguen a generar sus hijos.

4. Búsqueda en Anchura Iterativa (de derecha a izquierda)

Este algoritmo ejecuta iterativamente varias búsquedas primero en anchura, de manera que entre iteración e iteración se incrementa en una unidad el número máximo de hijos que se generan en cada expansión de un nodo padre. Al principio (en la primera iteración), únicamente un hijo es generado en cada expansión. En las figuras 2.5a y 2.5b se muestran los órdenes de expansión de nodos para cada una de las iteraciones de búsqueda primero en anchura que son necesarias para este ejemplo de búsqueda en anchura iterativa.

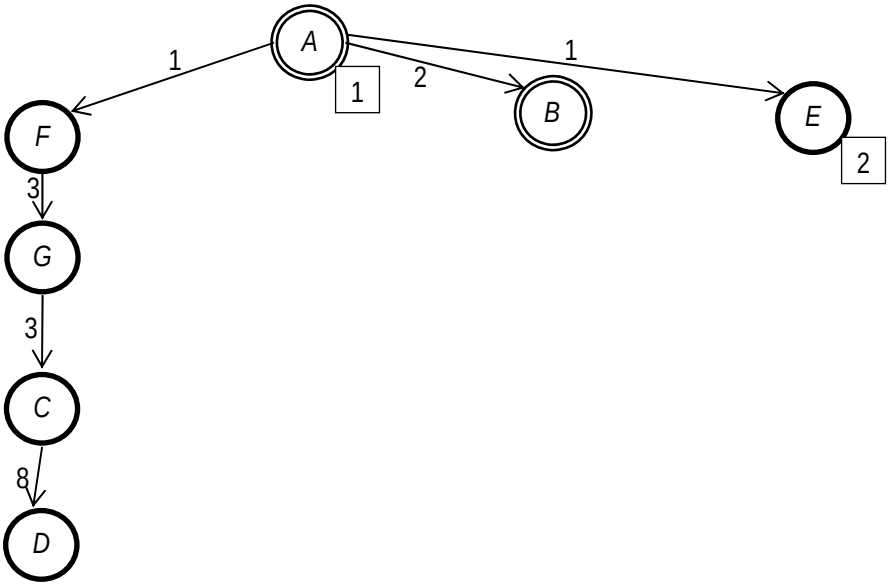


Figura 2.5a: Primera iteración de la búsqueda en anchura iterativa de derecha a izquierda, en la que como máximo un hijo es generado en cada expansión de un nodo padre. (Es decir, el nodo *B* no ha sido generado.)

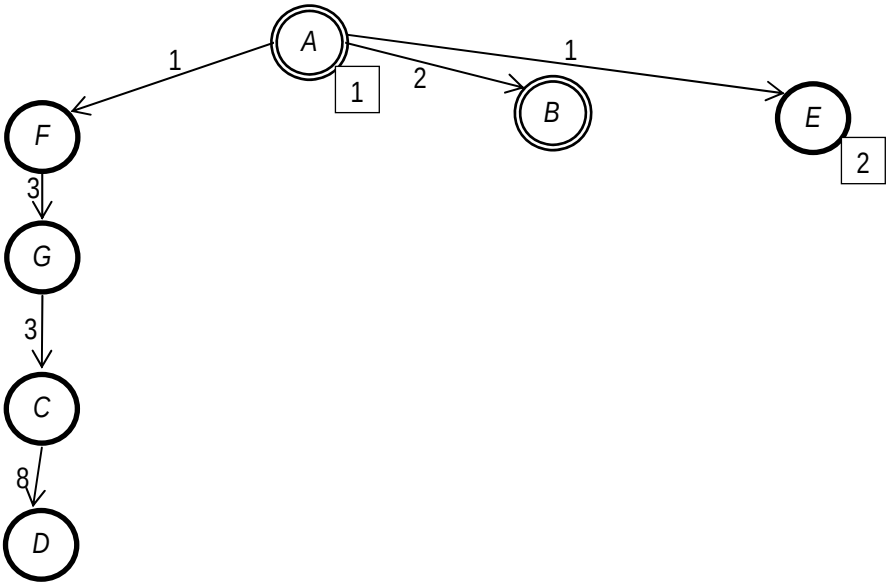


Figura 2.5b: Segunda iteración de la búsqueda en anchura iterativa de derecha a izquierda, en la que como máximo dos hijos son generados en cada expansión de un nodo padre. (Es decir, el nodo *B* sí ha sido generado.)

Observe que el nodo meta no es realmente expandido en esta segunda iteración (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto, el algoritmo termina en ese momento sin que se lleguen a generar sus hijos.

5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

Este algoritmo ejecuta iterativamente varias búsquedas primero en profundidad, de manera que entre iteración e iteración se incrementa en una unidad la profundidad límite. Al principio (en la primera iteración), la profundidad límite es igual a 1, así que sólo se podrá expandir el nodo inicial, cuya profundidad es igual a 0. En la figura 2.6a se muestran los órdenes de expansión de nodos para la única iteración de búsqueda primero en profundidad que es necesaria para este ejemplo de búsqueda en profundidad iterativa.

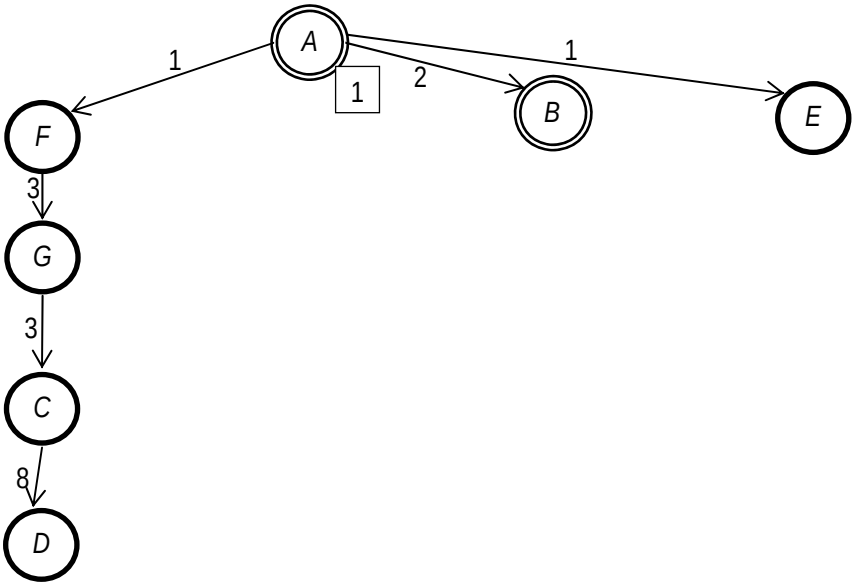


Figura 2.6a: Primera iteración de la búsqueda en profundidad iterativa de izquierda a derecha, en la que la profundidad límite es igual a 1 y únicamente son expandidos nodos con una profundidad máxima de 0.

Observe que realmente el nodo meta no es expandido en esta última iteración porque se encuentre a la profundidad límite. Esta condición no se llega a comprobar, ya que justo antes se averigua que el nodo es meta y, por tanto, el algoritmo termina.

EJERCICIO 3:

Considere el espacio de búsqueda de la figura 3.1, que tiene forma de árbol, donde el nodo raíz del árbol es el nodo inicial, existe un único nodo meta y cada operador tiene asociado un coste. Describa cuál es el contenido de ABIERTA, previamente a cada extracción de un nodo de la misma, a partir de cada uno de los métodos siguientes de búsqueda sin información del dominio:

1. Búsqueda Primero en Anchura (de izquierda a derecha)
2. Búsqueda Primero en Profundidad (de derecha a izquierda)
3. Búsqueda de Coste Uniforme
4. Búsqueda en Anchura Iterativa (de derecha a izquierda)
5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

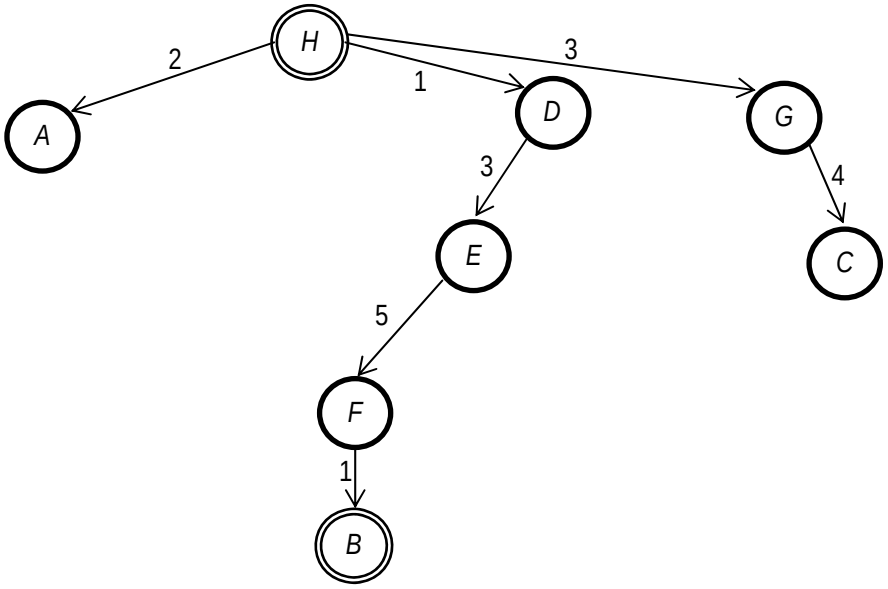


Figura 3.1: Árbol de búsqueda en el que el nodo inicial es *H*, el nodo meta es *B* y el coste de cada operador aparece al lado del arco que lo representa.

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 3:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- Cada uno de los cinco apartados, correspondiente a un método de búsqueda concreto, se puntúa sobre 2 puntos.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no gestiona ABIERTA en la forma debida: la puntuación del apartado bajaría 1.6 puntos si la respuesta dada varía significativamente de la correcta, mientras que si la respuesta dada varía de la correcta como consecuencia de un despiste no conceptual entonces la puntuación del apartado bajaría sólo 0.4 puntos en vez de los 1.6 puntos mencionados.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no finaliza cuando es debido, la puntuación del apartado bajaría 0.4 puntos. (Esto es independiente de la gestión de ABIERTA dada como respuesta, que ya ha sido valorada anteriormente con un máximo de 1.6 puntos.)

SOLUCIÓN DEL EJERCICIO 3 (por Severino Fernández Galán):

1. Búsqueda Primero en Anchura (de izquierda a derecha)

Este algoritmo usa ABIERTA como una cola, de manera que siempre se saca el primer nodo de la cola y se introducen sus hijos al final de la misma. El contenido de ABIERTA previamente a cada extracción de un nodo es el siguiente:

Antes de sacar *H*: {*H*}
Antes de sacar *A*: {*A, D, G*}
Antes de sacar *D*: {*D, G*}
Antes de sacar *G*: {*G, E*}
Antes de sacar *E*: {*E, C*}
Antes de sacar *C*: {*C, F*}
Antes de sacar *F*: {*F*}
Antes de sacar *B*: {*B*}
META ENCONTRADA

Observe que, tras cada expansión del nodo sacado de ABIERTA, sus nodos hijos más a la izquierda se introducen en ABIERTA antes que los situados más a la derecha.

2. Búsqueda Primero en Profundidad (de derecha a izquierda)

Este algoritmo usa ABIERTA como una pila, de manera que siempre se saca el primer nodo de la pila y se introducen sus hijos al principio de la misma. El contenido de ABIERTA previamente a cada extracción de un nodo es el siguiente:

Antes de sacar *H*: {*H*}
Antes de sacar *G*: {*G, D, A*}
Antes de sacar *C*: {*C, D, A*}
Antes de sacar *D*: {*D, A*}
Antes de sacar *E*: {*E, A*}
Antes de sacar *F*: {*F*, *A*}
Antes de sacar *B*: {*B, A*}
META ENCONTRADA

Observe que, tras cada expansión del nodo sacado de ABIERTA, sus nodos hijos más a la derecha se introducen en ABIERTA después que los situados más a la izquierda.

3. Búsqueda de Coste Uniforme

Este algoritmo mantiene ABIERTA ordenada según el coste del camino desde cada nodo al nodo inicial. En este ejercicio desharemos de forma arbitraria los posibles empates que surjan al determinar qué nodo se debe sacar de ABIERTA. El contenido de ABIERTA previamente a cada extracción de un nodo es el siguiente:

Antes de sacar *H*: {*H*(0)}
Antes de sacar *D*: {*D*(1), *A*(2), *G*(3)}
Antes de sacar *A*: {*A*(2), *G*(3), *E*(4)}
Antes de sacar *G*: {*G*(3), *E*(4)}
Antes de sacar *E*: {*E*(4), *C*(7)}
Antes de sacar *C*: {*C*(7), *F*(9)}
Antes de sacar *F*: {*F*(9)}
Antes de sacar *B*: {*B*(10)}
META ENCONTRADA

Observe que, tras cada expansión de un nodo, sus nodos hijos son introducidos en ABIERTA, donde todos los nodos quedan siempre ordenados por coste creciente. Para facilitar el seguimiento del

algoritmo, entre paréntesis y al lado de cada nodo de ABIERTA se especifica el coste del camino para ir desde dicho nodo hasta el nodo inicial.

4. Búsqueda en Anchura Iterativa (de derecha a izquierda)

Debido a que este algoritmo es una iteración de la búsqueda primero en anchura, los dos gestionan ABIERTA del mismo modo, es decir, como una cola. La diferencia reside en que en la búsqueda en anchura iterativa en cada iteración se fija un número máximo de hijos que pueden ser generados al expandir un nodo padre. Este número empieza valiendo 1 y es incrementado en una unidad al principio de cada nueva iteración.

Iteración 1 (cada expansión de un nodo padre genera como máximo un hijo):

Antes de sacar *H*: {*H*}
Antes de sacar *G*: {*G*}
Antes de sacar *C*: {*C*}

Iteración 2 (cada expansión de un nodo padre genera como máximo dos hijos):

Antes de sacar *H*: {*H*}
Antes de sacar *G*: {*G, D*}
Antes de sacar *D*: {*D, C*}
Antes de sacar *C*: {*C, E*}
Antes de sacar *E*: {*E*}
Antes de sacar *F*: {*F*}
Antes de sacar *B*: {*B*}
META ENCONTRADA

5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

Debido a que este algoritmo es una iteración de la búsqueda primero en profundidad, los dos gestionan ABIERTA del mismo modo, es decir, como una pila. La diferencia reside en que en la búsqueda en profundidad iterativa en cada iteración se fija una profundidad límite propia. Este número empieza valiendo 1, lo cual permite expandir únicamente el nodo inicial, y es incrementado en una unidad al principio de cada nueva iteración.

Iteración 1 (profundidad límite igual a 1):

Antes de sacar *H*: {*H*}
Antes de sacar *A*: {*A, D, G*}
Nótese que *A* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *D*: {*D, G*}
Nótese que *D* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *G*: {*G*}
Nótese que *G* no es expandido por coincidir su profundidad con la profundidad límite.

Iteración 2 (profundidad límite igual a 2):

Antes de sacar *H*: {*H*}
Antes de sacar *A*: {*A, D, G*}
Antes de sacar *D*: {*D, G*}
Antes de sacar *E*: {*E, G*}
Nótese que *E* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *G*: {*G*}
Antes de sacar *C*: {*C*}
Nótese que *C* no es expandido por coincidir su profundidad con la profundidad límite.

Iteración 3 (profundidad límite igual a 3):

Antes de sacar *H*: {*H*}
Antes de sacar *A*: {*A*, *D*, *G*}
Antes de sacar *D*: {*D*, *G*}
Antes de sacar *E*: {*E*, *G*}
Antes de sacar *F*: {*F*, *G*}

Nótese que *F* no es expandido por coincidir su profundidad con la profundidad límite.

Antes de sacar *G*: {*G*}
Antes de sacar *C*: {*C*}

Iteración 4 (profundidad límite igual a 4):

Antes de sacar *H*: {*H*}
Antes de sacar *A*: {*A*, *D*, *G*}
Antes de sacar *D*: {*D*, *G*}
Antes de sacar *E*: {*E*, *G*}
Antes de sacar *F*: {*F*, *G*}
Antes de sacar *B*: {*B*, *G*}
META ENCONTRADA

EJERCICIO 4:

Considere el espacio de búsqueda de la figura 4.1, que tiene forma de árbol, donde el nodo raíz del árbol es el nodo inicial, existe un único nodo meta y cada operador tiene asociado un coste. Describa cuál es el contenido de TABLA_A, posteriormente a cada expansión de un nodo, a partir de cada uno de los métodos siguientes de búsqueda sin información del dominio:

1. Búsqueda Primero en Anchura (de izquierda a derecha)
2. Búsqueda Primero en Profundidad (de derecha a izquierda)
3. Búsqueda de Coste Uniforme
4. Búsqueda en Anchura Iterativa (de derecha a izquierda)
5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

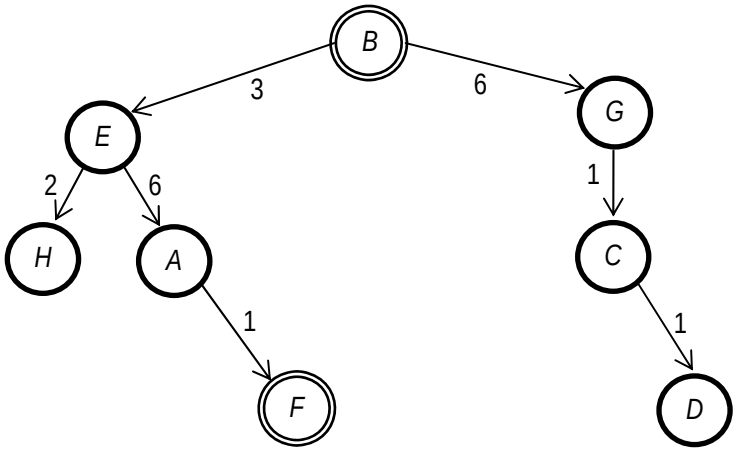


Figura 4.1: Árbol de búsqueda en el que el nodo inicial es *B*, el nodo meta es *F* y el coste de cada operador aparece al lado del arco que lo representa.

Para cada nodo de TABLA_A incluya la siguiente información: su nodo padre y el coste al nodo inicial. (En este ejercicio no es necesario incluir en TABLA_A información sobre los hijos de cada nodo expandido, ya que sólo existe un camino desde cada nodo al nodo inicial y, por tanto, el mejor camino desde cada nodo al nodo inicial no cambia a lo largo del proceso de búsqueda.)

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 4:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

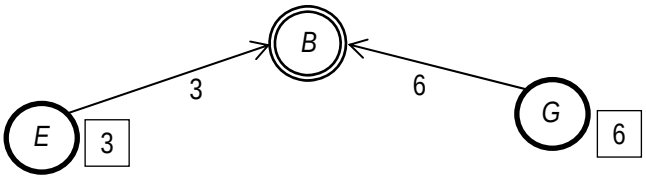
- Cada uno de los cinco apartados, correspondiente a un método de búsqueda concreto, se puntúa sobre 2 puntos.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no gestiona TABLA_A en la forma debida: la puntuación del apartado bajaría 1.6 puntos si la respuesta dada varía significativamente de la correcta, mientras que si la respuesta dada varía de la correcta como consecuencia de un despiste no conceptual entonces la puntuación del apartado bajaría sólo 0.4 puntos en vez de los 1.6 puntos mencionados.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no finaliza cuando es debido, la puntuación del apartado bajaría 0.4 puntos. (Esto es independiente de la gestión de TABLA_A dada como respuesta, que ya ha sido valorada anteriormente con un máximo de 1.6 puntos.)

SOLUCIÓN DEL EJERCICIO 4 (por Severino Fernández Galán):

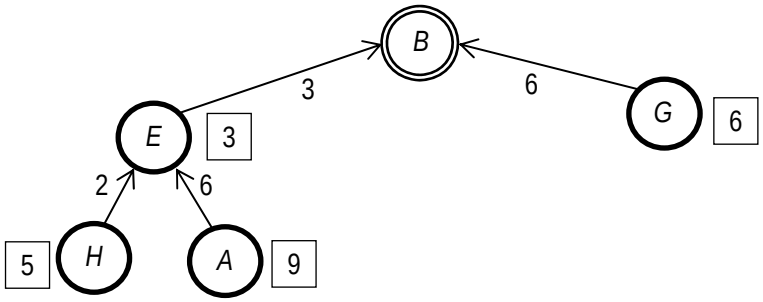
De cara a ilustrar la respuesta convenientemente, incluimos la información de TABLA_A gráficamente: por un lado, para cada nodo generado en una expansión trazamos un arco ascendente a su padre y, por otro lado, indicamos el coste al nodo inicial al lado de cada nodo generado.

1. Búsqueda Primero en Anchura (de izquierda a derecha)

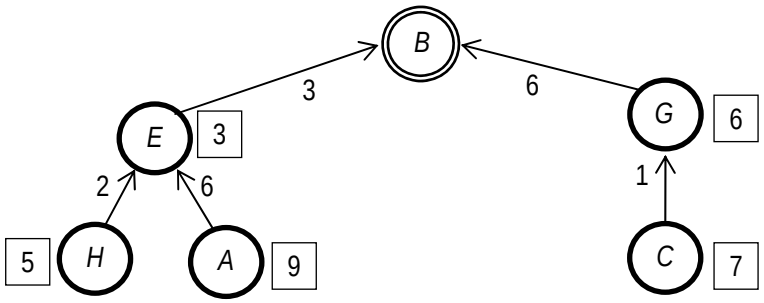
- Inicialmente, *B* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *B*. A continuación expandimos el nodo inicial, generando sus nodos hijos. Desde cada nodo hijo trazamos un nodo ascendente a su nodo padre. Al lado de cada nodo hijo indicamos el coste desde dicho nodo al nodo inicial.



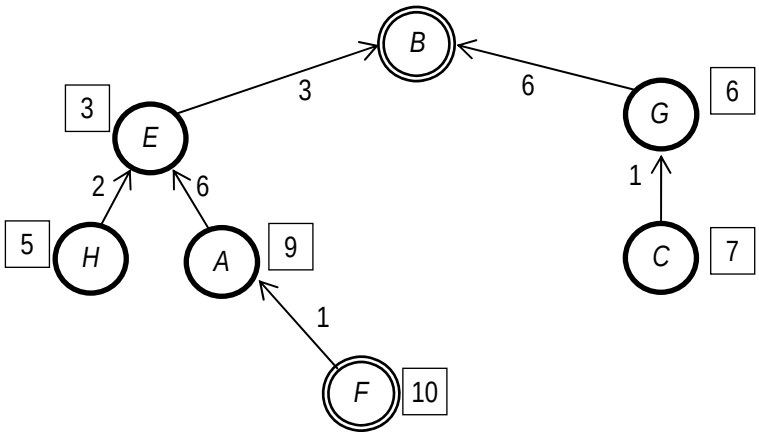
- Expandimos *E*:



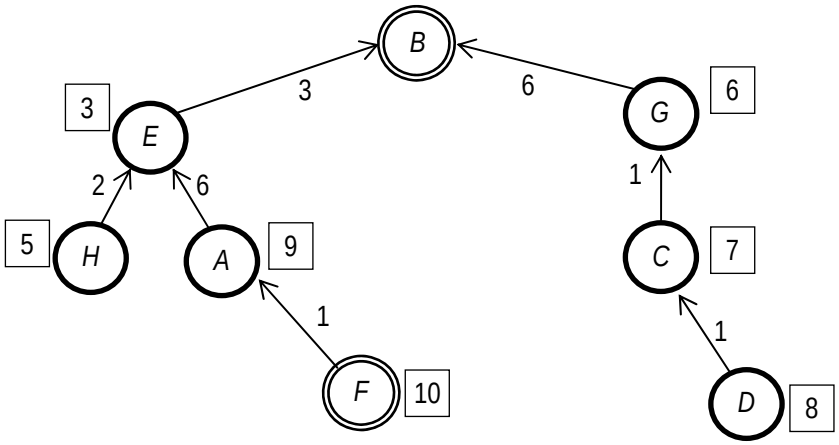
- Expandimos *G*:



- Expandimos *H* y TABLA_A no varía. A continuación expandimos *A*:



- Expandimos *C*:

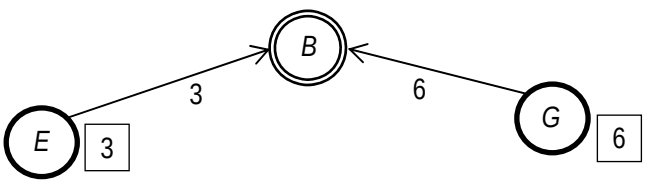


- A continuación elegiríamos *F* para su expansión y llegaríamos a la meta. El camino hallado hasta la meta tiene coste 10.

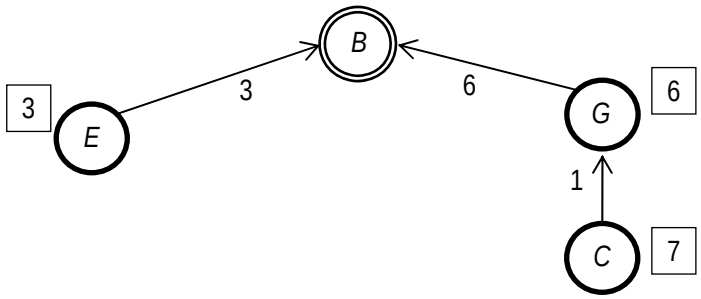
2. Búsqueda Primero en Profundidad (de derecha a izquierda)

Cuando sea necesario, en este algoritmo aplicaremos la función LimpiarTABLA_A (véase texto base, página 318). Tras la extracción de un nodo de ABIERTA y su posible expansión, dicha función elimina aquellos nodos que ya no son necesarios en el proceso de búsqueda de la meta. Esto permite preservar la linealidad de la complejidad espacial de este algoritmo con respecto a la profundidad de la solución.

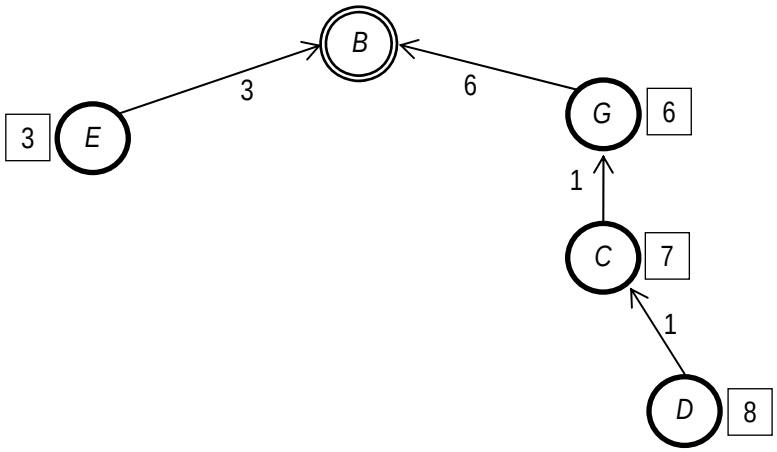
- Inicialmente, *B* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *B*. A continuación, expandimos el nodo inicial.



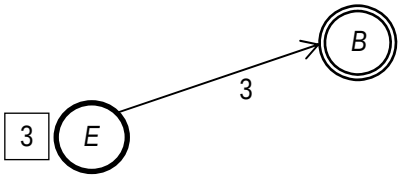
- Expandimos *G*:



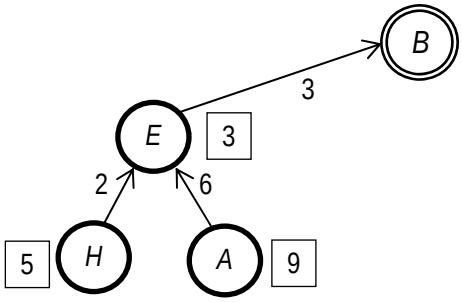
- Expandimos *C*:



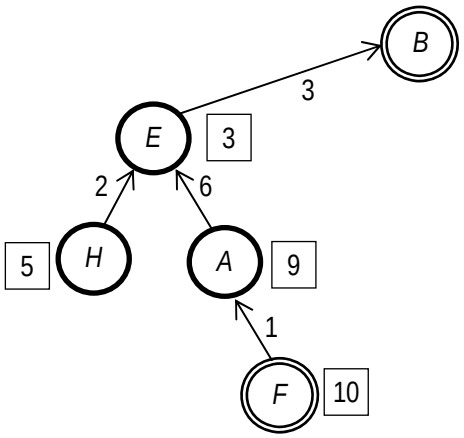
- Expandimos *D*. Seguidamente aplicamos LimpiarTABLA_A a *D*, es decir, sacamos *D* de TABLA_A por no tener hijos en ABIERTA. Por el mismo motivo sacamos de TABLA_A los nodos *C* y *G*.



- Expandimos *E*:



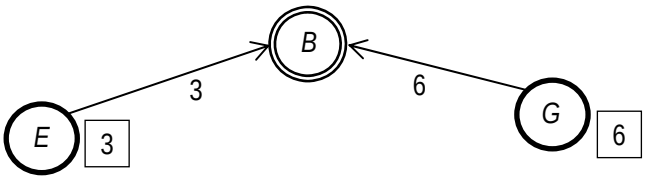
- Expandimos *A*:



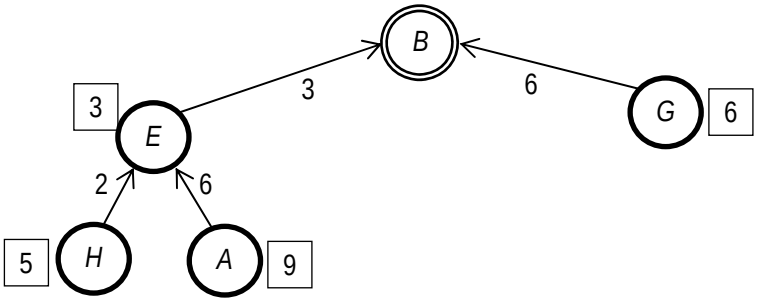
- A continuación elegiríamos *F* para su expansión y llegaríamos a la meta. El camino hallado hasta la meta tiene coste 10.

3. Búsqueda de Coste Uniforme

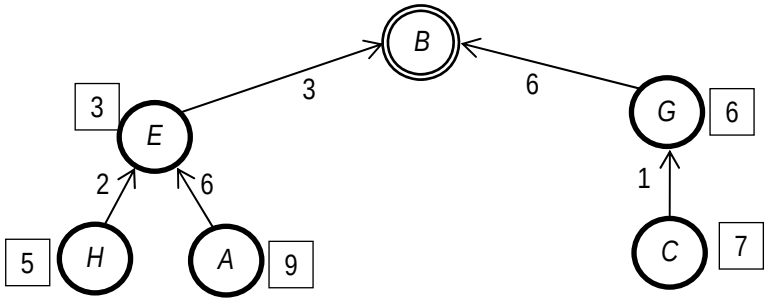
- Inicialmente, *B* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *B*. A continuación, expandimos el nodo inicial.



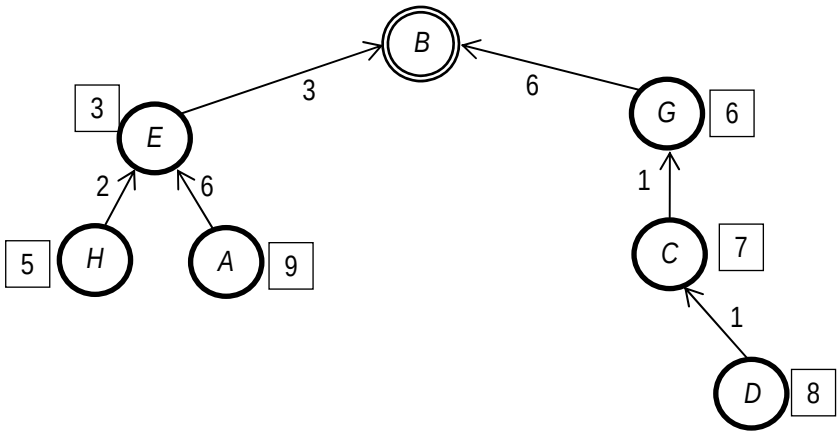
- Expandimos *E*:



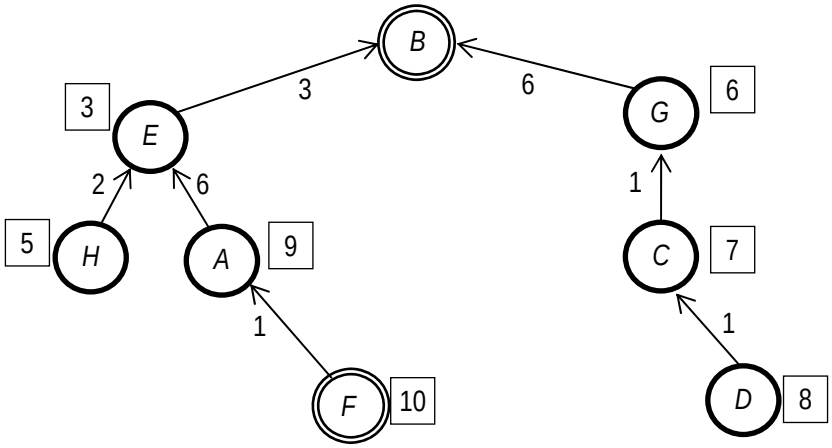
- Expandimos *H*, con lo que TABLA_A no cambia. A continuación, expandimos *G*:



- Expandimos *C*:



- Expandimos *D*, con lo que TABLA_A no cambia. A continuación, expandimos *A*:

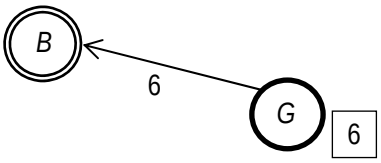


- Intentamos expandir *F*, con lo que hemos alcanzado la meta. El coste del camino solución hallado es 10.

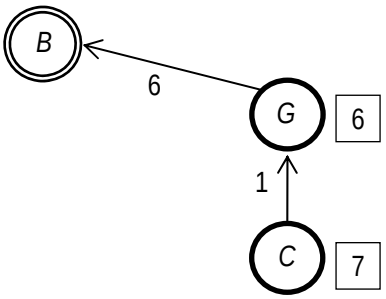
4. Búsqueda en Anchura Iterativa (de derecha a izquierda)

Iteración 1 (se genera como máximo 1 nodo hijo en cada expansión de un nodo padre):

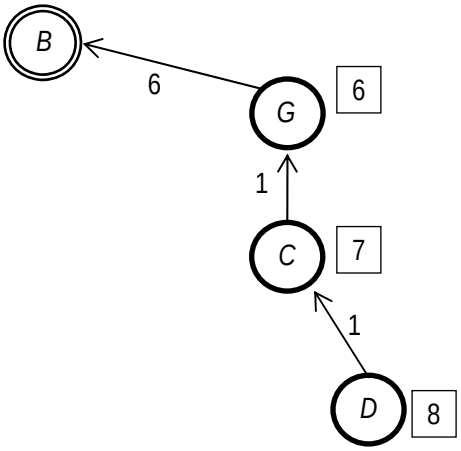
- Inicialmente, *B* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *B*. A continuación, expandimos el nodo inicial:



- Expandimos *G*:



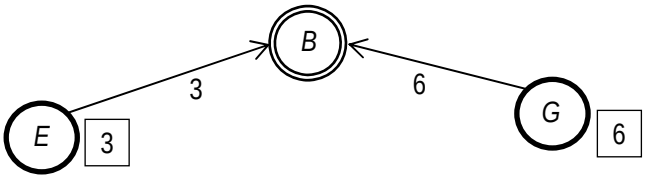
- Expandimos *C*:



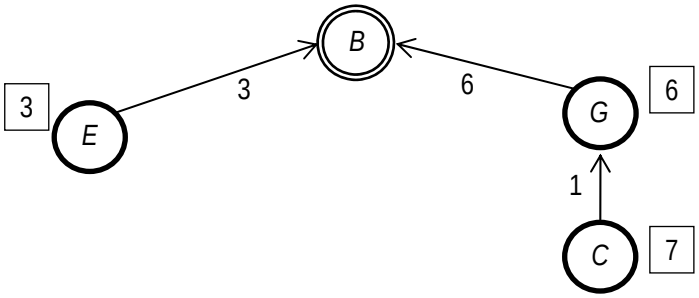
- Al expandir *D* finaliza la iteración actual.

<u>Iteración 2 (se generan como máximo 2 nodos hijos en cada expansión de un nodo padre)</u>:

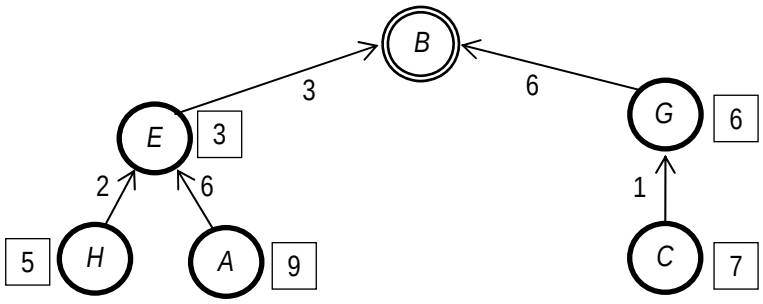
- Inicialmente, *B* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *B*. A continuación, expandimos el nodo inicial:



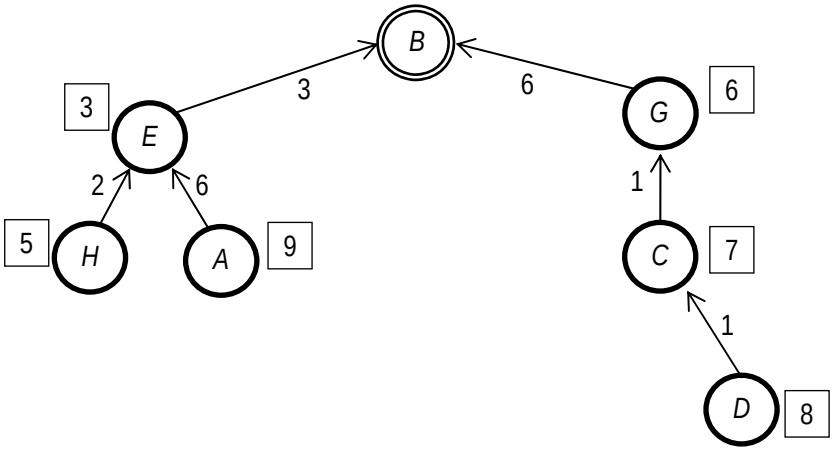
- Expandimos G:



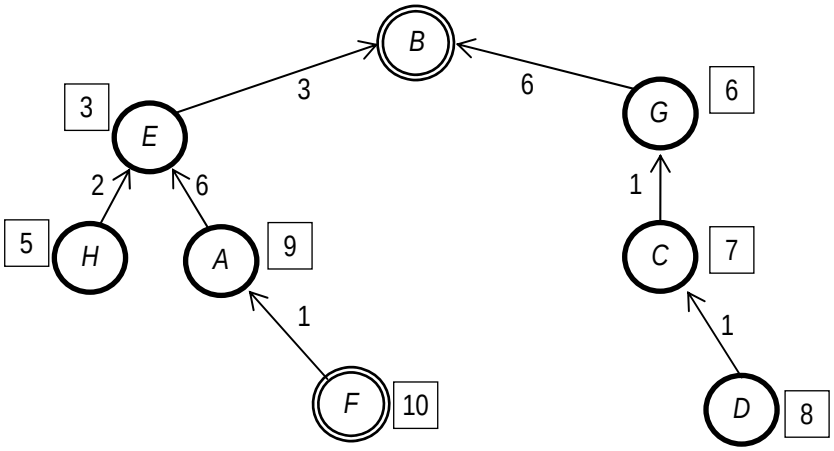
- Expandimos *E*:



- Expandimos *C*:



- Expandimos *A*:

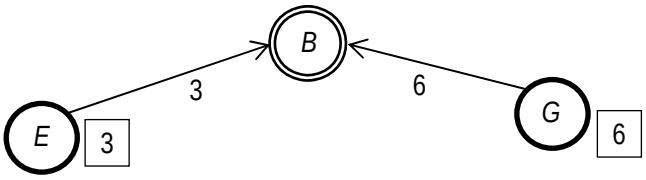


- Tras expandir *H* y *D*, TABLA_A no cambia. Finalmente, intentamos expandir *F* y alcanzamos la meta. El camino encontrado hasta el nodo inicial tiene coste 10.

5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

Iteración 1 (profundidad límite igual a 1):

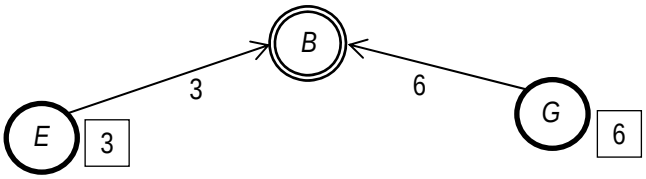
- Inicialmente, *B* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *B*. A continuación, expandimos el nodo inicial:



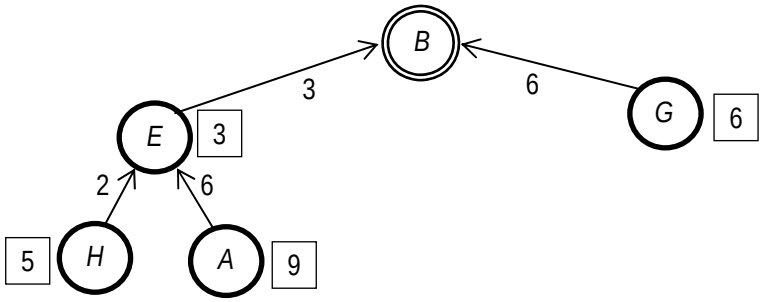
- Tras comprobar que *E* no es nodo meta, se le aplica la función LimpiarTABLA_A por estar en la profundidad límite y es sacado de TABLA_A. De igual manera, tras comprobar que *G* no es nodo meta, se le aplica la función LimpiarTABLA_A por estar en la profundidad límite y es sacado de TABLA_A. Seguidamente, tras aplicarle la función LimpiarTABLA_A, *B* es sacado de TABLA_A por no tener hijos en ABIERTA. A continuación pasamos a la siguiente iteración.

Iteración 2 (profundidad límite igual a 2):

- Inicialmente, *B* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *B*. A continuación, expandimos el nodo inicial:

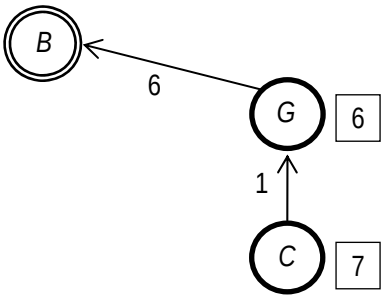


- Expandimos *E*:



- Tras comprobar que *H* no es nodo meta, se le aplica la función LimpiarTABLA_A por encontrarse en la profundidad límite y es sacado de TABLA_A. Del mismo modo, tras comprobar que *A* no es nodo meta, se le aplica la función LimpiarTABLA_A por estar en la profundidad límite y es sacado de TABLA_A. A continuación, tras aplicarle la función LimpiarTABLA_A, *E* es sacado de TABLA_A por no tener hijos en ABIERTA.

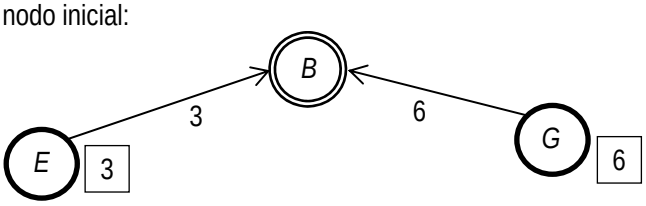
- Expandimos *G*:



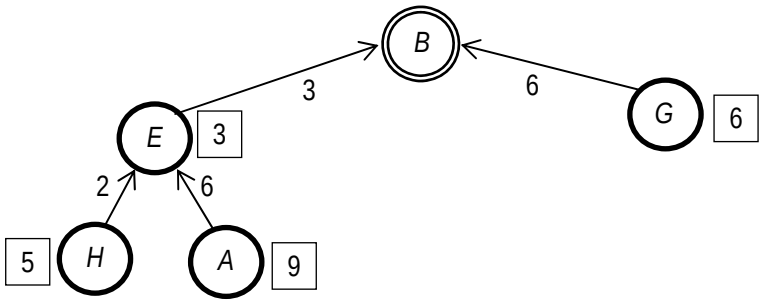
- Tras comprobar que *C* no es nodo meta, se le aplica la función LimpiarTABLA_A por estar en la profundidad límite y es sacado de TABLA_A. Tras aplicarle la función LimpiarTABLA_A, *G* es sacado de TABLA_A por no tener hijos en ABIERTA. Lo mismo ocurre posteriormente con *B*, por lo que pasamos a la siguiente iteración.

Iteración 3 (profundidad límite igual a 3):

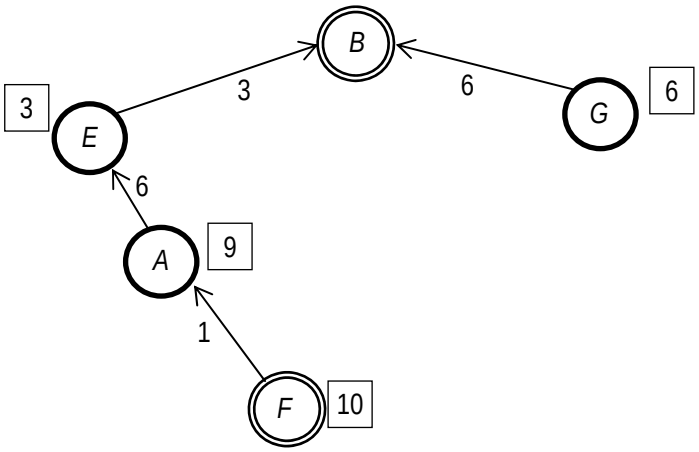
- Inicialmente, *B* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *B*. A continuación, expandimos el nodo inicial:



- Expandimos *E*:



- Expandimos *H*. Seguidamente, aplicamos la función LimpiarTABLA_A a *H* por no tener hijos en ABIERTA y es sacado de TABLA_A. A continuación, expandimos *A*:
:



- Tras elegir *F* para su expansión y comprobar que es nodo meta, finaliza el algoritmo habiendo hallado un camino entre el nodo inicial y el nodo meta de coste 10.

EJERCICIO 5:

Considere el grafo de la figura 5.1, donde el nodo inicial es n_1 y donde el nodo meta es n_{10}. Cada arco u operador lleva asociado su coste y en cada nodo aparece la estimación de la menor distancia desde ese nodo a una meta. Aplique paso a paso el algoritmo A* al grafo dado, indicando de forma razonada la siguiente información en cada paso del algoritmo:

1. Qué nodo es expandido.
2. Cuál es el contenido de ABIERTA tras la expansión del nodo, indicando el valor de la función de evaluación heurística para cada nodo de ABIERTA.
3. Cuál es el contenido de TABLA_A tras la expansión del nodo. Para cada nodo de TABLA_A incluya la siguiente información:
    a) Su nodo padre que indique el camino de menor coste hasta el nodo inicial encontrado hasta el momento
    b) El coste del camino de menor coste hasta el nodo inicial encontrado hasta el momento
    c) Sus nodos hijos (si el nodo de TABLA_A actual ya ha sido expandido)

Por último, ¿cuál es el camino solución hallado y su coste?

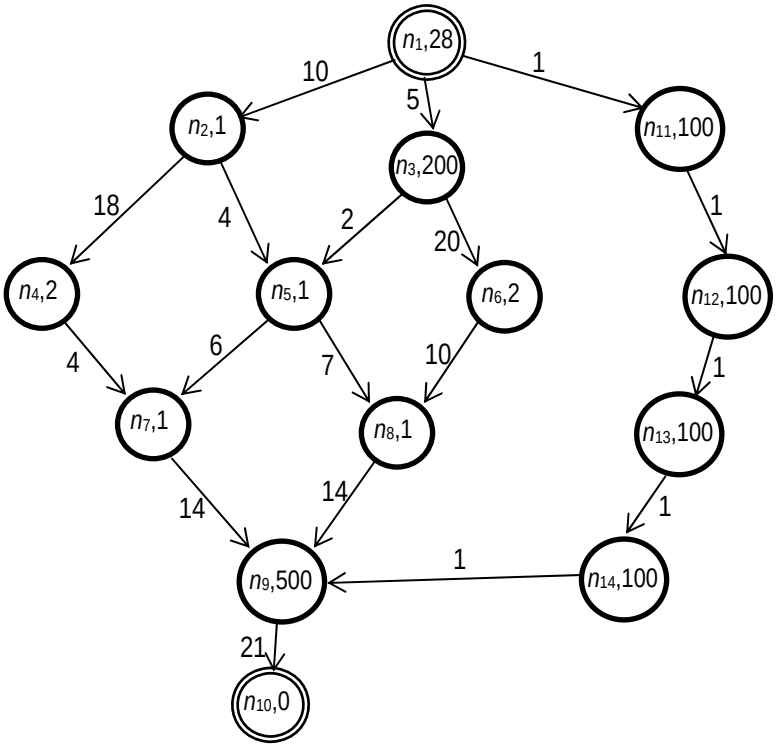


Figura 5.1: Grafo de búsqueda en el que el nodo inicial es n_1, el nodo meta es n_{10}, el coste de cada operador aparece al lado del arco que lo representa y al lado de cada nodo aparece la estimación de la menor distancia desde ese nodo a una meta.

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 5:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- El orden seguido en la expansión de los nodos se puntúa sobre 1.5 puntos.
- La forma en que se gestiona ABIERTA se puntúa sobre 3.75 puntos. Se hará especial énfasis en comprobar qué nodos hay en ABIERTA en cada paso del algoritmo y qué valores de la función de evaluación heurística se les asignan.

- La forma en que se gestiona TABLA_A se puntúa sobre 3.75 puntos. Se hará especial énfasis en comprobar qué nodos hay en TABLA_A en cada paso del algoritmo y qué padre mejor se les asigna (teniendo en cuenta las posibles reorientaciones o rectificaciones de enlaces)
- La correcta terminación del algoritmo se puntúa sobre 1 punto. Se hará especial énfasis en comprobar cuándo termina el algoritmo y qué camino solución devuelve.

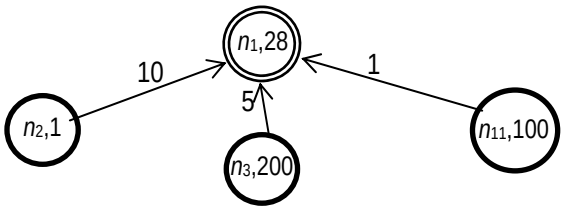
SOLUCIÓN DEL EJERCICIO 5 (por Severino Fernández Galán):

De cara a ilustrar la respuesta convenientemente, incluimos la información pedida sobre TABLA_A gráficamente. Para ello es necesario trazar, para cada nodo generado en una expansión, un arco ascendente a su padre expandido; además, hay que anotar para cada nodo su mejor padre encontrado hasta el momento (punto 3a del enunciado). De esta manera, siguiendo cada arco al mejor padre, se puede saber cuál es el mejor camino encontrado hasta el momento desde cada nodo al nodo inicial (punto 3b del enunciado). Además, los arcos ascendentes que llegan a un nodo ya expandido lo enlazan a sus nodos hijos (punto 3c del enunciado).

- PASO 0. El nodo inicial n_1 es introducido en ABIERTA y en TABLA_A, con lo que tenemos la siguiente situación:

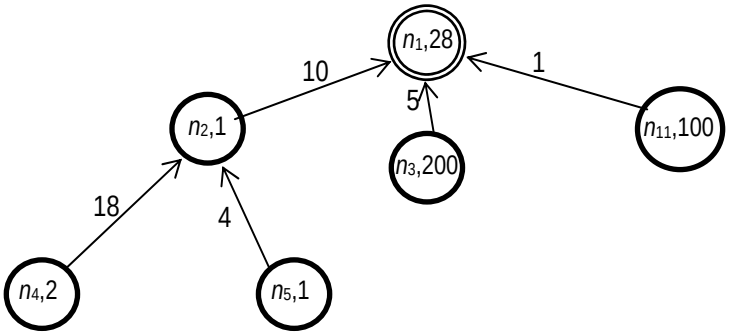
ABIERTA = {$\underline{n_1(0+28)}$}

- PASO 1. Expandimos el nodo n_1 de ABIERTA. Tras la expansión, la situación es la siguiente:



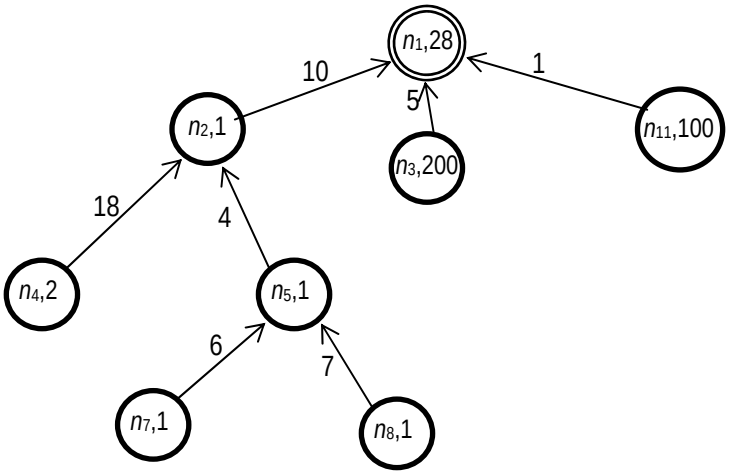
ABIERTA = {$\underline{n_2(10+1)}$, $n_{11}(1+100)$, $n_3(5+200)$}

- PASO 2. Expandimos n_2 por ser el nodo de ABIERTA con menor valor de la función de evaluación heurística, $f=g+h$ (al ser g=10 y h=1).



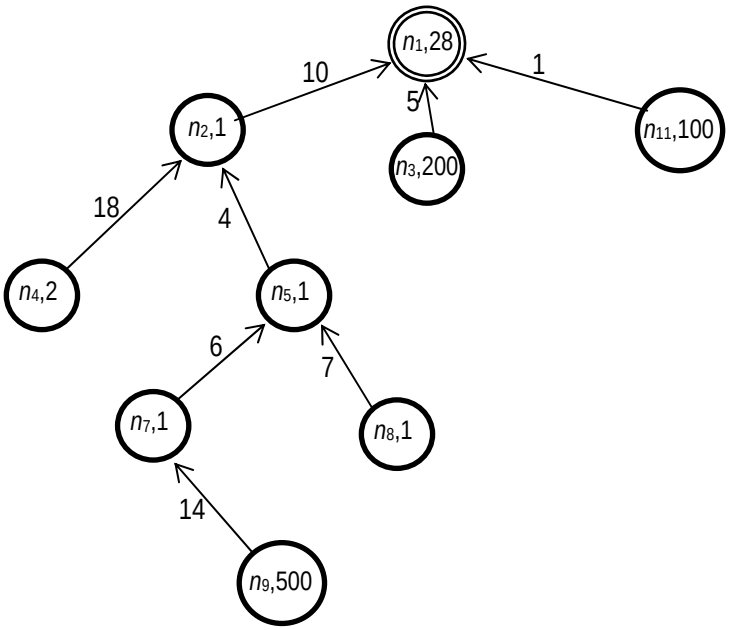
ABIERTA = {$\underline{n_5(14+1)}$, $n_4(28+2)$, $n_{11}(1+100)$, $n_3(5+200)$}

- PASO 3. Expandimos n_5.



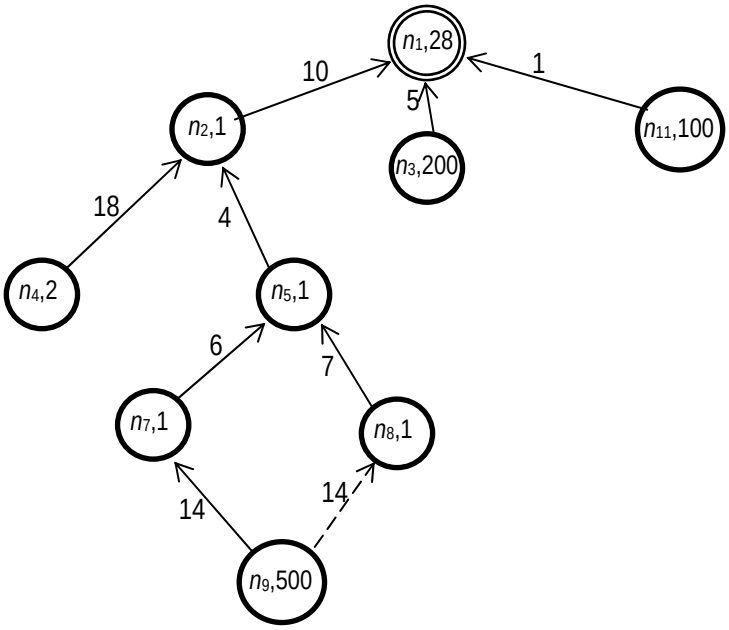
ABIERTA = {$\underline{n_7(20+1)}$, $n_8(21+1)$, $n_4(28+2)$, $n_{11}(1+100)$, $n_3(5+200)$}

- PASO 4. Expandimos n_7.



ABIERTA = {$\underline{n_8(21+1)}$, $n_4(28+2)$, $n_{11}(1+100)$, $n_3(5+200)$, $n_9(34+500)$}

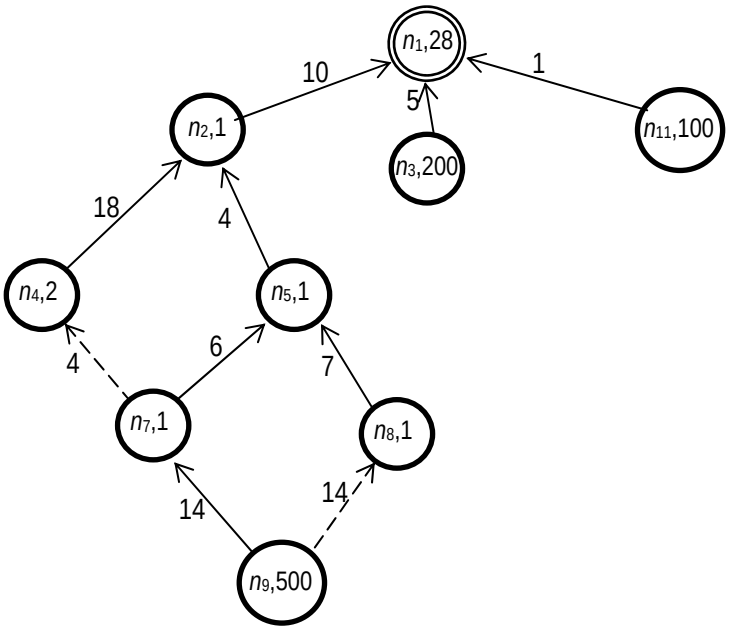
- PASO 5. Expandimos n_8.



ABIERTA = {$\underline{n_4(28+2)}$, $n_{11}(1+100)$, $n_3(5+200)$, $n_9(34+500)$}

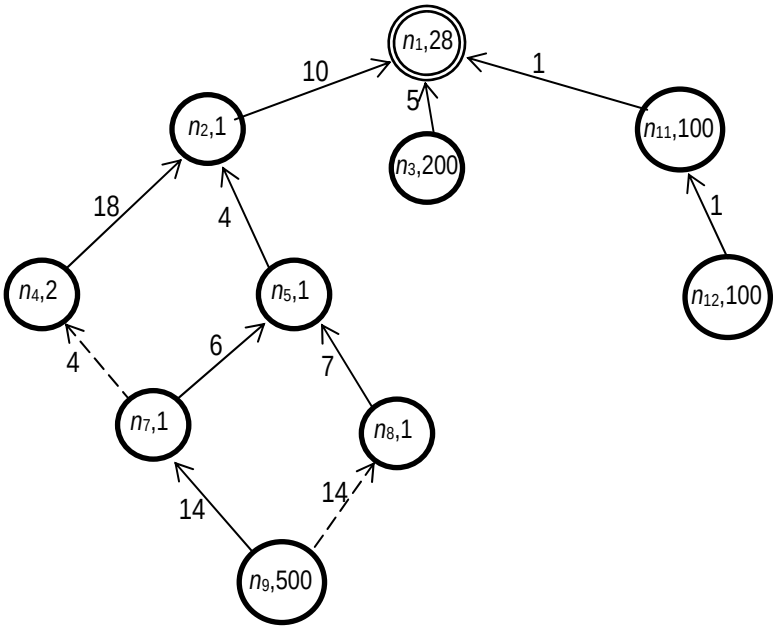
Observe que el mejor camino desde n_9 al nodo inicial lo marca su padre n_7 (coste 14+6+4+10=34) y no su padre n_8 (coste 14+7+4+10=35). Por ello, el arco ascendente de n_9 a n_7 se marca con trazo continuo y el arco ascendente de n_9 a n_8 se marca con trazo discontinuo. Es importante darse cuenta que el conjunto de arcos con trazo continuo formará siempre un árbol en el grafo parcial de búsqueda desarrollado hasta el momento.

- PASO 6. Expandimos n_4.



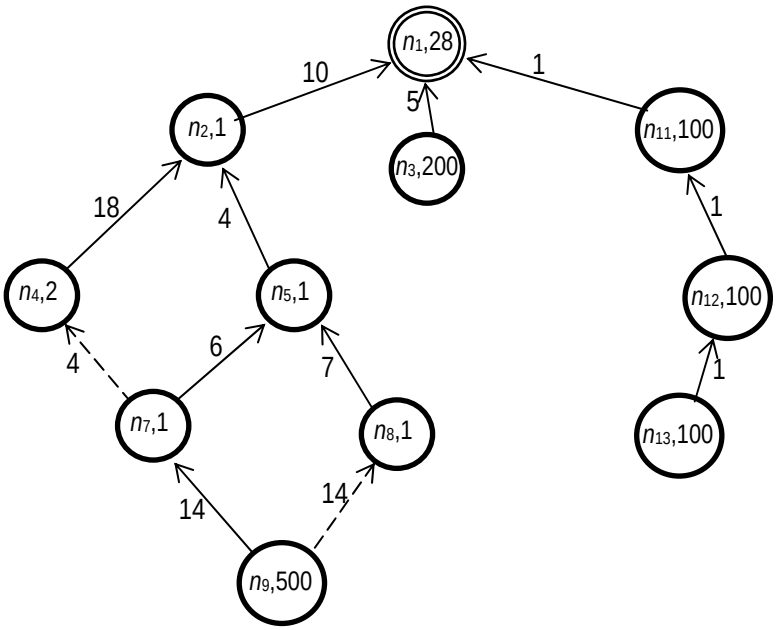
ABIERTA = {$\underline{n_{11}(1+100)}$, $n_3(5+200)$, $n_9(34+500)$}

- PASO 7. Expandimos n_{11}.



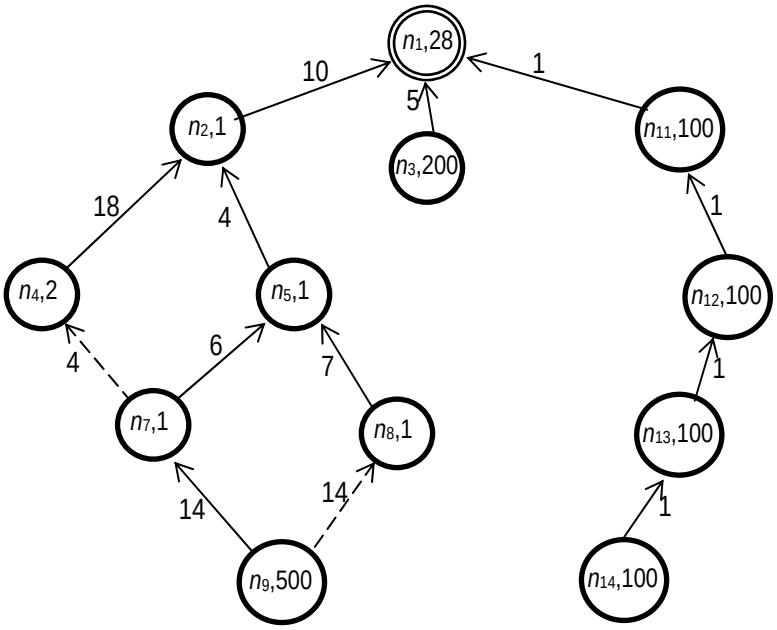
ABIERTA = {$\underline{n_{12}(2+100)}$, $n_3(5+200)$, $n_9(34+500)$}

- PASO 8. Expandimos n_{12}.



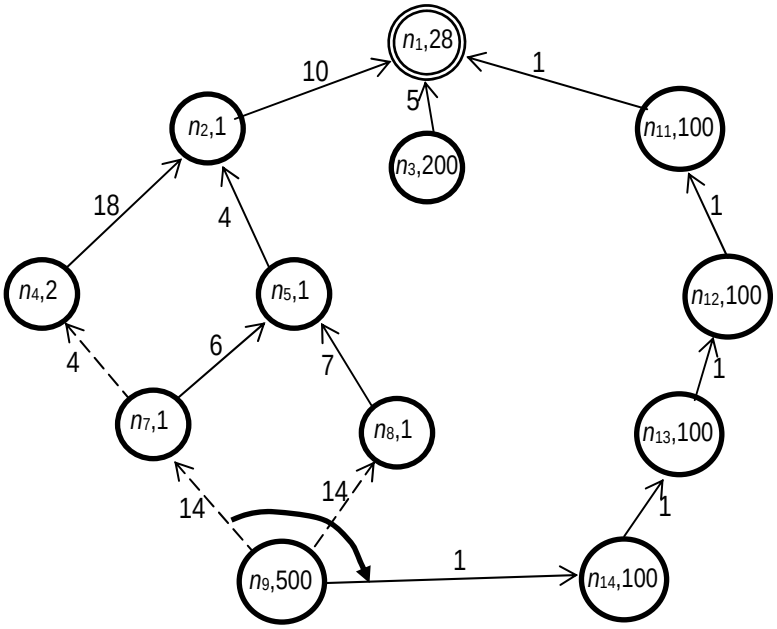
ABIERTA = {$\underline{n_{13}(3+100)}$, $n_3(5+200)$, $n_9(34+500)$}

- PASO 9. Expandimos n_{13}.



ABIERTA = {$\underline{n_{14}(4+100)}$, $n_3(5+200)$, $n_9(34+500)$}

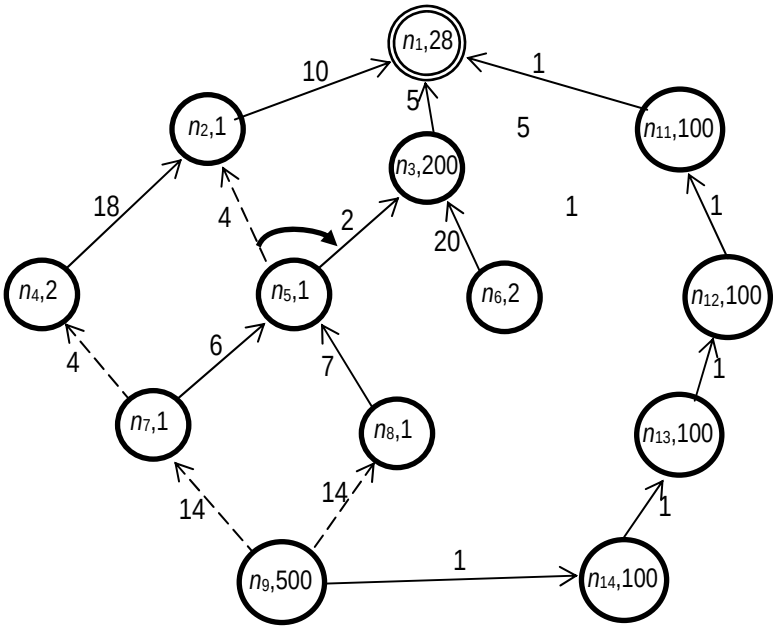
- PASO 10. Expandimos n_{14}.



ABIERTA = {$\underline{n_3(5+200)}$, $n_9(5+500)$}

Observe que ha habido una reorientación del mejor padre de n_9, que antes era n_7 y ahora pasa a ser n_{14}. El nuevo mejor coste de n_9 al nodo inicial, $g(n_9)$, es ahora 1+1+1+1+1=5, que se puede calcular a partir de los costes de los mejores arcos ascendentes hallados hasta el momento: $n_9 \rightarrow n_{14}$, $n_{14} \rightarrow n_{13}$, $n_{13} \rightarrow n_{12}$, $n_{12} \rightarrow n_{11}$ y $n_{11} \rightarrow n_1$.

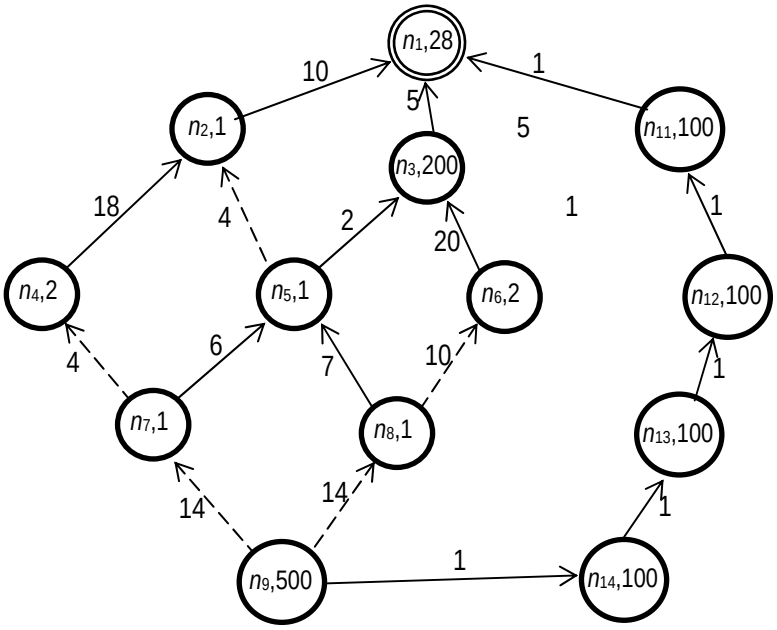
- PASO 11. Expandimos n_3.



ABIERTA = {$\underline{n_6(25+2)}$, $n_9(5+500)$}

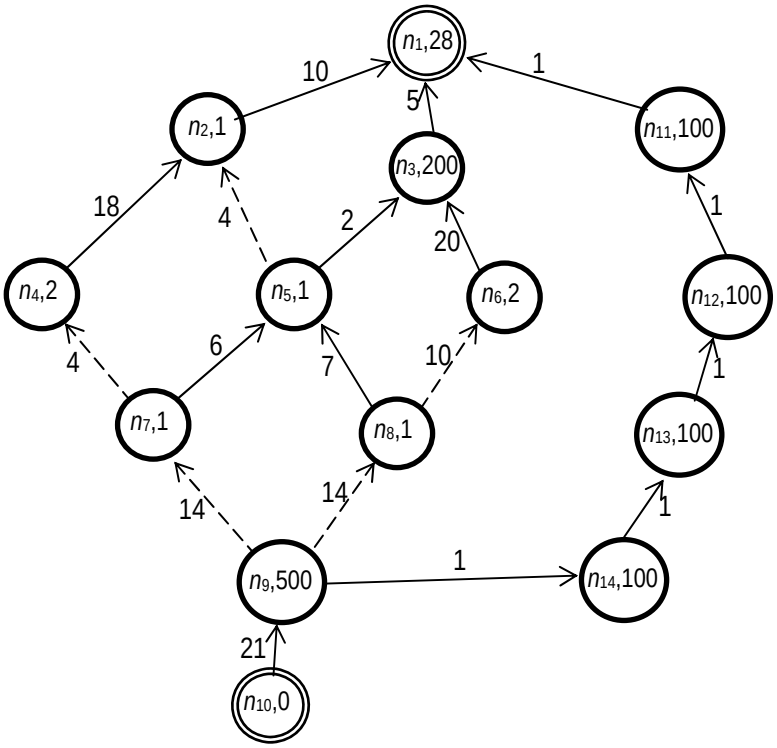
Obsérvese que hay una reorientación del mejor padre de n_5.

- PASO 12. Expandimos n_6.



ABIERTA = {$\underline{n_9(5+500)}$}

- PASO 13. Expandimos n_9.



ABIERTA = {$\underline{n_{10}(26+0)}$}

- PASO 14. Intentamos expandir n_{10} y alcanzamos una meta, con lo que el algoritmo termina. El camino solución encontrado es: $n_1 \rightarrow n_{11} \rightarrow n_{12} \rightarrow n_{13} \rightarrow n_{14} \rightarrow n_9 \rightarrow n_{10}$, cuyo coste es 26.

EJERCICIO 6:

Considere el grafo de la figura 6.1, donde el nodo inicial es *D* y donde los nodos meta son desconocidos. Cada arco u operador lleva asociado su coste y en cada nodo aparece su valor de la función de evaluación heurística (que hay que minimizar). Aplique paso a paso el algoritmo de escalada o máximo gradiente al grafo dado. Para ello indique de forma razonada qué nodo se expande en cada paso y cuál es el nodo final devuelto por el algoritmo. Utilice como *criterio de selección* el de mejor vecino. Utilice como *criterio de terminación* el que no se hayan producido mejoras durante los cinco últimos pasos del algoritmo.

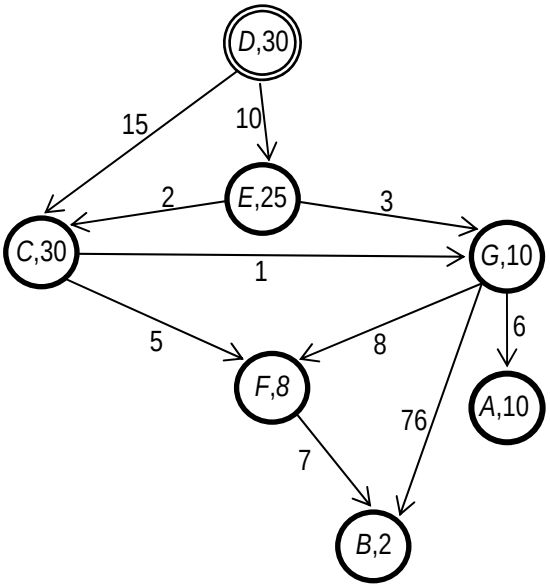


Figura 6.1: Grafo de búsqueda en el que el nodo inicial es *D*, los nodos meta son desconocidos, el coste de cada operador aparece al lado del arco que lo representa y al lado de cada nodo aparece el valor de su función de evaluación heurística (que hay que minimizar).

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 6:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- La correcta aplicación en cada paso del algoritmo del *criterio de selección* del vecino o hijo del nodo actual (qué vecino o hijo del nodo actual es considerado como candidato para sustituirlo) se puntúa sobre 4.5 puntos.
- La correcta aplicación en cada paso del algoritmo del *criterio de aceptación* del vecino o hijo seleccionado (si el vecino o hijo candidato sustituye o no finalmente al nodo actual) se puntúa sobre 4.5 puntos.
- La correcta aplicación del *criterio de finalización* del algoritmo se puntúa sobre 1 punto.

SOLUCIÓN DEL EJERCICIO 6 (por Severino Fernández Galán):

- PASO 1: Al principio expandimos el nodo inicial *D*, generando sus nodos hijos {*C*(30), *E*(25)}. Seleccionamos el nodo *E* por ser el mejor de los nodos hijos. A continuación aceptamos el nodo *E* como nuevo nodo actual en sustitución de *D*, debido a que el valor de la función de evaluación heurística de *E* es mejor o igual que el de *D* ($25 \leq 30$).

- PASO 2: Expandimos el nodo actual *E* y generamos sus hijos: {*C*(30), *G*(10)}. Seleccionamos el nodo *G* por ser el mejor de los nodos hijos. Seguidamente aceptamos el nodo *G* como nuevo nodo actual en sustitución de *E*, debido a que el valor de la función de evaluación heurística de *G* es mejor o igual que el de *E* ($10 \leq 25$).

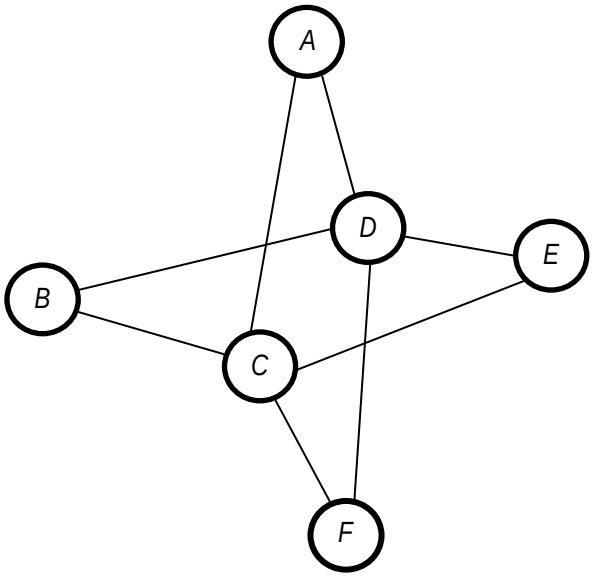
- PASO 3: Expandimos el nodo actual *G* y generamos sus hijos: {*F*(8), *B*(2), *A*(10)}. Seleccionamos el nodo *B* por ser el mejor de los nodos hijos. A continuación aceptamos el nodo *B* como nuevo nodo actual en sustitución de *G*, debido a que el valor de la función de evaluación heurística de *B* es mejor o igual que el de *G* ($2 \leq 10$).

Dado que *B* no tiene sucesores, la búsqueda terminaría. El algoritmo devolvería el nodo *B*(2) como mejor nodo encontrado.

AÑO 2015

EJERCICIO 1:

Dibuje mediante un grafo dirigido o describa detalladamente mediante una tabla el espacio de estados (o espacio de búsqueda) completo para el *problema del coloreado de grafos* descrito más adelante. Para ello especifique: el conjunto de todos los estados posibles, el estado inicial, el o los estados meta, los operadores aplicables a cada estado y el coste asociado a cada operador. En el problema del coloreado de grafos se dispone de un grafo no dirigido cuyos nodos hay que colorear con los colores {1, 2, …, *c*–1, *c*}, de manera que no haya dos nodos vecinos con el mismo color. Suponga que hay que colorear el grafo de la figura siguiente utilizando dos colores: {1, 2}.



Suponga además que, dado un estado (o grafo en el que todo nodo tiene asignado un color), los posibles operadores que se pueden aplicar sobre dicho estado son de la forma [*x*, *y*], donde "*x*" representa un nodo cualquiera del grafo e "*y*" representa cualquier color diferente al color actual del nodo representado por "*x*". El efecto del operador [*x*, *y*] sobre el estado actual consiste en cambiar el color actual del nodo representado por "*x*" por el nuevo color representado por "*y*". Suponga finalmente que el estado inicial está compuesto por el grafo de la figura coloreado del siguiente modo: {*color*(*A*)=1, *color*(*B*)=1, *color*(*C*)=2, *color*(*D*)=1, *color*(*E*)=1, *color*(*F*)=2}, donde *color*(*N*) representa el color asignado al nodo *N*. ¿Cuál es la solución menos costosa para este problema si el estado inicial y los operadores posibles son los descritos con anterioridad? Un estado meta sería una asignación de colores a los nodos del grafo tal que no haya dos nodos vecinos con el mismo color.

NOTA IMPORTANTE: Dado el alto número de estados y operadores de este problema, se recomienda optar por desarrollar la tabla asociada al espacio de búsqueda completo y no optar por dibujar el grafo dirigido equivalente.

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 1:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- Los estados se especifican correctamente: 2.5 puntos
- Los operadores se especifican correctamente: 2.5 puntos
- Los costes de los operadores se especifican correctamente: 1 punto
- El espacio de búsqueda se dibuja (mediante un grafo dirigido) o se describe (mediante una tabla) correctamente: 3 puntos
- La solución de menor coste se especifica correctamente: 1 punto

SOLUCIÓN DEL EJERCICIO 1 (por Severino Fernández Galán):

Para representar los estados posibles utilizaremos la siguiente notación. El estado del problema lo representamos como:

$$\{color(A), color(B), color(C), color(D), color(E), color(F)\},$$

donde $color(N) \in \{1, 2\}$ $\forall N \in \{A, B, C, D, E, F\}$. Por ejemplo, el estado {1, 1, 1, 1, 1, 2} representa el grafo en que todos los nodos tienen color "1" a excepción del nodo *F*, que tiene color "2". Nótese que existen $2^6 = 64$ estados posibles en el problema planteado.

El estado inicial descrito en el enunciado se representa como {1, 1, 2, 1, 1, 2}. Por otro lado, existen dos estados meta: {1, 1, 2, 2, 1, 1} y {2, 2, 1, 1, 2, 2}.

Dado un estado cualquiera, hay seis operadores aplicables al mismo: [*A*, *y*], [*B*, *y*], [*C*, *y*], [*D*, *y*], [*E*, *y*] y [*F*, *y*], donde "*y*" toma el valor "1" o "2" dependiendo de si el estado actual tiene en el nodo correspondiente un valor de "2" o "1" respectivamente. Cada uno de estos seis operadores consiste en cambiar de color uno de los seis nodos del grafo. Se puede considerar que cada operador tiene coste unidad, así que el coste de un camino es el número de arcos que lo componen.

En la tabla 1.1 figuran los estados posibles del sistema, los operadores que se pueden aplicar a cada estado posible y los estados resultantes de aplicar cada operador. Se ha resaltado en color tanto el estado inicial (rojo) como los estados meta (verde).

De forma alternativa, la información incluida en la tabla 1.1 se podría representar mediante un grafo dirigido. Dada la recomendación hecha en la nota del enunciado, no dibujamos aquí esta opción.

La solución menos costosa para el problema planteado en el enunciado consiste en: cambiar el color del nodo *D* (operador [*D*, 2]) y cambiar el color del nodo *F* (operador [*F*, 1]), bien en este orden o en el orden contrario. El coste para llegar desde el estado inicial al estado meta {1, 1, 2, 2, 1, 1} mediante cualquiera de los dos caminos mencionados es 2.

ESTADOS	OPERADORES					
	[A, y]	[B, y]	[C, y]	[D, y]	[E, y]	[F, y]
111111	211111	121111	112111	111211	111121	111112
111112	211112	121112	112112	111212	111122	111111
111121	211121	121121	112121	111221	111111	111122
111122	211122	121122	112122	111222	111112	111121
111211	211211	121211	112211	111111	111221	111212
111212	211212	121212	112212	111112	111222	111211
111221	211221	121221	112221	111121	111211	111222
111222	211222	121222	112222	111122	111212	111221
112111	212111	122111	111111	112211	112121	112112
112112	212112	122112	111112	112212	112122	112111
112121	212121	122121	111121	112221	112111	112122
112122	212122	122122	111122	112222	112112	112121
112211	ESTADO META					
112212	212212	122212	111212	112112	112222	112211
112221	212221	122221	111221	112121	112211	112222
112222	212222	122222	111222	112122	112212	112221
121111	221111	111111	122111	121211	121121	121112
121112	221112	111112	122112	121212	121122	121111
121121	221121	111121	122121	121221	121111	121122
121122	221122	111122	122122	121222	121112	121121
121211	221211	111211	122211	121111	121221	121212
121212	221212	111212	122212	121112	121222	121211
121221	221221	111221	122221	121121	121211	121222
121222	221222	111222	122222	121122	121212	121221
122111	222111	112111	121111	122211	122121	122112
122112	222112	112112	121112	122212	122122	122111
122121	222121	112121	121121	122221	122111	122122
122122	222122	112122	121122	122222	122112	122121
122211	222211	112211	121211	122111	122221	122212
122212	222212	112212	121212	122112	122222	122211
122221	222221	112221	121221	122121	122211	122222
122222	222222	112222	121222	122122	122212	122221
211111	111111	221111	212111	211211	211121	211112
211112	111112	221112	212112	211212	211122	211111
211121	111121	221121	212121	211221	211111	211122
211122	111122	221122	212122	211222	211112	211121
211211	111211	221211	212211	211111	211221	211212
211212	111212	221212	212212	211112	211222	211211
211221	111221	221221	212221	211121	211211	211222
211222	111222	221222	212222	211122	211212	211221
212111	112111	222111	211111	212211	212121	212112
212112	112112	222112	211112	212212	212122	212111
212121	112121	222121	211121	212221	212111	212122
212122	112122	222122	211122	212222	212112	212121
212211	112211	222211	211211	212111	212221	212212
212212	112212	222212	211212	212112	212222	212211
212221	112221	222221	211221	212121	212211	212222
212222	112222	222222	211222	212122	212212	212221
221111	121111	211111	222111	221211	221121	221112
221112	121112	211112	222112	221212	221122	221111
221121	121121	211121	222121	221221	221111	221122
221122	ESTADO META					
221211	121211	211211	222211	221111	221221	221212
221212	121212	211212	222212	221112	221222	221211
221221	121221	211221	222221	221121	221211	221222
221222	121222	211222	222222	221122	221212	221221
222111	122111	212111	221111	222211	222121	222112
222112	122112	212112	221112	222212	222122	222111
222121	122121	212121	221121	222221	222111	222122
222122	122122	212122	221122	222222	222112	222121
222211	122211	212211	221211	222111	222221	222212
222212	122212	212212	221212	222112	222222	222211
222221	122221	212221	221221	222121	222211	222222
222222	122222	212222	221222	222122	222212	222221

Tabla 1.1: Estados posibles, operadores aplicables a cada estado y estados resultantes de aplicar cada operador para el problema del coloreado de grafos planteado en este ejercicio. El estado inicial se resalta en rojo y los estados meta se resaltan en verde.

EJERCICIO 2:

Considere el espacio de búsqueda de la figura 2.1, que tiene forma de árbol, donde el nodo raíz del árbol es el nodo inicial, existe un único nodo meta y cada operador tiene asociado un coste. Explique razonadamente en qué orden se expandirían los nodos de dicho árbol de búsqueda a partir de cada uno de los métodos siguientes de búsqueda sin información del dominio:

1. Búsqueda Primero en Anchura (de izquierda a derecha)
2. Búsqueda Primero en Profundidad (de derecha a izquierda)
3. Búsqueda de Coste Uniforme
4. Búsqueda en Anchura Iterativa (de derecha a izquierda)
5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

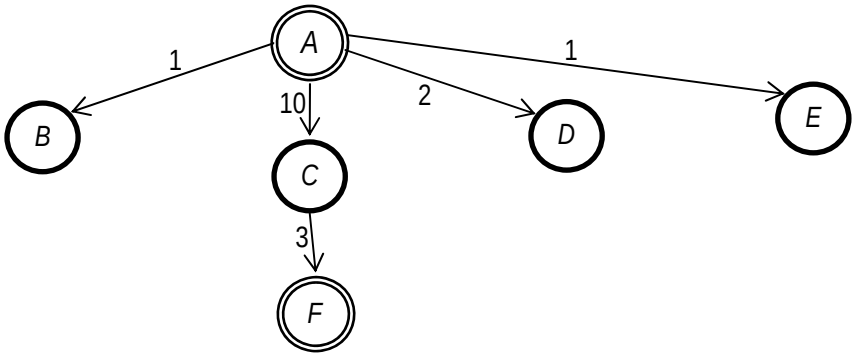


Figura 2.1: Árbol de búsqueda en el que el nodo inicial es *A*, el nodo meta es *F* y el coste de cada operador aparece al lado del arco que lo representa.

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 2:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- Cada uno de los cinco apartados, correspondiente a un método de búsqueda concreto, se puntúa sobre 2 puntos.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no expande los nodos en el orden debido: la puntuación del apartado bajaría 1.6 puntos si el orden dado como respuesta varía significativamente del correcto, mientras que si el orden dado como respuesta varía del correcto como consecuencia de un despiste no conceptual entonces la puntuación del apartado bajaría sólo 0.4 puntos en vez de los 1.6 puntos mencionados.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no finaliza cuando es debido, la puntuación del apartado bajaría 0.4 puntos. (Esto es independiente del orden de expansión de nodos dado como respuesta, que ya ha sido valorado anteriormente con un máximo de 1.6 puntos.)

SOLUCIÓN DEL EJERCICIO 2 (por Severino Fernández Galán):

1. Búsqueda Primero en Anchura (de izquierda a derecha)

Este algoritmo explora el árbol de búsqueda por niveles de profundidad, así que el orden de expansión de los nodos de izquierda a derecha sería el reflejado en la figura 2.2.

2. Búsqueda Primero en Profundidad (de derecha a izquierda)

Este algoritmo explora el árbol de búsqueda bajando de nivel siempre que sea posible. Si no es posible, se sube al nodo más cercano al nodo actual desde el que poder seguir bajando de nivel. El orden de expansión de los nodos de derecha a izquierda según este algoritmo se dibuja en la figura 2.3.

3. Búsqueda de Coste Uniforme

Este algoritmo explora el árbol de búsqueda expandiendo aquel nodo disponible cuyo coste al nodo inicial sea el menor. El orden de expansión de nodos según este criterio se indica en la figura 2.4. Obsérvese que después de la primera expansión (nodo *A*) hay un empate entre los nodos *B* y *E*, cuyos costes al nodo inicial son iguales a 1 en los dos casos. Nosotros hemos elegido el nodo *B* para realizar la segunda expansión, pero también se podría haber elegido el otro nodo, *E*. Los posibles empates posteriores también los desharíamos de esta forma arbitraria.

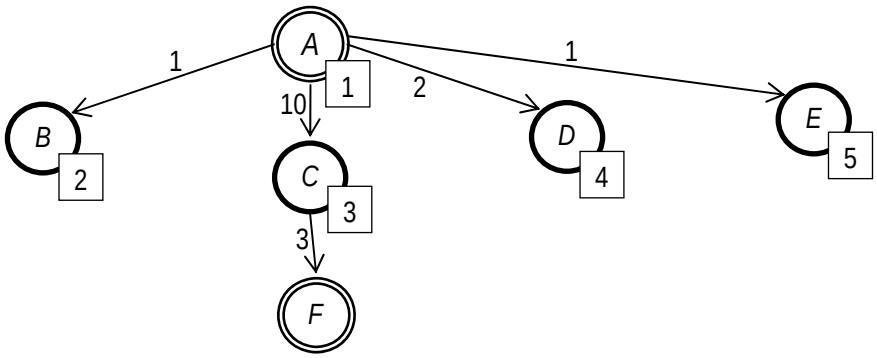


Figura 2.2: Orden de expansión de nodos para el árbol de la figura 2.1 según la búsqueda primero en anchura (de izquierda a derecha). Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto, el algoritmo termina en ese momento sin que se lleguen a generar sus hijos.

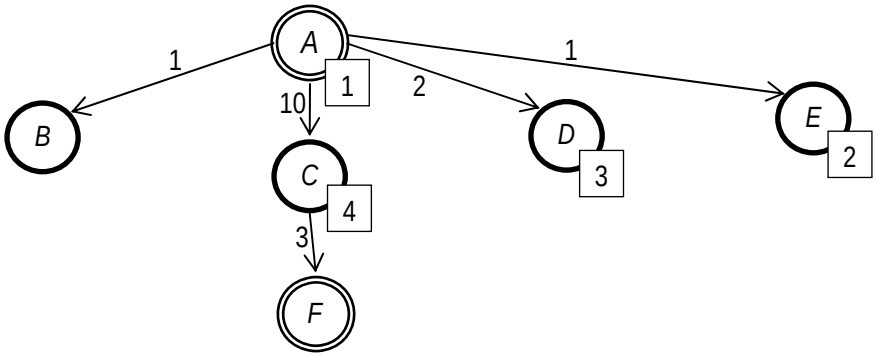


Figura 2.3: Orden de expansión de nodos para el árbol de la figura 2.1 según la búsqueda primero en profundidad (de derecha a izquierda). Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto, el algoritmo termina en ese momento sin que se lleguen a generar sus hijos.

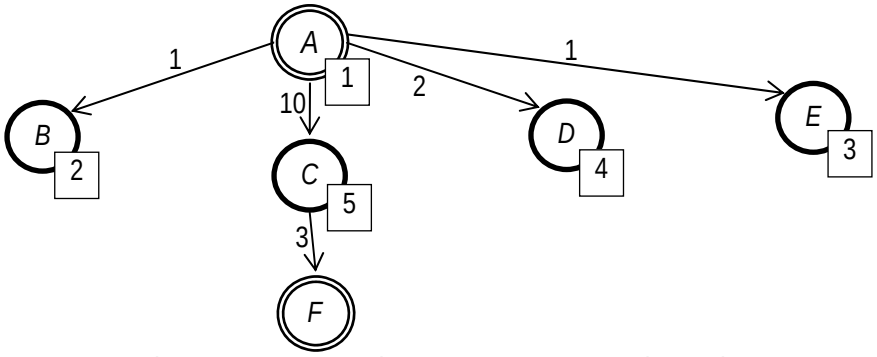


Figura 2.4: Orden de expansión de nodos para el árbol de la figura 2.1 según la búsqueda de coste uniforme. Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto, el algoritmo termina en ese momento sin que se lleguen a generar sus hijos.

4. Búsqueda en Anchura Iterativa (de derecha a izquierda)

Este algoritmo ejecuta iterativamente varias búsquedas primero en anchura, de manera que entre iteración e iteración se incrementa en una unidad el número máximo de hijos que se generan en cada expansión de un nodo padre. Al principio (en la primera iteración), únicamente un hijo es generado en cada expansión. En las figuras 2.5a, 2.5b y 2.5c se muestran los órdenes de expansión de nodos para cada una de las iteraciones de búsqueda primero en anchura que son necesarias para este ejemplo de búsqueda en anchura iterativa.

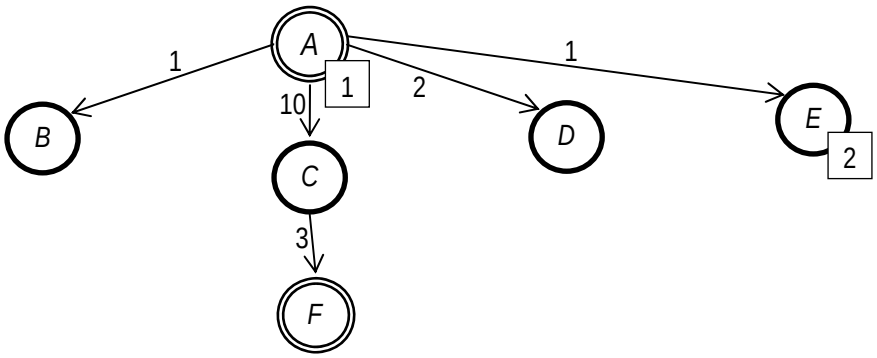


Figura 2.5a: Primera iteración de la búsqueda en anchura iterativa de derecha a izquierda, en la que como máximo un hijo es generado en cada expansión de un nodo padre.

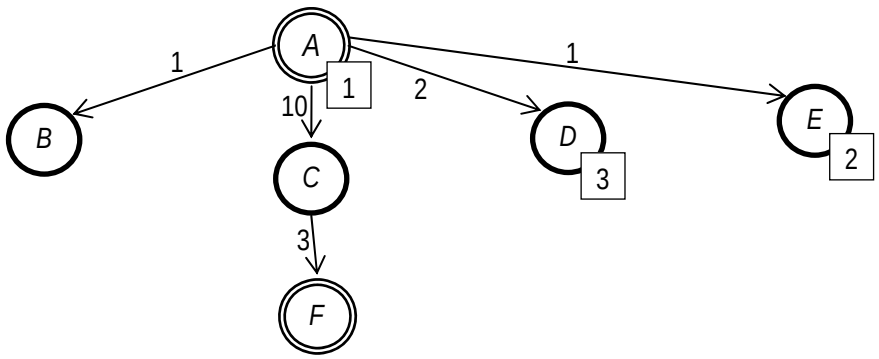


Figura 2.5b: Segunda iteración de la búsqueda en anchura iterativa de derecha a izquierda, en la que como máximo dos hijos son generados en cada expansión de un nodo padre.

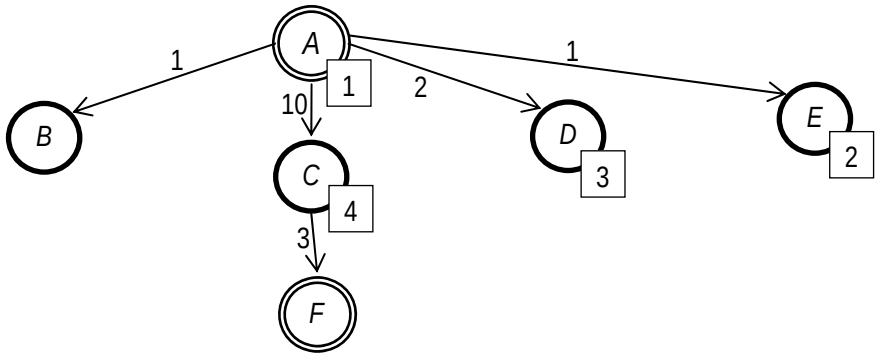


Figura 2.5c: Tercera iteración de la búsqueda en anchura iterativa de derecha a izquierda, en la que como máximo tres hijos son generados en cada expansión de un nodo padre.

Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto, el algoritmo termina en ese momento sin que se lleguen a generar sus hijos.

5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

Este algoritmo ejecuta iterativamente varias búsquedas primero en profundidad, de manera que entre iteración e iteración se incrementa en una unidad la profundidad límite. Al principio (en la primera iteración), la profundidad límite es igual a 1, así que sólo se podrá expandir el nodo inicial, cuya profundidad es igual a 0. En las figuras 2.6a y 2.6b se muestran los órdenes de expansión de nodos para las dos iteraciones de búsqueda primero en profundidad que son necesarias para este ejemplo de búsqueda en profundidad iterativa.

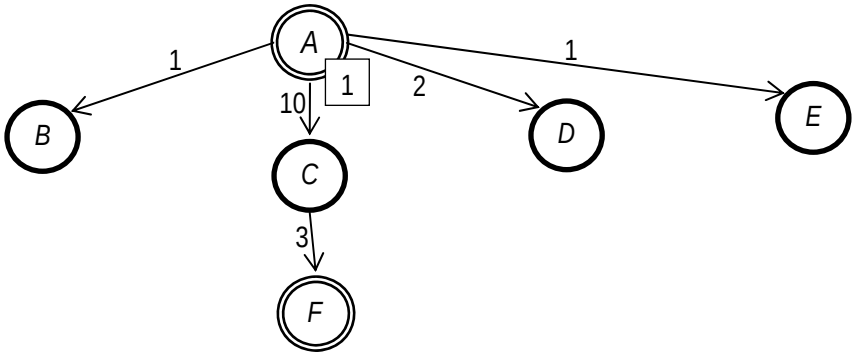


Figura 2.6a: Primera iteración de la búsqueda en profundidad iterativa de izquierda a derecha, en la que la profundidad límite es igual a 1 y únicamente son expandidos nodos con una profundidad máxima de 0.

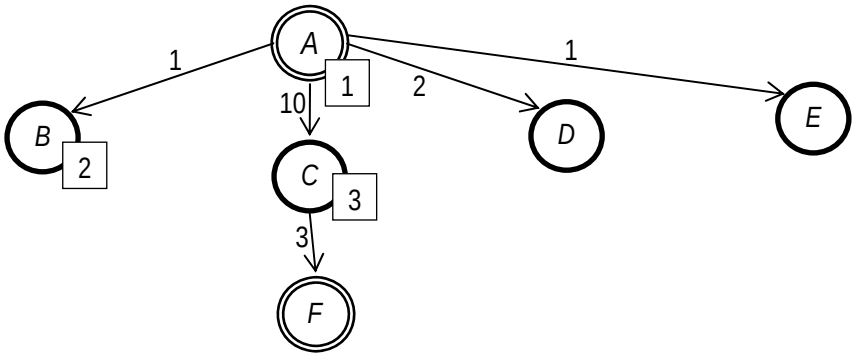


Figura 2.6b: Segunda iteración de la búsqueda en profundidad iterativa de izquierda a derecha, en la que la profundidad límite es igual a 2 y únicamente son expandidos nodos con una profundidad máxima de 1.

Observe que realmente el nodo meta no es expandido en esta última iteración porque se encuentre a la profundidad límite. Esta condición no se llega a comprobar, ya que justo antes se averigua que el nodo es meta y, por tanto, el algoritmo termina.

EJERCICIO 3:

Considere el espacio de búsqueda de la figura 3.1, que tiene forma de árbol, donde el nodo raíz del árbol es el nodo inicial, existe un único nodo meta y cada operador tiene asociado un coste. Describa cuál es el contenido de ABIERTA, previamente a cada extracción de un nodo de la misma, a partir de cada uno de los métodos siguientes de búsqueda sin información del dominio:

1. Búsqueda Primero en Anchura (de izquierda a derecha)
2. Búsqueda Primero en Profundidad (de derecha a izquierda)
3. Búsqueda de Coste Uniforme
4. Búsqueda en Anchura Iterativa (de derecha a izquierda)
5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

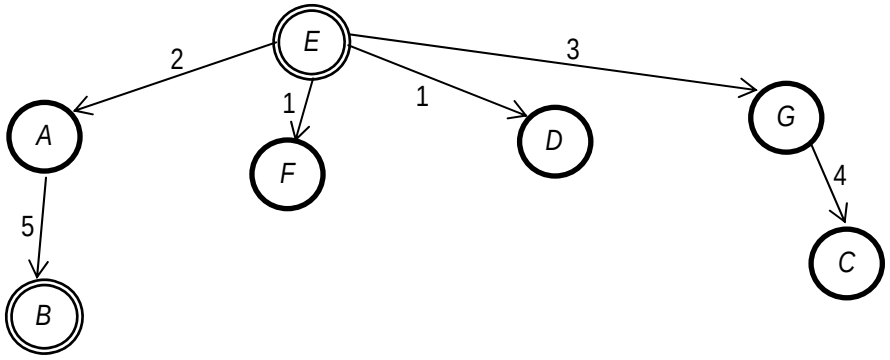


Figura 3.1: Árbol de búsqueda en el que el nodo inicial es *E*, el nodo meta es *B* y el coste de cada operador aparece al lado del arco que lo representa.

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 3:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- Cada uno de los cinco apartados, correspondiente a un método de búsqueda concreto, se puntúa sobre 2 puntos.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no gestiona ABIERTA en la forma debida: la puntuación del apartado bajaría 1.6 puntos si la respuesta dada varía significativamente de la correcta, mientras que si la respuesta dada varía de la correcta como consecuencia de un despiste no conceptual entonces la puntuación del apartado bajaría sólo 0.4 puntos en vez de los 1.6 puntos mencionados.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no finaliza cuando es debido, la puntuación del apartado bajaría 0.4 puntos. (Esto es independiente de la gestión de ABIERTA dada como respuesta, que ya ha sido valorada anteriormente con un máximo de 1.6 puntos.)

SOLUCIÓN DEL EJERCICIO 3 (por Severino Fernández Galán):

1. Búsqueda Primero en Anchura (de izquierda a derecha)

Este algoritmo usa ABIERTA como una cola, de manera que siempre se saca el primer nodo de la cola y se introducen sus hijos al final de la misma. El contenido de ABIERTA previamente a cada extracción de un nodo es el siguiente:

Antes de sacar *E*: {*E*}
Antes de sacar *A*: {*A, F, D, G*}
Antes de sacar *F*: {*F, D, G, B*}
Antes de sacar *D*: {*D, G, B*}
Antes de sacar *G*: {*G, B*}
Antes de sacar *B*: {*B, C*}
META ENCONTRADA

Observe que, tras cada expansión del nodo sacado de ABIERTA, sus nodos hijos más a la izquierda se introducen en ABIERTA antes que los situados más a la derecha.

2. Búsqueda Primero en Profundidad (de derecha a izquierda)

Este algoritmo usa ABIERTA como una pila, de manera que siempre se saca el primer nodo de la pila y se introducen sus hijos al principio de la misma. El contenido de ABIERTA previamente a cada extracción de un nodo es el siguiente:

Antes de sacar *E*: {*E*}
Antes de sacar *G*: {*G, D, F, A*}
Antes de sacar *C*: {*C, D, F, A*}
Antes de sacar *D*: {*D, F, A*}
Antes de sacar *F*: {*F, A*}
Antes de sacar *A*: {*A*}
Antes de sacar *B*: {*B*}
META ENCONTRADA

Observe que, tras cada expansión del nodo sacado de ABIERTA, sus nodos hijos más a la derecha se introducen en ABIERTA después que los situados más a la izquierda.

3. Búsqueda de Coste Uniforme

Este algoritmo mantiene ABIERTA ordenada según el coste del camino desde cada nodo al nodo inicial. En este ejercicio desharemos de forma arbitraria los posibles empates que surjan al determinar qué nodo se debe sacar de ABIERTA. El contenido de ABIERTA previamente a cada extracción de un nodo es el siguiente:

Antes de sacar *E*: {*E*(0)}
Antes de sacar *F*: {*F*(1), *D*(1), *A*(2), *G*(3)}
Antes de sacar *D*: {*D*(1), *A*(2), *G*(3)}
Antes de sacar *A*: {*A*(2), *G*(3)}
Antes de sacar *G*: {*G*(3), *B*(7)}
Antes de sacar *C*: {*C*(7), *B*(7)}
Antes de sacar *B*: {*B*(7)}
META ENCONTRADA

Observe que, tras cada expansión de un nodo, sus nodos hijos son introducidos en ABIERTA, donde todos los nodos quedan siempre ordenados por coste creciente. Para facilitar el seguimiento del algoritmo, entre paréntesis y al lado de cada nodo de ABIERTA se especifica el coste del camino para ir desde dicho nodo hasta el nodo inicial.

4. Búsqueda en Anchura Iterativa (de derecha a izquierda)

Debido a que este algoritmo es una iteración de la búsqueda primero en anchura, los dos gestionan ABIERTA del mismo modo, es decir, como una cola. La diferencia reside en que en la búsqueda en anchura iterativa en cada iteración se fija un número máximo de hijos que pueden ser generados al expandir un nodo padre. Este número empieza valiendo 1 y es incrementado en una unidad al principio de cada nueva iteración.

Iteración 1 (cada expansión de un nodo padre genera como máximo un hijo):

Antes de sacar *E*: {*E*}
Antes de sacar *G*: {*G*}
Antes de sacar *C*: {*C*}

Iteración 2 (cada expansión de un nodo padre genera como máximo dos hijos):

Antes de sacar *E*: {*E*}
Antes de sacar *G*: {*G, D*}
Antes de sacar *D*: {*D, C*}
Antes de sacar *C*: {*C*}

Iteración 3 (cada expansión de un nodo padre genera como máximo tres hijos):

Antes de sacar *E*: {*E*}
Antes de sacar *G*: {*G, D, F*}
Antes de sacar *D*: {*D, F, C*}
Antes de sacar *F*: {*F*, *C*}
Antes de sacar *C*: {*C*}

Iteración 4 (cada expansión de un nodo padre genera como máximo cuatro hijos):

Antes de sacar *E*: {*E*}
Antes de sacar *G*: {*G, D, F, A*}
Antes de sacar *D*: {*D, F, A, C*}
Antes de sacar *F*: {*F*, *A*, *C*}
Antes de sacar *A*: {*A*, *C*}
Antes de sacar *C*: {*C*, *B*}
Antes de sacar *B*: {*B*}
META ENCONTRADA

5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

Debido a que este algoritmo es una iteración de la búsqueda primero en profundidad, los dos gestionan ABIERTA del mismo modo, es decir, como una pila. La diferencia reside en que en la búsqueda en profundidad iterativa en cada iteración se fija una profundidad límite propia. Este número empieza valiendo 1, lo cual permite expandir únicamente el nodo inicial, y es incrementado en una unidad al principio de cada nueva iteración.

Iteración 1 (profundidad límite igual a 1):

Antes de sacar *E*: {*E*}
Antes de sacar *A*: {*A, F, D, G*}
Nótese que *A* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *F*: {*F*, *D, G*}
Nótese que *F* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *D*: {*D, G*}
Nótese que *D* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *G*: {*G*}
Nótese que *G* no es expandido por coincidir su profundidad con la profundidad límite.

Iteración 2 (profundidad límite igual a 2):

Antes de sacar *E*: {*E*}
Antes de sacar *A*: {*A*, *F*, *D*, *G*}
Antes de sacar *B*: {*B*, *F*, *D*, *G*}
META ENCONTRADA

EJERCICIO 4:

Considere el espacio de búsqueda de la figura 4.1, que tiene forma de árbol, donde el nodo raíz del árbol es el nodo inicial, existe un único nodo meta y cada operador tiene asociado un coste. Describa cuál es el contenido de TABLA_A, posteriormente a cada expansión de un nodo, a partir de cada uno de los métodos siguientes de búsqueda sin información del dominio:

1. Búsqueda Primero en Anchura (de izquierda a derecha)
2. Búsqueda Primero en Profundidad (de derecha a izquierda)
3. Búsqueda de Coste Uniforme
4. Búsqueda en Anchura Iterativa (de derecha a izquierda)
5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

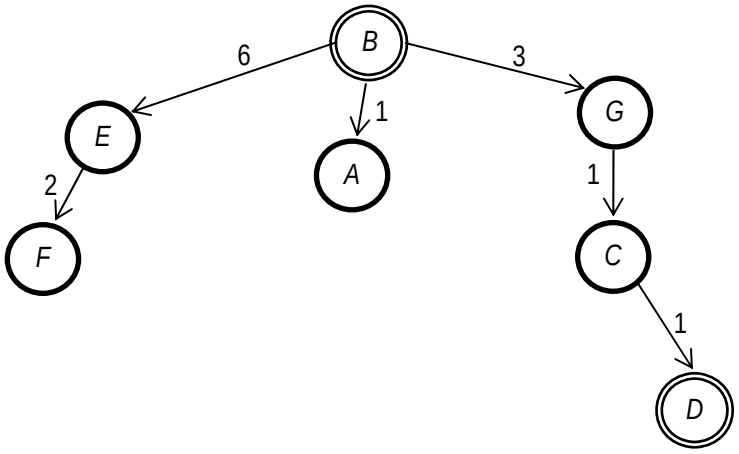


Figura 4.1: Árbol de búsqueda en el que el nodo inicial es *B*, el nodo meta es *D* y el coste de cada operador aparece al lado del arco que lo representa.

Para cada nodo de TABLA_A incluya la siguiente información: su nodo padre y el coste al nodo inicial. (En este ejercicio no es necesario incluir en TABLA_A información sobre los hijos de cada nodo expandido, ya que sólo existe un camino desde cada nodo al nodo inicial y, por tanto, el mejor camino desde cada nodo al nodo inicial no cambia a lo largo del proceso de búsqueda.)

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 4:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

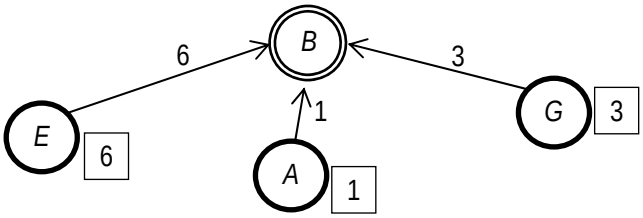
- Cada uno de los cinco apartados, correspondiente a un método de búsqueda concreto, se puntúa sobre 2 puntos.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no gestiona TABLA_A en la forma debida: la puntuación del apartado bajaría 1.6 puntos si la respuesta dada varía significativamente de la correcta, mientras que si la respuesta dada varía de la correcta como consecuencia de un despiste no conceptual entonces la puntuación del apartado bajaría sólo 0.4 puntos en vez de los 1.6 puntos mencionados.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no finaliza cuando es debido, la puntuación del apartado bajaría 0.4 puntos. (Esto es independiente de la gestión de TABLA_A dada como respuesta, que ya ha sido valorada anteriormente con un máximo de 1.6 puntos.)

SOLUCIÓN DEL EJERCICIO 4 (por Severino Fernández Galán):

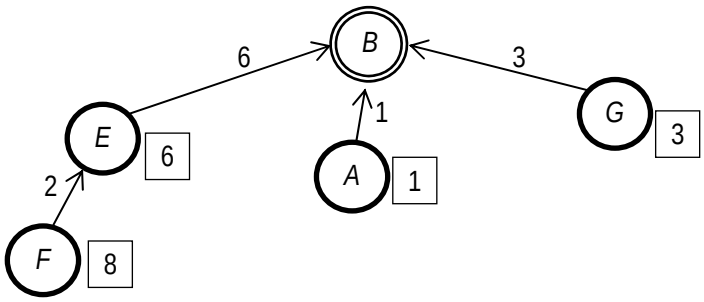
De cara a ilustrar la respuesta convenientemente, incluimos la información de TABLA_A gráficamente: por un lado, para cada nodo generado en una expansión trazamos un arco ascendente a su padre y, por otro lado, indicamos el coste al nodo inicial al lado de cada nodo generado.

1. Búsqueda Primero en Anchura (de izquierda a derecha)

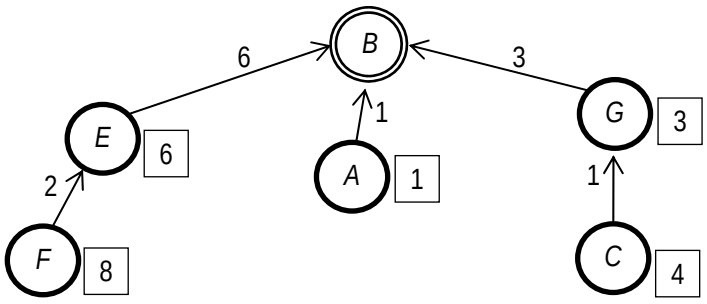
- Inicialmente, *B* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *B*. A continuación expandimos el nodo inicial, generando sus nodos hijos. Desde cada nodo hijo trazamos un nodo ascendente a su nodo padre. Al lado de cada nodo hijo indicamos el coste desde dicho nodo al nodo inicial.



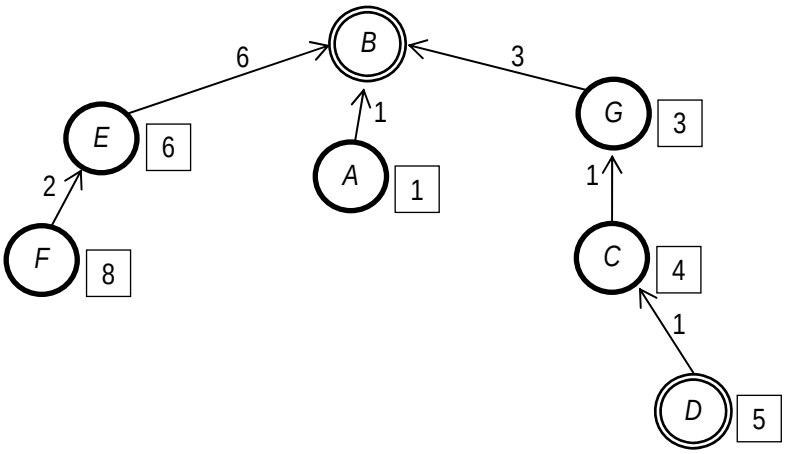
- Expandimos *E*:



- Expandimos *A* y TABLA_A no varía. A continuación expandimos *G*:



- Expandimos *F* y TABLA_A no varía. A continuación expandimos *C*:

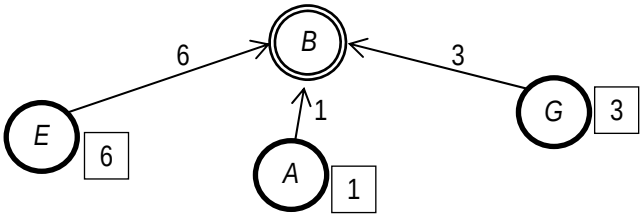


- A continuación elegiríamos *D* para su expansión y llegaríamos a la meta. El camino hallado hasta la meta tiene coste 5.

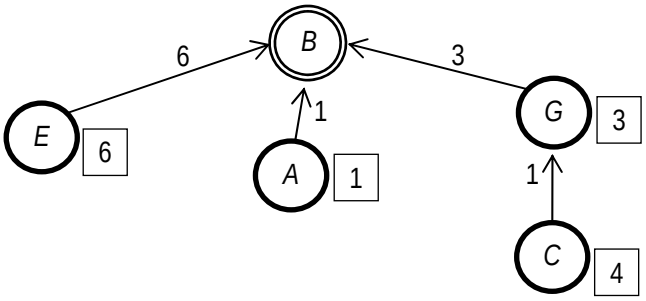
2. Búsqueda Primero en Profundidad (de derecha a izquierda)

Cuando sea necesario, en este algoritmo aplicaremos la función LimpiarTABLA_A (véase texto base, página 318). Tras la extracción de un nodo de ABIERTA y su posible expansión, dicha función elimina aquellos nodos que ya no son necesarios en el proceso de búsqueda de la meta. Esto permite preservar la linealidad de la complejidad espacial de este algoritmo con respecto a la profundidad de la solución.

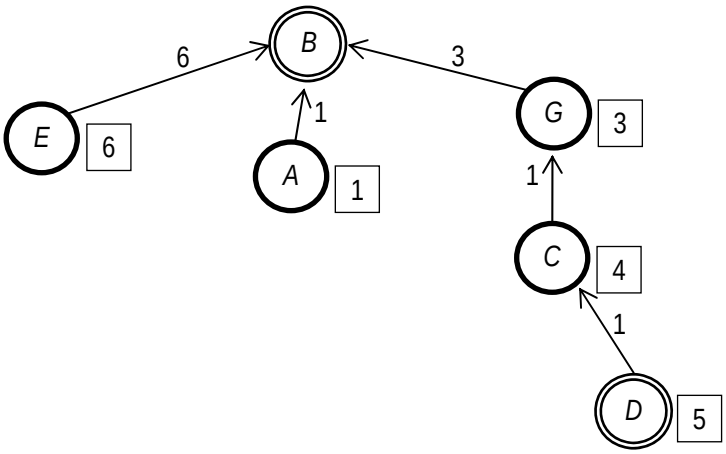
- Inicialmente, *B* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *B*. A continuación, expandimos el nodo inicial.



- Expandimos *G*:



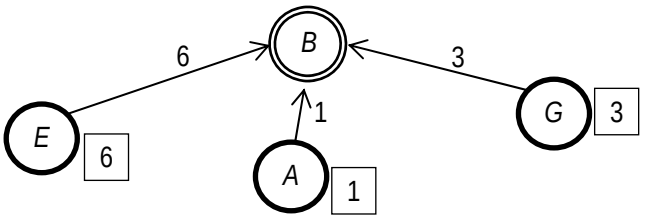
- Expandimos *C*:



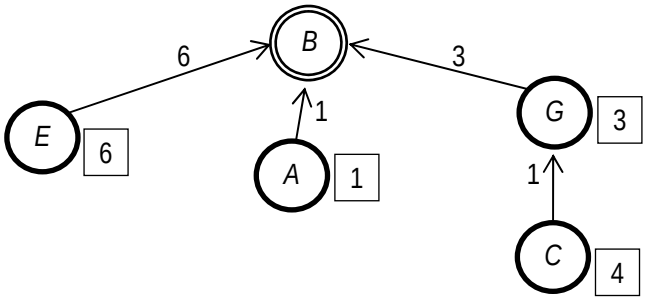
- A continuación elegiríamos *D* para su expansión y llegaríamos a la meta. El camino hallado hasta la meta tiene coste 5. Nótese que finalmente no ha sido necesario aplicar la función LimpiarTABLA_A a lo largo de este apartado.

3. Búsqueda de Coste Uniforme

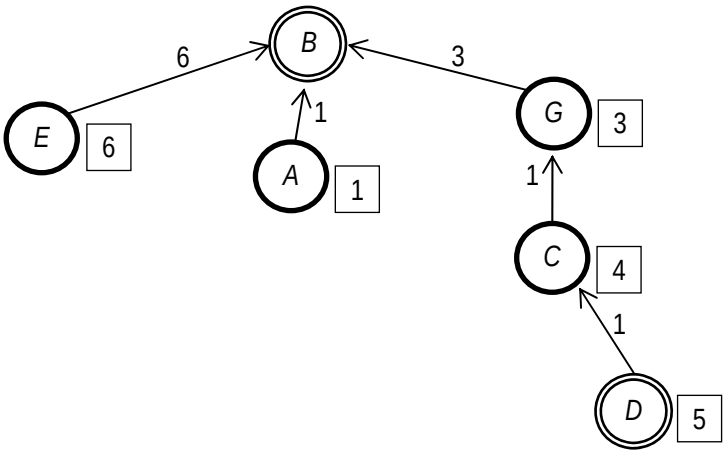
- Inicialmente, *B* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *B*. A continuación, expandimos el nodo inicial.



- Expandimos *A*, con lo que TABLA_A no cambia. A continuación expandimos *G*:



- Expandimos *C*:

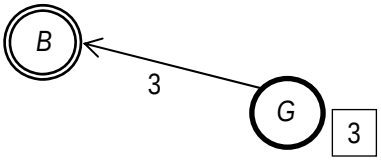


- Al intentar expandir *D*, alcanzamos la meta. El coste del camino solución hallado es 5.

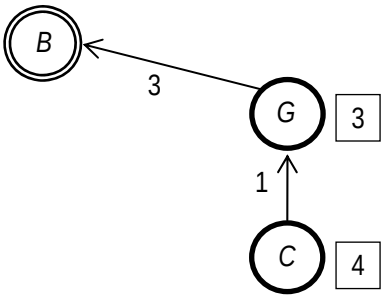
4. Búsqueda en Anchura Iterativa (de derecha a izquierda)

Iteración 1 (se genera como máximo 1 nodo hijo en cada expansión de un nodo padre):

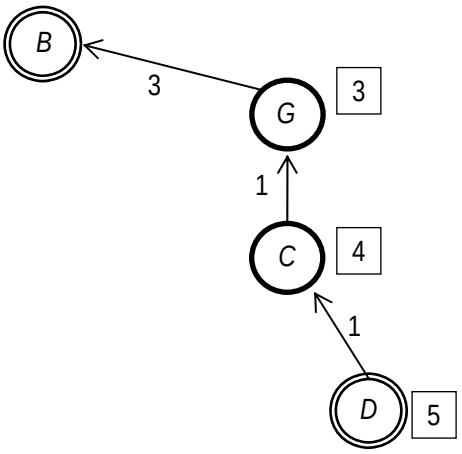
- Inicialmente, *B* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *B*. A continuación, expandimos el nodo inicial:



- Expandimos *G*:



- Expandimos *C*:

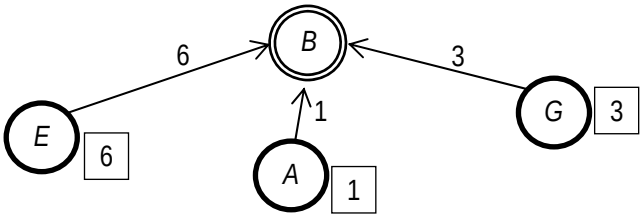


- Finalmente, al intentar expandir *D* alcanzamos la meta. El camino encontrado hasta el nodo inicial tiene coste 5.

5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

Iteración 1 (profundidad límite igual a 1):

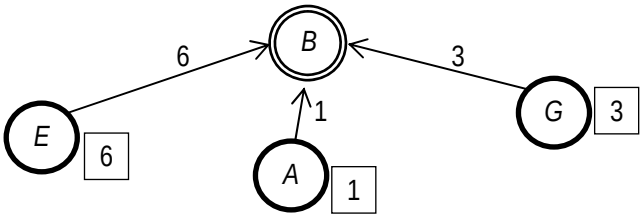
- Inicialmente, *B* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *B*. A continuación, expandimos el nodo inicial:



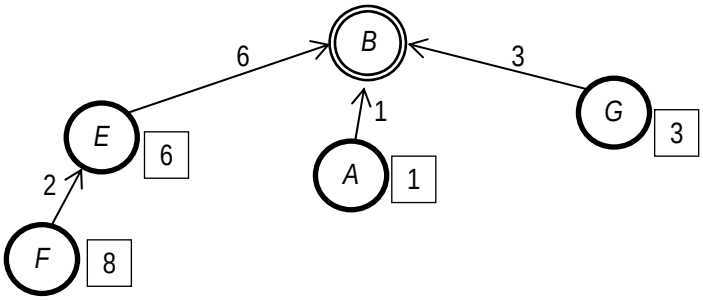
- Tras comprobar que *E* no es nodo meta, se le aplica la función LimpiarTABLA_A por estar en la profundidad límite y es sacado de TABLA_A. De igual manera, tras comprobar respectivamente que *A* y *G* no son nodo meta, se les aplica la función LimpiarTABLA_A por estar en la profundidad límite y son sacados de TABLA_A. Seguidamente, tras aplicarle la función LimpiarTABLA_A, *B* es sacado de TABLA_A por no tener hijos en ABIERTA. A continuación pasamos a la siguiente iteración.

Iteración 2 (profundidad límite igual a 2):

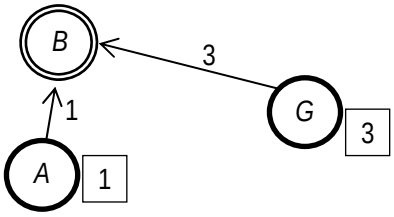
- Inicialmente, *B* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *B*. A continuación, expandimos el nodo inicial:



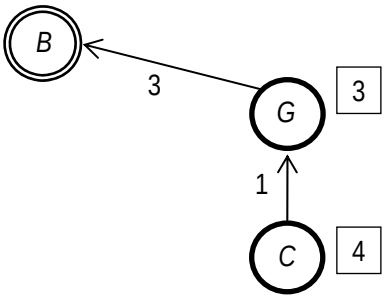
- Expandimos *E*:



- Tras comprobar que *F* no es nodo meta, se le aplica la función LimpiarTABLA_A por encontrarse en la profundidad límite y es sacado de TABLA_A. Del mismo modo, tras comprobar que *E* no tiene hijos en ABIERTA, se le aplica la función LimpiarTABLA_A y es sacado de TABLA_A.



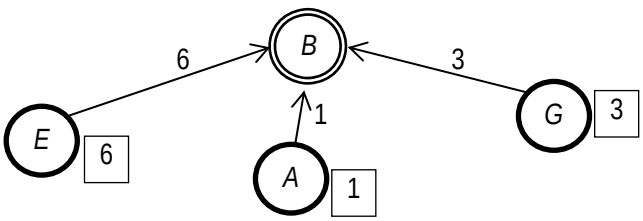
- Expandimos *A*. Seguidamente, aplicamos la función LimpiarTABLA_A a *A* por no tener hijos en ABIERTA y es sacado de TABLA_A. A continuación, expandimos *G*:



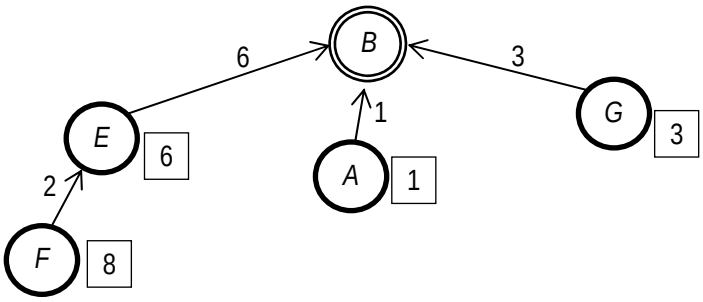
- Tras comprobar que *C* no es nodo meta, se le aplica la función LimpiarTABLA_A por estar en la profundidad límite y es sacado de TABLA_A. Tras aplicarle la función LimpiarTABLA_A, *G* es sacado de TABLA_A por no tener hijos en ABIERTA. Lo mismo ocurre posteriormente con *B*, por lo que pasamos a la siguiente iteración.

<u>Iteración 3 (profundidad límite igual a 3)</u>:

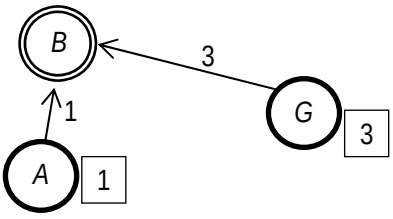
- Inicialmente, *B* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *B*. A continuación, expandimos el nodo inicial:



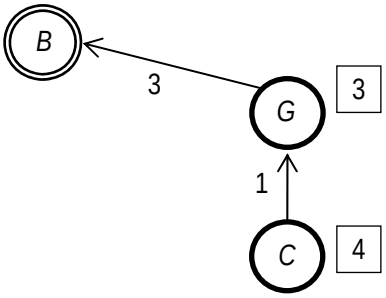
- Expandimos *E*:



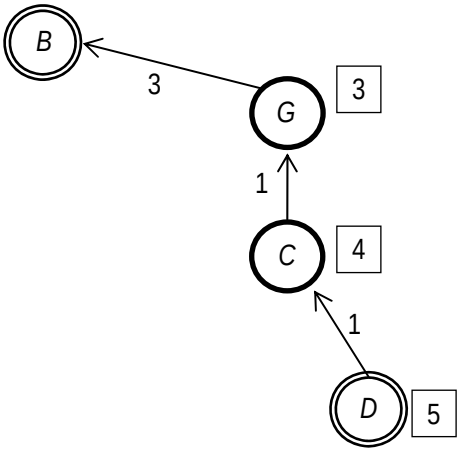
- Expandimos *F*. Seguidamente, aplicamos la función LimpiarTABLA_A a *F* por no tener hijos en ABIERTA y es sacado de TABLA_A. Del mismo modo, tras comprobar que *E* no tiene hijos en ABIERTA, se le aplica la función LimpiarTABLA_A y es sacado de TABLA_A.



- Expandimos *A*. Seguidamente, aplicamos la función LimpiarTABLA_A a *A* por no tener hijos en ABIERTA y es sacado de TABLA_A. A continuación, expandimos *G*:



- Expandimos C:



- Tras elegir *D* para su expansión y comprobar que es nodo meta, finaliza el algoritmo habiendo hallado un camino entre el nodo inicial y el nodo meta de coste 5.

EJERCICIO 5:

Considere el grafo de la figura 5.1, donde el nodo inicial es n_1 y donde el nodo meta es n_9. Cada arco u operador lleva asociado su coste y en cada nodo aparece la estimación de la menor distancia desde ese nodo a una meta. Aplique paso a paso el algoritmo A* al grafo dado, indicando de forma razonada la siguiente información en cada paso del algoritmo:

1. Qué nodo es expandido.
2. Cuál es el contenido de ABIERTA tras la expansión del nodo, indicando el valor de la función de evaluación heurística para cada nodo de ABIERTA.
3. Cuál es el contenido de TABLA_A tras la expansión del nodo. Para cada nodo de TABLA_A incluya la siguiente información:
    a) Su nodo padre que indique el camino de menor coste hasta el nodo inicial encontrado hasta el momento
    b) El coste del camino de menor coste hasta el nodo inicial encontrado hasta el momento
    c) Sus nodos hijos (si el nodo de TABLA_A actual ya ha sido expandido)

Por último, ¿cuál es el camino solución hallado y su coste?

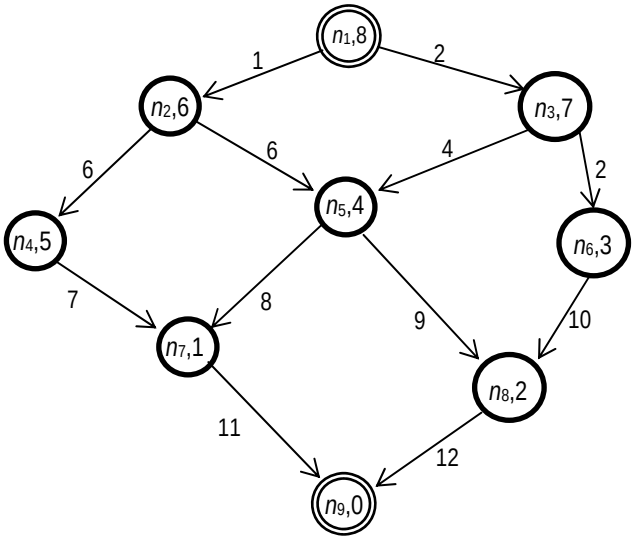


Figura 5.1: Grafo de búsqueda en el que el nodo inicial es n_1, el nodo meta es n_9, el coste de cada operador aparece al lado del arco que lo representa y al lado de cada nodo aparece la estimación de la menor distancia desde ese nodo a una meta.

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 5:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- El orden seguido en la expansión de los nodos se puntúa sobre 1.5 puntos.
- La forma en que se gestiona ABIERTA se puntúa sobre 3.75 puntos. Se hará especial énfasis en comprobar qué nodos hay en ABIERTA en cada paso del algoritmo y qué valores de la función de evaluación heurística se les asignan.
- La forma en que se gestiona TABLA_A se puntúa sobre 3.75 puntos. Se hará especial énfasis en comprobar qué nodos hay en TABLA_A en cada paso del algoritmo y qué padre mejor se les asigna (teniendo en cuenta las posibles reorientaciones o rectificaciones de enlaces)
- La correcta terminación del algoritmo se puntúa sobre 1 punto. Se hará especial énfasis en comprobar cuándo termina el algoritmo y qué camino solución devuelve.

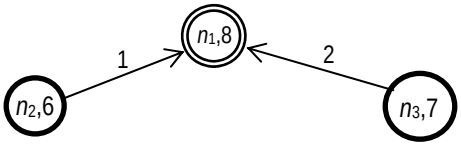
SOLUCIÓN DEL EJERCICIO 5 (por Severino Fernández Galán):

De cara a ilustrar la respuesta convenientemente, incluimos la información pedida sobre TABLA_A gráficamente. Para ello es necesario trazar, para cada nodo generado en una expansión, un arco ascendente a su padre expandido; además, hay que anotar para cada nodo su mejor padre encontrado hasta el momento (punto 3a del enunciado). De esta manera, siguiendo cada arco al mejor padre, se puede saber cuál es el mejor camino encontrado hasta el momento desde cada nodo al nodo inicial (punto 3b del enunciado). Además, los arcos ascendentes que llegan a un nodo ya expandido lo enlazan a sus nodos hijos (punto 3c del enunciado).

- PASO 0. El nodo inicial n_1 es introducido en ABIERTA y en TABLA_A, con lo que tenemos la siguiente situación:

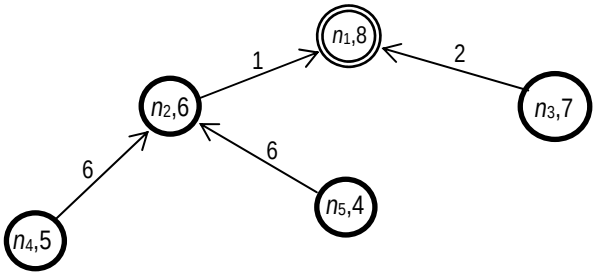
ABIERTA = {$\underline{n_1(0+8)}$}

- PASO 1. Expandimos el nodo n_1 de ABIERTA. Tras la expansión, la situación es la siguiente:



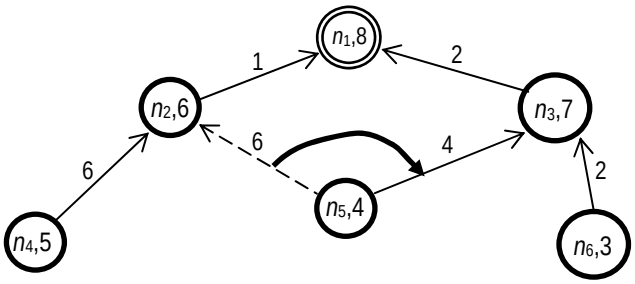
ABIERTA = {$\underline{n_2(1+6)}$, $n_3(2+7)$}

- PASO 2. Expandimos n_2 por ser el nodo de ABIERTA con menor valor de la función de evaluación heurística, $f=g+h$ (al ser $g=1$ y $h=6$).



ABIERTA = {$\underline{n_3(2+7)}$, $n_5(7+4)$, $n_4(7+5)$}

- PASO 3. Expandimos n_3.

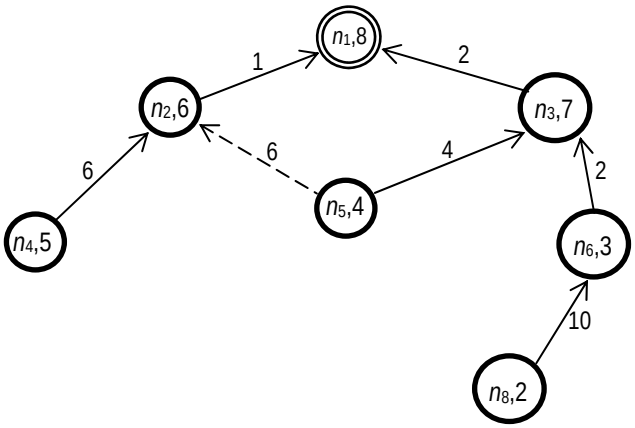


ABIERTA = {$\underline{n_6(4+3)}$, $n_5(6+4)$, $n_4(7+5)$}

Observe que el mejor camino desde n_5 al nodo inicial lo marca su padre n_3 (coste 4+2=6) y no su padre n_2 (coste 6+1=7). Por ello, el arco ascendente de n_5 a n_3 se marca con trazo continuo y el arco ascendente de n_5 a n_2 se marca con trazo discontinuo. Es importante darse cuenta que el conjunto de arcos con trazo continuo formará siempre un árbol en el grafo parcial de búsqueda desarrollado hasta el momento.

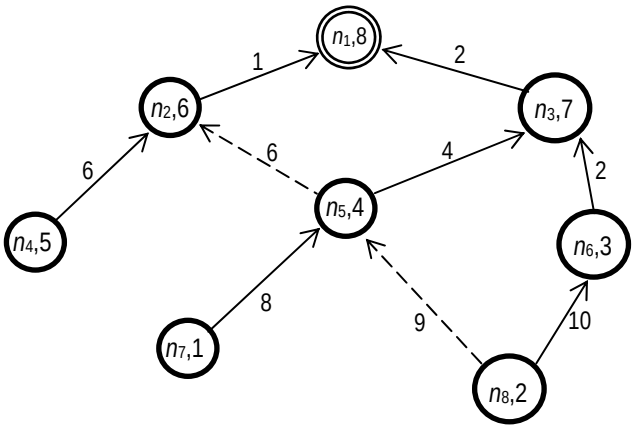
Observe también que ha habido una reorientación del mejor padre de n_5, que antes era n_2 y ahora pasa a ser n_3. El nuevo mejor coste de n_5 al nodo inicial, $g(n_5)$, es ahora 4+2=6, que se puede calcular a partir de los costes de los mejores arcos ascendentes hallados hasta el momento: $n_5 \rightarrow n_3$ y $n_3 \rightarrow n_1$.

- PASO 4. Expandimos n_6.



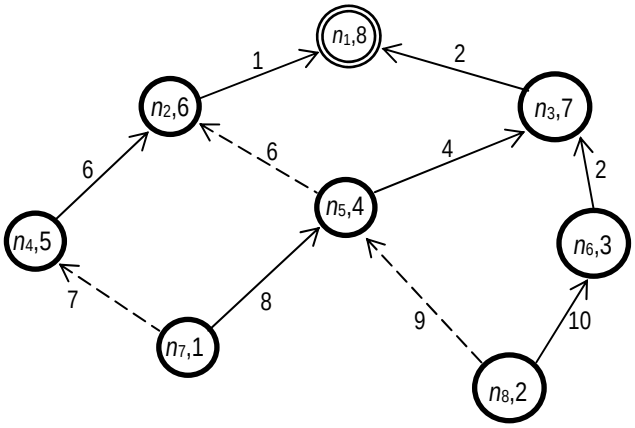
ABIERTA = {$\underline{n_5(6+4)}$, $n_4(7+5)$, $n_8(14+2)$}

- PASO 5. Expandimos n_5.



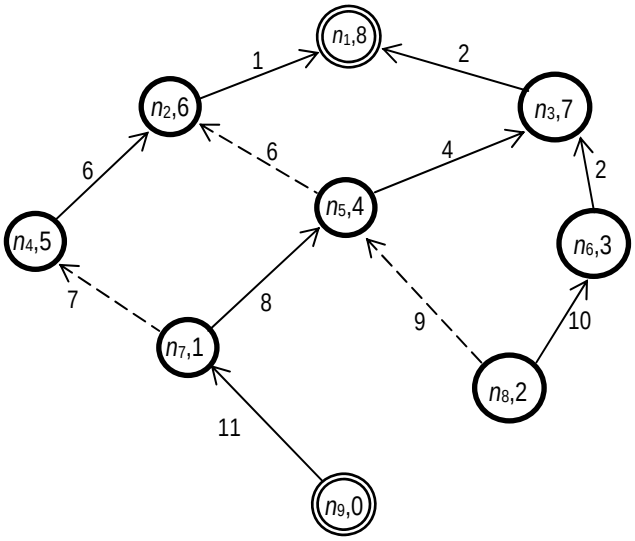
ABIERTA = {$\underline{n_4(7+5)}$, $n_7(14+1)$, $n_8(14+2)$}

- PASO 6. Expandimos n_4.



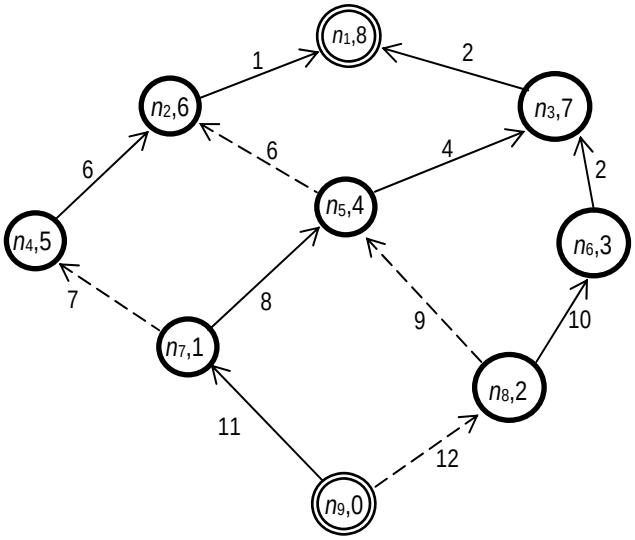
ABIERTA = {$\underline{n_7(14+1)}$, $n_8(14+2)$}

- PASO 7. Expandimos n_7.



ABIERTA = {$\underline{n_8(14+2)}$, $n_9(25+0)$}

- PASO 8. Expandimos n_8.



ABIERTA = {$\underline{n_9(25+0)}$}

- PASO 9. Intentamos expandir n_9 y alcanzamos una meta, con lo que el algoritmo termina. El camino solución encontrado es: $n_1 \rightarrow n_3 \rightarrow n_5 \rightarrow n_7 \rightarrow n_9$, cuyo coste es 25.

EJERCICIO 6:

Considere el grafo de la figura 6.1, donde el nodo inicial es *D* y donde los nodos meta son desconocidos. Cada arco u operador lleva asociado su coste y en cada nodo aparece su valor de la función de evaluación heurística (que hay que minimizar). Aplique paso a paso el algoritmo de escalada o máximo gradiente al grafo dado. Para ello indique de forma razonada qué nodo se expande en cada paso y cuál es el nodo final devuelto por el algoritmo. Utilice como *criterio de selección* el de mejor vecino. Utilice como *criterio de terminación* el que no se hayan producido mejoras durante los cinco últimos pasos del algoritmo.

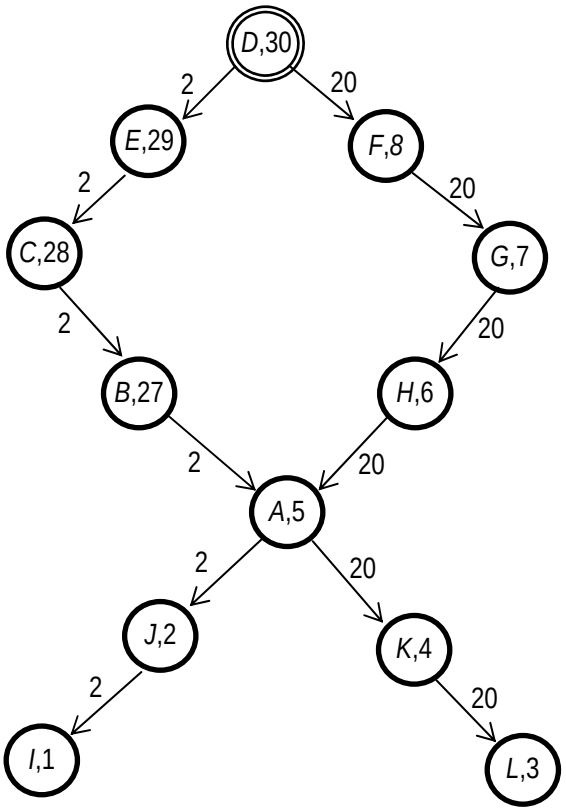


Figura 6.1: Grafo de búsqueda en el que el nodo inicial es *D*, los nodos meta son desconocidos, el coste de cada operador aparece al lado del arco que lo representa y al lado de cada nodo aparece el valor de su función de evaluación heurística (que hay que minimizar).

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 6:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- La correcta aplicación en cada paso del algoritmo del *criterio de selección* del vecino o hijo del nodo actual (qué vecino o hijo del nodo actual es considerado como candidato para sustituirlo) se puntúa sobre 4.5 puntos.
- La correcta aplicación en cada paso del algoritmo del *criterio de aceptación* del vecino o hijo seleccionado (si el vecino o hijo candidato sustituye o no finalmente al nodo actual) se puntúa sobre 4.5 puntos.
- La correcta aplicación del *criterio de finalización* del algoritmo se puntúa sobre 1 punto.

SOLUCIÓN DEL EJERCICIO 6 (por Severino Fernández Galán):

- PASO 1: Al principio expandimos el nodo inicial *D*, generando sus nodos hijos {*E*(29), *F*(8)}. Seleccionamos el nodo *F* por ser el mejor de los nodos hijos. A continuación aceptamos el nodo *F* como nuevo nodo actual en sustitución de *D*, debido a que el valor de la función de evaluación heurística de *F* es mejor o igual que el de *D* ($8 \leq 30$).

- PASO 2: Expandimos el nodo actual *F* y generamos sus hijos: {*G*(7)}. Seleccionamos el nodo *G* por ser el mejor de los nodos hijos. Seguidamente aceptamos el nodo *G* como nuevo nodo actual en sustitución de *F*, debido a que el valor de la función de evaluación heurística de *G* es mejor o igual que el de *F* ($7 \leq 8$).

- PASO 3: Expandimos el nodo actual *G* y generamos sus hijos: {*H*(6)}. Seleccionamos el nodo *H* por ser el mejor de los nodos hijos. Seguidamente aceptamos el nodo *H* como nuevo nodo actual en sustitución de *G*, debido a que el valor de la función de evaluación heurística de *H* es mejor o igual que el de *G* ($6 \leq 7$).

- PASO 4: Expandimos el nodo actual *H* y generamos sus hijos: {*A*(5)}. Seleccionamos el nodo *A* por ser el mejor de los nodos hijos. Seguidamente aceptamos el nodo *A* como nuevo nodo actual en sustitución de *H*, debido a que el valor de la función de evaluación heurística de *A* es mejor o igual que el de *H* ($5 \leq 6$).

- PASO 5: Expandimos el nodo actual *A* y generamos sus hijos: {*J*(2), *K*(4)}. Seleccionamos el nodo *J* por ser el mejor de los nodos hijos. Seguidamente aceptamos el nodo *J* como nuevo nodo actual en sustitución de *A*, debido a que el valor de la función de evaluación heurística de *J* es mejor o igual que el de *A* ($2 \leq 5$).

- PASO 6: Expandimos el nodo actual *J* y generamos sus hijos: {*I*(1)}. Seleccionamos el nodo *I* por ser el mejor de los nodos hijos. A continuación aceptamos el nodo *I* como nuevo nodo actual en sustitución de *J*, debido a que el valor de la función de evaluación heurística de *I* es mejor o igual que el de *J* ($1 \leq 2$).

Dado que *I* no tiene sucesores, la búsqueda terminaría. El algoritmo devolvería el nodo *I*(1) como mejor nodo encontrado.

AÑO 2016

EJERCICIO 1:

Represente detalladamente mediante una tabla el espacio de estados (o espacio de búsqueda) completo para el *problema de los cántaros de agua* descrito más adelante. Para ello especifique: el conjunto de todos los estados posibles, el estado inicial, el o los estados meta, los operadores aplicables a cada estado y el coste asociado a cada operador. En el problema de los cántaros de agua se dispone de dos cántaros, uno de 4 litros y otro de 3 litros de capacidad, así como de un grifo para llenar los cántaros. Suponga que se parte de una situación inicial donde los dos cántaros están vacíos y se quiere llegar a una situación final en la que el cántaro de 4 litros esté lleno por la mitad y el de 3 litros esté vacío. Tenga en cuenta que no es posible técnicamente echar agua a un cántaro para que contenga una cantidad arbitraria de litros, por ejemplo, 0.233 litros; sólo es posible: vaciar un cántaro, llenar un cántaro con agua del grifo o, por último, verter agua de un cántaro a otro (hasta que se cumpla cualquiera de las dos condiciones siguientes: que el cántaro desde el que se vierte quede vacío o que el cántaro hacia el que se vierte quede lleno). Por otra parte, dibuje mediante un grafo dirigido el subconjunto del espacio de estados resultante de partir del estado inicial en busca del estado meta. ¿Cuál es la solución menos costosa para este problema, tal como ha sido descrito?

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 1:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- Los estados se especifican correctamente: 2.5 puntos
- Los operadores se especifican correctamente: 2.5 puntos
- Los costes de los operadores se especifican correctamente: 1 punto
- El espacio de búsqueda se dibuja (mediante un grafo dirigido) y se describe (mediante una tabla) correctamente: 3 puntos
- La solución de menor coste se especifica correctamente: 1 punto

SOLUCIÓN DEL EJERCICIO 1 (por Severino Fernández Galán):

Para representar los **estados** posibles utilizaremos la siguiente notación. El estado del problema lo representamos como:

(xy),

donde x expresa la cantidad de agua en litros contenida en el cántaro de 4 litros, mientras que y expresa la cantidad de agua en litros contenida en el cántaro de 3 litros. (Debido a las restricciones técnicas impuestas, tanto x como y serán números enteros tal que $x \in \{0,1,2,3,4\}$ e $y \in \{0,1,2,3\}$.) Por ejemplo, el estado (31) representa la situación en que el cántaro de 4 litros contiene 3 litros, mientras que el cántaro de 3 litros contiene 1 litro. Nótese que existen 5x4 = 20 estados posibles en el problema planteado.

El **estado inicial** descrito en el enunciado se representa como (00). Por otro lado, existe un único **estado meta**: (20).

Consideramos seis **operadores** posibles:

[1] Llenar el cántaro de 4 litros.

Precondición: $x<4$
Resultado: $x=4$

[2] Llenar el cántaro de 3 litros.

Precondición: $y<3$
Resultado: $y=3$

[3] Vaciar el cántaro de 4 litros.

Precondición: $x>0$
Resultado: $x=0$

[4] Vaciar el cántaro de 3 litros.

Precondición: $y>0$
Resultado: $y=0$

[5] Verter agua del cántaro de 4 litros al de 3 litros.

Precondición: $x>0$ e $y<3$
Resultado: $x=0$ o $y=3$ (lo que primero se dé)

[6] Verter agua del cántaro de 3 litros al de 4 litros.

Precondición: $x<4$ e $y>0$
Resultado: $x=4$ o $y=0$ (lo que primero se dé)

Se puede considerar que cada operador tiene **coste** unidad, así que el coste de un camino es el número de arcos que lo componen.

(NOTA: Consideramos también válida la opción de desdoblar los operadores [5] y [6] en dos operadores cada uno: un operador para el caso en que el cántaro desde el que se vierte queda vacío y otro operador para el caso en que el cántaro hacia el que se vierte quede lleno.)

En la tabla 1.1 figuran los estados posibles del sistema, los operadores que se pueden aplicar a cada estado posible y los estados resultantes de aplicar cada operador. Se ha resaltado en color tanto el estado inicial (rojo) como los estados meta (verde). Por otra parte, la figura 1.1 contiene un grafo dirigido con el subconjunto del espacio de estados resultante de partir del estado inicial en busca del estado meta. La **solución menos costosa** para el problema planteado está marcada en verde en la figura 1.1 y tiene un coste de 6.

ESTADOS	OPERADORES					
	[1]	[2]	[3]	[4]	[5]	[6]
(00)	(40)	(03)				
(01)	(41)	(03)		(00)		(10)
(02)	(42)	(03)		(00)		(20)
(03)	(43)			(00)		(30)
(10)	(40)	(13)	(00)		(01)	
(11)	(41)	(13)	(01)	(10)	(02)	(20)
(12)	(42)	(13)	(02)	(10)	(03)	(30)
(13)	(43)		(03)	(10)		(40)
(20)	ESTADO META					
(21)	(41)	(23)	(01)	(20)	(03)	(30)
(22)	(42)	(23)	(02)	(20)	(13)	(40)
(23)	(43)		(03)	(20)		(41)
(30)	(40)	(33)	(00)		(03)	
(31)	(41)	(33)	(01)	(30)	(13)	(40)
(32)	(42)	(33)	(02)	(30)	(23)	(41)
(33)	(43)		(03)	(30)		(42)
(40)		(43)	(00)		(13)	
(41)		(43)	(01)	(40)	(23)	
(42)		(43)	(02)	(40)	(33)	
(43)			(03)	(40)		

Tabla 1.1: Estados posibles, operadores aplicables a cada estado y estados resultantes de aplicar cada operador para el problema de los cántaros de agua planteado en este ejercicio. El estado inicial se resalta en rojo y el estado meta se resalta en verde.

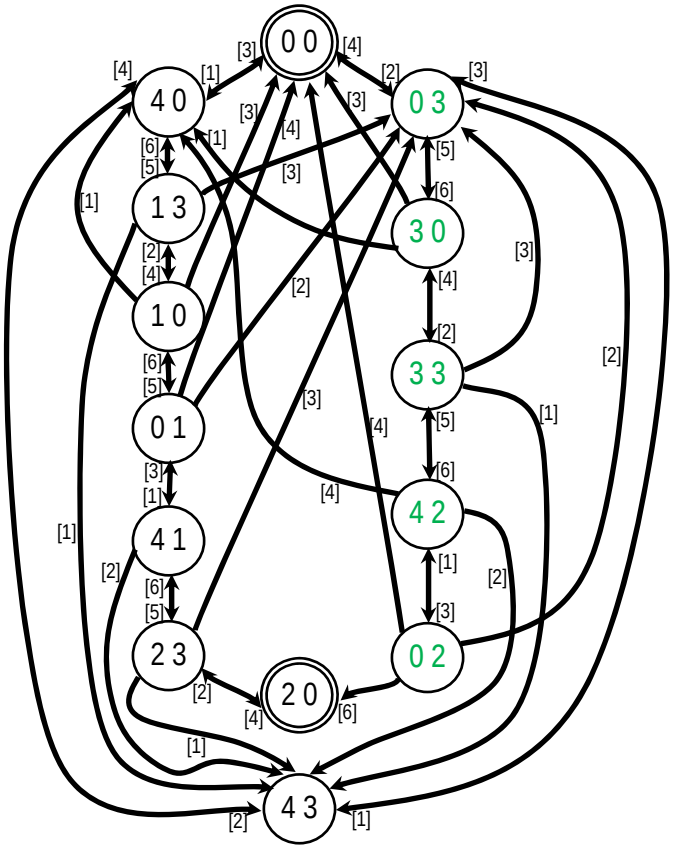


Figura 1.1: Grafo dirigido que constituye el subconjunto del espacio de estados resultante de partir del estado inicial (00) en busca del estado meta (20).

EJERCICIO 2:

Considere el espacio de búsqueda de la figura 2.1, que tiene forma de árbol, donde el nodo raíz del árbol es el nodo inicial, existe un único nodo meta y cada operador tiene asociado un coste. Explique razonadamente en qué orden se expandirían los nodos de dicho árbol de búsqueda a partir de cada uno de los métodos siguientes de búsqueda sin información del dominio:

1. Búsqueda Primero en Anchura (de izquierda a derecha)
2. Búsqueda Primero en Profundidad (de derecha a izquierda)
3. Búsqueda de Coste Uniforme
4. Búsqueda en Anchura Iterativa (de derecha a izquierda)
5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

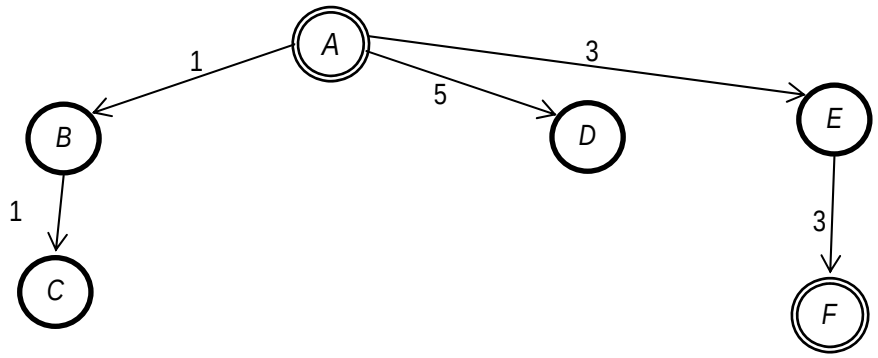


Figura 2.1: Árbol de búsqueda en el que el nodo inicial es *A*, el nodo meta es *F* y el coste de cada operador aparece al lado del arco que lo representa.

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 2:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- Cada uno de los cinco apartados, correspondiente a un método de búsqueda concreto, se puntúa sobre 2 puntos.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no expande los nodos en el orden debido: la puntuación del apartado bajaría 1.6 puntos si el orden dado como respuesta varía significativamente del correcto, mientras que si el orden dado como respuesta varía del correcto como consecuencia de un despiste no conceptual entonces la puntuación del apartado bajaría sólo 0.4 puntos en vez de los 1.6 puntos mencionados.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no finaliza cuando es debido, la puntuación del apartado bajaría 0.4 puntos. (Esto es independiente del orden de expansión de nodos dado como respuesta, que ya ha sido valorado anteriormente con un máximo de 1.6 puntos.)

SOLUCIÓN DEL EJERCICIO 2 (por Severino Fernández Galán):

1. Búsqueda Primero en Anchura (de izquierda a derecha)

Este algoritmo explora el árbol de búsqueda por niveles de profundidad, así que el orden de expansión de los nodos de izquierda a derecha sería el reflejado en la figura 2.2.

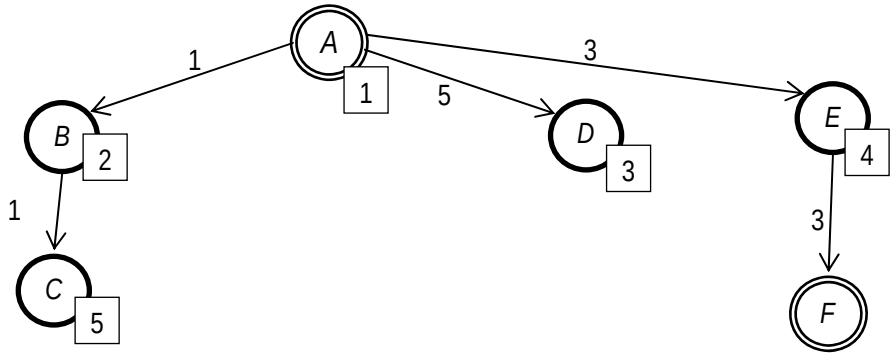


Figura 2.2: Orden de expansión de nodos para el árbol de la figura 2.1 según la búsqueda primero en anchura (de izquierda a derecha). Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto, el algoritmo termina en ese momento sin que se lleguen a generar sus hijos.

2. Búsqueda Primero en Profundidad (de derecha a izquierda)

Este algoritmo explora el árbol de búsqueda bajando de nivel siempre que sea posible. Si no es posible, se sube al nodo más cercano al nodo actual desde el que poder seguir bajando de nivel. El orden de expansión de los nodos de derecha a izquierda según este algoritmo se dibuja en la figura 2.3.

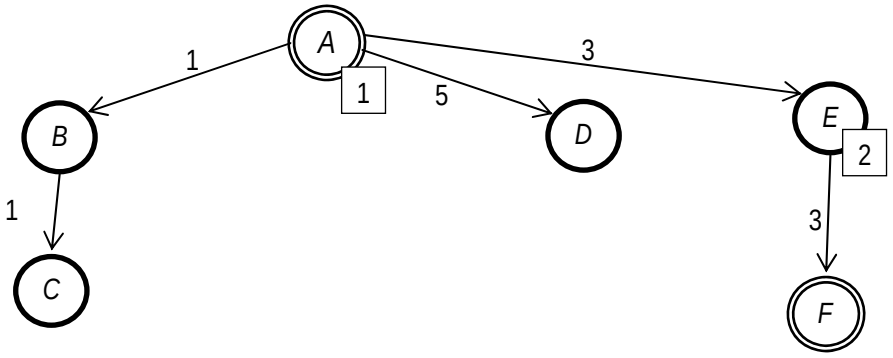


Figura 2.3: Orden de expansión de nodos para el árbol de la figura 2.1 según la búsqueda primero en profundidad (de derecha a izquierda). Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto, el algoritmo termina en ese momento sin que se lleguen a generar sus hijos.

3. Búsqueda de Coste Uniforme

Este algoritmo explora el árbol de búsqueda expandiendo aquel nodo disponible cuyo coste al nodo inicial sea el menor. El orden de expansión de nodos según este criterio se indica en la figura 2.4. Obsérvese que los empates, en caso de haberlos, serían deshechos de forma arbitraria.

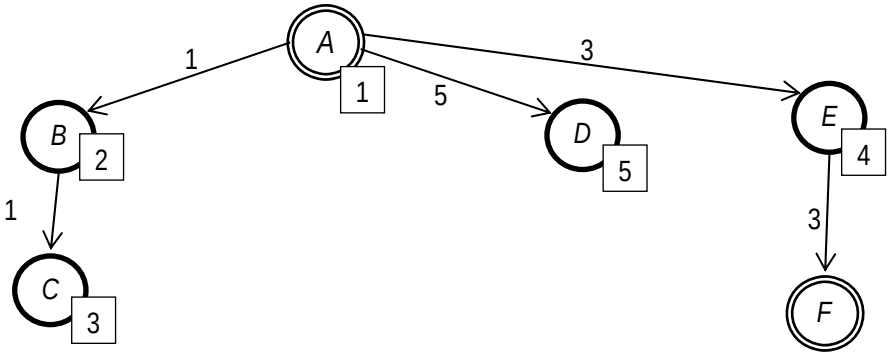


Figura 2.4: Orden de expansión de nodos para el árbol de la figura 2.1 según la búsqueda de coste uniforme. Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto, el algoritmo termina en ese momento sin que se lleguen a generar sus hijos.

4. Búsqueda en Anchura Iterativa (de derecha a izquierda)

Este algoritmo ejecuta iterativamente varias búsquedas primero en anchura, de manera que entre iteración e iteración se incrementa en una unidad el número máximo de hijos que se generan en cada expansión de un nodo padre. Al principio (en la primera iteración), únicamente un hijo es generado en cada expansión. En la figura 2.5a se muestra el orden de expansión de nodos para la única iteración de búsqueda primero en anchura que es necesaria para este ejemplo de búsqueda en anchura iterativa.

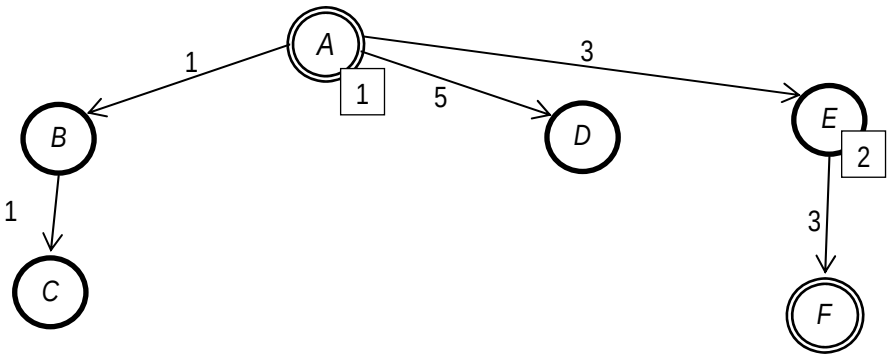


Figura 2.5a: Primera iteración de la búsqueda en anchura iterativa de derecha a izquierda, en la que como máximo un hijo es generado en cada expansión de un nodo padre.

Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto, el algoritmo termina en ese momento sin que se lleguen a generar sus hijos.

5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

Este algoritmo ejecuta iterativamente varias búsquedas primero en profundidad, de manera que entre iteración e iteración se incrementa en una unidad la profundidad límite. Al principio (en la primera iteración), la profundidad límite es igual a 1, así que sólo se podrá expandir el nodo inicial, cuya profundidad es igual a 0. En las figuras 2.6a y 2.6b se muestran los órdenes de expansión de nodos para las dos iteraciones de búsqueda primero en profundidad que son necesarias para este ejemplo de búsqueda en profundidad iterativa.

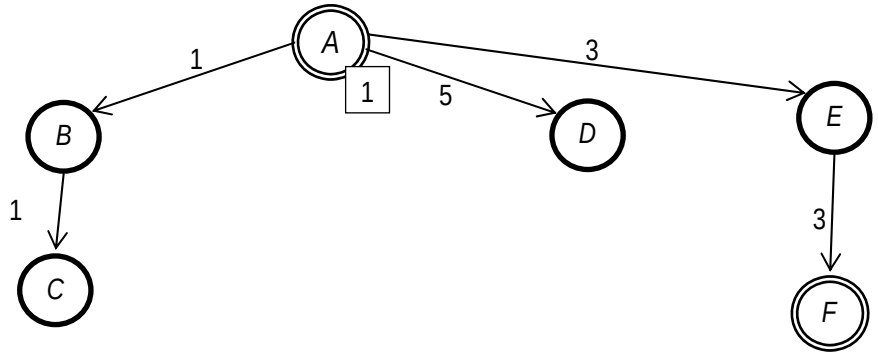


Figura 2.6a: Primera iteración de la búsqueda en profundidad iterativa de izquierda a derecha, en la que la profundidad límite es igual a 1 y únicamente son expandidos nodos con una profundidad máxima de 0.

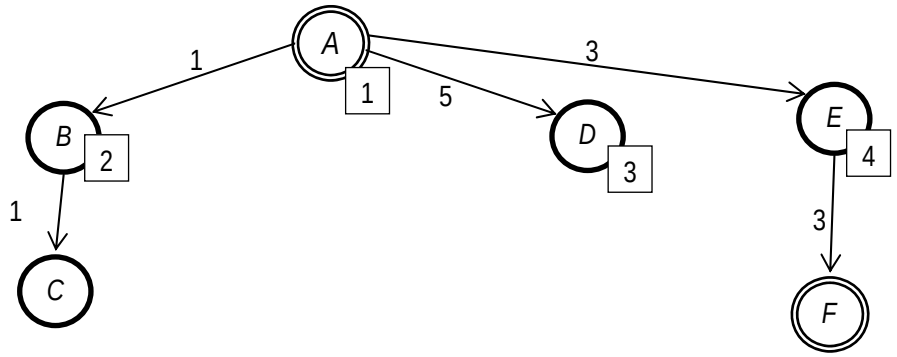


Figura 2.6b: Segunda iteración de la búsqueda en profundidad iterativa de izquierda a derecha, en la que la profundidad límite es igual a 2 y únicamente son expandidos nodos con una profundidad máxima de 1.

Observe que realmente el nodo meta no es expandido en esta última iteración porque se encuentre a la profundidad límite. Esta condición no se llega a comprobar, ya que justo antes se averigua que el nodo es meta y, por tanto, el algoritmo termina.

EJERCICIO 3:

Considere el espacio de búsqueda de la figura 3.1, que tiene forma de árbol, donde el nodo raíz del árbol es el nodo inicial, existe un único nodo meta y cada operador tiene asociado un coste. Describa cuál es el contenido de ABIERTA, previamente a cada extracción de un nodo de la misma, a partir de cada uno de los métodos siguientes de búsqueda sin información del dominio:

1. Búsqueda Primero en Anchura (de izquierda a derecha)
2. Búsqueda Primero en Profundidad (de derecha a izquierda)
3. Búsqueda de Coste Uniforme
4. Búsqueda en Anchura Iterativa (de derecha a izquierda)
5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

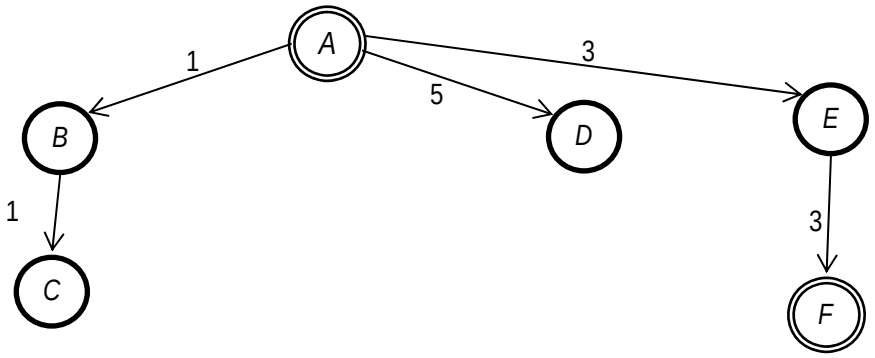


Figura 3.1: Árbol de búsqueda en el que el nodo inicial es *A*, el nodo meta es *F* y el coste de cada operador aparece al lado del arco que lo representa.

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 3:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- Cada uno de los cinco apartados, correspondiente a un método de búsqueda concreto, se puntúa sobre 2 puntos.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no gestiona ABIERTA en la forma debida: la puntuación del apartado bajaría 1.6 puntos si la respuesta dada varía significativamente de la correcta, mientras que si la respuesta dada varía de la correcta como consecuencia de un despiste no conceptual entonces la puntuación del apartado bajaría sólo 0.4 puntos en vez de los 1.6 puntos mencionados.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no finaliza cuando es debido, la puntuación del apartado bajaría 0.4 puntos. (Esto es independiente de la gestión de ABIERTA dada como respuesta, que ya ha sido valorada anteriormente con un máximo de 1.6 puntos.)

SOLUCIÓN DEL EJERCICIO 3 (por Severino Fernández Galán):

1. Búsqueda Primero en Anchura (de izquierda a derecha)

Este algoritmo usa ABIERTA como una cola, de manera que siempre se saca el primer nodo de la cola y se introducen sus hijos al final de la misma. El contenido de ABIERTA previamente a cada extracción de un nodo es el siguiente:

Antes de sacar *A*: {*A*}
Antes de sacar *B*: {*B, D, E*}
Antes de sacar *D*: {*D, E, C*}
Antes de sacar *E*: {*E, C*}
Antes de sacar *C*: {*C, F*}
Antes de sacar *F*: {*F*}
META ENCONTRADA

Observe que, tras cada expansión del nodo sacado de ABIERTA, sus nodos hijos más a la izquierda se introducen en ABIERTA antes que los situados más a la derecha.

2. Búsqueda Primero en Profundidad (de derecha a izquierda)

Este algoritmo usa ABIERTA como una pila, de manera que siempre se saca el primer nodo de la pila y se introducen sus hijos al principio de la misma. El contenido de ABIERTA previamente a cada extracción de un nodo es el siguiente:

Antes de sacar *A*: {*A*}
Antes de sacar *E*: {*E, D, B*}
Antes de sacar *F*: {*F, D, B*}
META ENCONTRADA

Observe que, tras cada expansión del nodo sacado de ABIERTA, sus nodos hijos más a la derecha se introducen en ABIERTA después que los situados más a la izquierda.

3. Búsqueda de Coste Uniforme

Este algoritmo mantiene ABIERTA ordenada según el coste del camino desde cada nodo al nodo inicial. En este ejercicio desharemos de forma arbitraria los posibles empates que surjan al determinar qué nodo se debe sacar de ABIERTA. El contenido de ABIERTA previamente a cada extracción de un nodo es el siguiente:

Antes de sacar *A*: {*A*(0)}
Antes de sacar *B*: {*B*(1), *E*(3), *D*(5)}
Antes de sacar *C*: {*C*(2), *E*(3), *D*(5)}
Antes de sacar *E*: {*E*(3), *D*(5)}
Antes de sacar *D*: {*D*(5), *F*(6)}
Antes de sacar *F*: {*F*(6)}
META ENCONTRADA

Observe que, tras cada expansión de un nodo, sus nodos hijos son introducidos en ABIERTA, donde todos los nodos quedan siempre ordenados por coste creciente. Para facilitar el seguimiento del algoritmo, entre paréntesis y al lado de cada nodo de ABIERTA se especifica el coste del camino para ir desde dicho nodo hasta el nodo inicial.

4. Búsqueda en Anchura Iterativa (de derecha a izquierda)

Debido a que este algoritmo es una iteración de la búsqueda primero en anchura, los dos gestionan ABIERTA del mismo modo, es decir, como una cola. La diferencia reside en que en la búsqueda en anchura iterativa en cada iteración se fija un número máximo de hijos que pueden ser generados al

expandir un nodo padre. Este número empieza valiendo 1 y es incrementado en una unidad al principio de cada nueva iteración.

Iteración 1 (cada expansión de un nodo padre genera como máximo un hijo):

Antes de sacar *A*: {*A*}
Antes de sacar *E*: {*E*}
Antes de sacar *F*: {*F*}
META ENCONTRADA

5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

Debido a que este algoritmo es una iteración de la búsqueda primero en profundidad, los dos gestionan ABIERTA del mismo modo, es decir, como una pila. La diferencia reside en que en la búsqueda en profundidad iterativa en cada iteración se fija una profundidad límite propia. Este número empieza valiendo 1, lo cual permite expandir únicamente el nodo inicial, y es incrementado en una unidad al principio de cada nueva iteración.

Iteración 1 (profundidad límite igual a 1):

Antes de sacar *A*: {*A*}
Antes de sacar *B*: {*B, D, E*}
Nótese que *B* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *D*: {*D, E*}
Nótese que *D* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *E*: {*E*}
Nótese que *E* no es expandido por coincidir su profundidad con la profundidad límite.

Iteración 2 (profundidad límite igual a 2):

Antes de sacar *A*: {*A*}
Antes de sacar *B*: {*B, D, E*}
Antes de sacar *C*: {*C, D, E*}
Nótese que *C* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *D*: {*D, E*}
Antes de sacar *E*: {*E*}
Antes de sacar *F*: {*F*}
META ENCONTRADA

EJERCICIO 4:

Considere el espacio de búsqueda de la figura 4.1, que tiene forma de árbol, donde el nodo raíz del árbol es el nodo inicial, existe un único nodo meta y cada operador tiene asociado un coste. Describa cuál es el contenido de TABLA_A, posteriormente a cada expansión de un nodo, a partir de cada uno de los métodos siguientes de búsqueda sin información del dominio:

1. Búsqueda Primero en Anchura (de izquierda a derecha)
2. Búsqueda Primero en Profundidad (de derecha a izquierda)
3. Búsqueda de Coste Uniforme
4. Búsqueda en Anchura Iterativa (de derecha a izquierda)
5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

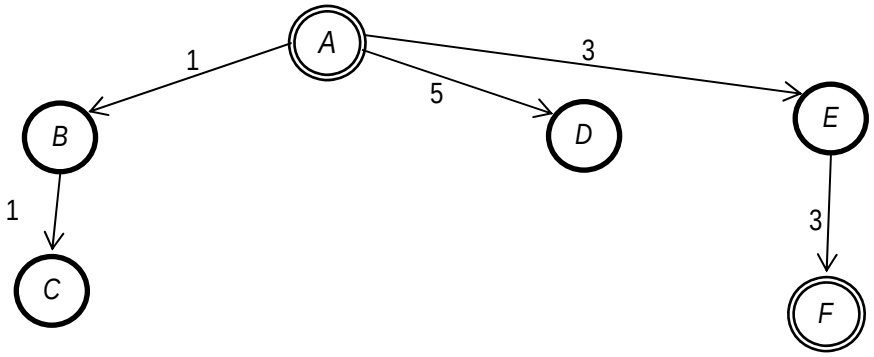


Figura 4.1: Árbol de búsqueda en el que el nodo inicial es *A*, el nodo meta es *F* y el coste de cada operador aparece al lado del arco que lo representa.

Para cada nodo de TABLA_A incluya la siguiente información: su nodo padre y el coste al nodo inicial. (En este ejercicio no es necesario incluir en TABLA_A información sobre los hijos de cada nodo expandido, ya que sólo existe un camino desde cada nodo al nodo inicial y, por tanto, el mejor camino desde cada nodo al nodo inicial no cambia a lo largo del proceso de búsqueda.)

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 4:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

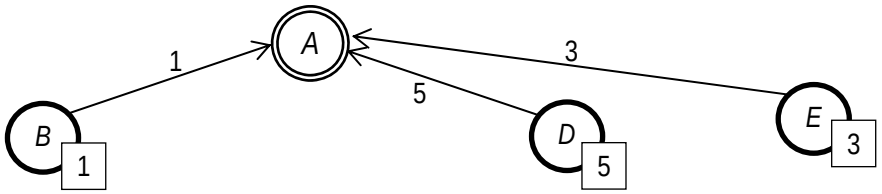
- Cada uno de los cinco apartados, correspondiente a un método de búsqueda concreto, se puntúa sobre 2 puntos.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no gestiona TABLA_A en la forma debida: la puntuación del apartado bajaría 1.6 puntos si la respuesta dada varía significativamente de la correcta, mientras que si la respuesta dada varía de la correcta como consecuencia de un despiste no conceptual entonces la puntuación del apartado bajaría sólo 0.4 puntos en vez de los 1.6 puntos mencionados.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no finaliza cuando es debido, la puntuación del apartado bajaría 0.4 puntos. (Esto es independiente de la gestión de TABLA_A dada como respuesta, que ya ha sido valorada anteriormente con un máximo de 1.6 puntos.)

SOLUCIÓN DEL EJERCICIO 4 (por Severino Fernández Galán):

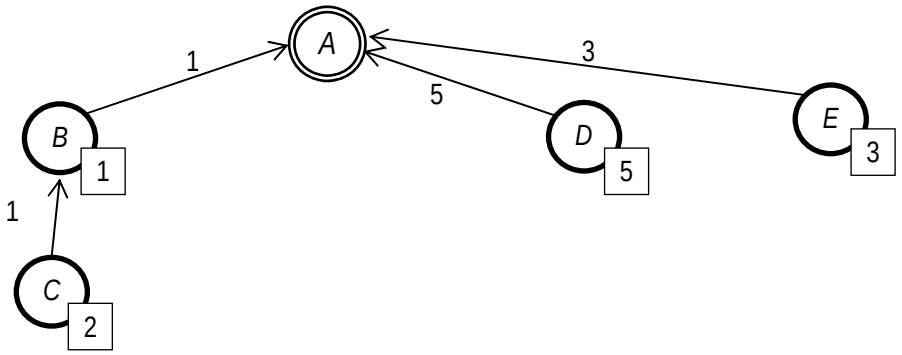
De cara a ilustrar la respuesta convenientemente, incluimos la información de TABLA_A gráficamente: por un lado, para cada nodo generado en una expansión trazamos un arco ascendente a su padre y, por otro lado, indicamos el coste al nodo inicial al lado de cada nodo generado.

1. Búsqueda Primero en Anchura (de izquierda a derecha)

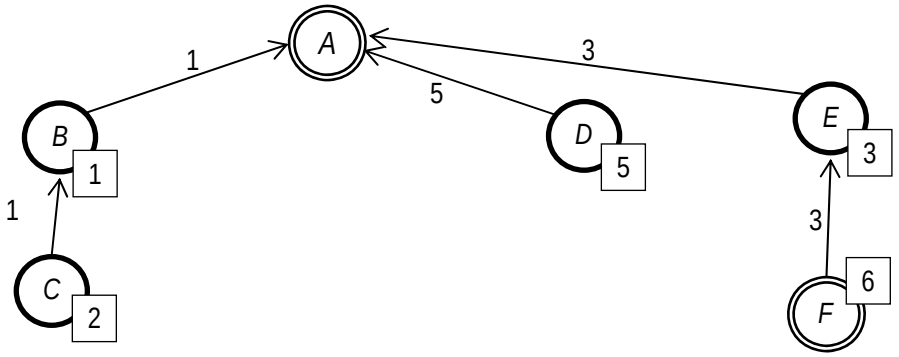
- Inicialmente, *A* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *A*. A continuación expandimos el nodo inicial, generando sus nodos hijos. Desde cada nodo hijo trazamos un nodo ascendente a su nodo padre. Al lado de cada nodo hijo indicamos el coste desde dicho nodo al nodo inicial.



- Expandimos *B*:



- Expandimos *D* y TABLA_A no varía. A continuación expandimos *E*:

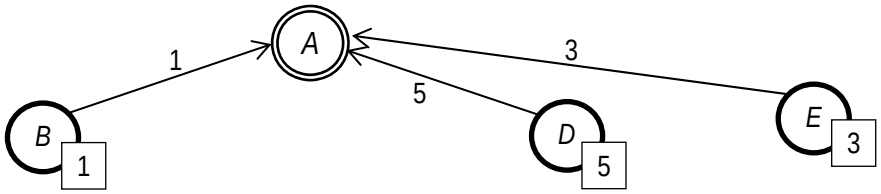


- Expandimos *C* y TABLA_A no varía. A continuación elegiríamos *F* para su expansión y llegaríamos a la meta. El camino hallado hasta la meta tiene coste 6.

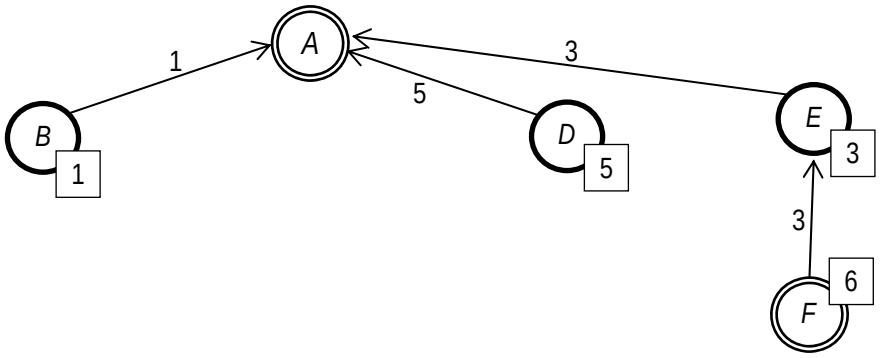
2. Búsqueda Primero en Profundidad (de derecha a izquierda)

Cuando sea necesario, en este algoritmo aplicaremos la función LimpiarTABLA_A (véase texto base, página 318). Tras la extracción de un nodo de ABIERTA y su posible expansión, dicha función elimina aquellos nodos que ya no son necesarios en el proceso de búsqueda de la meta. Esto permite preservar la linealidad de la complejidad espacial de este algoritmo con respecto a la profundidad de la solución.

- Inicialmente, *A* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *A*. A continuación, expandimos el nodo inicial.



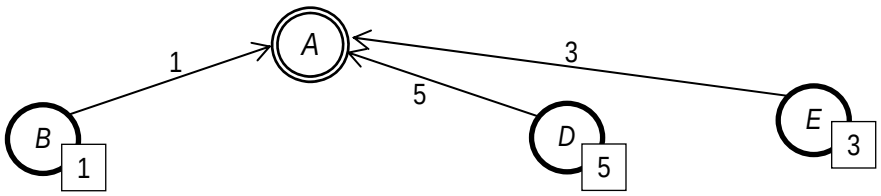
- Expandimos *E*:



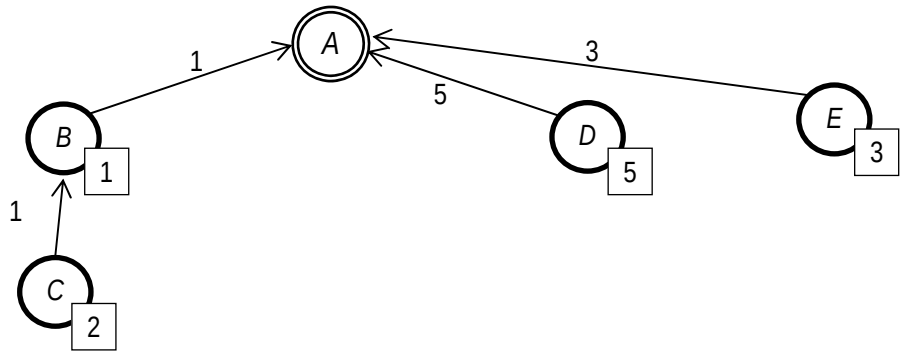
- A continuación elegiríamos *F* para su expansión y llegaríamos a la meta. El camino hallado hasta la meta tiene coste 6. Nótese que finalmente no ha sido necesario aplicar la función LimpiarTABLA_A a lo largo de este apartado.

3. Búsqueda de Coste Uniforme

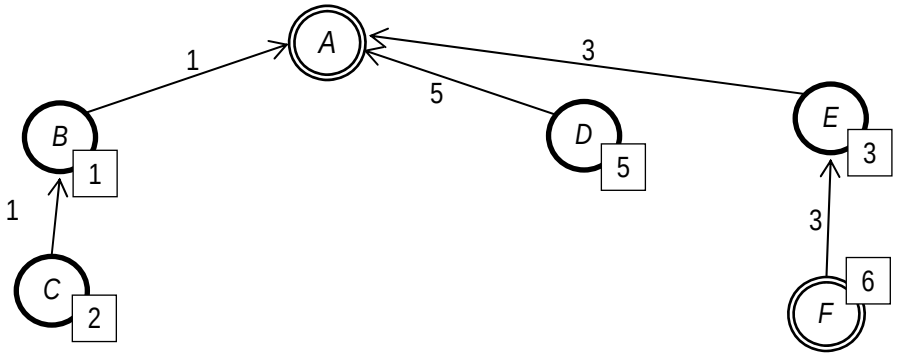
- Inicialmente, *A* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *A*. A continuación, expandimos el nodo inicial.



- Expandimos *B*:



- Expandimos *C* y TABLA_A no varía. A continuación expandimos *E*:

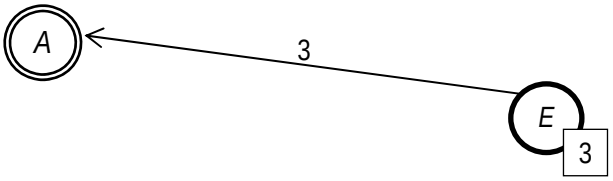


- Expandimos *D* y TABLA_A no varía. Al intentar expandir *F*, alcanzamos la meta. El coste del camino solución hallado es 6.

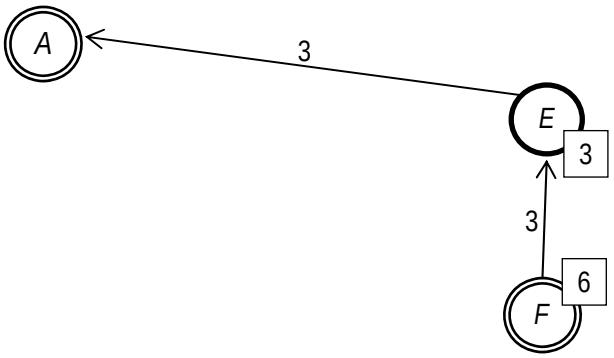
4. Búsqueda en Anchura Iterativa (de derecha a izquierda)

<u>Iteración 1 (se genera como máximo 1 nodo hijo en cada expansión de un nodo padre)</u>:

- Inicialmente, *A* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *A*. A continuación, expandimos el nodo inicial:



- Expandimos *E*:

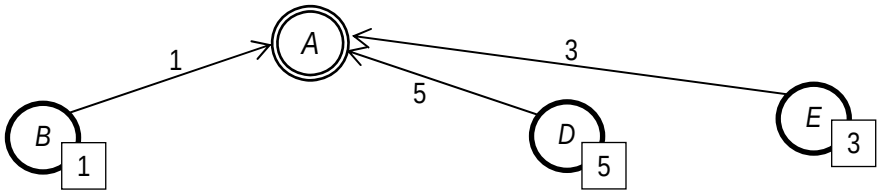


- Finalmente, al intentar expandir *F* alcanzamos la meta. El camino encontrado hasta el nodo inicial tiene coste 6.

5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

Iteración 1 (profundidad límite igual a 1):

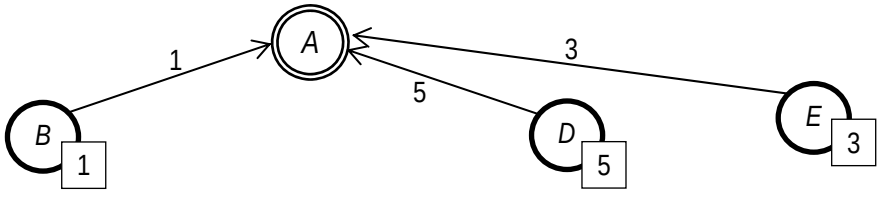
- Inicialmente, *A* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *A*. A continuación, expandimos el nodo inicial:



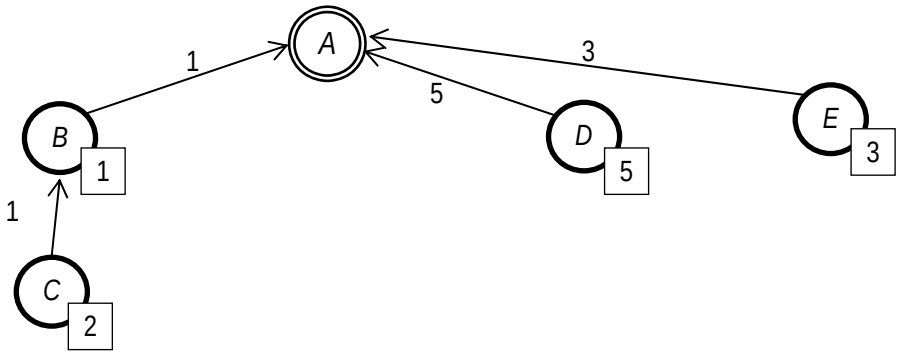
- Tras comprobar que *B* no es nodo meta, se le aplica la función LimpiarTABLA_A por estar en la profundidad límite y es sacado de TABLA_A. De igual manera, tras comprobar respectivamente que *D* y *E* no son nodo meta, se les aplica la función LimpiarTABLA_A por estar en la profundidad límite y son sacados de TABLA_A. Seguidamente, tras aplicarle la función LimpiarTABLA_A, *A* es sacado de TABLA_A por no tener hijos en ABIERTA. A continuación pasamos a la siguiente iteración.

Iteración 2 (profundidad límite igual a 2):

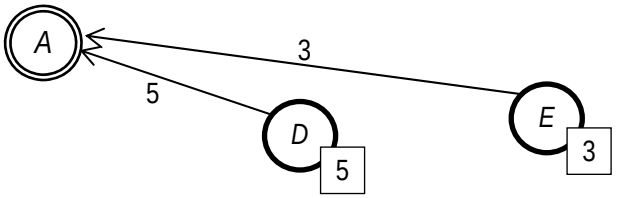
- Inicialmente, *A* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *A*. A continuación, expandimos el nodo inicial:



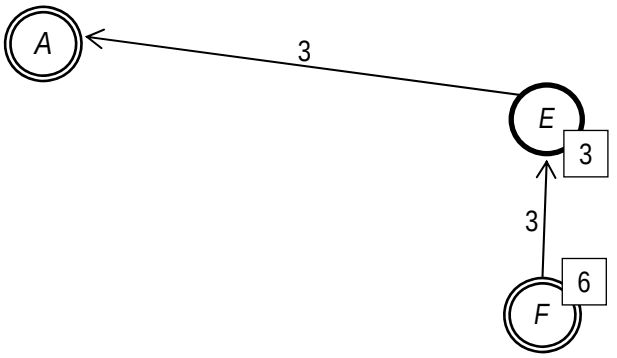
- Expandimos *B*:



- Tras comprobar que *C* no es nodo meta, se le aplica la función LimpiarTABLA_A por encontrarse en la profundidad límite y es sacado de TABLA_A. Del mismo modo, tras comprobar que *B* no tiene hijos en ABIERTA, se le aplica la función LimpiarTABLA_A y es sacado de TABLA_A.



- Expandimos *D*. Seguidamente, aplicamos la función LimpiarTABLA_A a *D* por no tener hijos en ABIERTA y es sacado de TABLA_A. A continuación, expandimos *E*:



- Tras elegir *F* para su expansión y comprobar que es nodo meta, finaliza el algoritmo habiendo hallado un camino entre el nodo inicial y el nodo meta de coste 6.

EJERCICIO 5:

Considere el grafo de la figura 5.1, donde el nodo inicial es n_1 y donde el nodo meta es n_6. Cada arco u operador lleva asociado su coste y en cada nodo aparece la estimación de la menor distancia desde ese nodo a una meta. Aplique paso a paso el algoritmo A* al grafo dado, indicando de forma razonada la siguiente información en cada paso del algoritmo:

1. Qué nodo es expandido.
2. Cuál es el contenido de ABIERTA tras la expansión del nodo, indicando el valor de la función de evaluación heurística para cada nodo de ABIERTA.
3. Cuál es el contenido de TABLA_A tras la expansión del nodo. Para cada nodo de TABLA_A incluya la siguiente información:
    a) Su nodo padre que indique el camino de menor coste hasta el nodo inicial encontrado hasta el momento
    b) El coste del camino de menor coste hasta el nodo inicial encontrado hasta el momento
    c) Sus nodos hijos (si el nodo de TABLA_A actual ya ha sido expandido)

Por último, ¿cuál es el camino solución hallado y su coste?

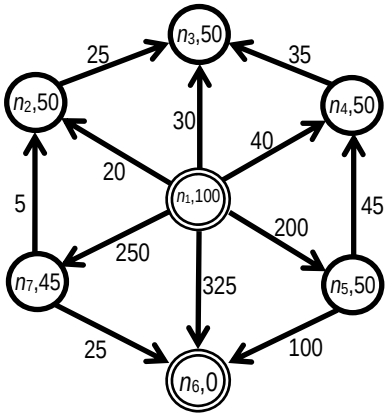


Figura 5.1: Grafo de búsqueda en el que el nodo inicial es n_1, el nodo meta es n_6, el coste de cada operador aparece al lado del arco que lo representa y al lado de cada nodo aparece la estimación de la menor distancia desde ese nodo a una meta.

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 5:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- El orden seguido en la expansión de los nodos se puntúa sobre 1.5 puntos.
- La forma en que se gestiona ABIERTA se puntúa sobre 3.75 puntos. Se hará especial énfasis en comprobar qué nodos hay en ABIERTA en cada paso del algoritmo y qué valores de la función de evaluación heurística se les asignan.
- La forma en que se gestiona TABLA_A se puntúa sobre 3.75 puntos. Se hará especial énfasis en comprobar qué nodos hay en TABLA_A en cada paso del algoritmo y qué padre mejor se les asigna (teniendo en cuenta las posibles reorientaciones o rectificaciones de enlaces)
- La correcta terminación del algoritmo se puntúa sobre 1 punto. Se hará especial énfasis en comprobar cuándo termina el algoritmo y qué camino solución devuelve.

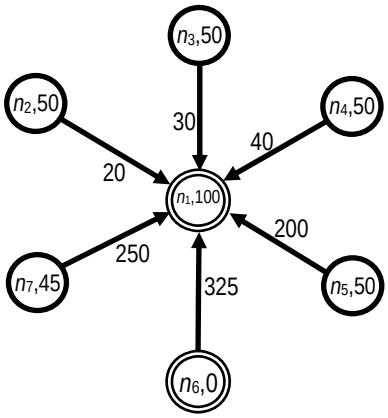
SOLUCIÓN DEL EJERCICIO 5 (por Severino Fernández Galán):

De cara a ilustrar la respuesta convenientemente, incluimos la información pedida sobre TABLA_A gráficamente. Para ello es necesario trazar, para cada nodo generado en una expansión, un arco ascendente a su padre expandido; además, hay que anotar para cada nodo su mejor padre encontrado hasta el momento (punto 3a del enunciado). De esta manera, siguiendo cada arco al mejor padre, se puede saber cuál es el mejor camino encontrado hasta el momento desde cada nodo al nodo inicial (punto 3b del enunciado). Además, los arcos ascendentes que llegan a un nodo ya expandido lo enlazan a sus nodos hijos (punto 3c del enunciado).

- PASO 0. El nodo inicial n_1 es introducido en ABIERTA y en TABLA_A, con lo que tenemos la siguiente situación:

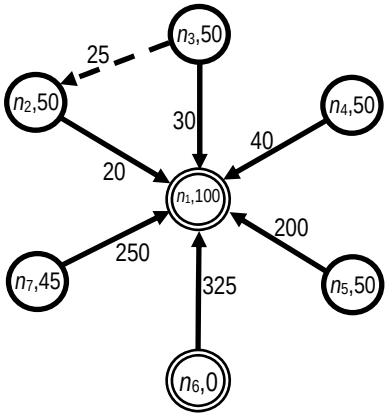
ABIERTA = {$\underline{n_1(0+100)}$}

- PASO 1. Expandimos el nodo n_1 de ABIERTA. Tras la expansión, la situación es la siguiente:



ABIERTA = {$\underline{n_2(20+50)}$, $n_3(30+50)$, $n_4(40+50)$, $n_5(200+50)$, $n_7(250+45)$, $n_6(325+0)$}

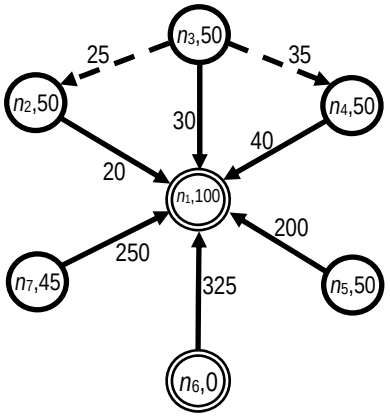
- PASO 2. Expandimos n_2 por ser el nodo de ABIERTA con menor valor de la función de evaluación heurística, $f=g+h$ (al ser g=20 y h=50).



ABIERTA = {$\underline{n_3(30+50)}$, $n_4(40+50)$, $n_5(200+50)$, $n_7(250+45)$, $n_6(325+0)$}

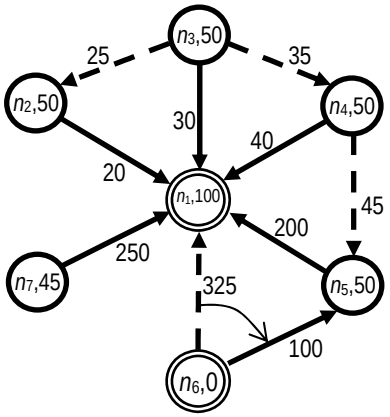
Observe que el mejor camino desde n_3 al nodo inicial lo marca su padre n_1 (coste 30) y no su padre n_2 (coste 25+20=45). Por ello, el arco ascendente de n_3 a n_1 se marca con trazo continuo y el arco ascendente de n_3 a n_2 se marca con trazo discontinuo. Es importante darse cuenta que el conjunto de arcos con trazo continuo formará siempre un árbol en el grafo parcial de búsqueda desarrollado hasta el momento.

- PASOS 3 y 4. Expandimos n_3 y nada cambia en TABLA_A. A continuación expandimos n_4.



ABIERTA = {$\underline{n_5(200+50)}$, $n_7(250+45)$, $n_6(325+0)$}

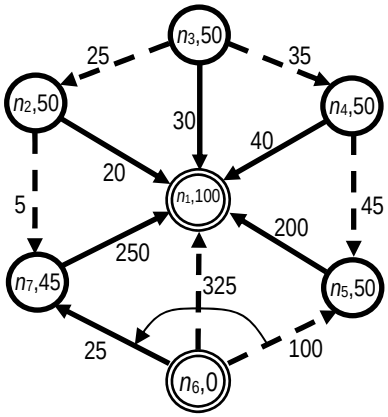
- PASO 5. Expandimos n_5.



ABIERTA = {$\underline{n_7(250+45)}$, $n_6(300+0)$}

Observe también que ha habido una reorientación del mejor padre de n_6, que antes era n_1 y ahora pasa a ser n_5. El nuevo mejor coste de n_6 al nodo inicial, $g(n_6)$, es ahora 100+200=300, que se puede calcular a partir de los costes de los mejores arcos ascendentes hallados hasta el momento: $n_6 \rightarrow n_5$ y $n_5 \rightarrow n_1$.

- PASO 6. Expandimos n_7.



ABIERTA = {n_6(275+0)}

- PASO 7. Intentamos expandir n_6 y alcanzamos una meta, con lo que el algoritmo termina. El camino solución encontrado es: $n_1 \rightarrow n_7 \rightarrow n_6$, cuyo coste es 250+25=275.

EJERCICIO 6:

Considere el grafo de la figura 6.1, donde el nodo inicial es *D* y donde los nodos meta son desconocidos. Cada arco u operador lleva asociado su coste y en cada nodo aparece su valor de la función de evaluación heurística (que hay que minimizar). Aplique paso a paso el algoritmo de escalada o máximo gradiente al grafo dado. Para ello indique de forma razonada qué nodo se expande en cada paso y cuál es el nodo final devuelto por el algoritmo. Utilice como *criterio de selección* el de mejor vecino. Utilice como *criterio de terminación* el que no se hayan producido mejoras durante los tres últimos pasos del algoritmo.

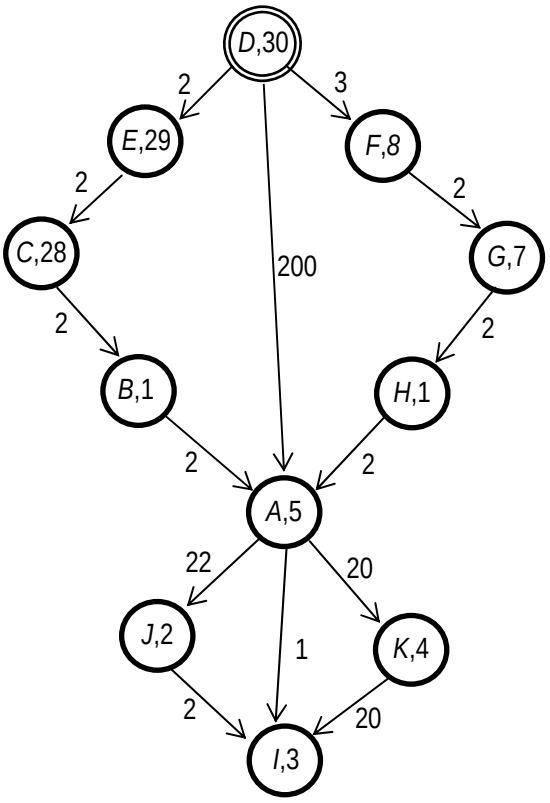


Figura 6.1: Grafo de búsqueda en el que el nodo inicial es *D*, los nodos meta son desconocidos, el coste de cada operador aparece al lado del arco que lo representa y al lado de cada nodo aparece el valor de su función de evaluación heurística (que hay que minimizar).

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 6:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- La correcta aplicación en cada paso del algoritmo del *criterio de selección* del vecino o hijo del nodo actual (qué vecino o hijo del nodo actual es considerado como candidato para sustituirlo) se puntúa sobre 4.5 puntos.
- La correcta aplicación en cada paso del algoritmo del *criterio de aceptación* del vecino o hijo seleccionado (si el vecino o hijo candidato sustituye o no finalmente al nodo actual) se puntúa sobre 4.5 puntos.
- La correcta aplicación del *criterio de finalización* del algoritmo se puntúa sobre 1 punto.

SOLUCIÓN DEL EJERCICIO 6 (por Severino Fernández Galán):

- PASO 1: Al principio expandimos el nodo inicial *D*, generando sus nodos hijos {*E*(29), *A*(5), *F*(8)}. Seleccionamos el nodo *A* por ser el mejor de los nodos hijos. A continuación aceptamos el nodo *A* como nuevo nodo actual en sustitución de *D*, debido a que el valor de la función de evaluación heurística de *A* es mejor o igual que el de *D* ($5 \leq 30$).

- PASO 2: Expandimos el nodo actual *A* y generamos sus hijos: {*J*(2), *I*(3), *K*(4)}. Seleccionamos el nodo *J* por ser el mejor de los nodos hijos. Seguidamente aceptamos el nodo *J* como nuevo nodo actual en sustitución de *A*, debido a que el valor de la función de evaluación heurística de *J* es mejor o igual que el de *A* ($2 \leq 5$).

- PASO 3: Expandimos el nodo actual *J* y generamos sus hijos: {*I*(3)}. Seleccionamos el nodo *I* por ser el mejor de los nodos hijos. A continuación rechazamos el nodo *I* como nuevo nodo actual en sustitución de *J*, debido a que el valor de la función de evaluación heurística de *I* no es mejor o igual que el de *J* (no se cumple que $3 \leq 2$).

La búsqueda terminaría en este punto. El algoritmo devolvería el nodo *J*(2) como mejor nodo encontrado.

AÑO 2017

EJERCICIO 1:

Dibuje mediante un grafo dirigido o describa detalladamente mediante una tabla el espacio de estados (o espacio de búsqueda) completo para el *problema del viajante* descrito más adelante. Para ello especifique: el conjunto de todos los estados posibles, el estado inicial, el o los estados meta, los operadores aplicables a cada estado y el coste asociado a cada operador. En el problema del viajante se busca un recorrido de longitud mínima entre varias ciudades que las visite una sola vez, partiendo y llegando a la misma ciudad. Suponga que se parte de una situación inicial donde: (1) Existen cinco ciudades {A, B, C, D, E} cuyas coordenadas cartesianas son, respectivamente, {(0,0), (0,1), (1,1), (2,1), (2,0)} y (2) El recorrido inicial considerado es ADCBEA. Dado un recorrido cualquiera, consideraremos que se pueden generar recorridos hijos suyos en el espacio de búsqueda al intercambiar el orden en que son visitadas dos ciudades intermedias contiguas cualesquiera del recorrido dado. (Por ejemplo, en el recorrido ADCBEA las ciudades intermedias son {B, C, D, E} y se puede generar un recorrido hijo intercambiando las ciudades contiguas B y E, tal que se obtenga el recorrido ADCEBA.) Considere también que el estado meta es el recorrido más corto, es decir, cualquiera de los dos recorridos que se corresponden con el rectángulo formado por las cinco ciudades. ¿Cuál es la solución menos costosa para este problema, tal como ha sido descrito?

(NOTA1: Se recomienda suponer que la ciudad inicial de cualquier recorrido considerado es siempre la misma, por ejemplo, la ciudad A.)

(NOTA2: El tipo de representación y operadores elegidos en este ejercicio para el problema del viajante podrían ser utilizados, por ejemplo, en algoritmos de búsqueda local. Aunque estos algoritmos suelen asociar a cada recorrido su longitud total, en este ejercicio sólo estamos interesados en estudiar el espacio de estados global resultante, con lo que no es de nuestro interés dicha longitud.)

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 1:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- Los estados se especifican correctamente: 2.5 puntos
- Los operadores se especifican correctamente: 2.5 puntos
- Los costes de los operadores se especifican correctamente: 1 punto
- El espacio de búsqueda se dibuja (mediante un grafo dirigido) o se describe (mediante una tabla) correctamente: 3 puntos
- La solución de menor coste se especifica correctamente: 1 punto

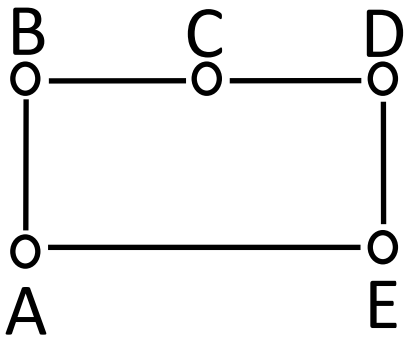
SOLUCIÓN DEL EJERCICIO 1 (por Severino Fernández Galán):

Para representar los **estados** posibles utilizaremos la siguiente notación. El estado del problema lo representamos como:

$$(A\ I_1\ I_2\ I_3\ I_4\ A),$$

donde A es la ciudad inicial y final del recorrido, mientras que la secuencia de variables $\{I_1, I_2, I_3, I_4\}$ representa cualquier permutación de las ciudades {B, C, D, E}. Por ejemplo, el estado (ADECBA) representa el estado en que el recorrido se inicia en A, sigue a D, sigue a E, sigue a C, sigue a B y acaba en A. Nótese que existen 4x3x2x1 = 24 estados posibles en el problema planteado.

El **estado inicial** descrito en el enunciado se representa como (ADCBEA). Por otro lado, existen dos **estados meta**, (ABCDEA) y (AEDCBA), que unen las ciudades formando el rectángulo siguiente:



Consideramos tres **operadores** posibles, según las condiciones dadas en el enunciado:

[1] Intercambiar I_1 e I_2: $(A\ I_1\ I_2\ I_3\ I_4\ A) \rightarrow (A\ I_2\ I_1\ I_3\ I_4\ A)$

[2] Intercambiar I_2 e I_3: $(A\ I_1\ I_2\ I_3\ I_4\ A) \rightarrow (A\ I_1\ I_3\ I_2\ I_4\ A)$

[3] Intercambiar I_3 e I_4: $(A\ I_1\ I_2\ I_3\ I_4\ A) \rightarrow (A\ I_1\ I_2\ I_4\ I_3\ A)$

Se puede considerar que cada operador tiene **coste** unidad, así que el coste de un camino es el número de arcos que lo componen.

En la tabla 1.1 figuran los estados posibles del sistema, los operadores que se pueden aplicar a cada estado posible y los estados resultantes de aplicar cada operador. Se ha resaltado en color tanto el estado inicial (rojo) como los estados meta (verde).

De forma alternativa, la información incluida en la tabla 1.1 se puede representar mediante un grafo dirigido tal como se muestra en la figura 1.1. En dicha figura no existen estados no permitidos (que marcaríamos con elipses discontinuas). Se ha utilizado color rojo para marcar el estado inicial y color verde para marcar los estados meta. Por otra parte, obsérvese que no existen estados aislados, que no sean alcanzables desde el estado inicial.

Las **soluciones menos costosas** para el problema planteado están marcadas en trazo discontinuo en la figura 1.1 y tienen un coste de 3:

(ADCBEA) → (ACDBEA) → (ACBDEA) → (ABCDEA)
(ADCBEA) → (ADBCEA) → (ABDCEA) → (ABCDEA)
(ADCBEA) → (ADCEBA) → (ADECBA) → (AEDCBA)

Los arcos correspondientes a los operadores [1], [2] y [3] han sido dibujados respectivamente con colores azul, **negro** y naranja, de cara a una mejor visualización.

NOTA:

También se considera válida la respuesta que supone que los recorridos $(A\ I_1\ I_2\ I_3\ I_4\ A)$ y $(A\ I_4\ I_3\ I_2\ I_1\ A)$ se corresponden con el mismo estado. De hecho, bajo esta suposición el número de estados posibles y el de estados meta se reduce a la mitad, lo cual constituye una importante ventaja cuando el número de ciudades crece.

ESTADOS	OPERADORES		
	[1]	[2]	[3]
(ABCDEA)	ESTADO META		
(ABCEDA)	(ACBEDA)	(ABECDA)	(ABCDEA)
(ABDCEA)	(ADBCEA)	(ABCDEA)	(ABDECA)
(ABDECA)	(ADBECA)	(ABEDCA)	(ABDCEA)
(ABECDA)	(AEBCDA)	(ABCEDA)	(ABEDCA)
(ABEDCA)	(AEBDCA)	(ABDECA)	(ABECDA)
(ACBDEA)	(ABCDEA)	(ACDBEA)	(ACBEDA)
(ACBEDA)	(ABCEDA)	(ACEBDA)	(ACBDEA)
(ACDBEA)	(ADCBEA)	(ACBDEA)	(ACDEBA)
(ACDEBA)	(ADCEBA)	(ACEDBA)	(ACDBEA)
(ACEBDA)	(AECBDA)	(ACBEDA)	(ACEDBA)
(ACEDBA)	(AECDBA)	(ACDEBA)	(ACEBDA)
(ADBCEA)	(ABDCEA)	(ADCBEA)	(ADBECA)
(ADBECA)	(ABDECA)	(ADEBCA)	(ADBCEA)
(ADCBEA)	(ACDBEA)	(ADBCEA)	(ADCEBA)
(ADCEBA)	(ACDEBA)	(ADECBA)	(ADCBEA)
(ADEBCA)	(AEDBCA)	(ADBECA)	(ADECBA)
(ADECBA)	(AEDCBA)	(ADCEBA)	(ADEBCA)
(AEBCDA)	(ABECDA)	(AECBDA)	(AEBDCA)
(AEBDCA)	(ABEDCA)	(AEDBCA)	(AEBCDA)
(AECBDA)	(ACEBDA)	(AEBCDA)	(AECDBA)
(AECDBA)	(ACEDBA)	(AEDCBA)	(AECBDA)
(AEDBCA)	(ADEBCA)	(AEBDCA)	(AEDCBA)
(AEDCBA)	ESTADO META		

Tabla 1.1: Estados posibles, operadores aplicables a cada estado y estados resultantes de aplicar cada operador para el problema del viajante planteado en este ejercicio. El estado inicial se resalta en rojo y los estados meta se resaltan en verde.

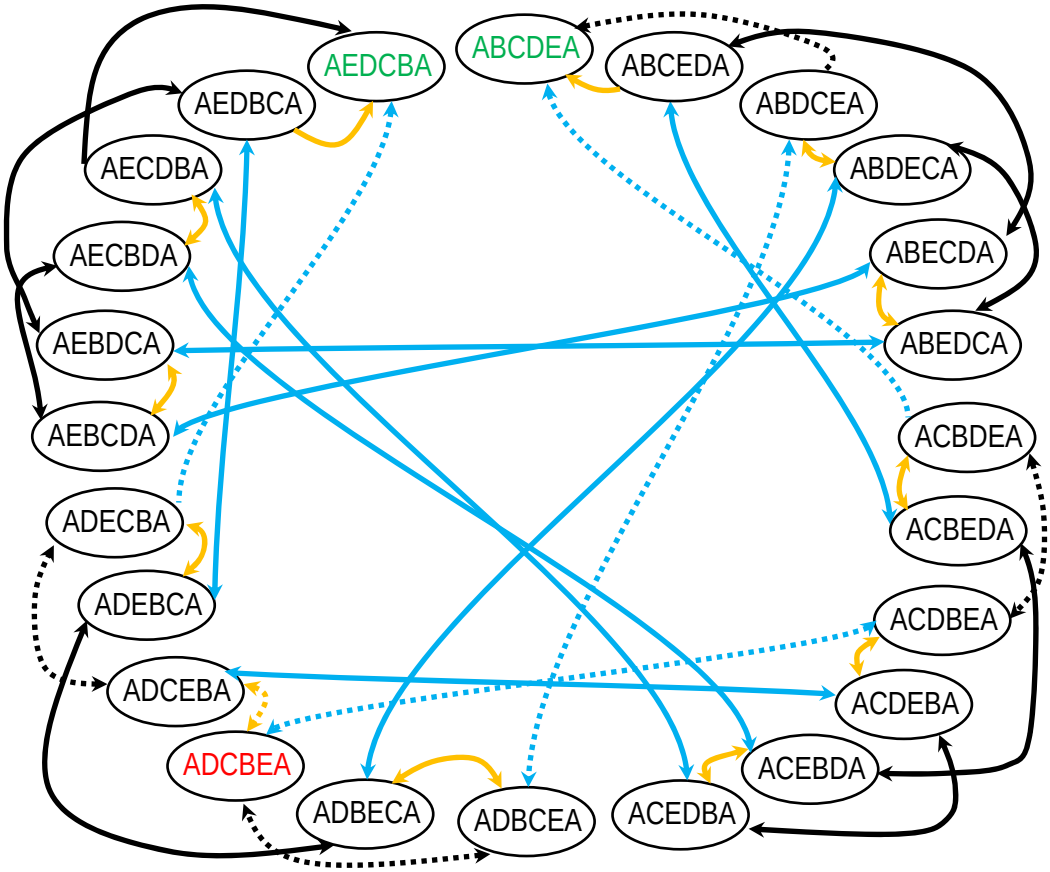


Figura 1.1: Grafo dirigido que representa el espacio de búsqueda para el problema del viajante. No existen estados no alcanzables desde el estado inicial, (ADCBEA), que queden aislados. Los caminos de menor coste hasta los estados meta, (ABCDEA) y (AEDCBA), se representan en trazo discontinuo. Los arcos correspondientes a los operadores [1], [2] y [3] han sido dibujados respectivamente con colores azul, negro y naranja, de cara a una mejor visualización.

EJERCICIO 2:

Considere el espacio de búsqueda de la figura 2.1, que tiene forma de árbol, donde el nodo raíz del árbol es el nodo inicial, existe un único nodo meta y cada operador tiene asociado un coste. Explique razonadamente en qué orden se expandirían los nodos de dicho árbol de búsqueda a partir de cada uno de los métodos siguientes de búsqueda sin información del dominio:

1. Búsqueda Primero en Anchura (de izquierda a derecha)
2. Búsqueda Primero en Profundidad (de derecha a izquierda)
3. Búsqueda de Coste Uniforme
4. Búsqueda en Anchura Iterativa (de derecha a izquierda)
5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

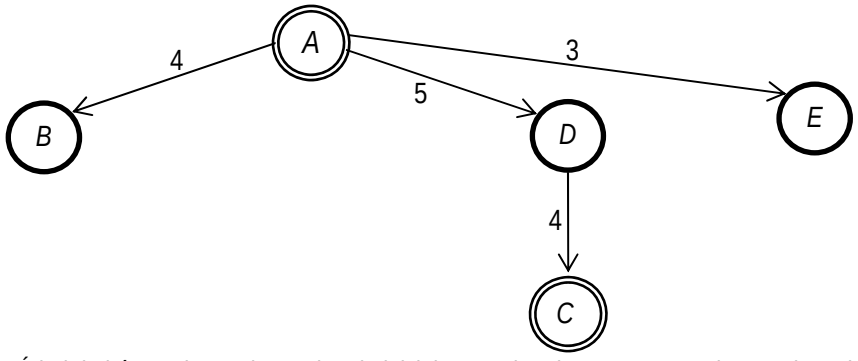


Figura 2.1: Árbol de búsqueda en el que el nodo inicial es *A*, el nodo meta es *C* y el coste de cada operador aparece al lado del arco que lo representa.

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 2:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- Cada uno de los cinco apartados, correspondiente a un método de búsqueda concreto, se puntúa sobre 2 puntos.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no expande los nodos en el orden debido: la puntuación del apartado bajaría 1.6 puntos si el orden dado como respuesta varía significativamente del correcto, mientras que si el orden dado como respuesta varía del correcto como consecuencia de un despiste no conceptual entonces la puntuación del apartado bajaría sólo 0.4 puntos en vez de los 1.6 puntos mencionados.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no finaliza cuando es debido, la puntuación del apartado bajaría 0.4 puntos. (Esto es independiente del orden de expansión de nodos dado como respuesta, que ya ha sido valorado anteriormente con un máximo de 1.6 puntos.)

SOLUCIÓN DEL EJERCICIO 2 (por Severino Fernández Galán):

1. Búsqueda Primero en Anchura (de izquierda a derecha)

Este algoritmo explora el árbol de búsqueda por niveles de profundidad, así que el orden de expansión de los nodos de izquierda a derecha sería el reflejado en la figura 2.2.

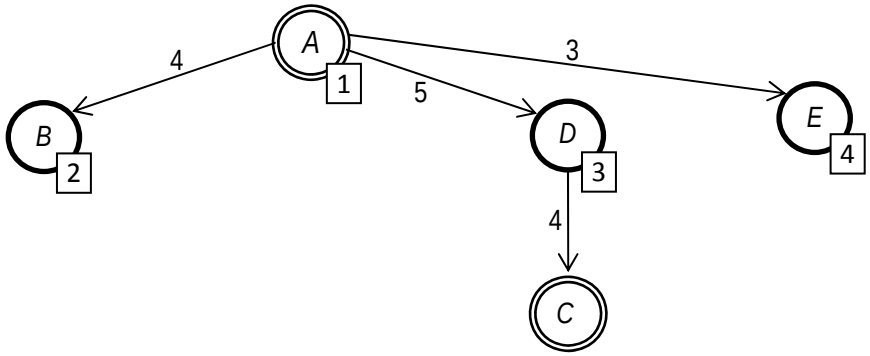


Figura 2.2: Orden de expansión de nodos para el árbol de la figura 2.1 según la búsqueda primero en anchura (de izquierda a derecha). Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto, el algoritmo termina en ese momento sin que se lleguen a generar sus hijos.

2. Búsqueda Primero en Profundidad (de derecha a izquierda)

Este algoritmo explora el árbol de búsqueda bajando de nivel siempre que sea posible. Si no es posible, se sube al nodo más cercano al nodo actual desde el que poder seguir bajando de nivel. El orden de expansión de los nodos de derecha a izquierda según este algoritmo se dibuja en la figura 2.3.

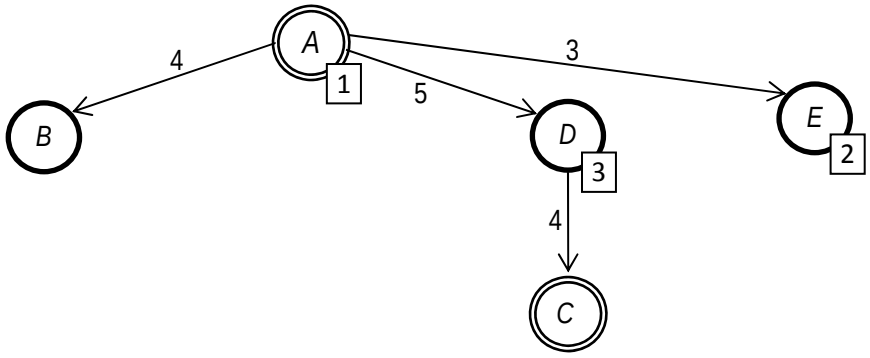


Figura 2.3: Orden de expansión de nodos para el árbol de la figura 2.1 según la búsqueda primero en profundidad (de derecha a izquierda). Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto, el algoritmo termina en ese momento sin que se lleguen a generar sus hijos.

3. Búsqueda de Coste Uniforme

Este algoritmo explora el árbol de búsqueda expandiendo aquel nodo disponible cuyo coste al nodo inicial sea el menor. El orden de expansión de nodos según este criterio se indica en la figura 2.4. Obsérvese que los empates, en caso de haberlos, serían deshechos de forma arbitraria.

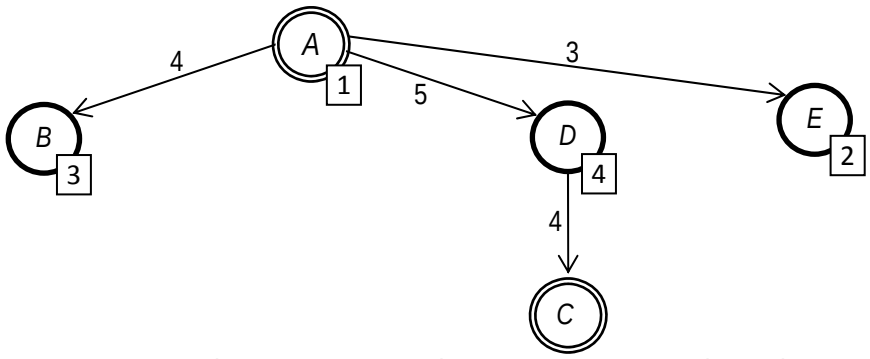


Figura 2.4: Orden de expansión de nodos para el árbol de la figura 2.1 según la búsqueda de coste uniforme. Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto, el algoritmo termina en ese momento sin que se lleguen a generar sus hijos.

4. Búsqueda en Anchura Iterativa (de derecha a izquierda)

Este algoritmo ejecuta iterativamente varias búsquedas primero en anchura, de manera que entre iteración e iteración se incrementa en una unidad el número máximo de hijos que se generan en cada expansión de un nodo padre. Al principio (en la primera iteración), únicamente un hijo es generado en cada expansión. En las figuras 2.5a y 2.5b se muestran los órdenes de expansión de nodos para las dos iteraciones de búsqueda primero en anchura que son necesarias para este ejemplo de búsqueda en anchura iterativa.

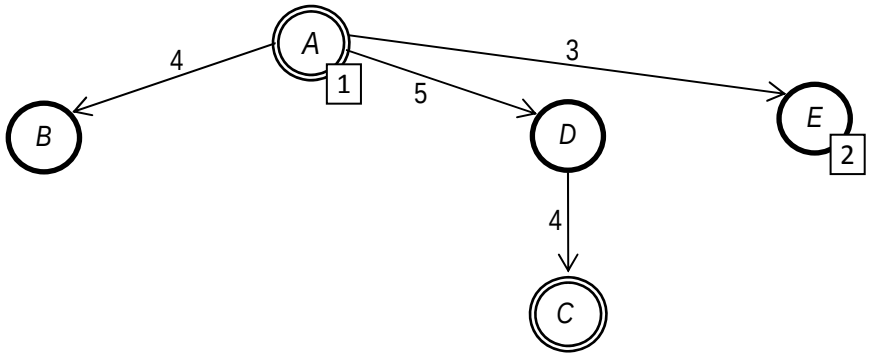


Figura 2.5a: Primera iteración de la búsqueda en anchura iterativa de derecha a izquierda, en la que como máximo un hijo es generado en cada expansión de un nodo padre.

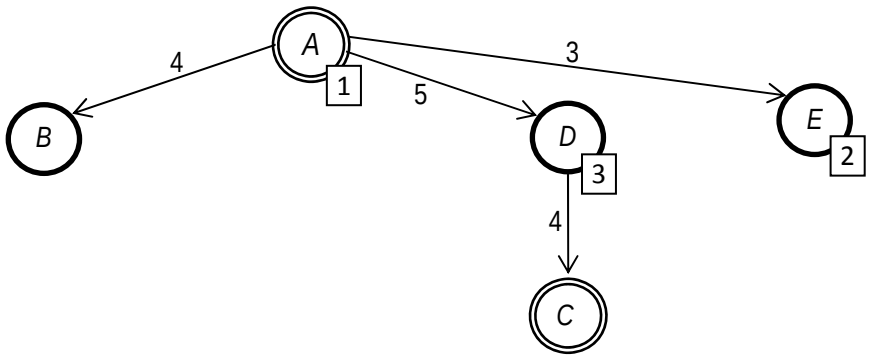


Figura 2.5b: Segunda iteración de la búsqueda en anchura iterativa de derecha a izquierda, en la que como máximo dos hijos son generados en cada expansión de un nodo padre. Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto, el algoritmo termina en ese momento sin que se lleguen a generar sus hijos.

5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

Este algoritmo ejecuta iterativamente varias búsquedas primero en profundidad, de manera que entre iteración e iteración se incrementa en una unidad la profundidad límite. Al principio (en la primera iteración), la profundidad límite es igual a 1, así que sólo se podrá expandir el nodo inicial, cuya profundidad es igual a 0. En las figuras 2.6a y 2.6b se muestran los órdenes de expansión de nodos para las dos iteraciones de búsqueda primero en profundidad que son necesarias para este ejemplo de búsqueda en profundidad iterativa.

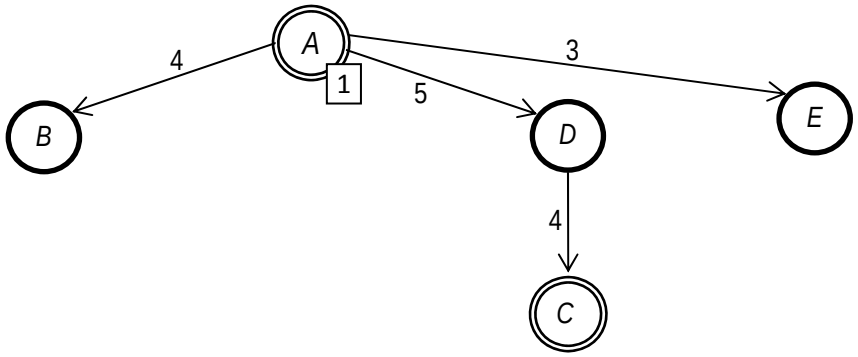


Figura 2.6a: Primera iteración de la búsqueda en profundidad iterativa de izquierda a derecha, en la que la profundidad límite es igual a 1 y únicamente son expandidos nodos con una profundidad máxima de 0.

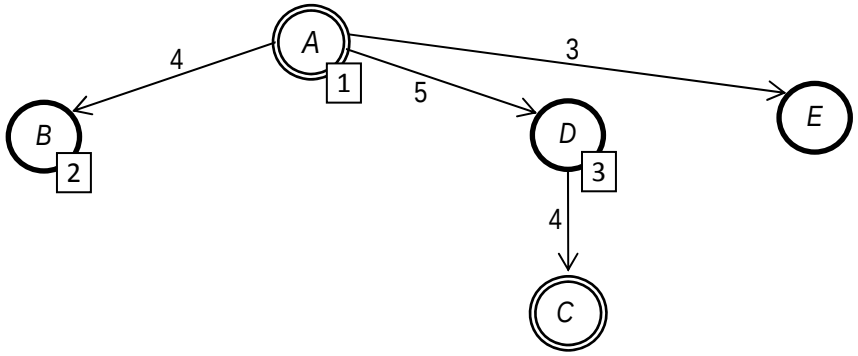


Figura 2.6b: Segunda iteración de la búsqueda en profundidad iterativa de izquierda a derecha, en la que la profundidad límite es igual a 2 y únicamente son expandidos nodos con una profundidad máxima de 1.

Observe que realmente el nodo meta no es expandido en esta última iteración porque se encuentre a la profundidad límite. Esta condición no se llega a comprobar, ya que justo antes se averigua que el nodo es meta y, por tanto, el algoritmo termina.

EJERCICIO 3:

Considere el espacio de búsqueda de la figura 3.1, que tiene forma de árbol, donde el nodo raíz del árbol es el nodo inicial, existe un único nodo meta y cada operador tiene asociado un coste. Describa cuál es el contenido de ABIERTA, previamente a cada extracción de un nodo de la misma, a partir de cada uno de los métodos siguientes de búsqueda sin información del dominio:

1. Búsqueda Primero en Anchura (de izquierda a derecha)
2. Búsqueda Primero en Profundidad (de derecha a izquierda)
3. Búsqueda de Coste Uniforme
4. Búsqueda en Anchura Iterativa (de derecha a izquierda)
5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

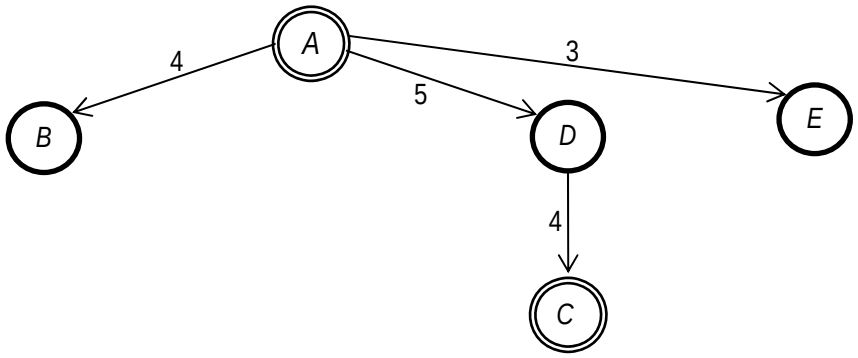


Figura 3.1: Árbol de búsqueda en el que el nodo inicial es *A*, el nodo meta es *C* y el coste de cada operador aparece al lado del arco que lo representa.

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 3:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- Cada uno de los cinco apartados, correspondiente a un método de búsqueda concreto, se puntúa sobre 2 puntos.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no gestiona ABIERTA en la forma debida: la puntuación del apartado bajaría 1.6 puntos si la respuesta dada varía significativamente de la correcta, mientras que si la respuesta dada varía de la correcta como consecuencia de un despiste no conceptual entonces la puntuación del apartado bajaría sólo 0.4 puntos en vez de los 1.6 puntos mencionados.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no finaliza cuando es debido, la puntuación del apartado bajaría 0.4 puntos. (Esto es independiente de la gestión de ABIERTA dada como respuesta, que ya ha sido valorada anteriormente con un máximo de 1.6 puntos.)

SOLUCIÓN DEL EJERCICIO 3 (por Severino Fernández Galán):

1. Búsqueda Primero en Anchura (de izquierda a derecha)

Este algoritmo usa ABIERTA como una cola, de manera que siempre se saca el primer nodo de la cola y se introducen sus hijos al final de la misma. El contenido de ABIERTA previamente a cada extracción de un nodo es el siguiente:

Antes de sacar *A*: {*A*}
Antes de sacar *B*: {*B, D, E*}
Antes de sacar *D*: {*D, E*}
Antes de sacar *E*: {*E, C*}
Antes de sacar *C*: {*C*}
META ENCONTRADA

Observe que, tras cada expansión del nodo sacado de ABIERTA, sus nodos hijos más a la izquierda se introducen en ABIERTA antes que los situados más a la derecha.

2. Búsqueda Primero en Profundidad (de derecha a izquierda)

Este algoritmo usa ABIERTA como una pila, de manera que siempre se saca el primer nodo de la pila y se introducen sus hijos al principio de la misma. El contenido de ABIERTA previamente a cada extracción de un nodo es el siguiente:

Antes de sacar *A*: {*A*}
Antes de sacar *E*: {*E, D, B*}
Antes de sacar *D*: {*D, B*}
Antes de sacar *C*: {*C, B*}
META ENCONTRADA

Observe que, tras cada expansión del nodo sacado de ABIERTA, sus nodos hijos más a la derecha se introducen en ABIERTA después que los situados más a la izquierda.

3. Búsqueda de Coste Uniforme

Este algoritmo mantiene ABIERTA ordenada según el coste del camino desde cada nodo al nodo inicial. En este ejercicio desharemos de forma arbitraria los posibles empates que surjan al determinar qué nodo se debe sacar de ABIERTA. El contenido de ABIERTA previamente a cada extracción de un nodo es el siguiente:

Antes de sacar *A*: {*A*(0)}
Antes de sacar *E*: {*E*(3), *B*(4), *D*(5)}
Antes de sacar *B*: {*B*(4), *D*(5)}
Antes de sacar *D*: {*D*(5)}
Antes de sacar *C*: {*C*(9)}
META ENCONTRADA

Observe que, tras cada expansión de un nodo, sus nodos hijos son introducidos en ABIERTA, donde todos los nodos quedan siempre ordenados por coste creciente. Para facilitar el seguimiento del algoritmo, entre paréntesis y al lado de cada nodo de ABIERTA se especifica el coste del camino para ir desde dicho nodo hasta el nodo inicial.

4. Búsqueda en Anchura Iterativa (de derecha a izquierda)

Debido a que este algoritmo es una iteración de la búsqueda primero en anchura, los dos gestionan ABIERTA del mismo modo, es decir, como una cola. La diferencia reside en que en la búsqueda en anchura iterativa en cada iteración se fija un número máximo de hijos que pueden ser generados al expandir un nodo padre. Este número empieza valiendo 1 y es incrementado en una unidad al principio de cada nueva iteración.

Iteración 1 (cada expansión de un nodo padre genera como máximo un hijo):

Antes de sacar *A*: {*A*}
Antes de sacar *E*: {*E*}

Iteración 2 (cada expansión de un nodo padre genera como máximo dos hijos):

Antes de sacar *A*: {*A*}
Antes de sacar *E*: {*E*, *D*}
Antes de sacar *D*: {*D*}
Antes de sacar *C*: {*C*}
META ENCONTRADA

5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

Debido a que este algoritmo es una iteración de la búsqueda primero en profundidad, los dos gestionan ABIERTA del mismo modo, es decir, como una pila. La diferencia reside en que en la búsqueda en profundidad iterativa en cada iteración se fija una profundidad límite propia. Este número empieza valiendo 1, lo cual permite expandir únicamente el nodo inicial, y es incrementado en una unidad al principio de cada nueva iteración.

Iteración 1 (profundidad límite igual a 1):

Antes de sacar *A*: {*A*}
Antes de sacar *B*: {*B, D, E*}
Nótese que *B* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *D*: {*D, E*}
Nótese que *D* no es expandido por coincidir su profundidad con la profundidad límite.
Antes de sacar *E*: {*E*}
Nótese que *E* no es expandido por coincidir su profundidad con la profundidad límite.

Iteración 2 (profundidad límite igual a 2):

Antes de sacar *A*: {*A*}
Antes de sacar *B*: {*B, D, E*}
Antes de sacar *D*: {*D, E*}
Antes de sacar *C*: {*C*, *E*}
META ENCONTRADA

EJERCICIO 4:

Considere el espacio de búsqueda de la figura 4.1, que tiene forma de árbol, donde el nodo raíz del árbol es el nodo inicial, existe un único nodo meta y cada operador tiene asociado un coste. Describa cuál es el contenido de TABLA_A, posteriormente a cada expansión de un nodo, a partir de cada uno de los métodos siguientes de búsqueda sin información del dominio:

1. Búsqueda Primero en Anchura (de izquierda a derecha)
2. Búsqueda Primero en Profundidad (de derecha a izquierda)
3. Búsqueda de Coste Uniforme
4. Búsqueda en Anchura Iterativa (de derecha a izquierda)
5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

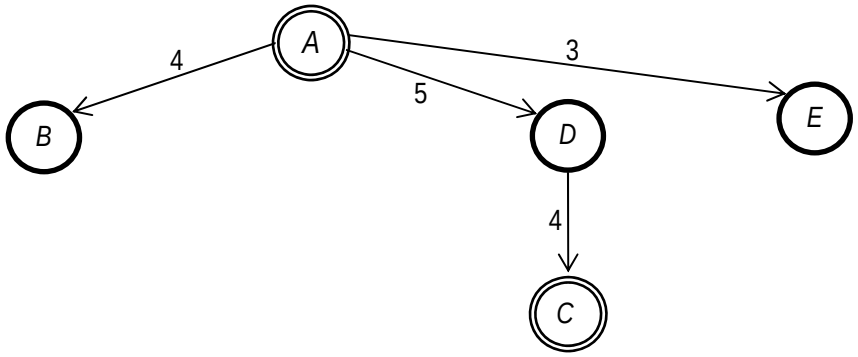


Figura 4.1: Árbol de búsqueda en el que el nodo inicial es *A*, el nodo meta es *C* y el coste de cada operador aparece al lado del arco que lo representa.

Para cada nodo de TABLA_A incluya la siguiente información: su nodo padre y el coste al nodo inicial. (En este ejercicio no es necesario incluir en TABLA_A información sobre los hijos de cada nodo expandido, ya que sólo existe un camino desde cada nodo al nodo inicial y, por tanto, el mejor camino desde cada nodo al nodo inicial no cambia a lo largo del proceso de búsqueda.)

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 4:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

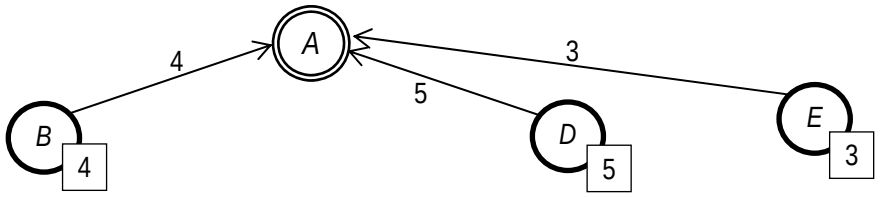
- Cada uno de los cinco apartados, correspondiente a un método de búsqueda concreto, se puntúa sobre 2 puntos.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no gestiona TABLA_A en la forma debida: la puntuación del apartado bajaría 1.6 puntos si la respuesta dada varía significativamente de la correcta, mientras que si la respuesta dada varía de la correcta como consecuencia de un despiste no conceptual entonces la puntuación del apartado bajaría sólo 0.4 puntos en vez de los 1.6 puntos mencionados.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no finaliza cuando es debido, la puntuación del apartado bajaría 0.4 puntos. (Esto es independiente de la gestión de TABLA_A dada como respuesta, que ya ha sido valorada anteriormente con un máximo de 1.6 puntos.)

SOLUCIÓN DEL EJERCICIO 4 (por Severino Fernández Galán):

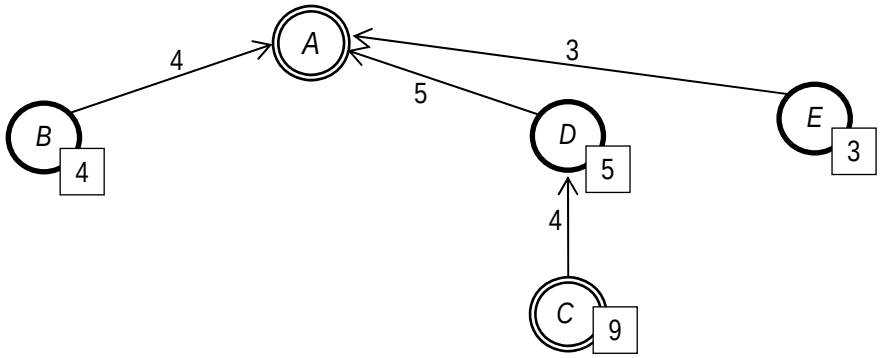
De cara a ilustrar la respuesta convenientemente, incluimos la información de TABLA_A gráficamente: por un lado, para cada nodo generado en una expansión trazamos un arco ascendente a su padre y, por otro lado, indicamos el coste al nodo inicial al lado de cada nodo generado.

1. Búsqueda Primero en Anchura (de izquierda a derecha)

- Inicialmente, *A* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *A*. A continuación expandimos el nodo inicial, generando sus nodos hijos. Desde cada nodo hijo trazamos un nodo ascendente a su nodo padre. Al lado de cada nodo hijo indicamos el coste desde dicho nodo al nodo inicial.



- Expandimos *B* y TABLA_A no varía. A continuación expandimos *D*:

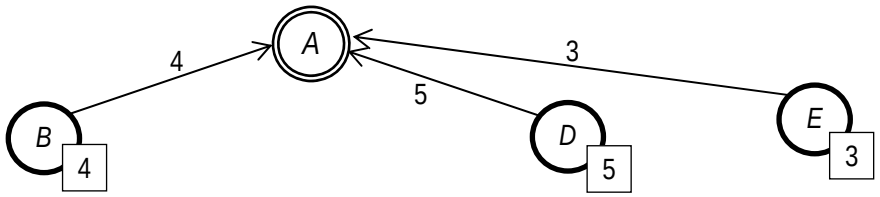


- Expandimos *E* y TABLA_A no varía. A continuación elegiríamos *C* para su expansión y llegaríamos a la meta. El camino hallado hasta la meta tiene coste 9.

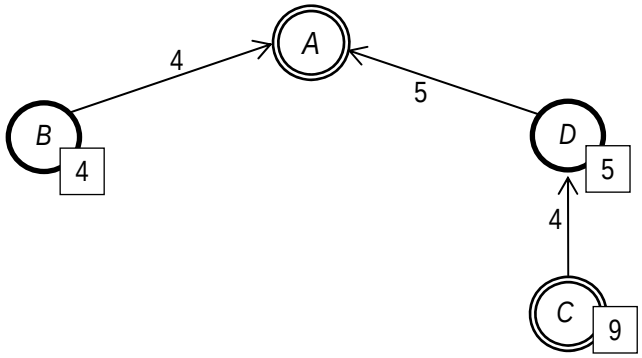
2. Búsqueda Primero en Profundidad (de derecha a izquierda)

Cuando sea necesario, en este algoritmo aplicaremos la función LimpiarTABLA_A (véase texto base, página 318). Tras la extracción de un nodo de ABIERTA y su posible expansión, dicha función elimina aquellos nodos que ya no son necesarios en el proceso de búsqueda de la meta. Esto permite preservar la linealidad de la complejidad espacial de este algoritmo con respecto a la profundidad de la solución.

- Inicialmente, *A* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *A*. A continuación, expandimos el nodo inicial.



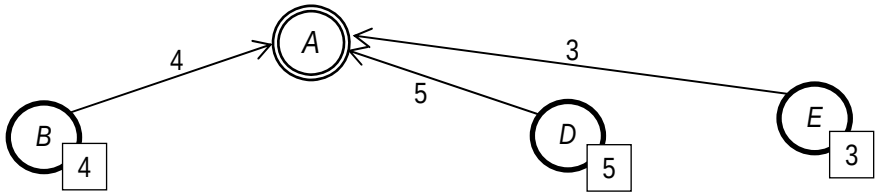
- Expandimos *E*. Seguidamente, aplicamos la función LimpiarTABLA_A a *E* por no tener hijos en ABIERTA y es sacado de TABLA_A. A continuación expandimos *D*:



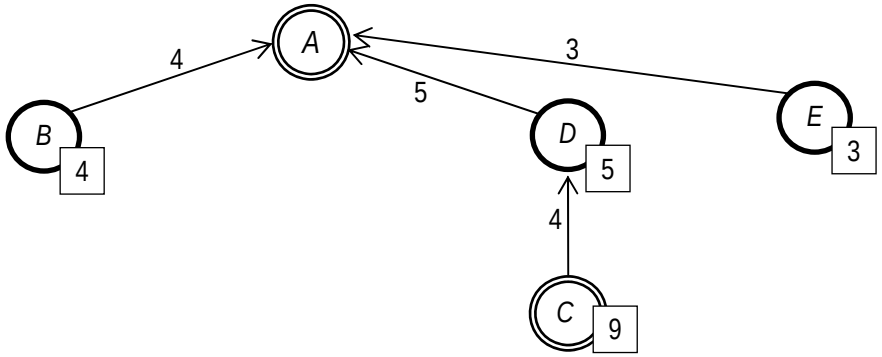
- A continuación elegiríamos *C* para su expansión y llegaríamos a la meta. El camino hallado hasta la meta tiene coste 9.

3. Búsqueda de Coste Uniforme

- Inicialmente, *A* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *A*. A continuación, expandimos el nodo inicial.



- Expandimos *E* y TABLA_A no varía. A continuación expandimos *B* y TABLA_A tampoco varía. Seguidamente expandimos *D*:

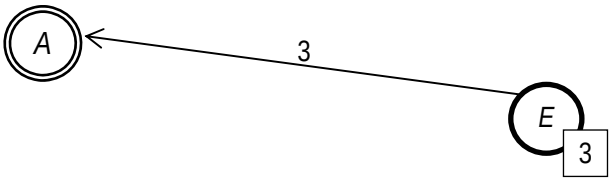


- Al intentar expandir *C*, alcanzamos la meta. El coste del camino solución hallado es 9.

4. Búsqueda en Anchura Iterativa (de derecha a izquierda)

<u>Iteración 1 (se genera como máximo 1 nodo hijo en cada expansión de un nodo padre)</u>:

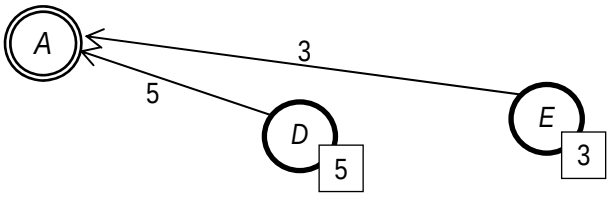
- Inicialmente, *A* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *A*. A continuación, expandimos el nodo inicial:



- Expandimos *E*, que no tiene hijos, con lo que TABLA_A no cambia y esta iteración termina.

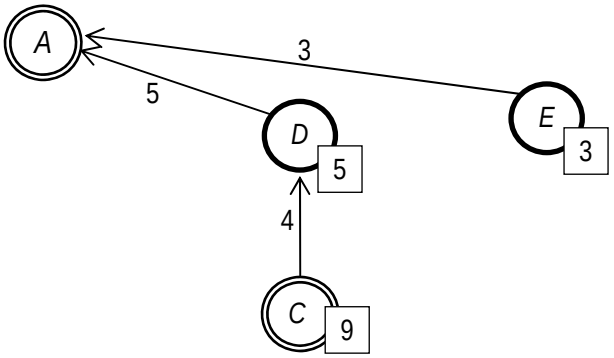
Iteración 2 (se generan como máximo 2 nodos hijo en cada expansión de un nodo padre):

- Inicialmente, *A* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *A*. A continuación, expandimos el nodo inicial:



- Expandimos *E*, que no tiene hijos, con lo que TABLA_A no cambia.

- Expandimos *D*:

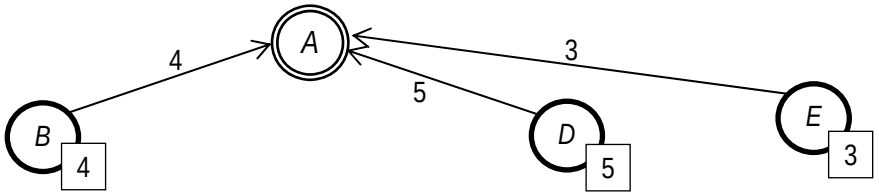


- Finalmente, al intentar expandir *C* alcanzamos la meta. El camino encontrado hasta el nodo inicial tiene coste 9.

5. Búsqueda en Profundidad Iterativa (de izquierda a derecha)

Iteración 1 (profundidad límite igual a 1):

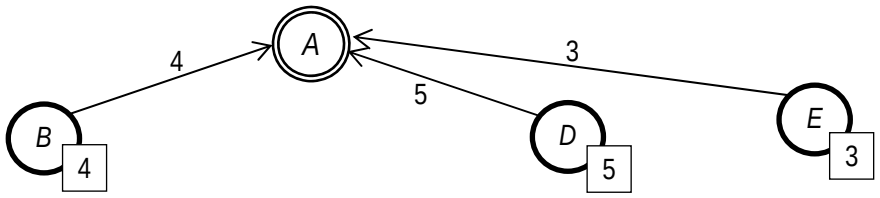
- Inicialmente, *A* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *A*. A continuación, expandimos el nodo inicial:



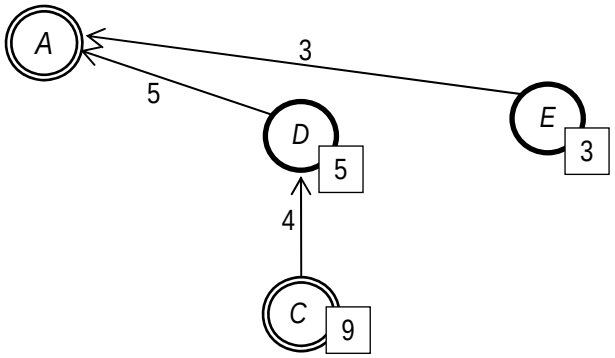
- Tras comprobar que *B* no es nodo meta, se le aplica la función LimpiarTABLA_A por estar en la profundidad límite y es sacado de TABLA_A. De igual manera, tras comprobar respectivamente que *D* y *E* no son nodo meta, se les aplica la función LimpiarTABLA_A por estar en la profundidad límite y son sacados de TABLA_A. Seguidamente, tras aplicarle la función LimpiarTABLA_A, *A* es sacado de TABLA_A por no tener hijos en ABIERTA. A continuación pasamos a la siguiente iteración.

Iteración 2 (profundidad límite igual a 2):

- Inicialmente, *A* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *A*. A continuación, expandimos el nodo inicial:



- Expandimos *B*. Seguidamente, aplicamos la función LimpiarTABLA_A a *B* por no tener hijos en ABIERTA y es sacado de TABLA_A. A continuación, expandimos *D*:



- Tras elegir *C* para su expansión y comprobar que es nodo meta, finaliza el algoritmo habiendo hallado un camino entre el nodo inicial y el nodo meta de coste 9.

EJERCICIO 5:

Considere el grafo de la figura 5.1, donde el nodo inicial es n_1 y donde el nodo meta es n_6. Cada arco u operador lleva asociado su coste y en cada nodo aparece la estimación de la menor distancia desde ese nodo a una meta. Aplique paso a paso el algoritmo A* al grafo dado, indicando de forma razonada la siguiente información en cada paso del algoritmo:

1. Qué nodo es expandido.
2. Cuál es el contenido de ABIERTA tras la expansión del nodo, indicando el valor de la función de evaluación heurística para cada nodo de ABIERTA.
3. Cuál es el contenido de TABLA_A tras la expansión del nodo. Para cada nodo de TABLA_A incluya la siguiente información:
    a) Su nodo padre que indique el camino de menor coste hasta el nodo inicial encontrado hasta el momento
    b) El coste del camino de menor coste hasta el nodo inicial encontrado hasta el momento
    c) Sus nodos hijos (si el nodo de TABLA_A actual ya ha sido expandido)

Por último, ¿cuál es el camino solución hallado y su coste?

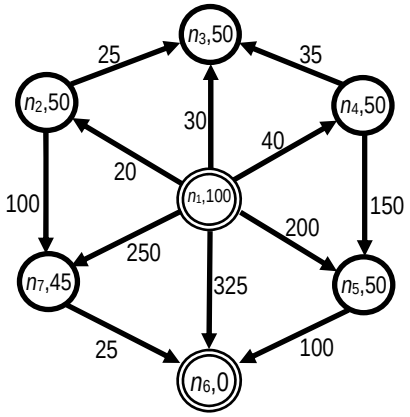


Figura 5.1: Grafo de búsqueda en el que el nodo inicial es n_1, el nodo meta es n_6, el coste de cada operador aparece al lado del arco que lo representa y al lado de cada nodo aparece la estimación de la menor distancia desde ese nodo a una meta.

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 5:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- El orden seguido en la expansión de los nodos se puntúa sobre 1.5 puntos.
- La forma en que se gestiona ABIERTA se puntúa sobre 3.75 puntos. Se hará especial énfasis en comprobar qué nodos hay en ABIERTA en cada paso del algoritmo y qué valores de la función de evaluación heurística se les asignan.
- La forma en que se gestiona TABLA_A se puntúa sobre 3.75 puntos. Se hará especial énfasis en comprobar qué nodos hay en TABLA_A en cada paso del algoritmo y qué padre mejor se les asigna (teniendo en cuenta las posibles reorientaciones o rectificaciones de enlaces)
- La correcta terminación del algoritmo se puntúa sobre 1 punto. Se hará especial énfasis en comprobar cuándo termina el algoritmo y qué camino solución devuelve.

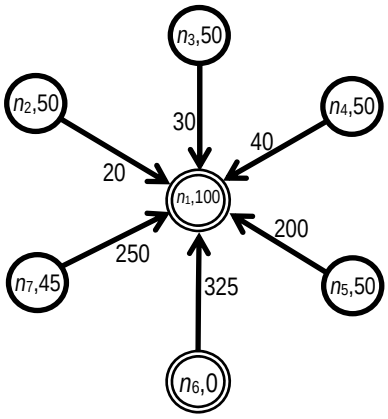
SOLUCIÓN DEL EJERCICIO 5 (por Severino Fernández Galán):

De cara a ilustrar la respuesta convenientemente, incluimos la información pedida sobre TABLA_A gráficamente. Para ello es necesario trazar, para cada nodo generado en una expansión, un arco ascendente a su padre expandido; además, hay que anotar para cada nodo su mejor padre encontrado hasta el momento (punto 3a del enunciado). De esta manera, siguiendo cada arco al mejor padre, se puede saber cuál es el mejor camino encontrado hasta el momento desde cada nodo al nodo inicial (punto 3b del enunciado). Además, los arcos ascendentes que llegan a un nodo ya expandido lo enlazan a sus nodos hijos (punto 3c del enunciado).

- PASO 0. El nodo inicial n_1 es introducido en ABIERTA y en TABLA_A, con lo que tenemos la siguiente situación:

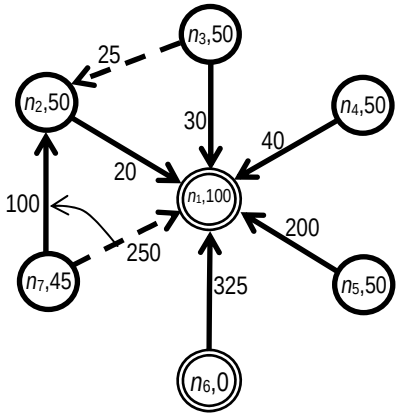
ABIERTA = {$\underline{n_1(0+100)}$}

- PASO 1. Expandimos el nodo n_1 de ABIERTA. Tras la expansión, la situación es la siguiente:



ABIERTA = {$\underline{n_2(20+50)}$, $n_3(30+50)$, $n_4(40+50)$, $n_5(200+50)$, $n_7(250+45)$, $n_6(325+0)$}

- PASO 2. Expandimos n_2 por ser el nodo de ABIERTA con menor valor de la función de evaluación heurística, $f=g+h$ (al ser $g=20$ y $h=50$).

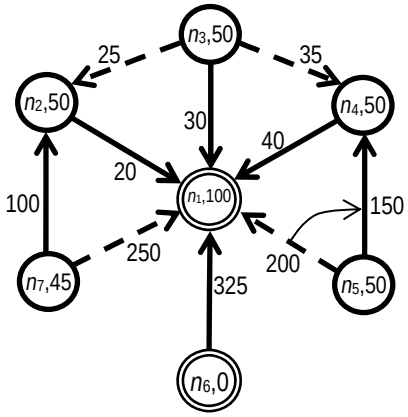


ABIERTA = {$\underline{n_3(30+50)}$, $n_4(40+50)$, $n_7(120+45)$, $n_5(200+50)$, $n_6(325+0)$}

Observe que el mejor camino desde n_3 al nodo inicial lo marca su padre n_1 (coste 30) y no su padre n_2 (coste 25+20=45). Por ello, el arco ascendente de n_3 a n_1 se marca con trazo continuo y el arco ascendente de n_3 a n_2 se marca con trazo discontinuo. Es importante darse cuenta que el conjunto de arcos con trazo continuo formará siempre un árbol en el grafo parcial de búsqueda desarrollado hasta el momento.

Observe también que ha habido una reorientación del mejor padre de n_7, que antes era n_1 y ahora pasa a ser n_2. El nuevo mejor coste de n_7 al nodo inicial, $g(n_7)$, es ahora 100+20=120, que se puede calcular a partir de los costes de los mejores arcos ascendentes hallados hasta el momento: $n_7 \rightarrow n_2$ y $n_2 \rightarrow n_1$.

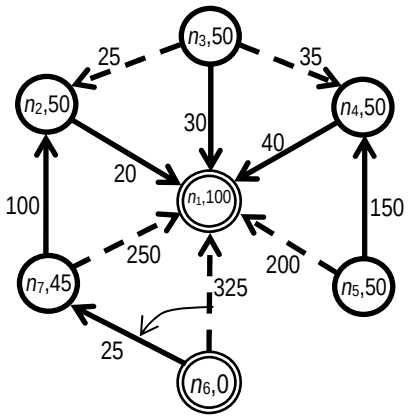
- PASOS 3 y 4. Expandimos n_3 y nada cambia en TABLA_A. A continuación expandimos n_4.



ABIERTA = {$\underline{n_7(120+45)}$, $n_5(190+50)$, $n_6(325+0)$}

Observe que ha habido una reorientación del mejor padre de n_5, que antes era n_1 y ahora pasa a ser n_4. El nuevo mejor coste de n_5 al nodo inicial, $g(n_5)$, es ahora 150+40=190, que se puede calcular a partir de los costes de los mejores arcos ascendentes hallados hasta el momento: $n_5 \rightarrow n_4$ y $n_4 \rightarrow n_1$.

- PASO 5. Expandimos n_7.



ABIERTA = {$\underline{n_6(145+0)}$, $n_5(190+50)$}

Observe que ha habido una reorientación del mejor padre de n_6, que antes era n_1 y ahora pasa a ser n_7. El nuevo mejor coste de n_6 al nodo inicial, $g(n_6)$, es ahora 25+100+20=145, que se puede calcular a partir de los costes de los mejores arcos ascendentes hallados hasta el momento: $n_6 \rightarrow n_7$, $n_7 \rightarrow n_2$ y $n_2 \rightarrow n_1$.

- PASO 6. Intentamos expandir n_6 y alcanzamos una meta, con lo que el algoritmo termina. El camino solución encontrado es: $n_1 \rightarrow n_2 \rightarrow n_7 \rightarrow n_6$, cuyo coste es 20+100+25=145.

EJERCICIO 6:

Considere el grafo de la figura 6.1, donde el nodo inicial es *D* y donde los nodos meta son desconocidos. Cada arco u operador lleva asociado su coste y en cada nodo aparece su valor de la función de evaluación heurística (que hay que minimizar). Aplique paso a paso el algoritmo de escalada o máximo gradiente al grafo dado. Para ello indique de forma razonada qué nodo se expande en cada paso y cuál es el nodo final devuelto por el algoritmo. Utilice como *criterio de selección* el de mejor vecino. Utilice como *criterio de terminación* el que no se hayan producido mejoras durante los tres últimos pasos del algoritmo.

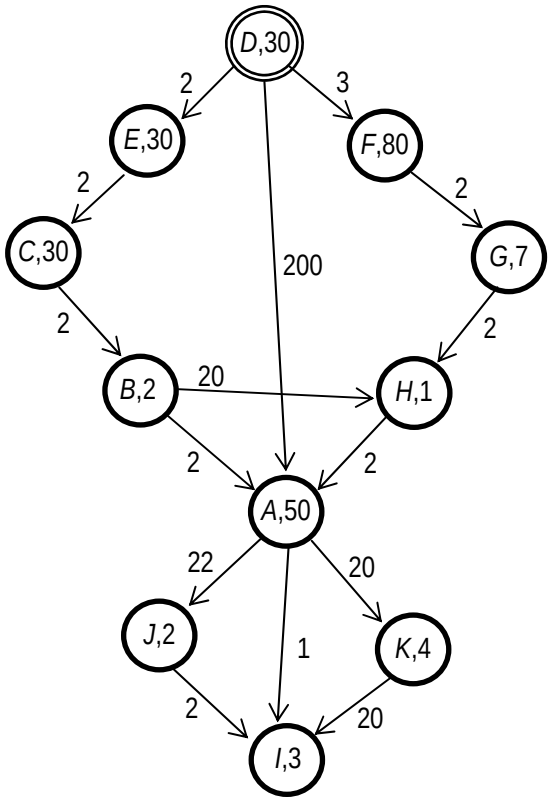


Figura 6.1: Grafo de búsqueda en el que el nodo inicial es *D*, los nodos meta son desconocidos, el coste de cada operador aparece al lado del arco que lo representa y al lado de cada nodo aparece el valor de su función de evaluación heurística (que hay que minimizar).

CRITERIOS DE EVALUACIÓN DEL EJERCICIO 6:

La evaluación sobre 10 puntos de este ejercicio se realizaría atendiendo a los siguientes criterios:

- La correcta aplicación en cada paso del algoritmo del *criterio de selección* del vecino o hijo del nodo actual (qué vecino o hijo del nodo actual es considerado como candidato para sustituirlo) se puntúa sobre 4.5 puntos.
- La correcta aplicación en cada paso del algoritmo del *criterio de aceptación* del vecino o hijo seleccionado (si el vecino o hijo candidato sustituye o no finalmente al nodo actual) se puntúa sobre 4.5 puntos.
- La correcta aplicación del *criterio de finalización* del algoritmo se puntúa sobre 1 punto.

SOLUCIÓN DEL EJERCICIO 6 (por Severino Fernández Galán):

- PASO 1: Al principio expandimos el nodo inicial *D*, generando sus nodos hijos {*E*(30), *A*(50), *F*(80)}. Seleccionamos el nodo *E* por ser el mejor de los nodos hijos. A continuación aceptamos el nodo *E* como nuevo nodo actual en sustitución de *D*, debido a que el valor de la función de evaluación heurística de *E* es mejor o igual que el de *D* ($30 \leq 30$).

- PASO 2: Expandimos el nodo actual *E* y generamos sus hijos: {*C*(30)}. Seleccionamos el nodo *C* por ser el mejor de los nodos hijos. Seguidamente aceptamos el nodo *C* como nuevo nodo actual en sustitución de *E*, debido a que el valor de la función de evaluación heurística de *C* es mejor o igual que el de *E* ($30 \leq 30$). (Observe que la condición de terminación del algoritmo todavía no se cumple tras este paso, ya que sólo hemos completado dos pasos sin mejora, cuando la condición de terminación del enunciado nos indica que tienen que ser tres.)

- PASO 3: Expandimos el nodo actual *C* y generamos sus hijos: {*B*(2)}. Seleccionamos el nodo *B* por ser el mejor de los nodos hijos. A continuación aceptamos el nodo *B* como nuevo nodo actual en sustitución de *C*, debido a que el valor de la función de evaluación heurística de *B* es mejor o igual que el de *C* (se cumple que $2 \leq 30$).

- PASO 4: Expandimos el nodo actual *B*, generando sus nodos hijos {*A*(50), *H*(1)}. Seleccionamos el nodo *H* por ser el mejor de los nodos hijos. A continuación aceptamos el nodo *H* como nuevo nodo actual en sustitución de *B*, debido a que el valor de la función de evaluación heurística de *H* es mejor o igual que el de *B* ($1 \leq 2$).

- PASO 5: Expandimos el nodo actual *H* y generamos sus hijos: {*A*(50)}. Seleccionamos el nodo *A* por ser el mejor de los nodos hijos. A continuación rechazamos el nodo *A* como nuevo nodo actual en sustitución de *H*, debido a que el valor de la función de evaluación heurística de *A* no es mejor o igual que el de *H* (no se cumple que $50 \leq 1$).

La búsqueda terminaría en este punto. El algoritmo devolvería el nodo *H*(1) como mejor nodo encontrado.