Asignatura de Grado: FUNDAMENTOS DE INTELIGENCIA ARTIFICIAL
Curso: 2020 - 2021
Equipo docente: Severino Fernández Galán y Ángeles Manjarrés Riesco

Soluciones de la Prueba de Evaluación Continua 1 (PEC1): *BÚSQUEDA EN UN ESPACIO DE ESTADOS*

1. INTRODUCCIÓN

Esta asignatura requiere la realización de dos actividades evaluables para las que no será necesario que el alumno acuda al Centro Asociado, ya que podrán realizarse a distancia. El objetivo de estas actividades evaluables es afianzar y poner en práctica los contenidos teóricos de la asignatura.

La primera de las actividades evaluables está relacionada con los métodos de búsqueda en espacios de estados. El tiempo estimado para la realización de la misma es de 15 horas. La entrega del material elaborado por el alumno como contestación a las actividades evaluables se realizará a través del curso virtual y un profesor tutor se encargará de su corrección. El plazo de entrega finaliza el día 1 de abril de 2021.

La nota final de las actividades evaluables será la media de las puntuaciones obtenidas en cada una de las dos actividades evaluables y constituirá un 20% de la nota final de la asignatura (siempre que la nota de la prueba presencial sea igual o superior a 4 –sobre 10–). Es importante tener en cuenta que sólo se corregirán las actividades evaluables una vez durante el curso (previamente a la convocatoria de junio). Por tanto, la nota asignada a las actividades evaluables de cara a junio será la única válida tanto para la convocatoria de junio como para la de septiembre. En caso de que el alumno no realice la entrega de actividades evaluables de cara a la convocatoria de junio, se le asignará un cero al 20% de la nota final correspondiente a las actividades evaluables, tanto para la convocatoria de junio como para la de septiembre.

2. ENUNCIADO DE LA PEC 1: "BÚSQUEDA EN UN ESPACIO DE ESTADOS"

La actividad evaluable sobre búsqueda en un espacio de estados consistirá en la realización por parte del alumno de seis ejercicios prácticos. La nota sobre 10 de esta actividad evaluable se calcula del siguiente modo:

$$\text{NOTA} = 0.125 \cdot nota_1 + 0.1 \cdot nota_2 + 0.2 \cdot nota_3 + 0.2 \cdot nota_4 + 0.3 \cdot nota_5 + 0.075 \cdot nota_6,$$

donde $nota_i$ representa la nota sobre 10 del ejercicio i.

El alumno deberá seguir los contenidos del texto base de la asignatura a la hora de dar respuesta a estos ejercicios, cuya temática y criterios de evaluación se especifican a continuación.

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 puzzle 2x3* 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 puzzle 2x3 existe un conjunto de 5 fichas cuadradas de idéntico tamaño, que se pueden mover en una cuadrícula 2x3. Estas fichas están numeradas del 1 al 5 y en todo momento queda una casilla vacía a la que se puede desplazar horizontal o verticalmente cualquier ficha contigua del puzzle. Suponga que se parte de la situación inicial siguiente:

4	1	3
	2	5

Dado un estado cualquiera del puzzle, se pueden generar estados hijos suyos en el espacio de búsqueda al desplazar a la casilla vacía cualquiera de sus fichas contiguas vertical u horizontalmente. Considere también que el estado meta del puzzle es el siguiente:

1	2	3
4	5	

¿Cuál es la solución menos costosa para este problema, tal como ha sido descrito?

NOTA:

Dado el amplio número de estados diferentes en este problema, resulta muy difícil representar en una tabla o en un grafo el conjunto total de estados del espacio de búsqueda. Por ello, se pide únicamente dibujar el subgrafo resultante de partir del estado inicial y generar aquellos estados cuya profundidad sea menor o igual que el coste del mejor camino a la meta.

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 la matriz:

m_{11}	m_{12}	m_{13}
m_{21}	m_{22}	m_{23}

donde $m_{ij} \in \{\varnothing, 1, 2, 3, 4, 5\}$ es el contenido de la casilla (i, j), con $i \in \{1, 2\}$ y $j \in \{1, 2, 3\}$. La secuencia de variables $\{m_{11}, m_{12}, m_{13}, m_{21}, m_{22}, m_{23}\}$ puede almacenar una permutación cualquiera de los valores $\{\varnothing, 1, 2, 3, 4, 5\}$. Nótese que existen 6x5x4x3x2x1 = 720 estados posibles en el problema planteado.

El **estado inicial** descrito en el enunciado se representa como:

4	1	3
	2	5

Por otro lado, el **estado meta** es:

1	2	3
4	5	

Consideramos cuatro **operadores** posibles {N, S, E, O}, correspondientes al desplazamiento de la casilla vacía hacia el Norte, Sur, Este u Oeste respectivamente. Se puede considerar que cada operador tiene **coste** unidad, así que el coste de un camino es el número de arcos que lo componen.

Según las condiciones expuestas en la nota del enunciado, el **subespacio de búsqueda pedido** que parte del estado inicial se puede representar mediante un grafo dirigido tal como se muestra en la figura 1.1. Se ha utilizado color rojo para marcar el estado inicial y color verde para marcar el estado meta. Por otra parte, obsérvese que no todos los estados posibles son alcanzables desde el estado inicial. Por ejemplo, desde el estado inicial del enunciado no se podría alcanzar el estado siguiente:

4	1	5
	2	3

La **solución menos costosa** para el problema planteado está marcada en trazo discontinuo en la figura 1.1 y tiene un coste de 4.

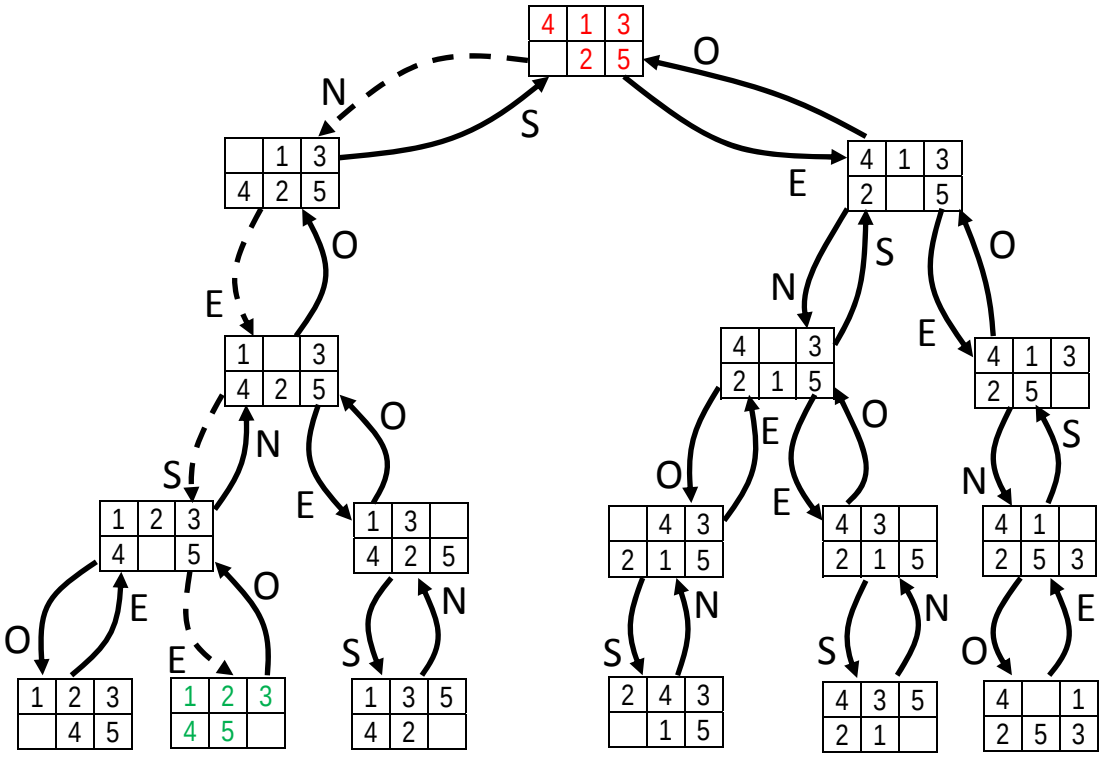


Figura 1.1: Grafo dirigido que representa el subespacio de búsqueda pedido en el enunciado para el problema del puzzle 2x3. El camino de menor coste hasta el estado meta se representan en trazo discontinuo.

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)

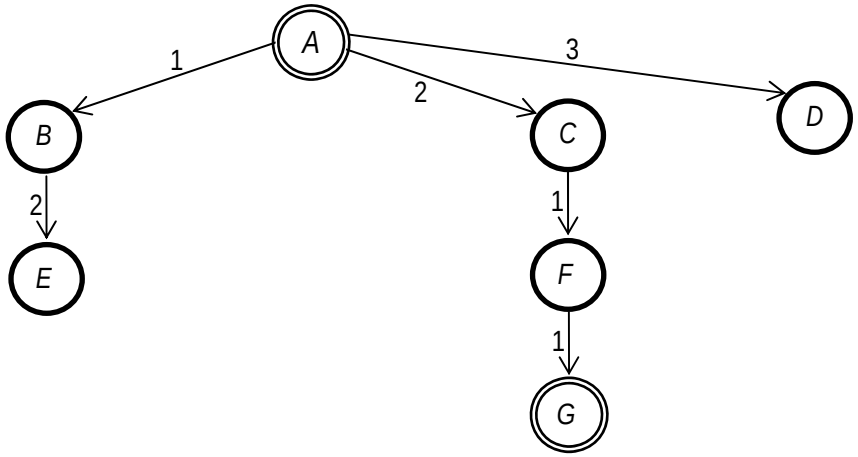


Figura 2.1: Árbol de búsqueda en el que el nodo inicial es *A*, el nodo meta es *G* y el coste de cada operador aparece al lado del arco que lo representa.

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.

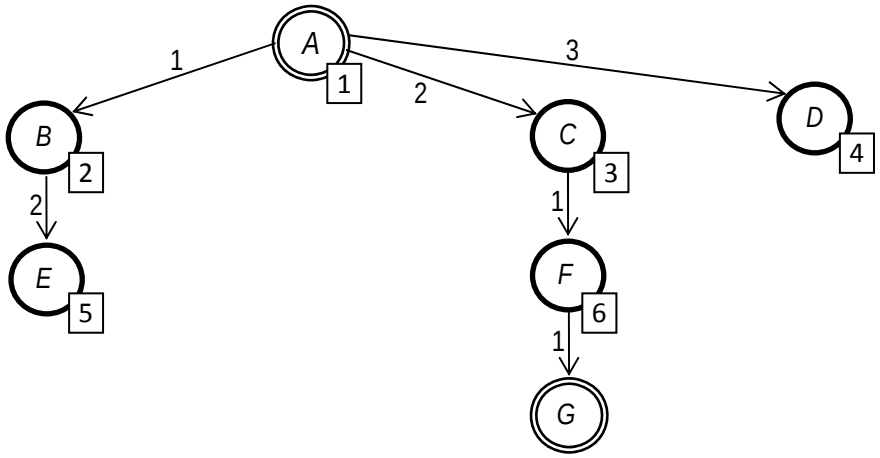


Figura 2.2: Orden de expansión de nodos para el árbol de la figura 2.1 según la búsqueda primero en anchura (de izquierda a derecha). Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto,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

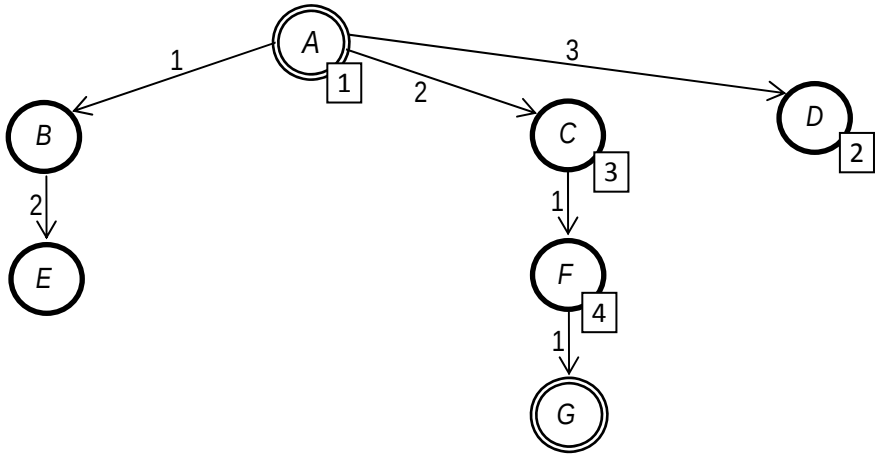


Figura 2.3: Orden de expansión de nodos para el árbol de la figura 2.1 según la búsqueda primero en profundidad (de derecha a izquierda). Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por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

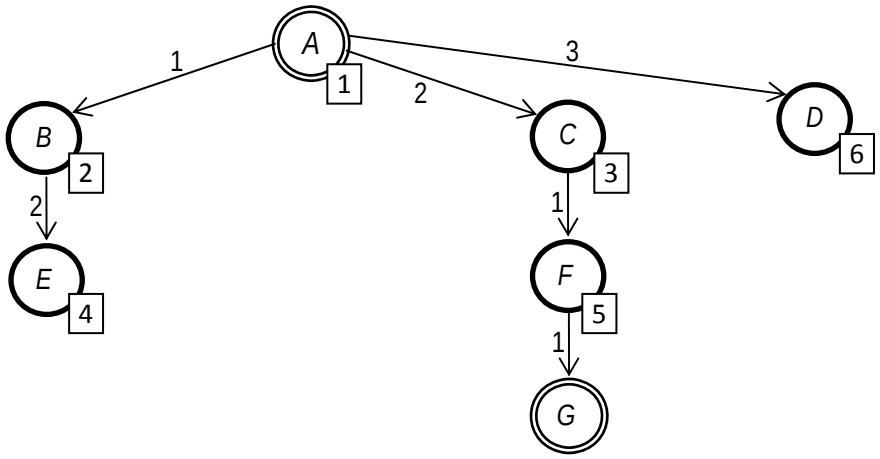


Figura 2.4: Orden de expansión de nodos para el árbol de la figura 2.1 según la búsqueda de coste uniforme. Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto, el algoritmo termina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

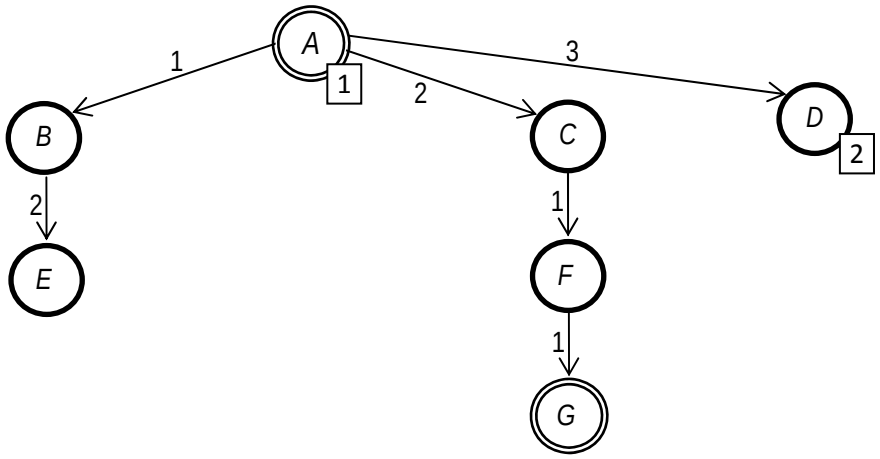


Figura 2.5a: Primera iteración de la búsqueda en anchura iterativa de derecha a izquierda, en la que como máximo un hijo es generado en cada expansión de un nodo padre.

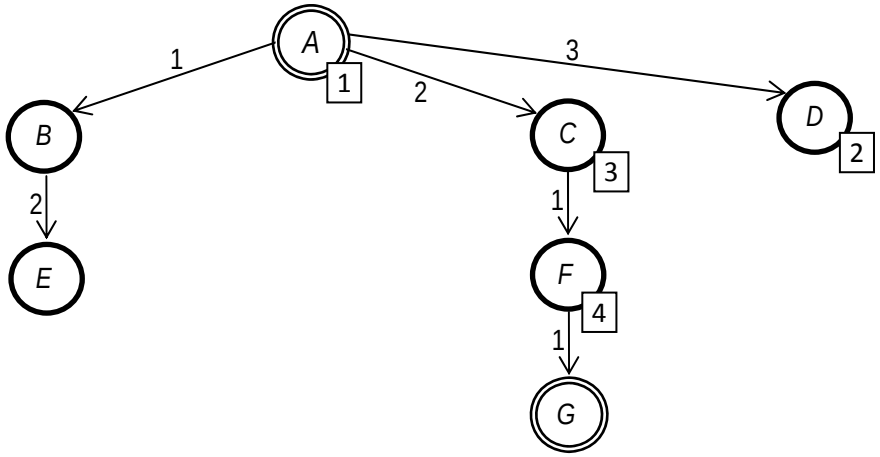


Figura 2.5b: Segunda iteración de la búsqueda en anchura iterativa de derecha a izquierda, en la que como máximo dos hijos son generados en cada expansión de un nodo padre. Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se 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, 2.6b y 2.6c se muestran los órdenes de expansión de nodos para las tres iteraciones de búsqueda primero en profundidad que son necesarias para este ejemplo de búsqueda en profundidad iterativa.

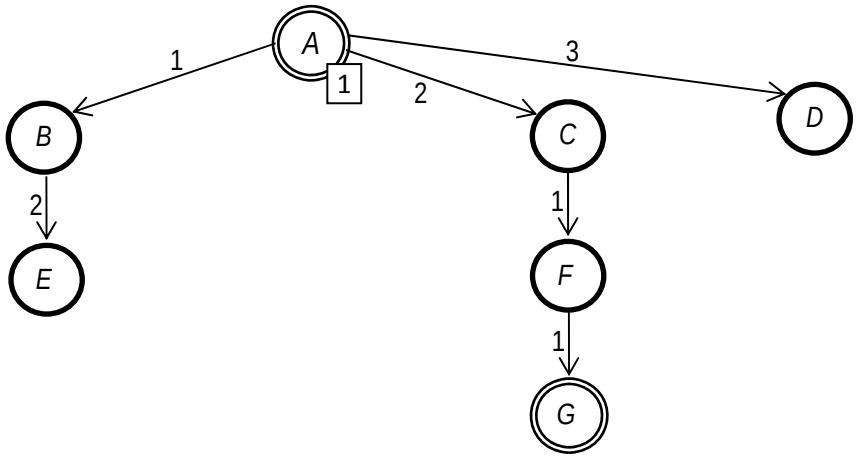


Figura 2.6a: Primera iteración de la búsqueda en profundidad iterativa de izquierda a derecha, en la que la profundidad límite es igual a 1 y únicamente son expandidos nodos con una profundidad máxima de 0.

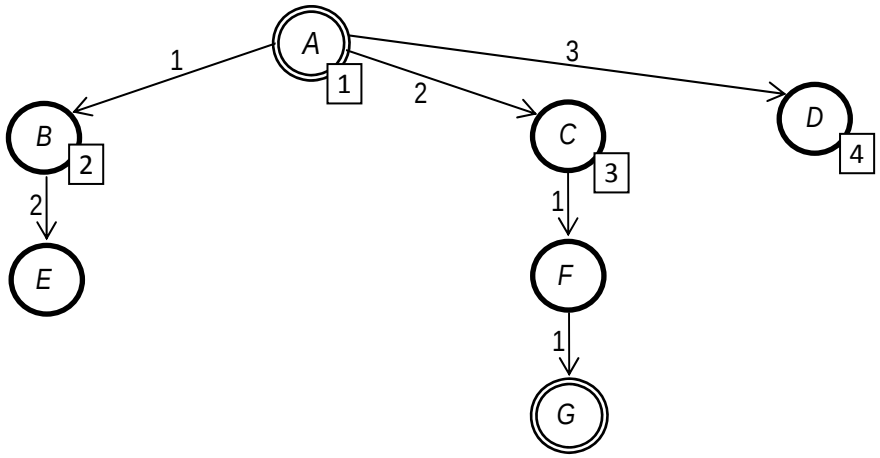


Figura 2.6b: Segunda iteración de la búsqueda en profundidad iterativa de izquierda a derecha, en la que la profundidad límite es igual a 2 y únicamente son expandidos nodos con una profundidad máxima de 1.

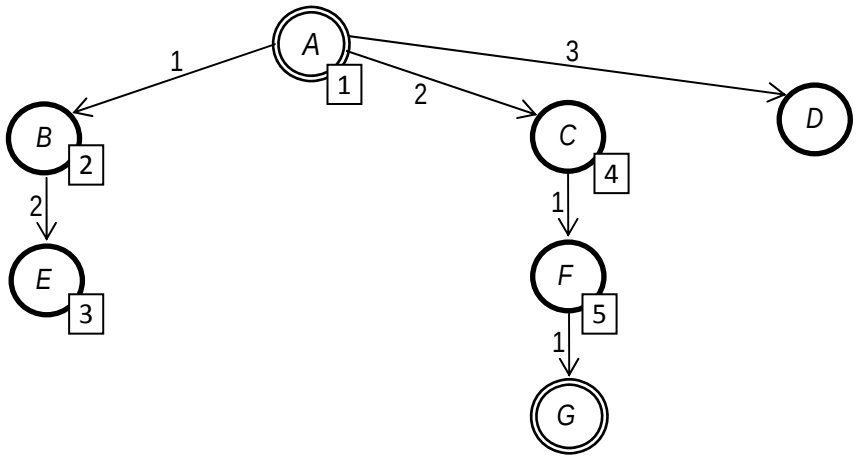


Figura 2.6c: Tercera iteración de la búsqueda en profundidad iterativa de izquierda a derecha, en la que la profundidad límite es igual a 3 y únicamente son expandidos nodos con una profundidad máxima de 2.

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)

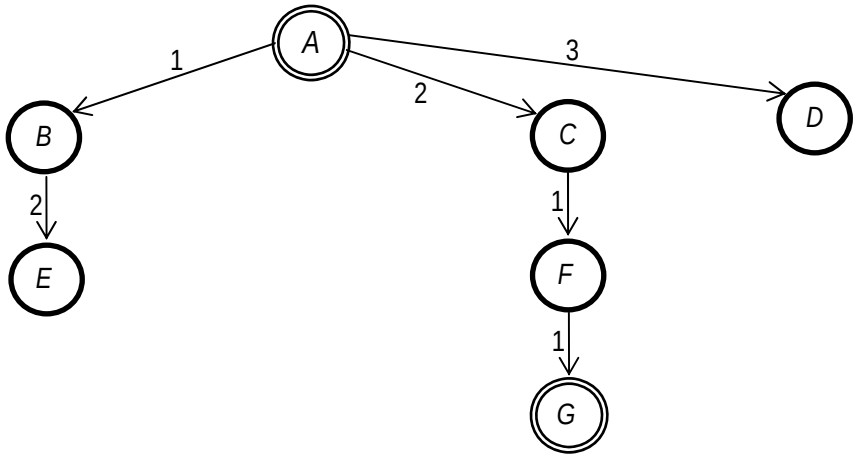


Figura 3.1: Árbol de búsqueda en el que el nodo inicial es *A*, el nodo meta es *G* y el coste de cada operador aparece al lado del arco que lo representa.

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, C, D*}
Antes de sacar *C*: {*C*, *D, E*}
Antes de sacar *D*: {*D*, *E, F*}
Antes de sacar *E*: {*E, F*}
Antes de sacar *F*: {*F*}
Antes de sacar *G*: {*G*}
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 *D*: {*D, C, B*}
Antes de sacar *C*: {*C, B*}
Antes de sacar *F*: {*F, B*}
Antes de sacar *G*: {*G, 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), *C*(2), *D*(3)}
Antes de sacar *C*: {*C*(2), *E*(3), *D*(3)}
Antes de sacar *E*: {*E*(3), *F*(3), *D*(3)} (empate resuelto arbitrariamente a favor de *E*)
Antes de sacar *F*: {*F*(3), *D*(3)} (empate resuelto arbitrariamente a favor de *F*)
Antes de sacar *D*: {*D*(3), *G*(4)}
Antes de sacar *G*: {*G*(4)}
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 *D*: {*D*}

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

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

Iteración 3 (profundidad límite igual a 3):

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

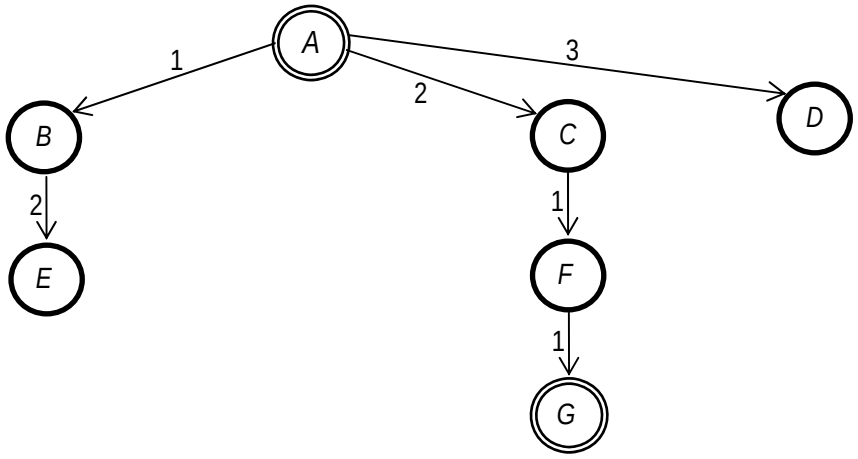


Figura 4.1: Árbol de búsqueda en el que el nodo inicial es *A*, el nodo meta es *G* y el coste de cada operador aparece al lado del arco que lo representa.

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:

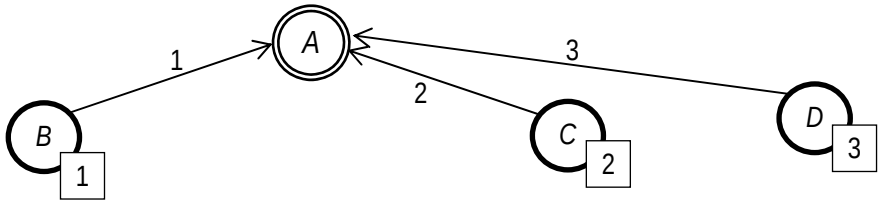
- Cada uno de los cinco apartados, correspondiente a un método de búsqueda concreto, se puntúa sobre 2 puntos.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no gestiona TABLA_A en la forma debida: la puntuación del apartado bajaría 1.6 puntos si la respuesta dada varía significativamente 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):

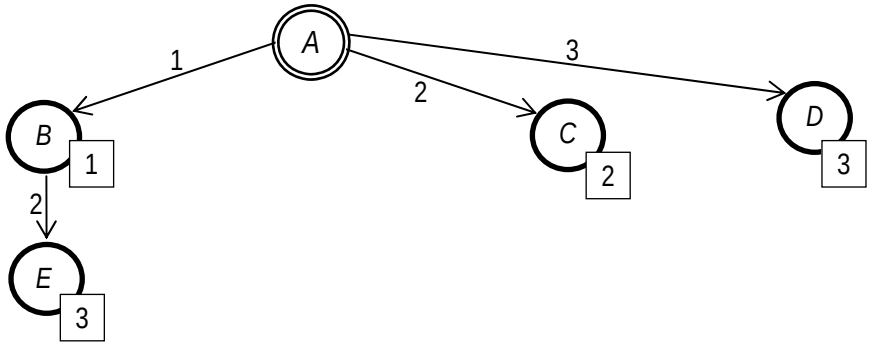
De cara a ilustrar la respuesta convenientemente, incluimos la información de TABLA_A gráficamente: por un lado, para cada nodo generado en una expansión trazamos un arco ascendente a su padre y, por otro lado, indicamos el coste al nodo inicial al lado de cada nodo generado.

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

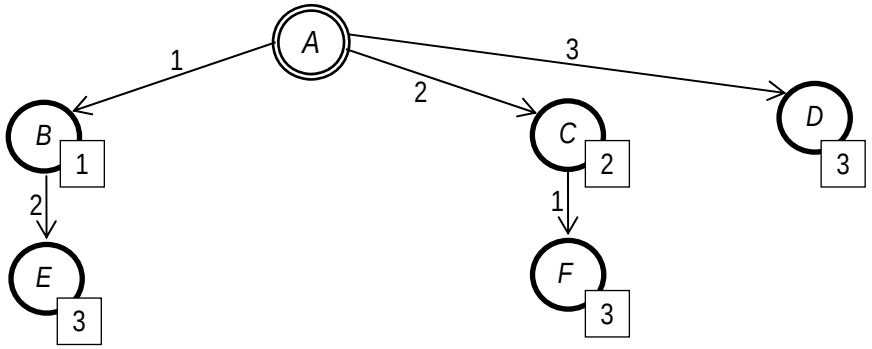
- Inicialmente, *A* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *A*. A continuación expandimos el nodo inicial, generando sus nodos hijos. Desde cada nodo hijo trazamos un nodo ascendente a su nodo padre. Al lado de cada nodo hijo indicamos el coste desde dicho nodo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



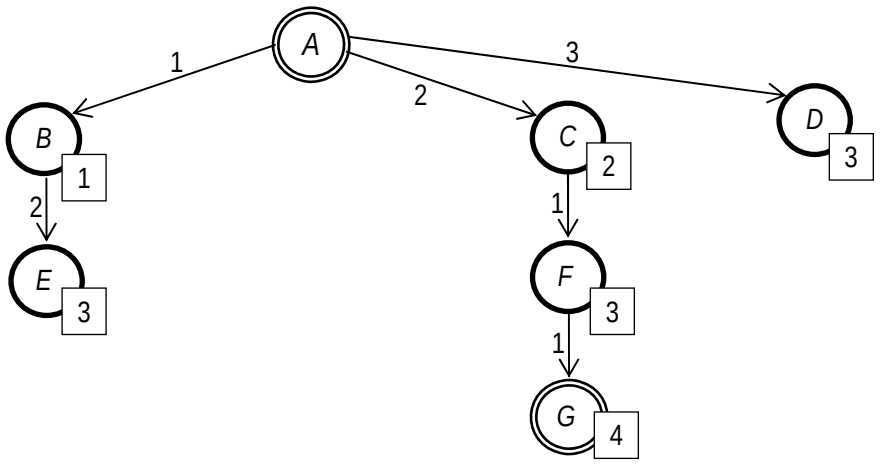
- Expandimos *B*:



- Expandimos *C*:



- Expandimos *D* y TABLA_A no varía. Del mismo modo, expandimos *E* y TABLA_A no varía. A continuación expandimos *F*:

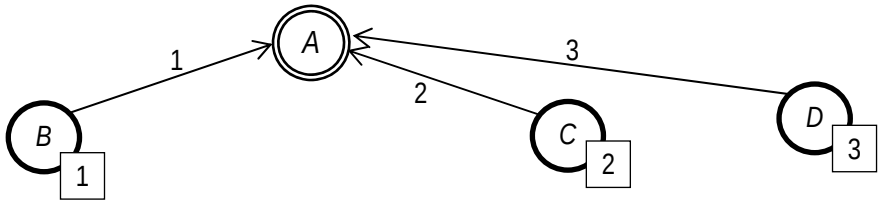


- Seguidamente, elegiríamos *G* para su expansión y llegaríamos a la meta. El camino hallado hasta la meta tiene coste 4.

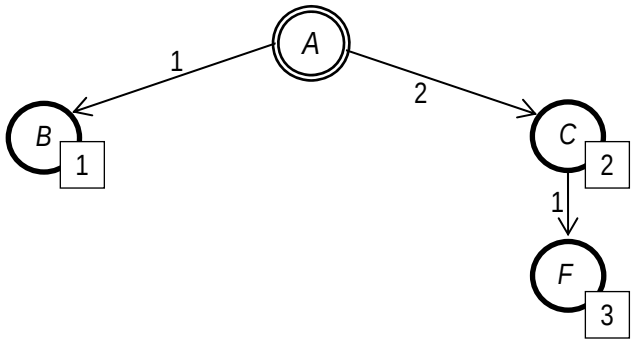
2. Búsqueda Primero en Profundidad (de derecha a izquierda)

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.

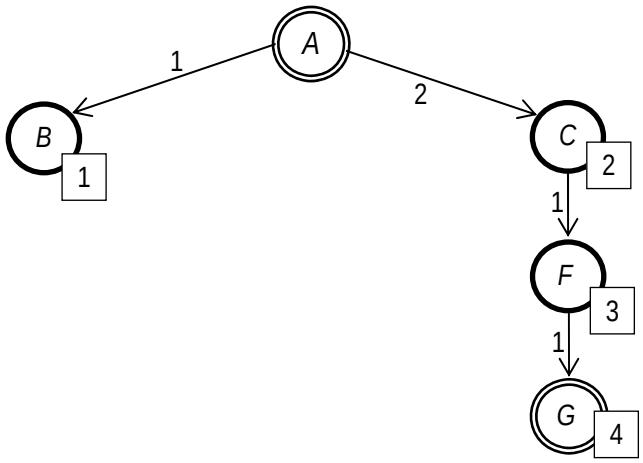
- Inicialmente, *A* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *A*. A continuación, expandimos el nodo inicial.



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



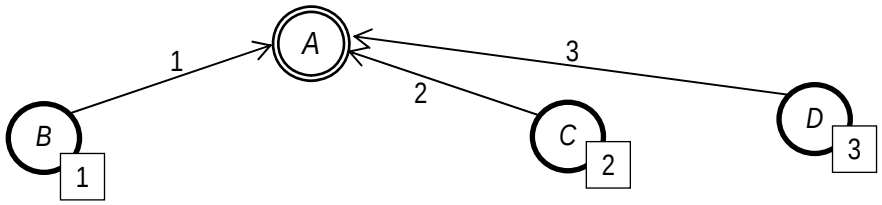
- Expandimos *F*:



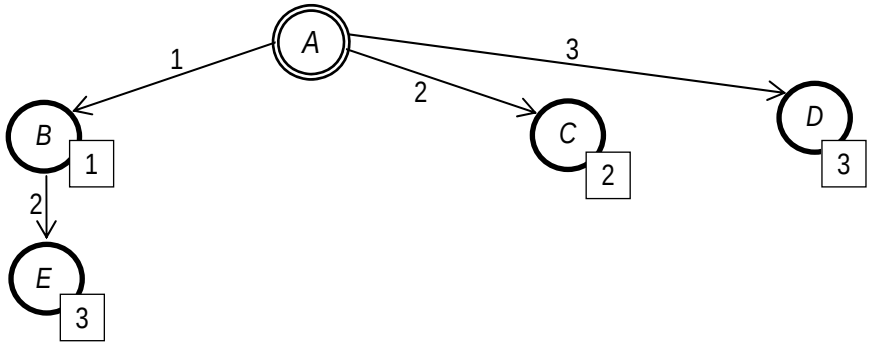
- A continuación elegiríamos *G* para su expansión y llegaríamos a la meta. El camino hallado hasta la meta tiene coste 4.

3. Búsqueda de Coste Uniforme

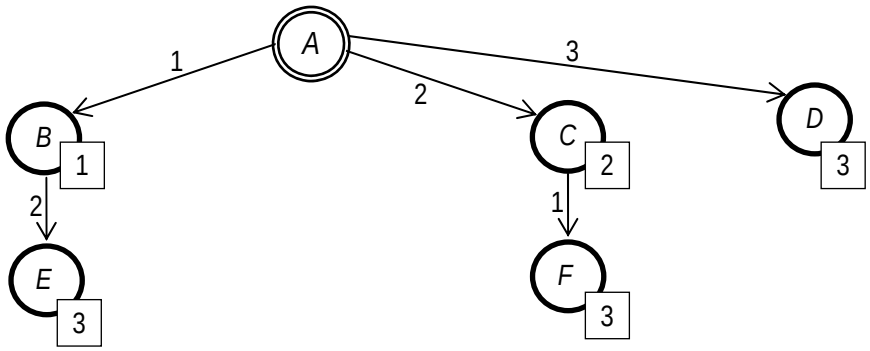
- Inicialmente, *A* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *A*. A continuación, expandimos el nodo inicial.



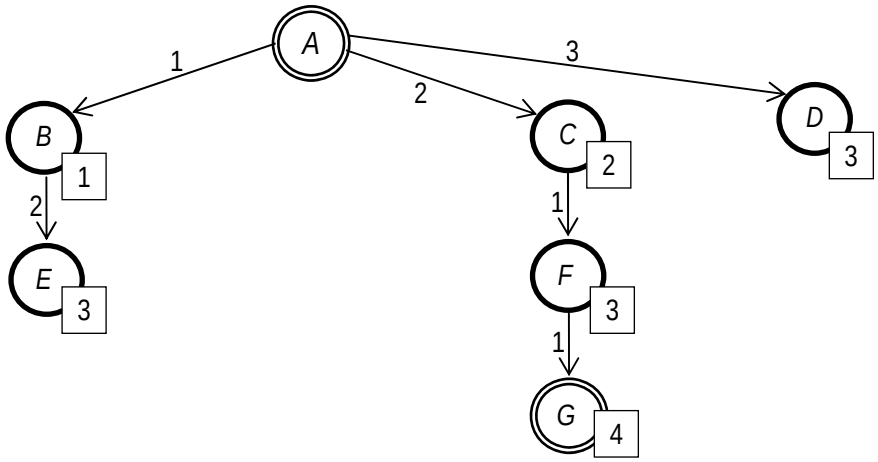
- Expandimos *B*:



- Expandimos *C*:



- Expandimos arbitrariamente *E* para deshacer el triple empate y TABLA_A no varía. Seguidamente, expandimos arbitrariamente *F* para deshacer el doble empate:

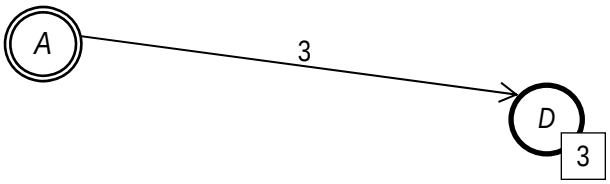


- Expandimos *D* y TABLA_A no varía. A continuación, al intentar expandir *G*, alcanzamos la meta. El coste del camino solución hallado es 4.

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):

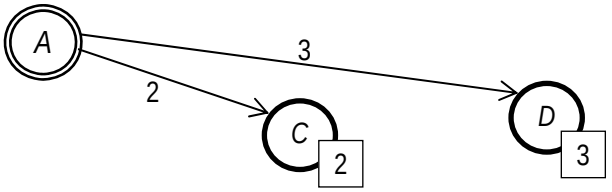
- Inicialmente, *A* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *A*. A continuación, expandimos el nodo inicial:



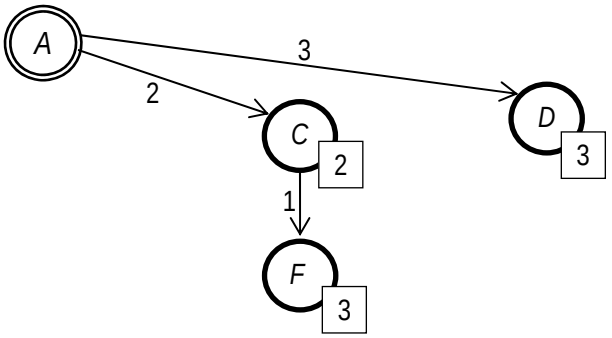
- Expandimos *D*, que no tiene hijos, con lo que TABLA_A no cambia y esta iteración termina.

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

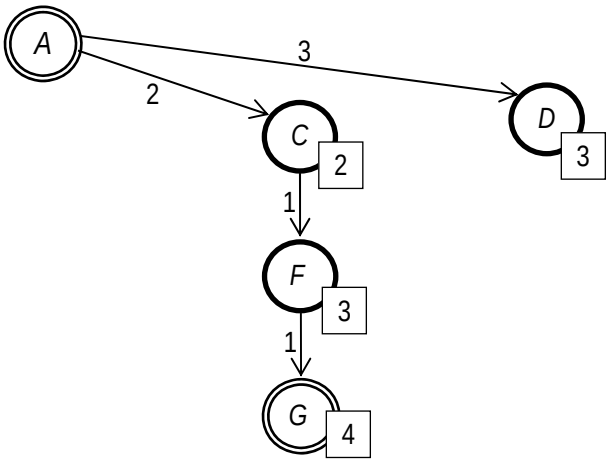
- Inicialmente, *A* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *A*. A continuación, expandimos el nodo inicial:



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



- Expandimos *F*:

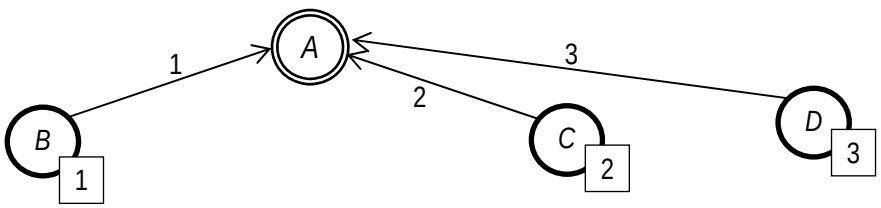


- Finalmente, al intentar expandir *G* alcanzamos la meta. El camino encontrado hasta el nodo inicial tiene coste 4.

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

Iteración 1 (profundidad límite igual a 1):

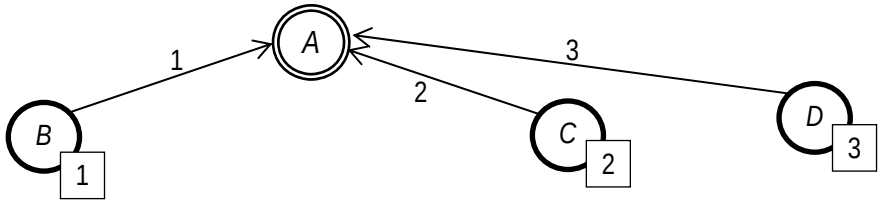
- Inicialmente, *A* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *A*. A continuación, expandimos el nodo inicial:



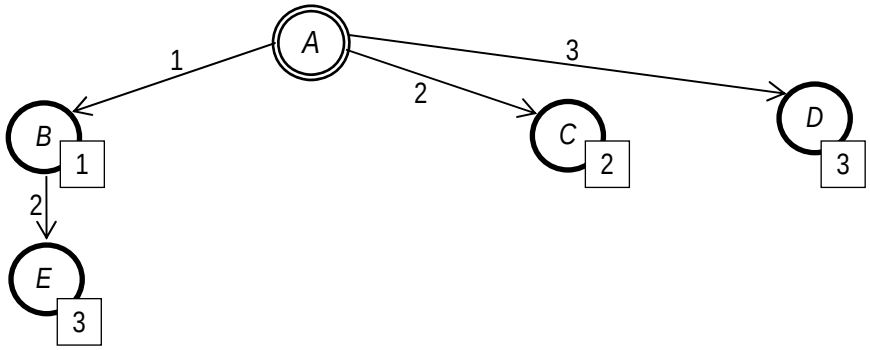
- Tras comprobar que *B* no es nodo meta, se le aplica la función LimpiarTABLA_A por estar en la profundidad límite y es sacado de TABLA_A. De igual manera, tras comprobar respectivamente que *C* y *D* no son nodo meta, se les aplica la función LimpiarTABLA_A por estar en la profundidad límite y son 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):

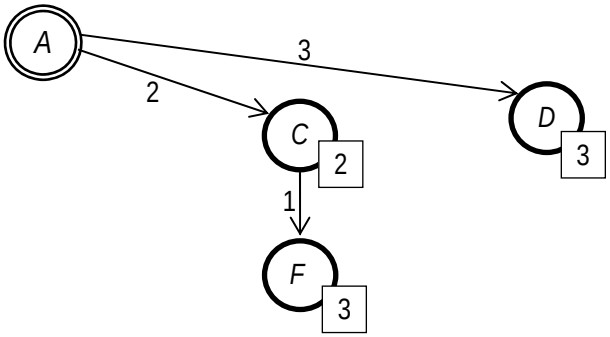
- Inicialmente, *A* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *A*. A continuación, expandimos el nodo inicial:



- Expandimos *B*:



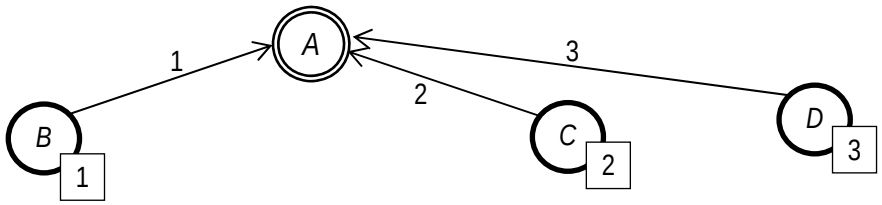
- Tras comprobar que *E* no es nodo meta, se le aplica la función LimpiarTABLA_A por estar en la profundidad límite y es sacado de TABLA_A. Igualmente, se aplica la función LimpiarTABLA_A a *B*, que es sacado de TABLA_A por no tener hijos en ABIERTA. A continuación, expandimos *C*:



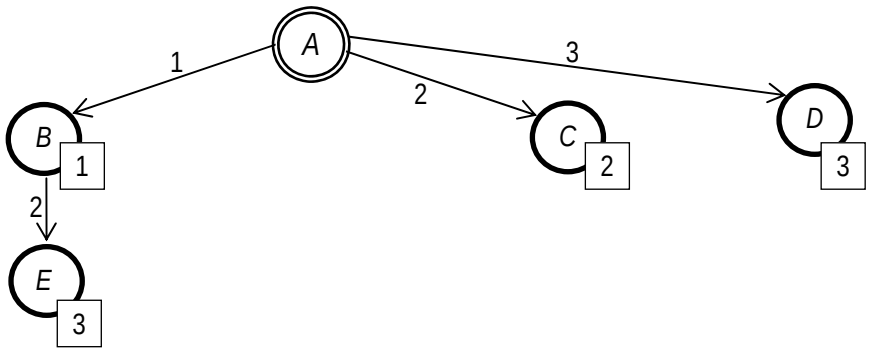
- Al no ser *F* nodo meta, la función LimpiarTABLA_A lo elimina de TABLA_A por estar en la produndidad límite. Seguidamente, LimpiarTABLA_A elimina a *C* de TABLA_A por no tener hijos en ABIERTA. A continuación, tras elegir *D* para su expansión, la función LimpiarTABLA_A lo saca de TABLA_A por no tener hijos en ABIERTA. La presente iteración finaliza cuando *A* es sacado de TABLA_A por LimpiarTABLA_A, dado que no tiene hijos en ABIERTA.

Iteración 3 (profundidad límite igual a 3):

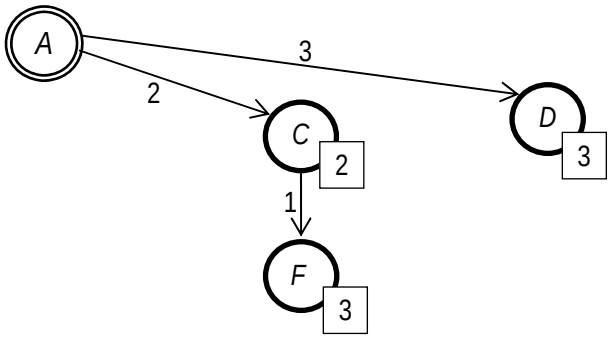
- Inicialmente, *A* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *A*. A continuación, expandimos el nodo inicial:



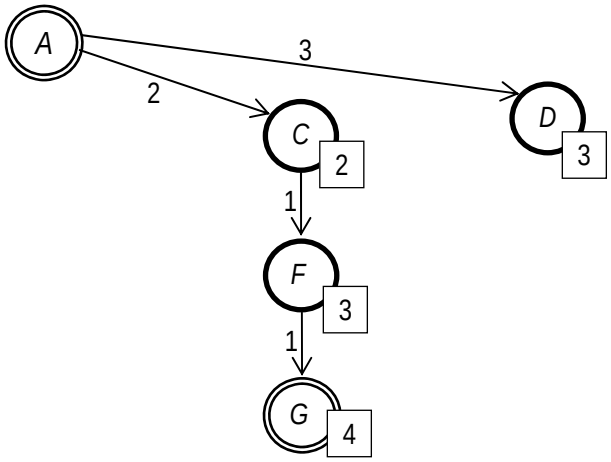
- Expandimos *B*:



- Expandimos *E*, al que se le aplica la función LimpiarTABLA_A por no tener hijos en ABIERTA y es sacado de TABLA_A. Por el mismo motivo, *B* es sacado de TABLA_A. A continuación, expandimos *C*:



- Expandimos *F*:



- Finalmente, tras comprobar que *G* es nodo meta, finaliza el algoritmo habiendo hallado un camino entre el nodo inicial y el nodo meta de coste 4.

EJERCICIO 5:

Considere el grafo de la figura 5.1, donde el nodo inicial es n_1 y donde el nodo meta es n_7. 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?

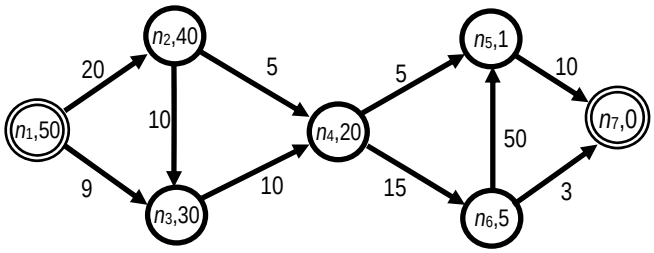


Figura 5.1: Grafo de búsqueda en el que el nodo inicial es n_1, el nodo meta es n_7, el coste de cada operador aparece al lado del arco que lo representa y al lado de cada nodo aparece la estimación de la menor distancia desde ese nodo a una meta.

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.

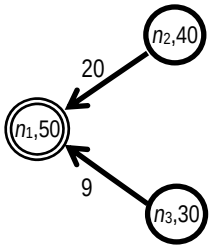
SOLUCIÓN DEL EJERCICIO 5 (por Severino Fernández Galán):

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:

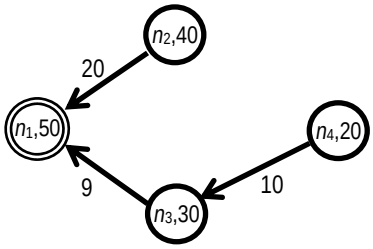
ABIERTA = {$\underline{n_1(0+50)}$}

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



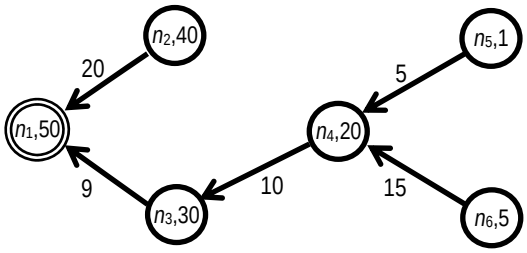
ABIERTA = {$\underline{n_3(9+30)}$, $n_2(20+40)$}

- 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 $g=9$ y $h=30$).



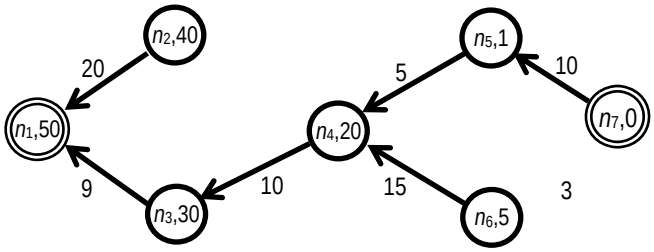
ABIERTA = {$\underline{n_4(19+20)}$, $n_2(20+40)$}

- PASO 3. Expandimos n_4.



ABIERTA = {$\underline{n_5(24+1)}$, $n_6(34+5)$, $n_2(20+40)$}

- PASO 4. Expandimos n_5.



ABIERTA = {$\underline{n_7(34+0)}$, $n_6(34+5)$, $n_2(20+40)$}

- PASO 6. Intentamos expandir n_7 y alcanzamos una meta, con lo que el algoritmo termina. El camino solución encontrado es: $n_1 \rightarrow n_3 \rightarrow n_4 \rightarrow n_5 \rightarrow n_7$, cuyo coste es 9+10+5+10=34. (Obsérvese que, a lo largo de todos los pasos del algoritmo, nunca ha sido necesario realizar ninguna reorientación de enlaces ascendentes al mejor padre desde ninguno de los hijos generados en cada expansión de un nodo padre.)

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 el valor de su 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 dos últimos pasos del algoritmo.

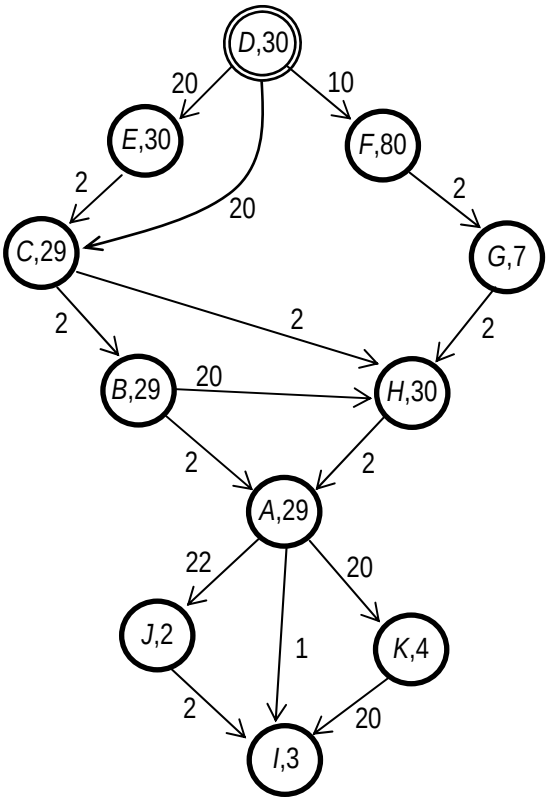


Figura 6.1: Grafo de búsqueda en el que el nodo inicial es *D*, los nodos meta son desconocidos, el coste de cada operador aparece al lado del arco que lo representa y al lado de cada nodo aparece el valor de su función de evaluación heurística (que hay que minimizar).

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*(29), *E*(30), *F*(80)}. Seleccionamos el nodo *C* por ser el mejor de los nodos hijos. A continuación aceptamos el nodo *C* como nuevo nodo actual en sustitución de *D*, debido a que el valor de la función de evaluación heurística de *C* es mejor o igual que el de *D* ($29 \leq 30$).

- PASO 2: Expandimos el nodo actual *C* y generamos sus hijos: {*B*(29), *H*(30)}. Seleccionamos el nodo *B* por ser el mejor de los nodos hijos. Seguidamente 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* ($29 \leq 29$). (Observe que la condición de terminación del algoritmo todavía no se cumple tras este paso, ya que sólo hemos completado un paso sin mejora, cuando la condición de terminación del enunciado nos indica que tienen que ser dos.)

- PASO 3: Expandimos el nodo actual *B* y generamos sus hijos: {*A*(29), *H*(30)}. 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 *B*, debido a que el valor de la función de evaluación heurística de *A* es mejor o igual que el de *B* (se cumple que $29 \leq 29$).

La búsqueda terminaría en este punto, ya que se han completado dos pasos sin mejora. El algoritmo devolvería el nodo *A*(29) como mejor nodo encontrado.