Asignatura de Grado: FUNDAMENTOS DE INTELIGENCIA ARTIFICIAL
Curso: 2017 - 2018
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 2018.

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 de las torres de Hanoi* 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 las torres de Hanoi se dispone de tres varillas verticales y un número *d* de discos de diferente radio. (Nosotros consideraremos que $d=3$.) En la varilla izquierda se apilan inicialmente todos los discos según radio decreciente en altura. El objetivo es pasar todos los discos de la varilla izquierda a una de las otras dos varillas aplicando las siguientes reglas: (1) un disco no puede descansar nunca sobre otro más pequeño, (2) sólo se puede mover un disco cada vez y (3) sólo se puede desplazar el disco superior de cada varilla. ¿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) 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 utilizamos la siguiente notación. Consideramos tres varillas verticales (izquierda, central y derecha) y tres discos $\{d_1, d_2, d_3\}$ ordenados según radio creciente. Es decir, d_1 es el disco menor, d_2 es el disco mediano y d_3 es el disco mayor. El estado del problema lo representamos como una lista con tres sublistas, donde cada sublista contiene los discos de una de las varillas en orden descendente en altura. Por ejemplo, si representamos cada disco simplemente por su subíndice, el estado (12, -, 3) representa el caso en que la varilla izquierda contiene el disco menor encima del mediano, la varilla central está vacía y la varilla derecha contiene el disco mayor. Nótese que, según las reglas impuestas en este problema, existen 27 estados posibles en el problema planteado.

El **estado inicial** descrito en el enunciado se representa como (123, -, -). Por otro lado, existen dos **estados meta**, (-, 123, -) y (-, -, 123).

Existen seis **operadores** posibles, $i \rightarrow j$, según las condiciones dadas en el enunciado. El operador $i \rightarrow j$ mueve el disco superior de la varilla i a la parte superior de la varilla j, donde i y j toman valores diferentes correspondientes a la varilla "izquierda", la "central" o la "derecha". Para que se pueda aplicar un operador $i \rightarrow j$ es necesario que exista al menos un disco en la varilla i y que su radio sea menor que el radio del disco superior de la varilla j (en caso de que la varilla j no esté vacía). Al trasladarse únicamente un disco en cada operación, 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 permitidos del sistema, los operadores que se pueden aplicar a cada estado permitido 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 se ha utilizado color rojo para marcar el estado inicial y se ha utilizado 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 tienen coste 7 y una de ellas está marcada en trazo discontinuo en la figura 1.1. Las dos soluciones menos costosas son las siguientes:

(123, -, -) → (23, -, 1) → (3, 2, 1) → (3, 12, -) → (-, 12, 3) → (1, 2, 3) → (1, -, 23) → (-, -, 123)
(123, -, -) → (23, 1, -) → (3, 1, 2) → (3, -, 12) → (-, 3, 12) → (1, 3, 2) → (1, 23, -) → (-, 123, -)

ESTADOS	OPERADORES					
	izq→cen	*izq→der*	*cen→izq*	*cen→der*	*der→izq*	*der→cen*
(123, -, -)	(23,1,-)	(23, -, 1)	No aplicable	No aplicable	No aplicable	No aplicable
(-, 123, -)	ESTADO META					
(-, -, 123)	ESTADO META					
(12,3,-)	(2, 13, -)	(2, 3, 1)	No aplicable	(12, -, 3)	No aplicable	No aplicable
(12, -, 3)	(2, 1, 3)	(2, -, 13)	No aplicable	No aplicable	No aplicable	(12, 3, -)
(13, 2, -)	(3, 12, -)	(3, 2, 1)	No aplicable	(13, -, 2)	No aplicable	No aplicable
(13, -, 2)	(3, 1, 2)	(3, -, 12)	No aplicable	No aplicable	No aplicable	(13, 2, -)
(3, 12, -)	No aplicable	(-, 12, 3)	(13, 2, -)	(3, 2, 1)	No aplicable	No aplicable
(-, 12, 3)	No aplicable	No aplicable	(1, 2, 3)	(-, 2, 13)	(3, 12, -)	No aplicable
(2, 13, -)	No aplicable	(-, 13, 2)	(12, 3, -)	(2, 3, 1)	No aplicable	No aplicable
(-, 13, 2)	No aplicable	No aplicable	(1, 3, 2)	(-, 3, 12)	(2, 13, -)	No aplicable
(-, 3, 12)	No aplicable	No aplicable	(3, -, 12)	No aplicable	(1, 3, 2)	(-, 13, 2)
(3, -, 12)	(-, 3, 12)	No aplicable	No aplicable	No aplicable	(13, -, 2)	(3, 1, 2)
(-, 2, 13)	No aplicable	No aplicable	(2, -, 13)	No aplicable	(1, 2, 3)	(-, 12, 3)
(2, -, 13)	(-, 2, 13)	No aplicable	No aplicable	No aplicable	(12, -, 3)	(2, 1, 3)
(1, 2, 3)	(-, 12, 3)	(-, 2, 13)	No aplicable	(1, -, 23)	No aplicable	No aplicable
(1, 3, 2)	(-, 13, 2)	(-, 3, 12)	No aplicable	No aplicable	No aplicable	(1, 23, -)
(1, 23, -)	(-, 123, -)	(-, 23, 1)	No aplicable	(1, 3, 2)	No aplicable	No aplicable
(1, -, 23)	(-, 1, 23)	(-, -, 123)	No aplicable	No aplicable	No aplicable	(1, 2, 3)
(2, 1, 3)	No aplicable	(-, 1, 23)	(12, - 3)	(2, -, 13)	No aplicable	No aplicable
(3, 1, 2)	No aplicable	No aplicable	(13, -, 2)	(3, -, 12)	(23, 1, -)	No aplicable
(23, 1, -)	No aplicable	(3, 1, 2)	(123, -, -)	(23, -, 1)	No aplicable	No aplicable
(-, 1, 23)	No aplicable	No aplicable	(1, -, 23)	(-, -, 123)	(2, 1, 3)	No aplicable
(3, 2, 1)	No aplicable	No aplicable	(23, -, 1)	No aplicable	(13, 2, -)	(3, 12, -)
(2, 3, 1)	(-, 23, 1)	No aplicable	No aplicable	No aplicable	(12, 3, -)	(2, 13, -)
(23, -, 1)	(3, 2, 1)	No aplicable	No aplicable	No aplicable	(123, -, -)	(23, 1, -)
(-, 23, 1)	No aplicable	No aplicable	(2, 3, 1)	No aplicable	(1, 23, -)	(-, 123, -)

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

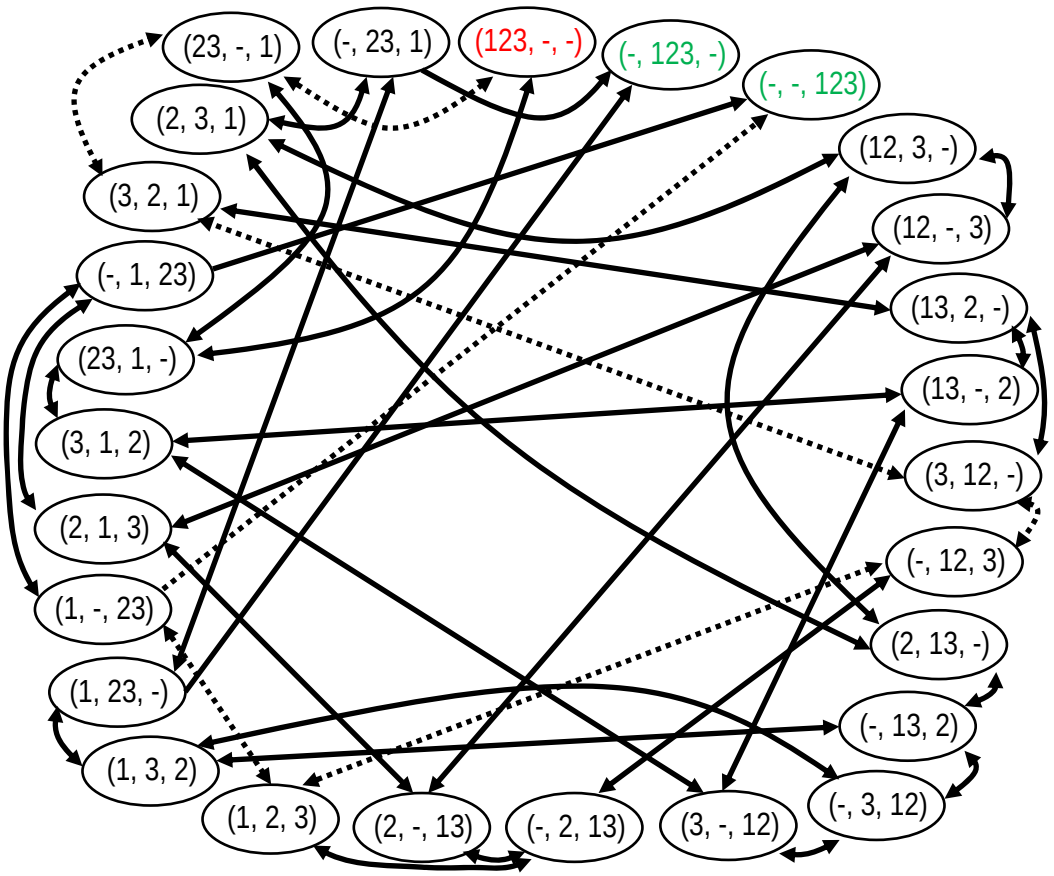


Figura 1.1: Grafo dirigido que representa el espacio de búsqueda para el problema del viajante. No existen estados no alcanzables desde el estado inicial, (123, -, -), que queden aislados. El camino de menor coste hasta el estado meta, (-, -, 123), se representan en trazo discontinuo.

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)

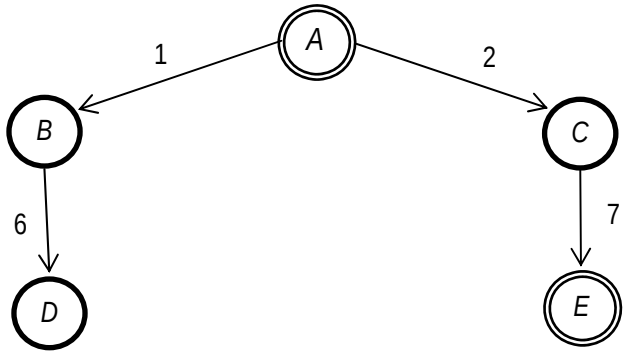


Figura 2.1: Árbol de búsqueda en el que el nodo inicial es *A*, el nodo meta es *E* y el coste de cada operador aparece al lado del arco que lo representa.

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.

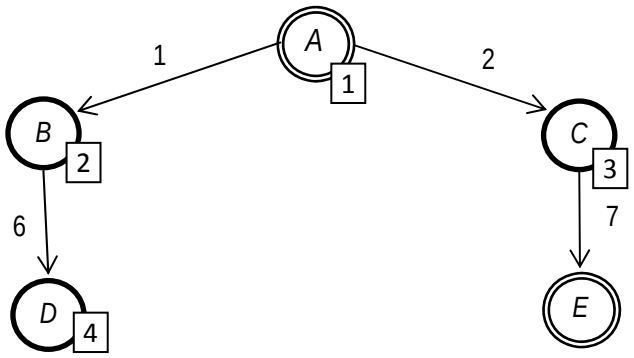


Figura 2.2: Orden de expansión de nodos para el árbol de la figura 2.1 según la búsqueda primero en anchura (de izquierda a derecha). Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto,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

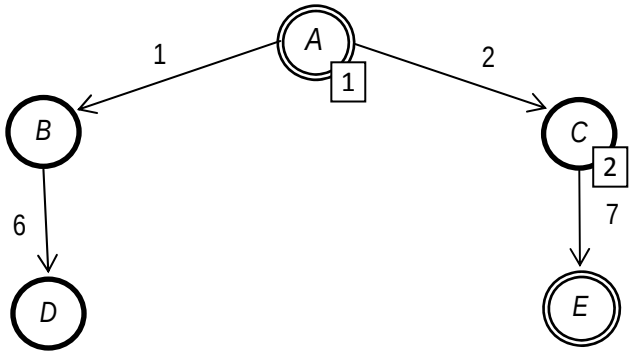


Figura 2.3: Orden de expansión de nodos para el árbol de la figura 2.1 según la búsqueda primero en profundidad (de derecha a izquierda). Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por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

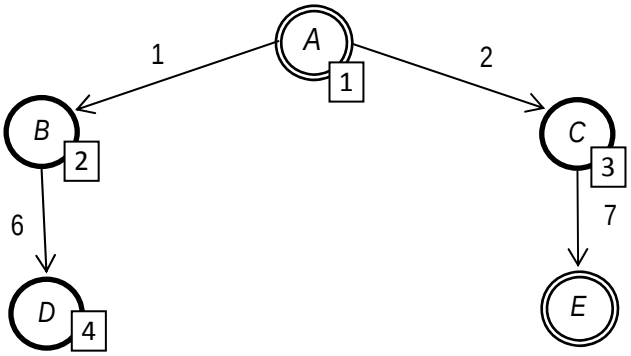


Figura 2.4: Orden de expansión de nodos para el árbol de la figura 2.1 según la búsqueda de coste uniforme. Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión se comprueba que es un nodo meta y, por tanto, el algoritmo termina 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 muestran los órdenes de expansión de nodos para la única iteración de búsqueda primero en anchura que es necesaria para este ejemplo de búsqueda en anchura iterativa.

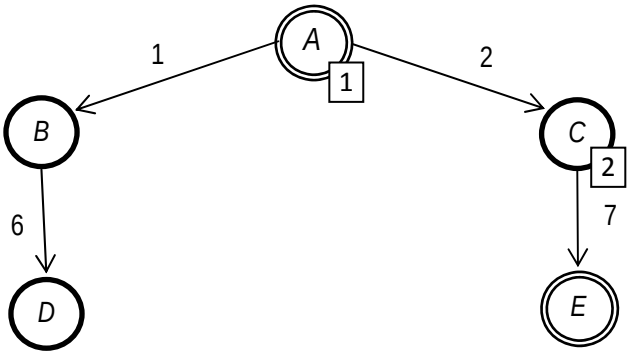


Figura 2.5a: Primera (y última) iteración de la búsqueda en anchura iterativa de derecha a izquierda, en la que como máximo un hijo es generado en cada expansión de un nodo padre. Observe que el nodo meta no es realmente expandido (es decir, sus hijos no son generados), ya que justo antes de su expansión 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.

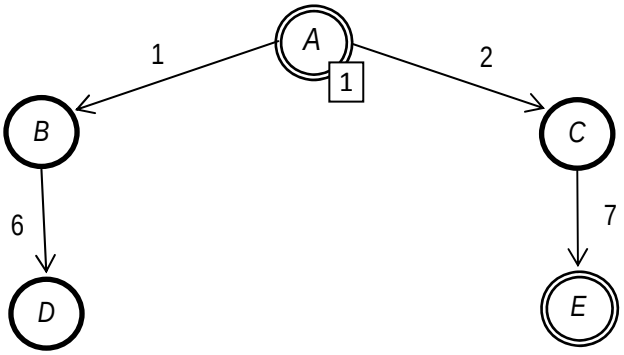


Figura 2.6a: Primera iteración de la búsqueda en profundidad iterativa de izquierda a derecha, en la que la profundidad límite es igual a 1 y únicamente son expandidos nodos con una profundidad máxima de 0.

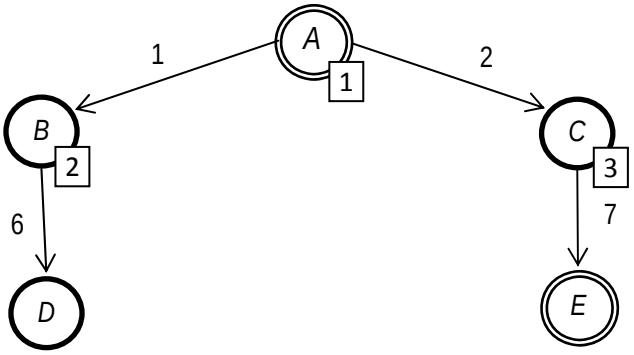


Figura 2.6b: Segunda iteración de la búsqueda en profundidad iterativa de izquierda a derecha, en la que la profundidad límite es igual a 2 y únicamente son expandidos nodos con una profundidad máxima de 1.

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)

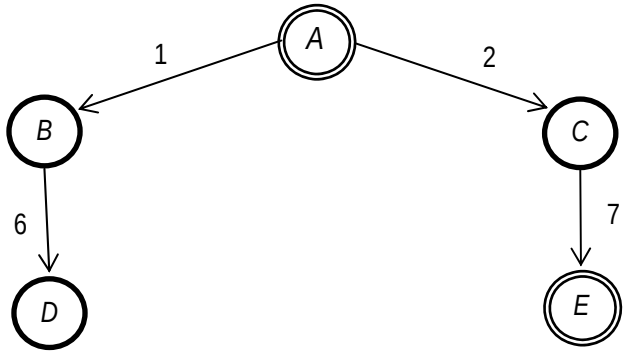


Figura 3.1: Árbol de búsqueda en el que el nodo inicial es *A*, el nodo meta es *E* y el coste de cada operador aparece al lado del arco que lo representa.

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*}
Antes de sacar *C*: {*C, D*}
Antes de sacar *D*: {*D, E*}
Antes de sacar *E*: {*E*}
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 *C*: {*C, B*}
Antes de sacar *E*: {*E, 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)}
Antes de sacar *C*: {*C*(2), *D*(7)}
Antes de sacar *D*: {*D*(7), *E*(9)}
Antes de sacar *E*: {*E*(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 *C*: {*C*}
Antes de sacar *E*: {*E*}
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*}
Nótese que *B* 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 2 (profundidad límite igual a 2):

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

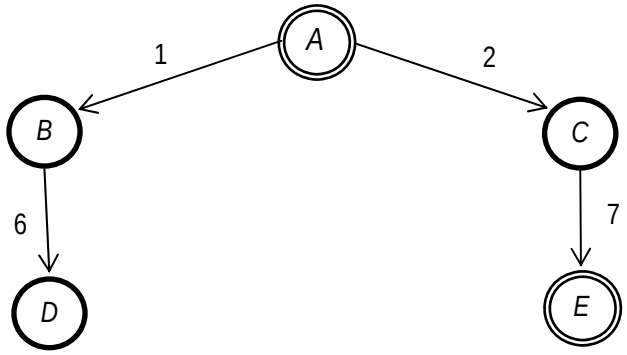


Figura 4.1: Árbol de búsqueda en el que el nodo inicial es *A*, el nodo meta es *E* y el coste de cada operador aparece al lado del arco que lo representa.

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:

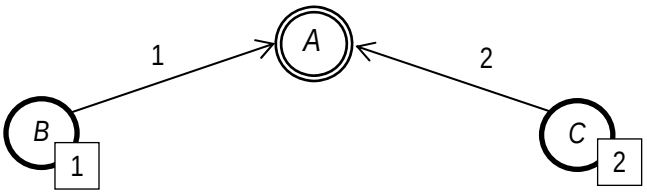
- Cada uno de los cinco apartados, correspondiente a un método de búsqueda concreto, se puntúa sobre 2 puntos.
- Si en cualquiera de los cinco apartados el algoritmo correspondiente no gestiona TABLA_A en la forma debida: la puntuación del apartado bajaría 1.6 puntos si la respuesta dada varía significativamente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

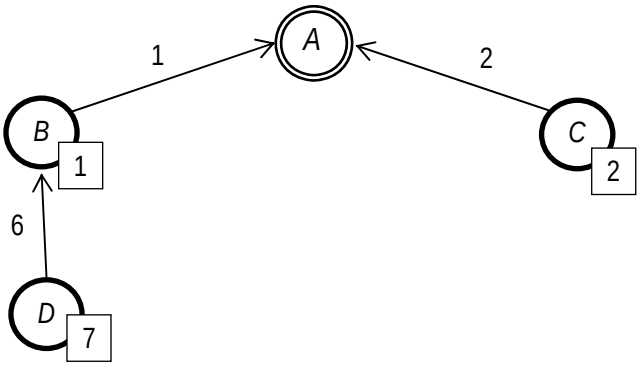
De cara a ilustrar la respuesta convenientemente, incluimos la información de TABLA_A gráficamente: por un lado, para cada nodo generado en una expansión trazamos un arco ascendente a su padre y, por otro lado, indicamos el coste al nodo inicial al lado de cada nodo generado.

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

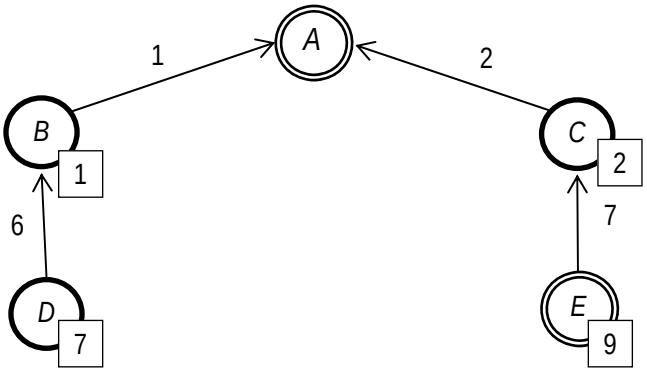
- Inicialmente, *A* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *A*. A continuación expandimos el nodo inicial, generando sus nodos hijos. Desde cada nodo hijo trazamos un nodo ascendente a su nodo padre. Al lado de cada nodo hijo indicamos el coste desde dicho nodo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



- A continuación expandimos *B*:



- A continuación expandimos *C*:

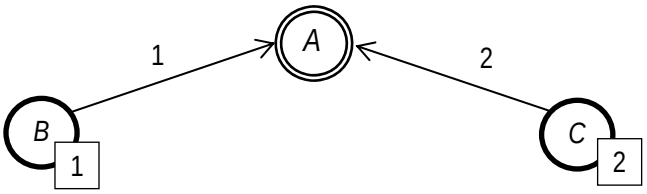


- Expandimos *D* y TABLA_A no varía. A continuación elegiríamos *E* para su expansión y llegaríamos a la meta. El camino hallado hasta la meta tiene coste 9.

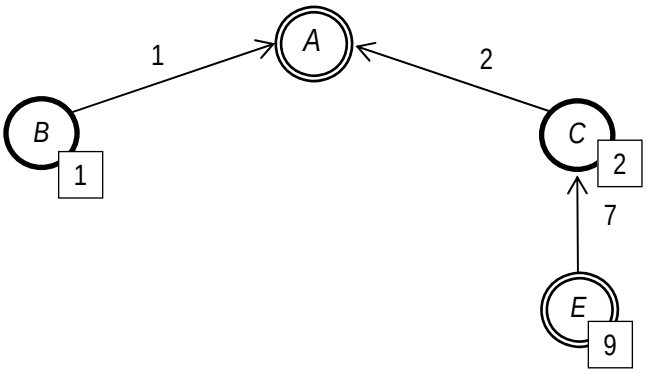
2. Búsqueda Primero en Profundidad (de derecha a izquierda)

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.



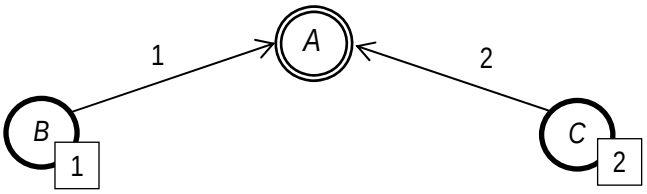
- A continuación expandimos *C*:



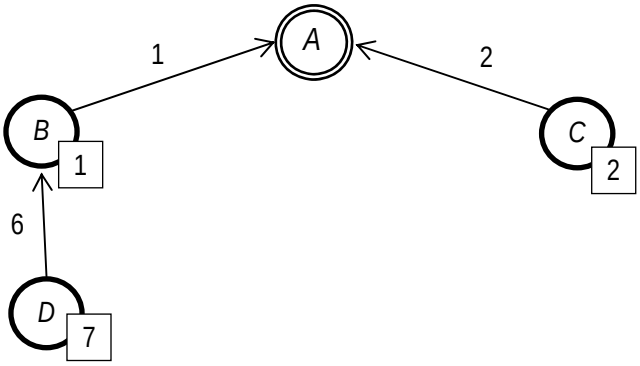
- A continuación elegiríamos *E* para su expansión y llegaríamos a la meta. El camino hallado hasta la meta tiene coste 9. (Observe que finalmente no ha sido necesario aplicar la función LimpiarTABLA_A en ningún paso de este apartado.)

3. Búsqueda de Coste Uniforme

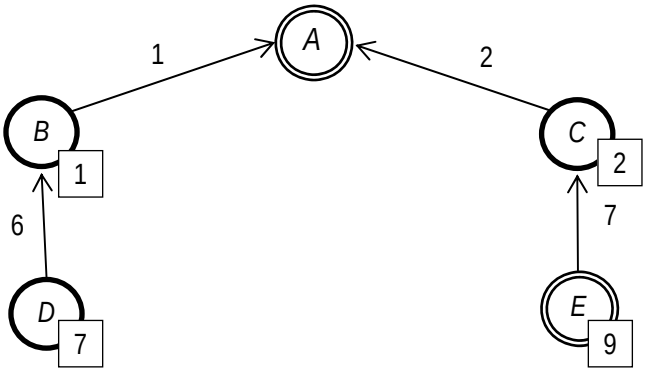
- Inicialmente, *A* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *A*. A continuación, expandimos el nodo inicial.



- Seguidamente expandimos *B*:



- A continuación expandimos *C*:

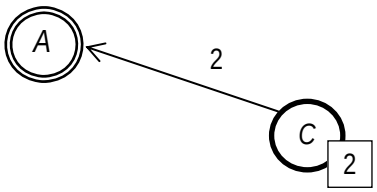


- Seguidamente expandimos *D* y TABLA_A no varía. Por último, al intentar expandir *E*, alcanzamos la meta. El coste del camino solución hallado es 9.

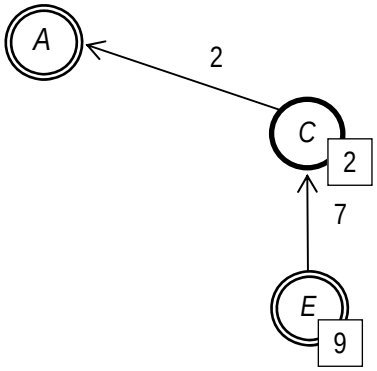
4. Búsqueda en Anchura Iterativa (de derecha a izquierda)

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:



- A continuación expandimos *C*:

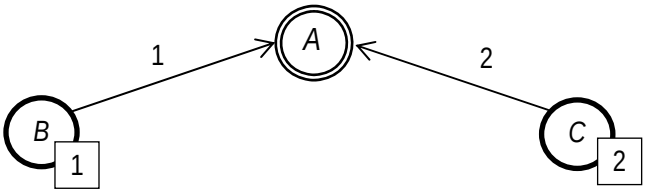


- Finalmente, al intentar expandir *E*, alcanzamos la meta. El camino encontrado hasta el nodo inicial tiene coste 9.

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

Iteración 1 (profundidad límite igual a 1):

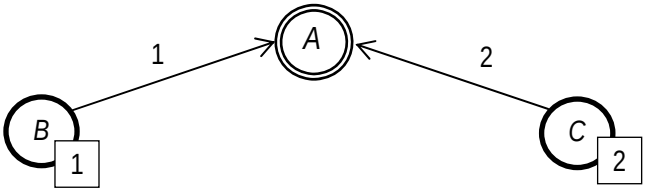
- Inicialmente, *A* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *A*. A continuación, expandimos el nodo inicial:



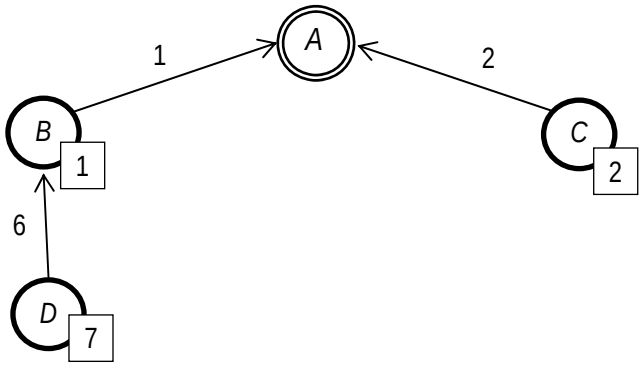
- Tras comprobar que *B* no es nodo meta, se le aplica la función LimpiarTABLA_A por estar en la profundidad límite y es sacado de TABLA_A. De igual manera, tras comprobar que *C* no es nodo meta, se le aplica la función LimpiarTABLA_A por estar en la profundidad límite y es sacado de TABLA_A. Por último,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

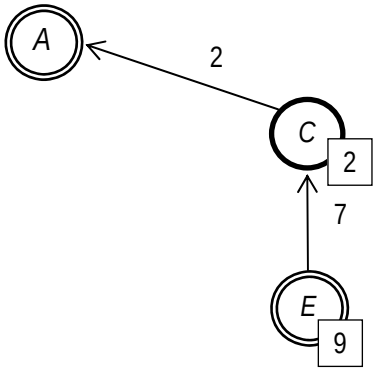
- Inicialmente, *A* sería incluido en TABLA_A. Gráficamente esto se correspondería con un único nodo *A*. A continuación, expandimos el nodo inicial:



- A continuación, expandimos *B*:



- Tras comprobar que *D* 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 expandimos *C*:



- Por último, tras elegir *E* 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?

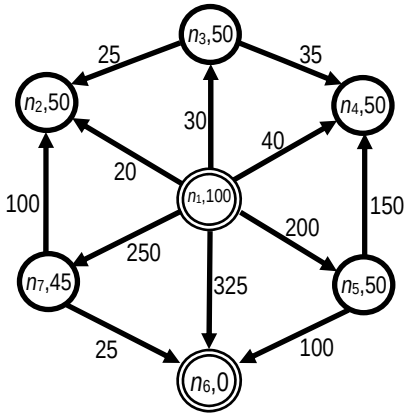


Figura 5.1: Grafo de búsqueda en el que el nodo inicial es n_1, el nodo meta es n_6, el coste de cada operador aparece al lado del arco que lo representa y al lado de cada nodo aparece la estimación de la menor distancia desde ese nodo a una meta.

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.

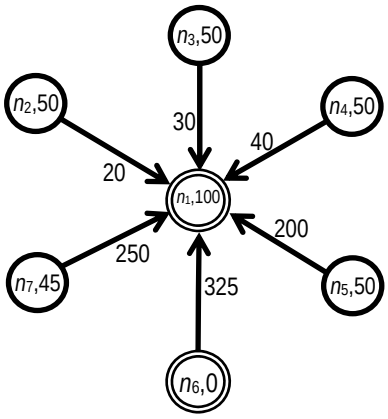
SOLUCIÓN DEL EJERCICIO 5 (por Severino Fernández Galán):

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:

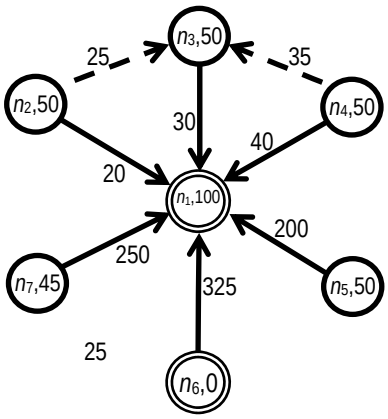
ABIERTA = {$\underline{n_1(0+100)}$}

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



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

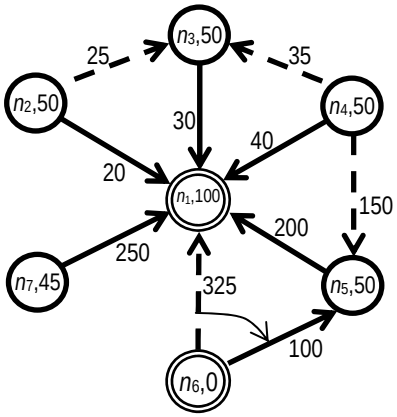
- PASOS 2 y 3. Expandimos n_2 por ser el nodo de ABIERTA con menor valor de la función de evaluación heurística, $f=g+h$ (al ser $g=20$ y $h=50$), tras lo cual nada cambia en TABLA_A. A continuación expandimos n_3:



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

Observe que el mejor camino desde n_2 al nodo inicial lo marca su padre n_1 (coste 20) y no su padre n_3 (coste 25+30=55). Por ello, el arco ascendente de n_2 a n_1 se marca con trazo continuo y el arco ascendente de n_2 a n_3 se marca con trazo discontinuo. Del mismo modo, el mejor camino desde n_4 al nodo inicial lo marca su padre n_1 (coste 40) y no su padre n_3 (coste 35+30=65). Por ello, el arco ascendente de n_4 a n_1 se marca con trazo continuo y el arco ascendente de n_4 a n_3 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.

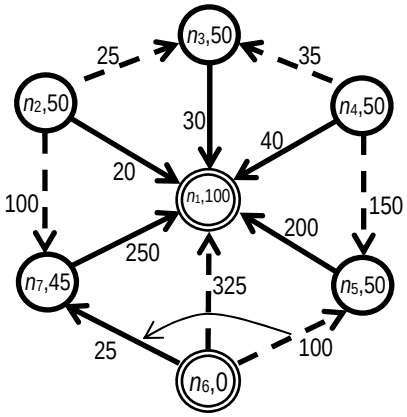
- PASOS 4 y 5. Expandimos n_4, tras lo cual nada cambia en TABLA_A. A continuación expandimos n_5:



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

Observe 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 = {$\underline{n_6(275+0)}$}

Observe que ha habido una reorientación del mejor padre de n_6, que antes era n_5 y ahora pasa a ser n_7. El nuevo mejor coste de n_6 al nodo inicial, $g(n_6)$, es ahora 25+250=275, que se puede calcular a partir de los costes de los mejores arcos ascendentes hallados hasta el momento: $n_6 \rightarrow n_7$ y $n_7 \rightarrow n_1$.

- 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.

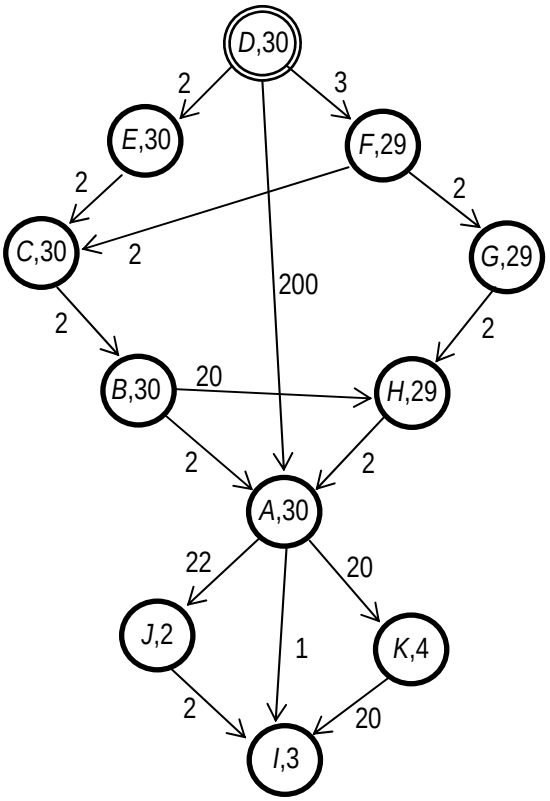


Figura 6.1: Grafo de búsqueda en el que el nodo inicial es *D*, los nodos meta son desconocidos, el coste de cada operador aparece al lado del arco que lo representa y al lado de cada nodo aparece el valor de su función de evaluación heurística (que hay que minimizar).

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*(30), *A*(30), *F*(29)}. 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* ($29 \leq 30$).

- PASO 2: Expandimos el nodo actual *F* y generamos sus hijos: {*C*(30), *G*(29)}. 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* ($29 \leq 29$). (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 tres.)

- PASO 3: Expandimos el nodo actual *G* y generamos sus hijos: {*H*(29)}. 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 *G*, debido a que el valor de la función de evaluación heurística de *H* es mejor o igual que el de *G* (se cumple que $29 \leq 29$). (Al igual que en el Paso 2, 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 4: Expandimos el nodo actual *H* y generamos sus hijos: {*A*(30)}. 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 $30 \leq 29$).

La búsqueda termina en este punto. El algoritmo devuelve el nodo *H*(29) como mejor nodo encontrado.