BÚSQUEDA

Vicente Martínez Orga
vicente.martinez@upm.es

Departamento de Inteligencia Artificial
Facultad de Informática
Universidad Politécnica de Madrid
Campus de Montegancedo sn,
28660 Boadilla del Monte, Madrid, Spain

LA BÚSQUEDA APARECE EN TODOS LOS TIPOS DE PROBLEMAS EN DONDE LA ADQUISICIÓN O RECUPERACIÓN DE INFORMACIÓN Y/O CONOCIMIENTOS CONSTITUYE UNA PARTE DE LOS MISMOS

BÚSQUEDA EN ESPACIO DE ESTADOS

CONSISTE EN DADO UN ESTADO INICIAL ALCANZAR UN ESTADO META PARA LO CUAL, ES NECESARIO APLICAR UNAS ACCIONES QUE NOS HAGAN TRANSITAR DE UN ESTADO A OTRO

GRAFO

CONJUNTO DE NODOS QUE REPRESENTAN ESTADOS O SITUACIONES, CONECTADOS A TRAVÉS DE ARCOS

AL NÚMERO DE SUCESORES DE UN NODO SE LE DENOMINA FACTOR DE RAMIFICACIÓN

ARBOL

ES UN GRAFO EN EL QUE CADA NODO, EXCEPTO EL NODO INICIAL, LLAMADO RAIZ, TIENE UN SOLO ANTECEDENTE

CUANDO UN NODO NO TIENE SUCESORES, SE LE LLAMA HOJA

A LOS NODOS SE LES SUELE ASOCIAR UN VALOR DENOMINADO “PESO”, QUE REPRESENTA UN COSTE ASOCIADO

A UNA SECUENCIA DE NODOS DONDE n ES SECUENCIA DE n – 1, SE LE DENOMINA CAMINO DE TAMAÑO m

COSTE DE UN CAMINO ES LA SUMA DE LOS COSTES DE TODOS LOS ARCOS DE ESTE CAMINO

A CADA ARCO QUE CONECTA DOS NODOS SE LE ASOCIA UN DIRECCIONADOR QUE NOS INDICA DE QUIEN ES DESCENDIENTE

LISTA

LISTA (O PILA) ES UN ELEMENTO QUE NOS PERMITE ALMACENAR LOS NODOS QUE SE VAN DESARROLLANDO EN UN ARBOL O GRAFO DE EXPLORACIÓN AL OBJETO DE PODER ESTABLECER EL RESULTADO FINAL DESPUES DEL PROCESO DE BÚSQUEDA

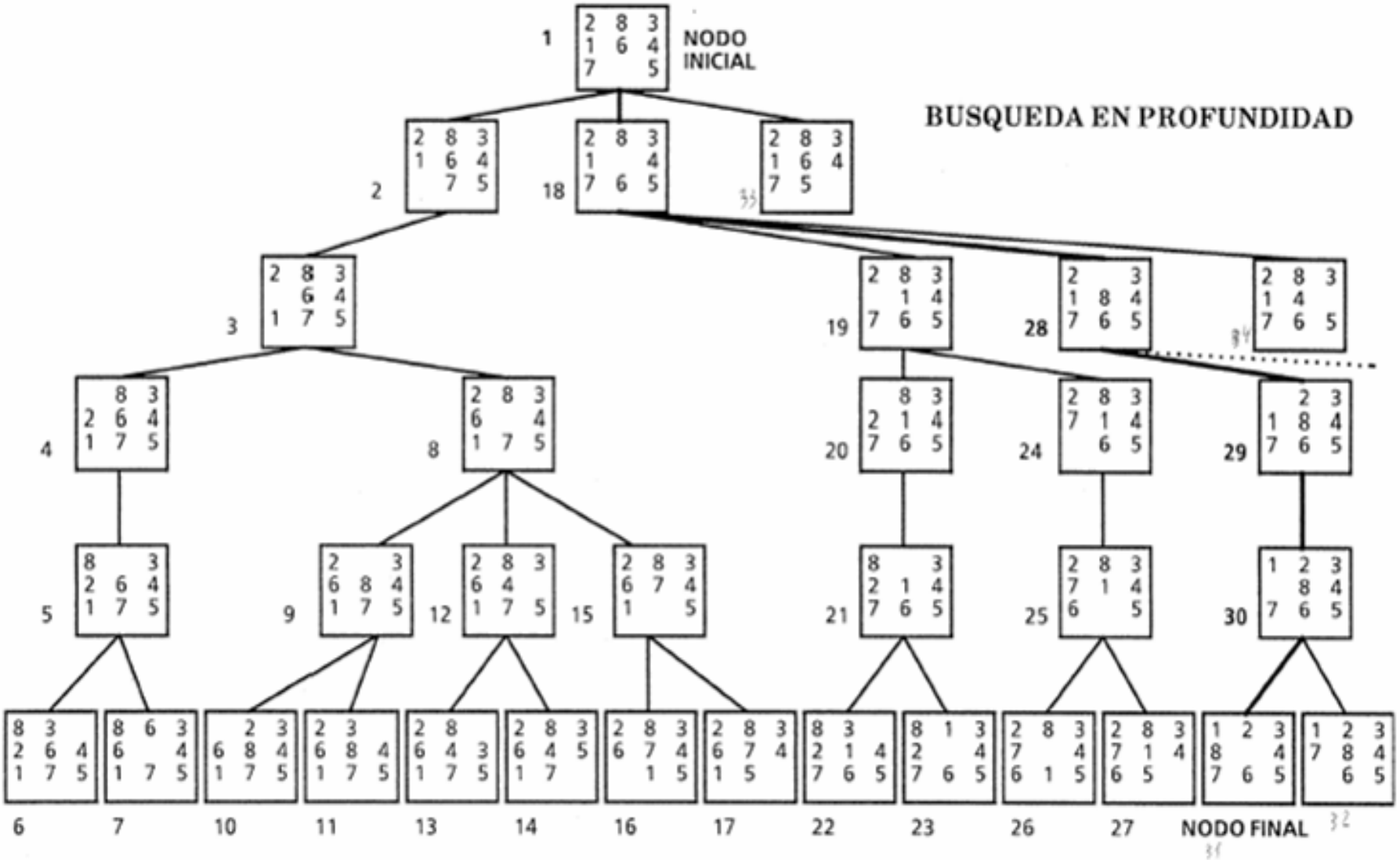
ALGUNAS VECES SE USAN DOS LISTAS (SE DENOMINAN ABIERTA Y CERRADA), EN OTRAS OCASIONES SOLO SE UTILIZA UNA (SE DENOMINA ABIERTA)

CLASES DE NODOS

a) NODOS QUE AUN NO HAN SIDO GENERADOS
b) NODOS GENERADOS Y NO EXPLORADOS (ABIERTOS)
c) NODOS QUE HAN SIDO EXPLORADOS PERO NO DESARROLLADOS
d) NODOS QUE HAN SIDO DESARROLLADOS (CERRADOS)

BÚSQUEDA EN PROFUNDIDAD

ESTA BÚSQUEDA DA PRIORIDAD A LOS NODOS DE NIVELES MAS PROFUNDOS EN EL GRAFO DE BÚSQUEDA


BUSQUEDA EN PROFUNDIDAD
1
NODO INICIAL
2
18
3
19
28
4
8
20
24
29
5
9
12
15
21
25
30
6
7
10
11
13
14
16
17
22
23
26
27
NODO FINAL

BÚSQUEDA CON RETROCESO

CONSISTE EN QUE CUANDO UN NODO SE SELECCIONA PARA LA EXPLORACIÓN, SOLO SE GENERA UNO DE SUS SUCESORES Y ESTE NODO GENERADO, A MENOS QUE SEA TERMINAL O FINAL SIN ÉXITO SE SOMETE DE NUEVO PARA LA EXPLORACIÓN

BÚSQUEDA CON RETROCESO

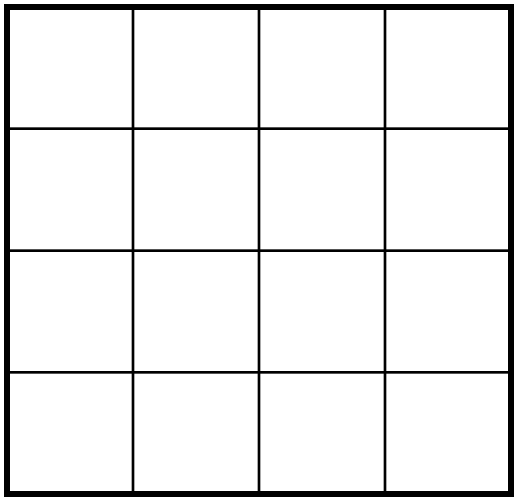
SI EL NODO GENERADO CUMPLE ALGÚN CRITERIO DE PARADA, EL PROGRAMA RETROCEDE AL ANTECESOR MAS CERCANO INEXPLORADO

BÚSQUEDA CON RETROCESO

4 REINAS EN UN TABLERO DE 4 x 4 DE FORMA QUE NO PUEDAN CAPTURARSE ENTRE SI, USANDO COMO ESTRATEGIA D E L C O N J U N T O C O N F L I C T O DESARROLLAR UN SP PARA COLOCAR LA BÚSQUEDA CON RETROCESO

BÚSQUEDA CON RETROCESO

BASE DE DATOS:

[i = 1, ]

BÚSQUEDA CON RETROCESO

BASE DE REGLAS:

R_{1j}: i = 1 → (PONER REINA EN FILA 1, COL, j) (i = 2)

R_{2j}: i = 2 → (PONER REINA EN FILA 2, COL, j) (i = 3)

R_{3j}: i = 3 → (PONER REINA EN FILA 3, COL, j) (i = 4)

R_{4j}: i = 4 → (PONER REINA EN FILA 4, COL, j) (i = 5)

R_5: i = 5 → fin

BÚSQUEDA CON RETROCESO

CRITERIO DE SELECCIÓN DE LAS REGLAS: SELECCIONAR LA REGLA CON MENOR DIAGONAL (ij)

ASI PARA LA FILA 1:

DIAGONAL (1,1) = 4 DIAGONAL (1,3) = 3

DIAGONAL (1,2) = 3 DIAGONAL (1,4) = 4

BÚSQUEDA CON RETROCESO

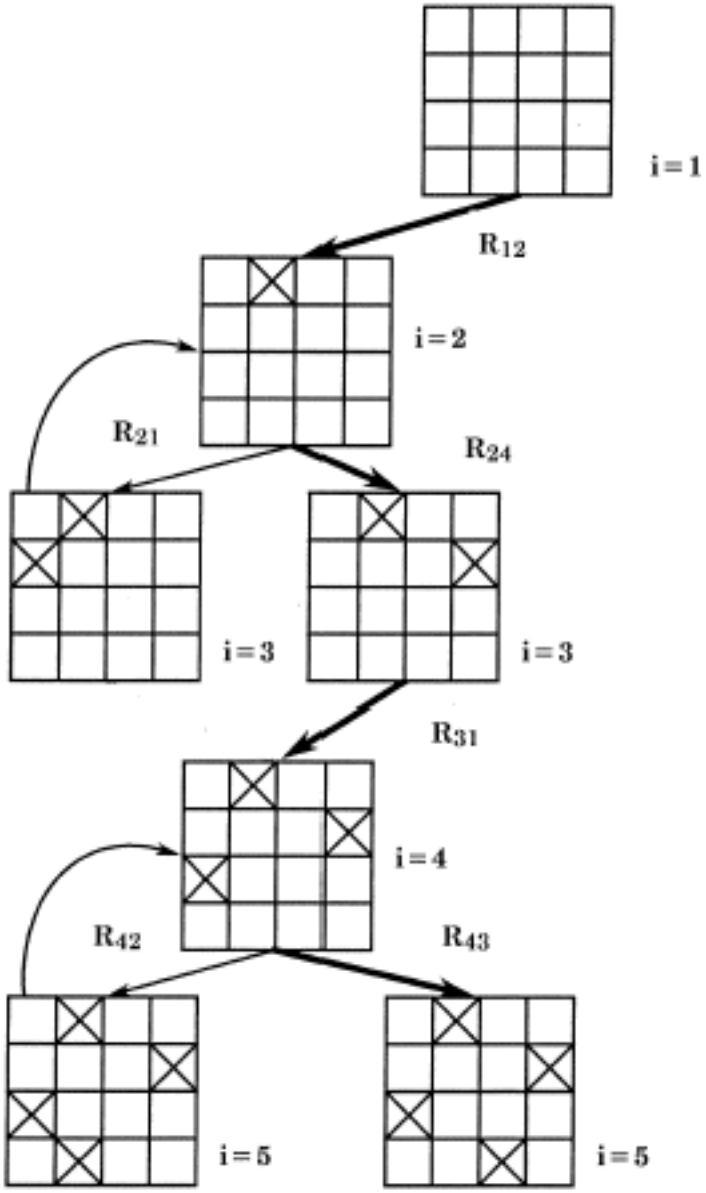
DE ACUERDO CON ESTE CRITERIO, LA ORDENACIÓN DE LAS REGLAS QUEDA:

(R12,R13,R11,R14)

(R21,R24,R22,R23)

(R31,R34,R32,R33)

(R42,R43,R41,R44)


i = 1
R12
i = 2
R21
R24
i = 3
i = 3
R31
i = 4
R42
R43
i = 5
i = 5

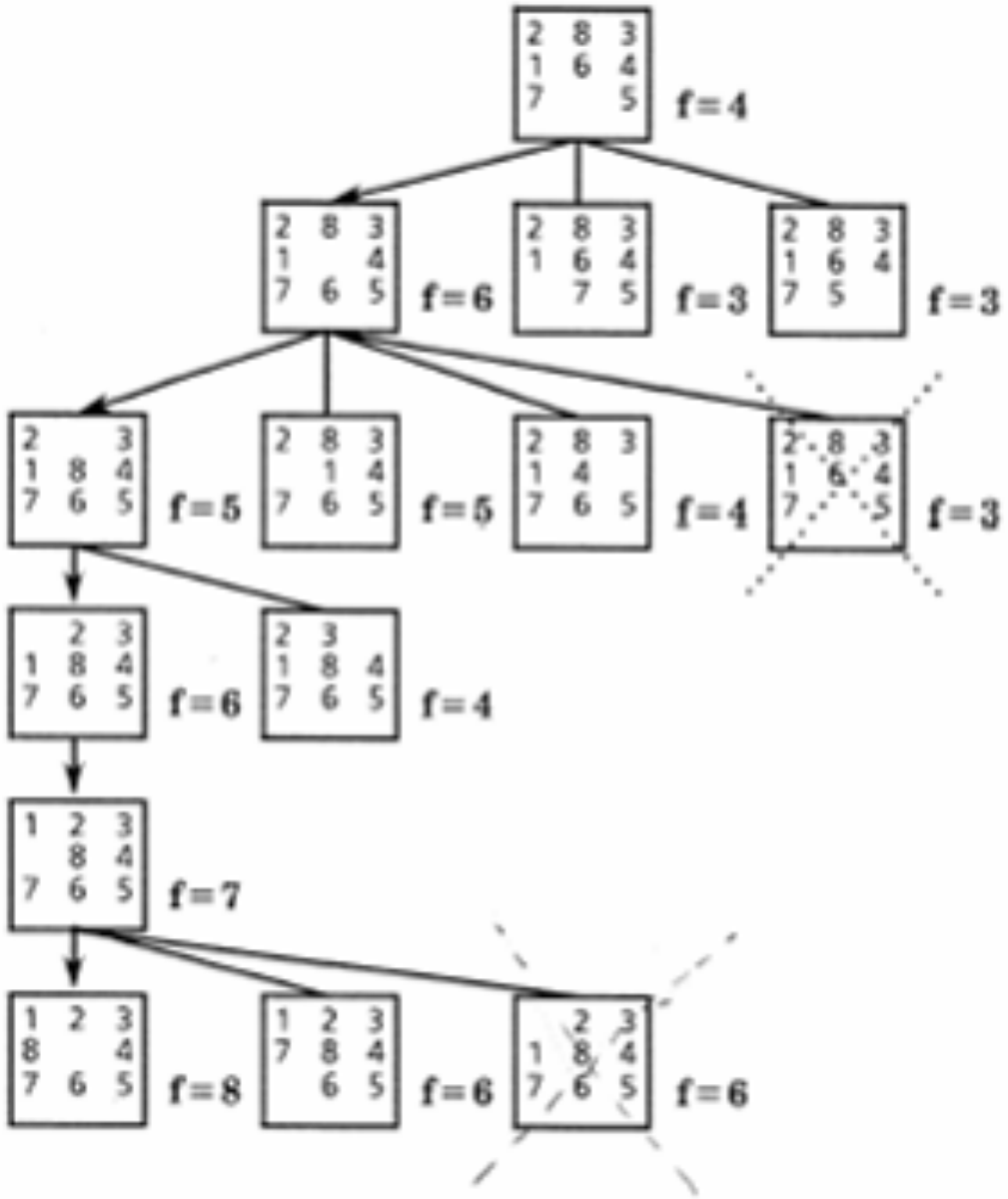
BÚSQUEDA EN ESCALADA

ESTA BÚSQUEDA PERMITE REDUCIR EL NÚMERO DE SECUENCIAS DE ACCIONES QUE DEBEN REALIZARSE ANTES DE ALCANZAR UNA SOLUCIÓN

DIFICULTADES DE LA BÚSQUEDA EN ESCALADA

MAXIMOS LOCALES: ES UN ESTADO MEJOR QUE LOS VECINOS PERO PEOR QUE OTROS MÁS ALEJADOS

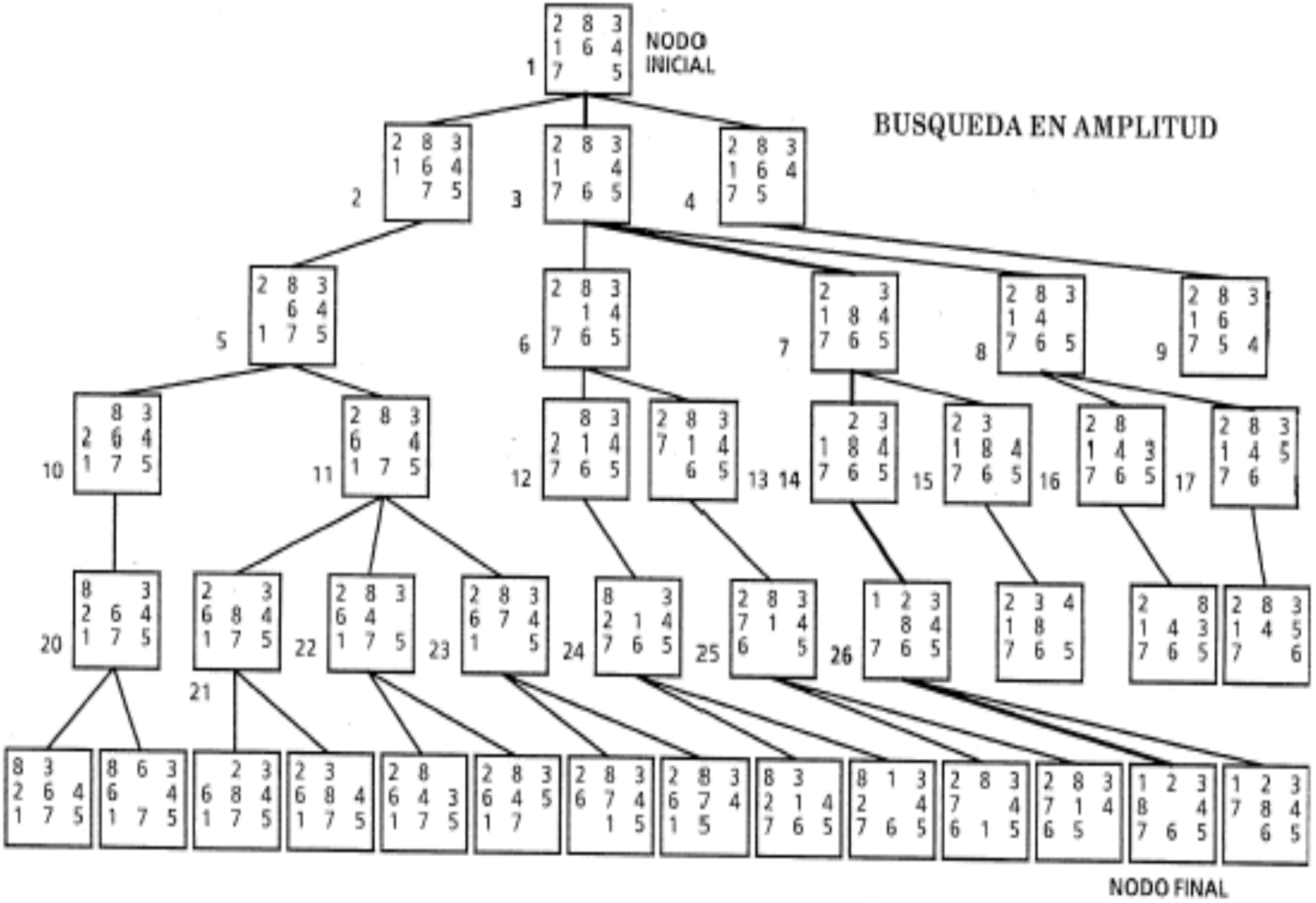
ALTIPLANICIES: ES UN ÁREA “LLANA” DEL ESPACIO DE BÚSQUEDA POR LO QUE EXISTE UN CONJUNTO COMPLETO DE ESTADOS VECINOS QUE TIENEN EL MISMO VALOR



BUSQUEDA MEDIANTE ESCALADA

BÚSQUEDA EN AMPLITUD

ESTA BÚSQUEDA DA PRIORIDAD A LOS NODOS DE NIVELES MÁS PRÓXIMOS AL NODO INICIAL EN EL GRAFO DE BÚSQUEDA


BUSQUEDA EN AMPLITUD
NODO INICIAL
NODO FINAL

BÚSQUEDA DIRIGIDA

ES SIMILAR A LA BÚSQUEDA EN AMPLITUD PORQUE PROGRESA NIVEL POR NIVEL, PERO SE DIFERENCIAN EN QUE EN ESTE CASO SOLO SE DESARROLLAN LOS NODOS MEJORES (QUE SE ESTABLEZCAN) PARA CADA NIVEL

BÚSQUEDA EN AMPLITUD Y PROFUNDIDAD

ESTA BÚSQUEDA TRATA DE DESARROLLAR LOS NODOS MÁS PROFUNDOS PERO CONDICIONADA AL “VALOR” QUE TIENEN LOS NODOS EN LA LISTA ABIERTA LO QUE LA HACE TRABAJAR SIMULTANEAMENTE EN AMPLITUD

PROCEDIMIENTO GENERAL DE BÚSQUEDA EN GRAFOS

1. CREAR UN GRAFO DE BÚSQUEDA CONSISTENTE ÚNICAMENTE DEL NODO INICIAL

2. CREAR LA LISTA CERRADA QUE INICIALMENTE ESTARA VACIA

PROCEDIMIENTO GENERAL DE BÚSQUEDA EN GRAFOS

3. SI LA LISTA ABIERTA ESTA VACIA, SALIR CON FRACASO

4. SELECCIONAR EL PRIMER NODO DE LA LISTA ABIERTA Y PASARLO A LA LISTA CERRADA, LLAMANDOLO n

PROCEDIMIENTO GENERAL DE BÚSQUEDA EN GRAFOS

5. SI n ES UN NODO META TERMINAR OBTENIENDO LA SOLUCIÓN A TRAVÉS DE LOS PUNTEROS DESDE n HASTA EL NODO INICIAL

6. EXPANDIR n GENERANDO EL CONJUNTO m DE SUS SUCESORES QUE NO SON ANTECESORES DE n

PROCEDIMIENTO GENERAL DE BÚSQUEDA EN GRAFOS

7. PONER UN PUNTERO HACIA n DESDE AQUELLOS ELEMENTOS DE m QUE NO ESTUVIESEN ANTERIORMENTE EN EL GRAFO AÑADIRLOS A ABIERTA

PROCEDIMIENTO GENERAL DE BÚSQUEDA EN GRAFOS

PARA CADA ELEMENTO DE m QUE ESTUVIESE ANTERIORMENTE EN ABIERTA O EN CERRADA, DECIDIR SI SE REDIRIGE O NO SU PUNTERO HACIA n

PROCEDIMIENTO GENERAL DE BÚSQUEDA EN GRAFOS

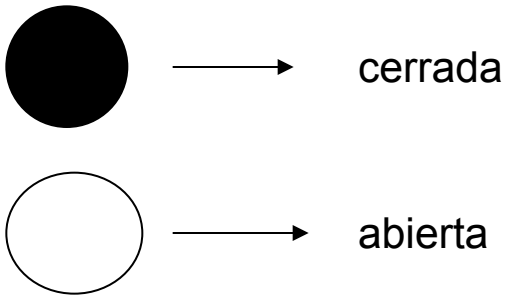
PARA CADA ELEMENTO DE m QUE ESTUVIESE ANTERIORMENTE EN CERRADA, DECIDIR PARA CADA UNO DE SUS DESCENDIENTES SI SE REDIRIGEN O NO SUS PUNTEROS

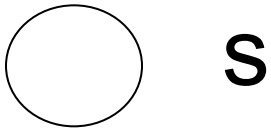
PROCEDIMIENTO GENERAL DE BÚSQUEDA EN GRAFOS

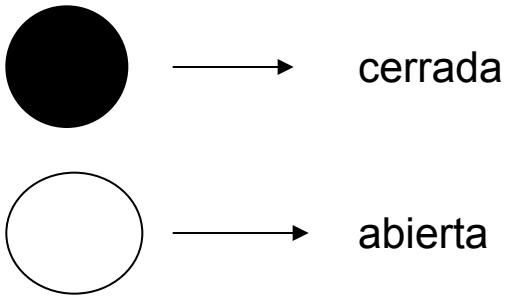
8. REORDENAR LA LISTA ABIERTA DE ACUERDO A UN ESQUEMA ARBITRARIO O A UN MÉRITO HEURÍSTICO

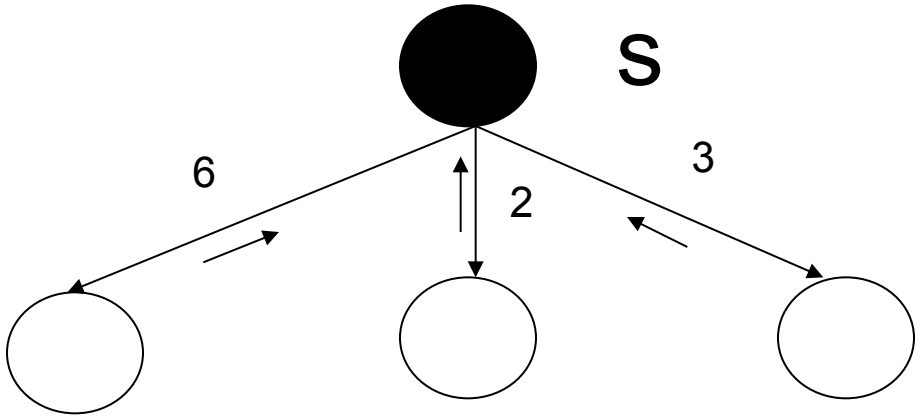
9. IR A 3

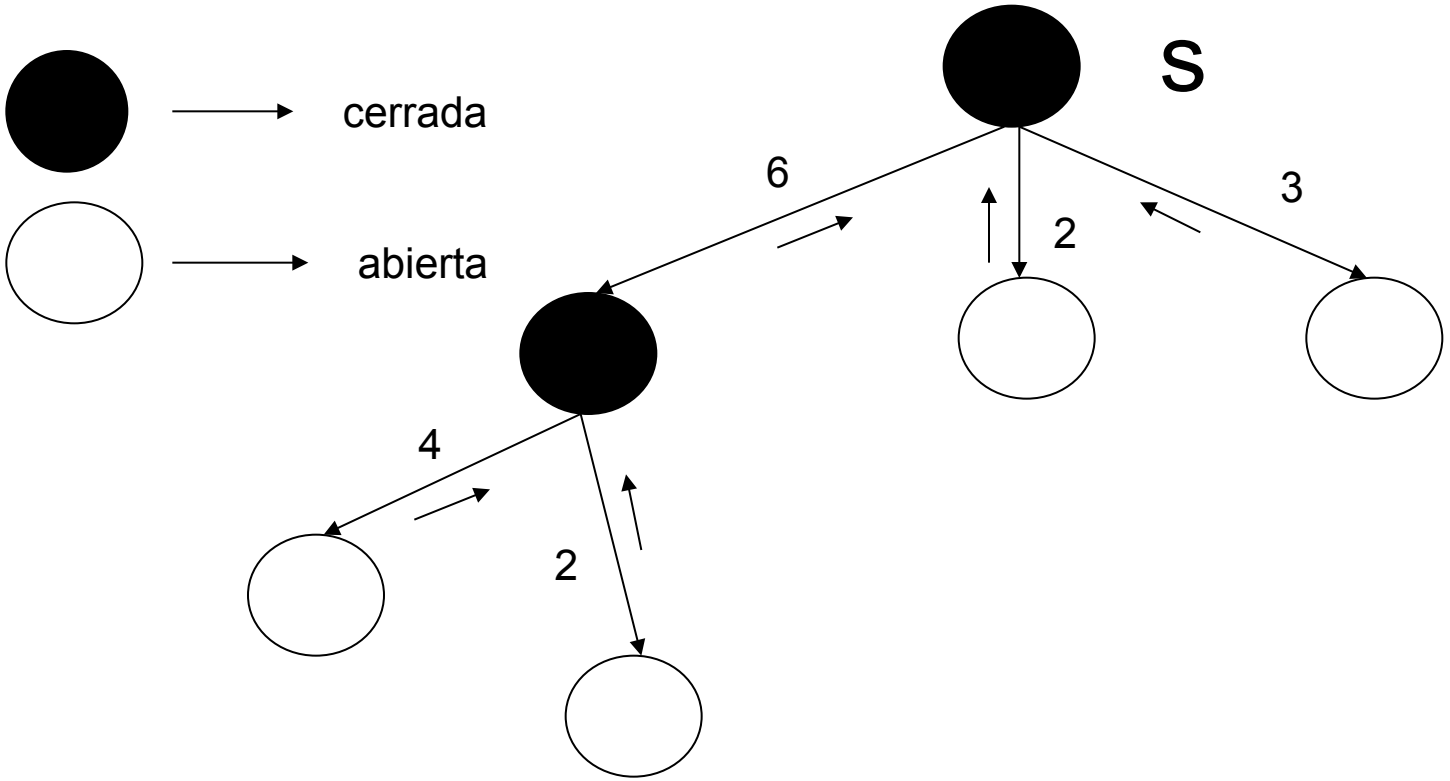
EJEMPLO:

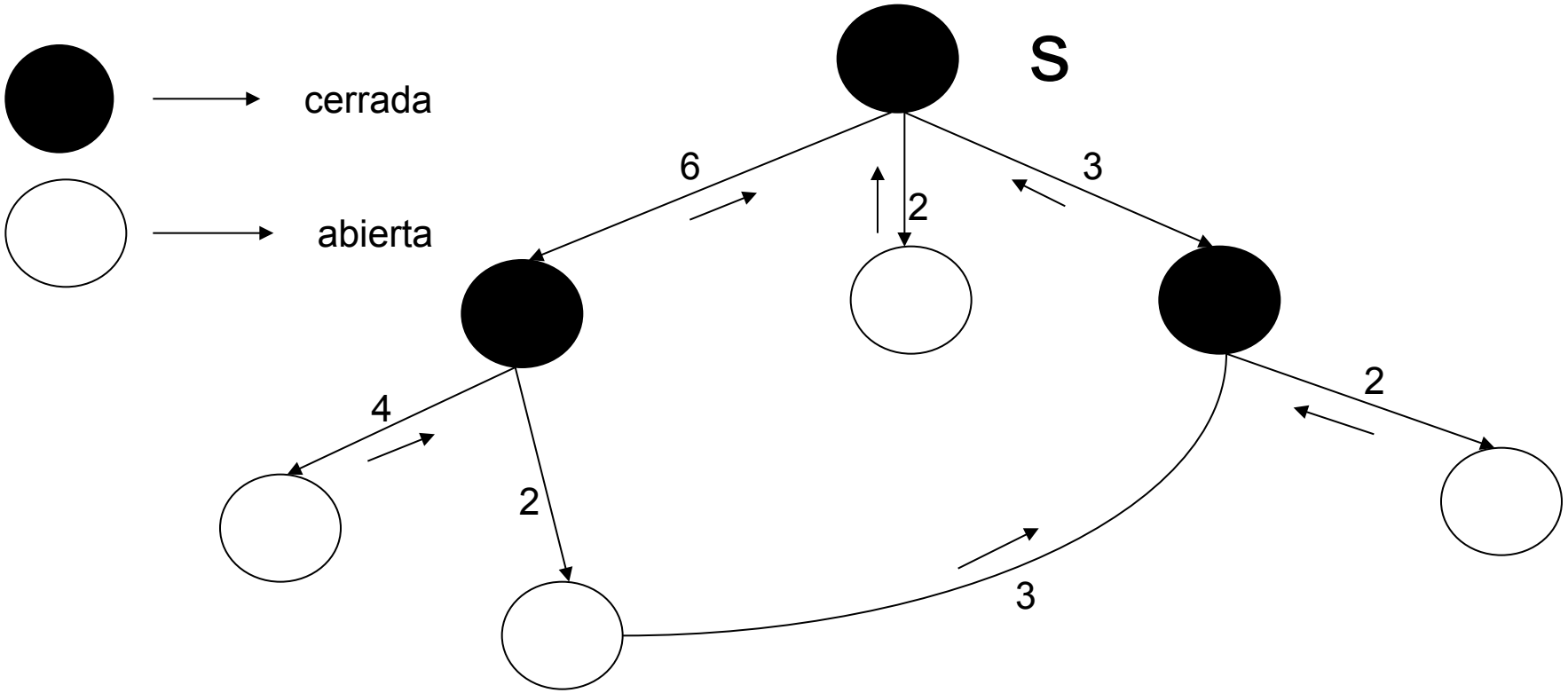


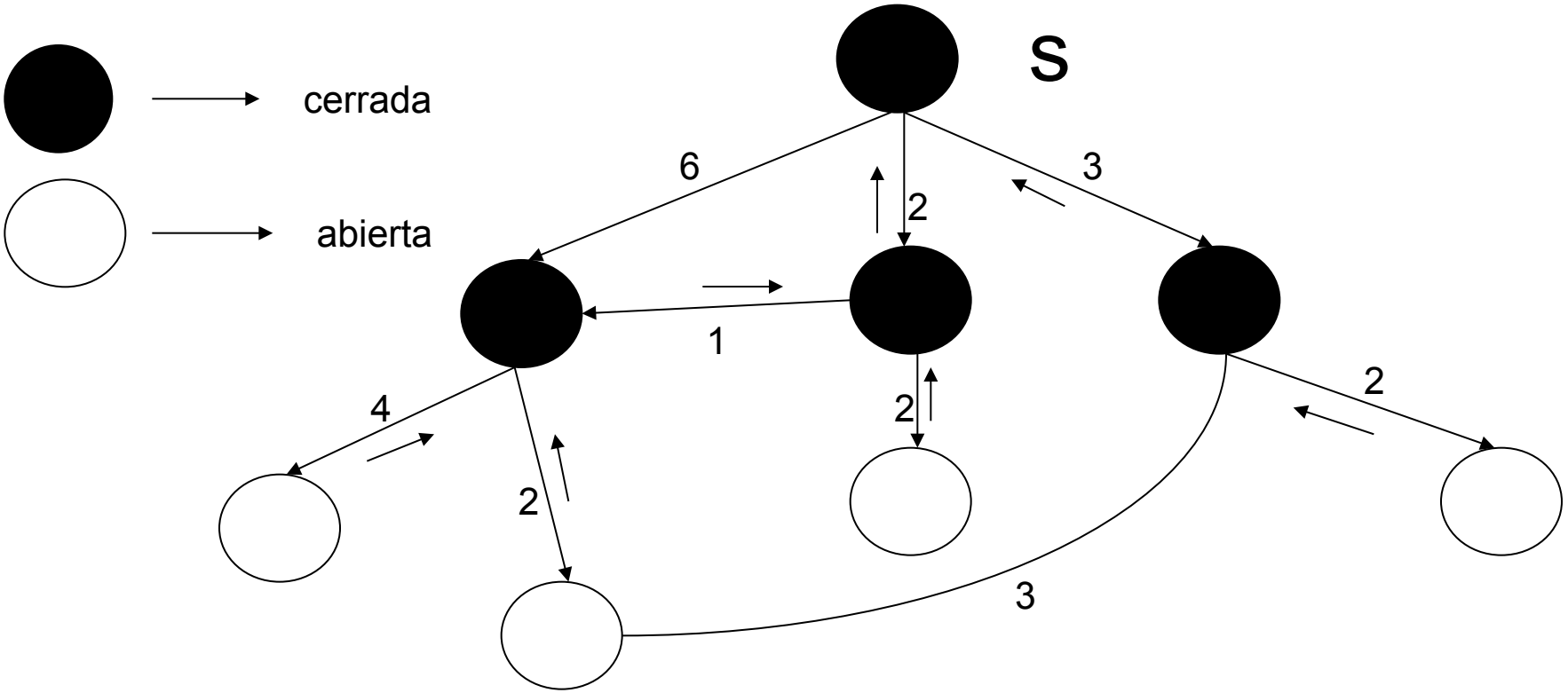



cerrada
abierta

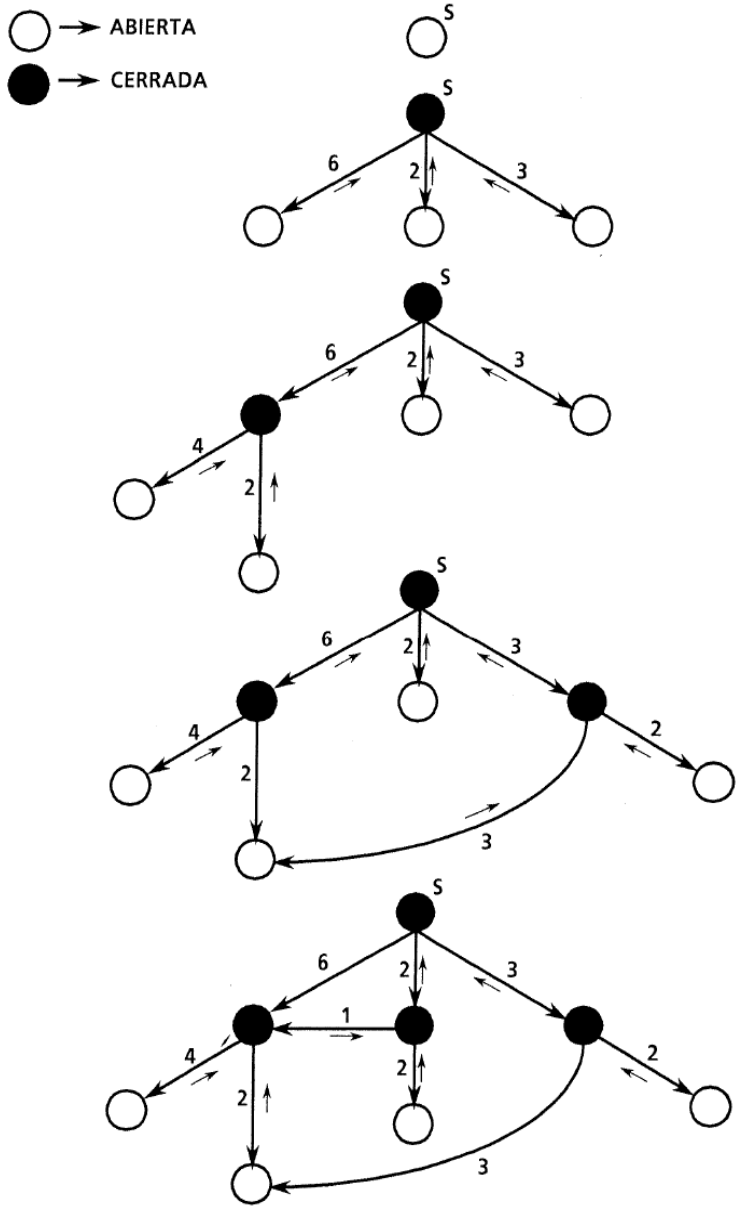

S
6
2
3


cerrada
abierta
S
6
2
3
4
2


cerrada
abierta
S
6
2
3
4
2
2
3


cerrada
abierta
S
6
2
3
1
4
2
2
2
3

EJEMPLO:



EL MEJOR PRIMERO

EN ESTE PROCEDIMIENTO SE ESCOGE PARA SU DESARROLLO EL NODO MAS PROMETEDOR, PARA ELLO ES NECESARIO ASIGNAR A CADA NODO DEL ARBOL UN VALOR EN BASE A UNA FUNCIÓN HEURÍSTICA

ALGORITMO A

EL ALGORITMO **A** ES EL PROCEDIMIENTO GENERAL DE BÚSQUEDA EN GRAFOS UTILIZANDO UNA FUNCIÓN PARA REALIZAR LA REORDENACIÓN DE LOS NODOS EN LA LISTA ABIERTA

ALGORITMO A

$$f^*(n) = g^*(n) + h^*(n)$$

$f^*(n)$ ES EL COSTE REAL DE UN CAMINO ÓPTIMO DESDE EL NODO INICIAL **s** A UN NODO META, RESTRINGIDO A QUE DICHO NODO PASE POR EL NODO **n**

ALGORITMO A

SE DEFINE UNA FUNCIÓN f QUE SEA UN ESTIMADOR DE f* DE LA SIGUIENTE FORMA:

$$f(n) = g(n) + h(n)$$

g ES UN ESTIMADOR DE g*

h ES UN ESTIAMDOR DE h*

ALGORITMO A

g SE DEFINE COMO EL COSTE DEL CAMINO QUE VA DESDE EL NODO INICIAL **s** AL NODO **n** EN EL ARBOL DE BÚSQUEDA, ESTE CAMINO ES EL DE COSTE MÁS BAJO ENCONTRADO HASTA EL MOMENTO

h ES UNA FUNCIÓN HEURÍSTICA

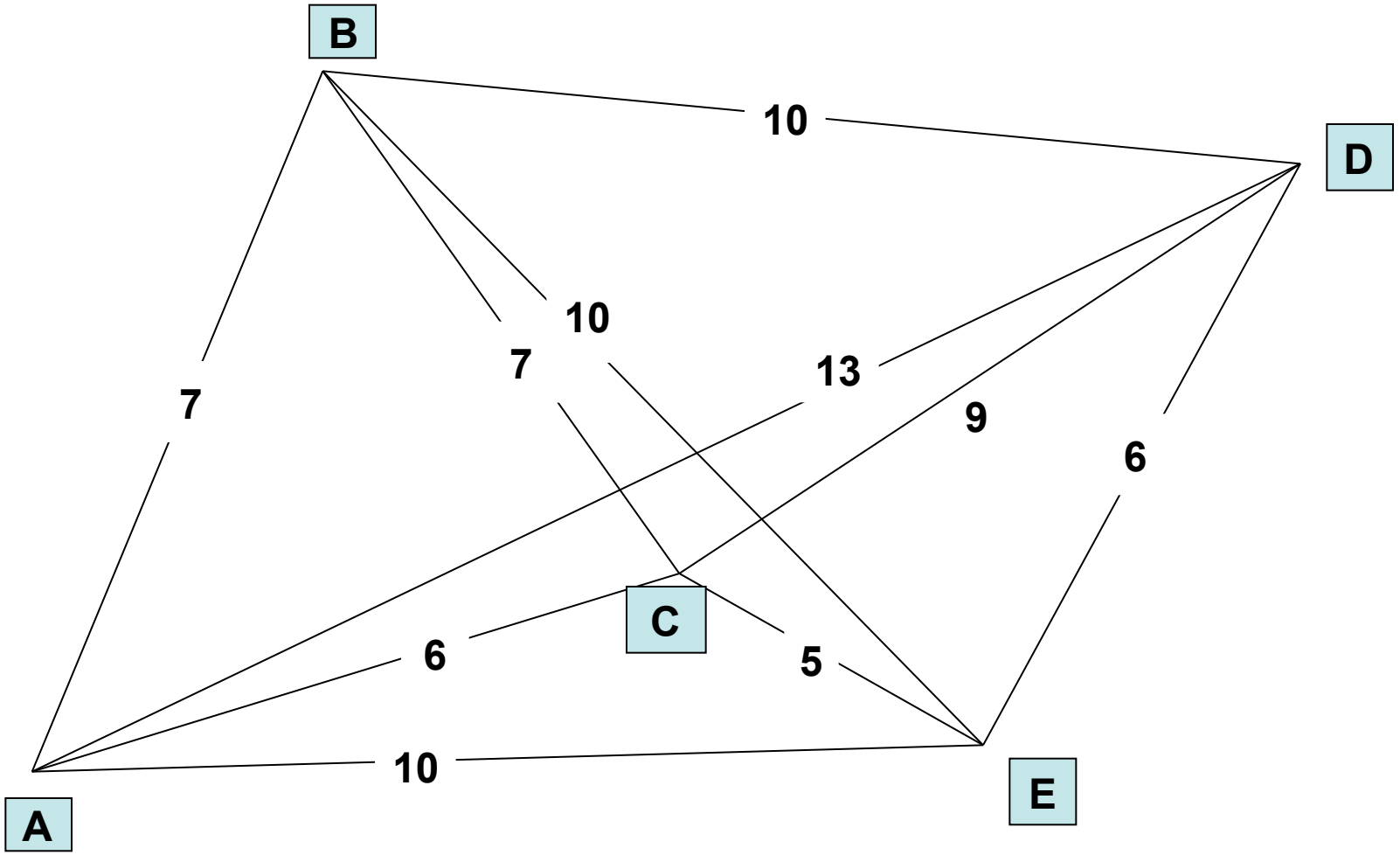
ALGORITMO A

EL ALGORÍTMO **A** SELECCIONA PARA SER EXPANDIDO EL NODO EN ABIERTA QUE TIENE UN VALOR MÁS PEQUEÑO DE LA FUNCIÓN **f**

ALGORITMO A*

SI h(n) ES MENOR O IGUAL QUE h*(n) PARA TODO **n** ENTONCES EL ALGORITMO **A** SE LLAMA **A***

ALGORITMO A*



ALGORÍTMO A*

Un vendedor tiene que visitar cada una de las cinco ciudades de la figura anterior, una sola vez cada una, a excepción de la ciudad de partida que es A a la que tenemos que regresar después de haber visitado el resto de las ciudades

ALGORÍTMO A*

Desarrollar un Sistema de Producción que resuelva el ejercicio anterior, usando el algoritmo A* para resolver el conjunto conflicto, como estrategia de control.

ALGORÍTMO A*

BASE DE HECHOS

La ciudad A

ALGORÍTMO A*

BASE DE REGLAS

R1. **~**(B) → ($B)

R2. **~**(C) → ($C)

R3. **~**(D) → ($D)

R4. **~**(E) → ($E)

R5. (B), (C), (D), (E), **~**(A$A) → ($A)

R6. (A$A) → FIN

ALGORÍTMO A*

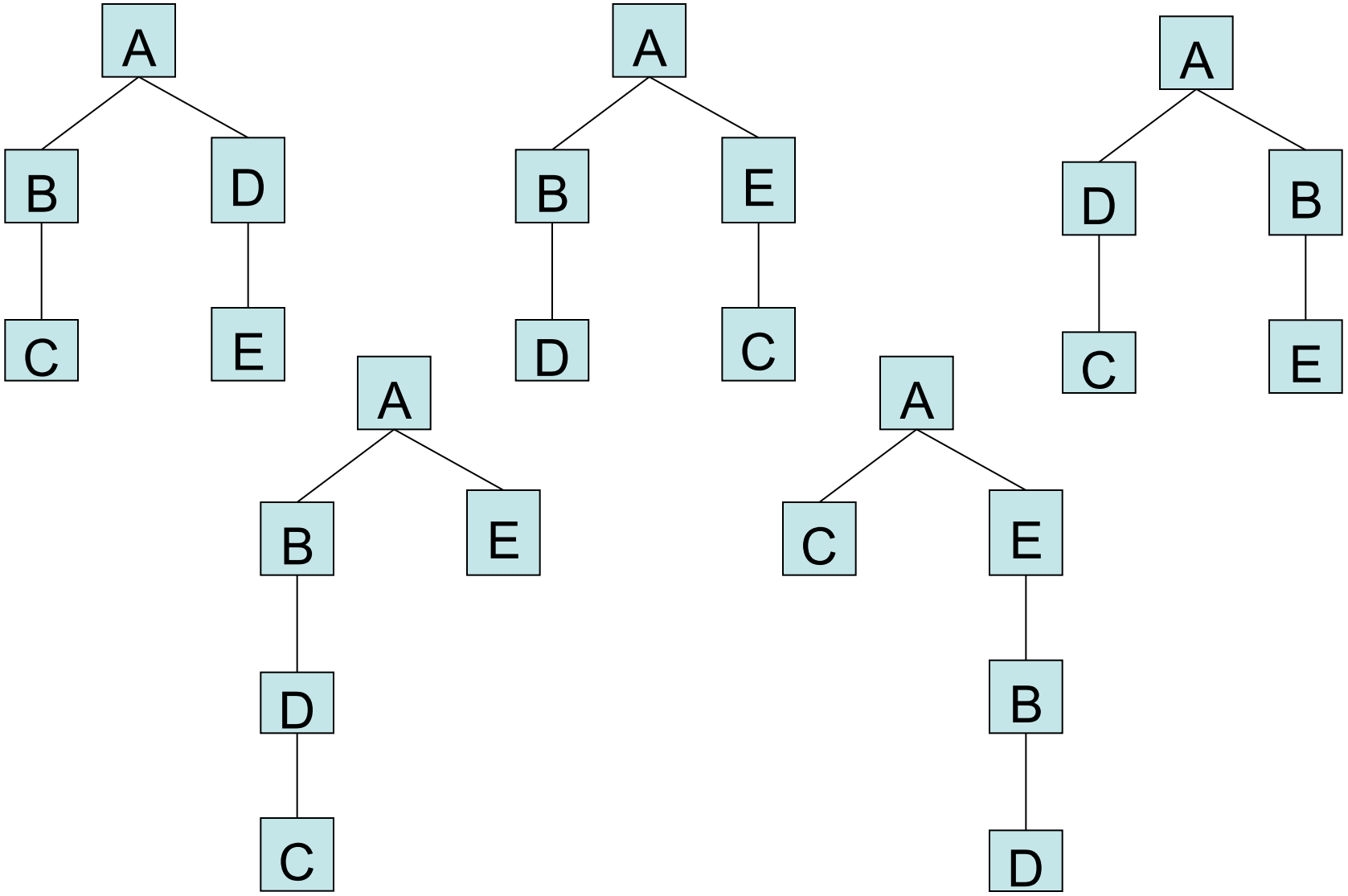
CADA VEZ QUE TENGAMOS QUE RESOLVER EL CONJUNTO CONFLICTO LO HAREMOS A TRAVÉS DEL ARBOL DE MÁXIMO ALCANCE

ALGORÍTMO A*

EL ARBOL DE MÁXIMO ALCANCE CONSITE EN PARTIENDO DEL NODO **A** CREAR UN ARBOL, DE COSTE MÍNIMO, QUE VISITE TODOS LOS NODOS (NO SE CONTEMPLAN LAS CIUDADES QUE SE ENCUENTRAN EN EL CAMINO RECORRIDO, A EXCEPCIÓN DEL NODO FINAL (HOJA) DE ESE CAMINO), Y QUE UNO DE SUS NODOS HOJA SEA EL ÚLTIMO NODO DEL CAMINO RECORRIDO

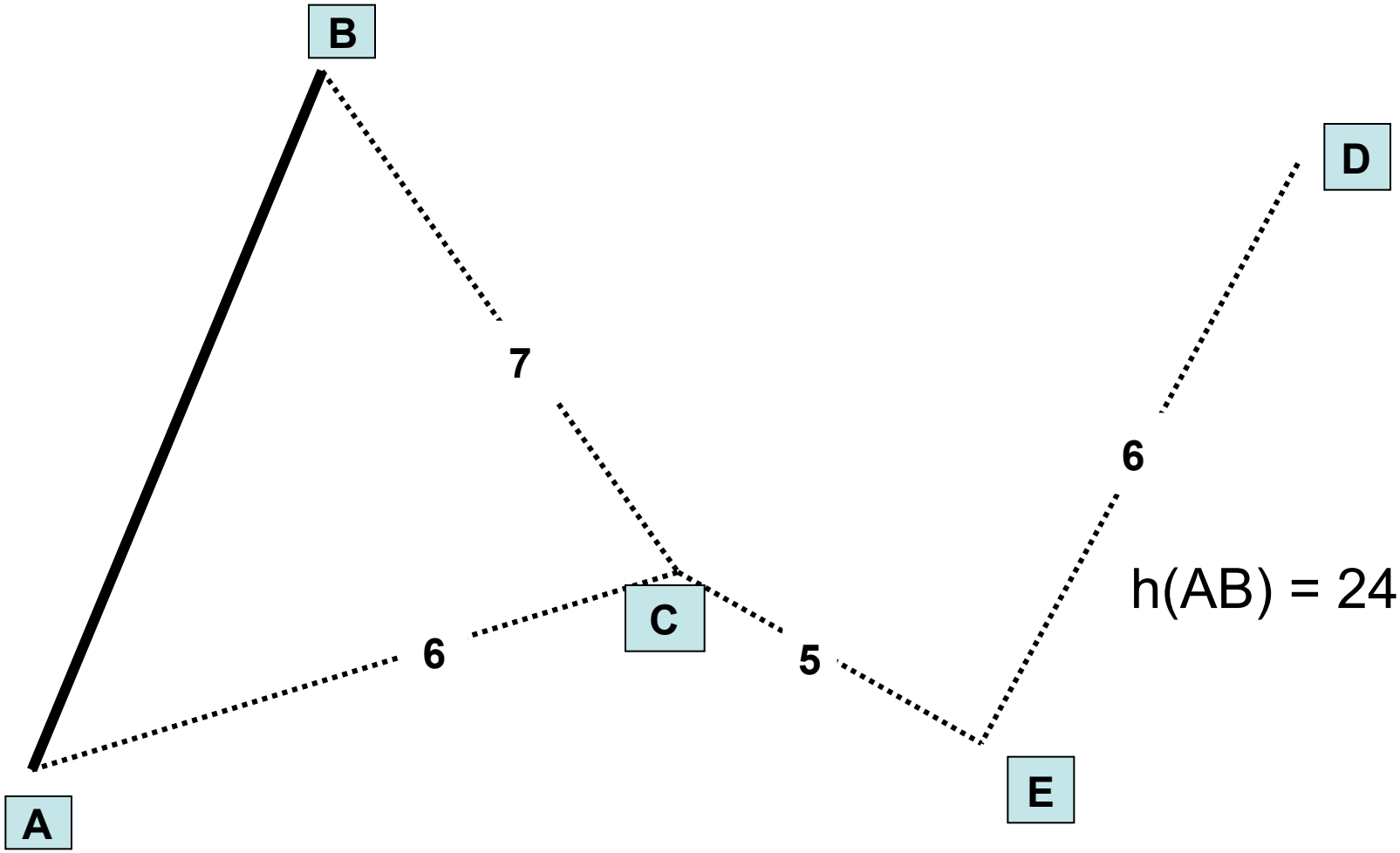
Por ejemplo si decidiéramos visitar C, en el primer instante, algunos de los árboles que cumplen la condición de máximo alcance son:

ALGORÍTMO A*

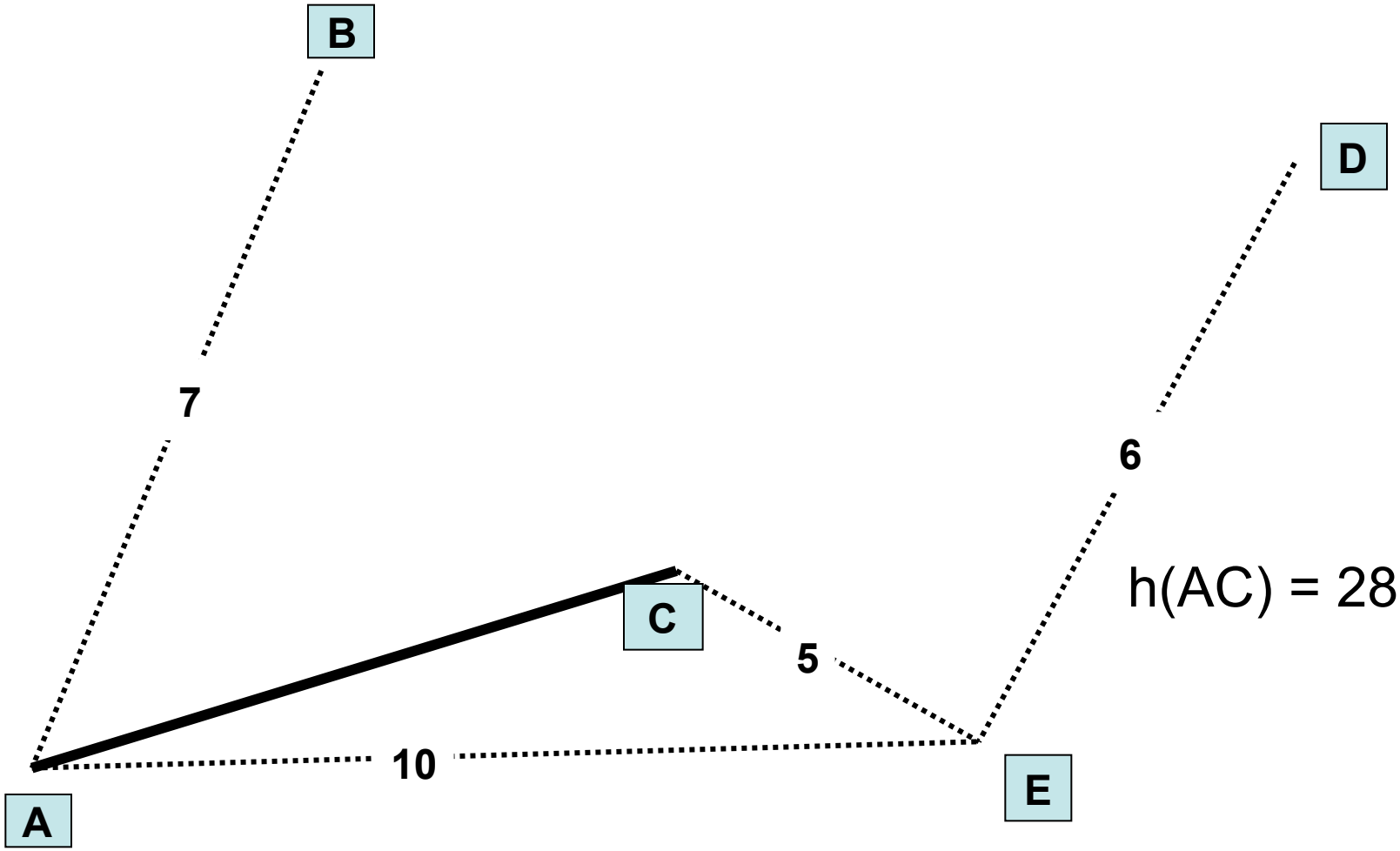


Los árboles de máximo alcance para cada una de las posibilidades existentes son:

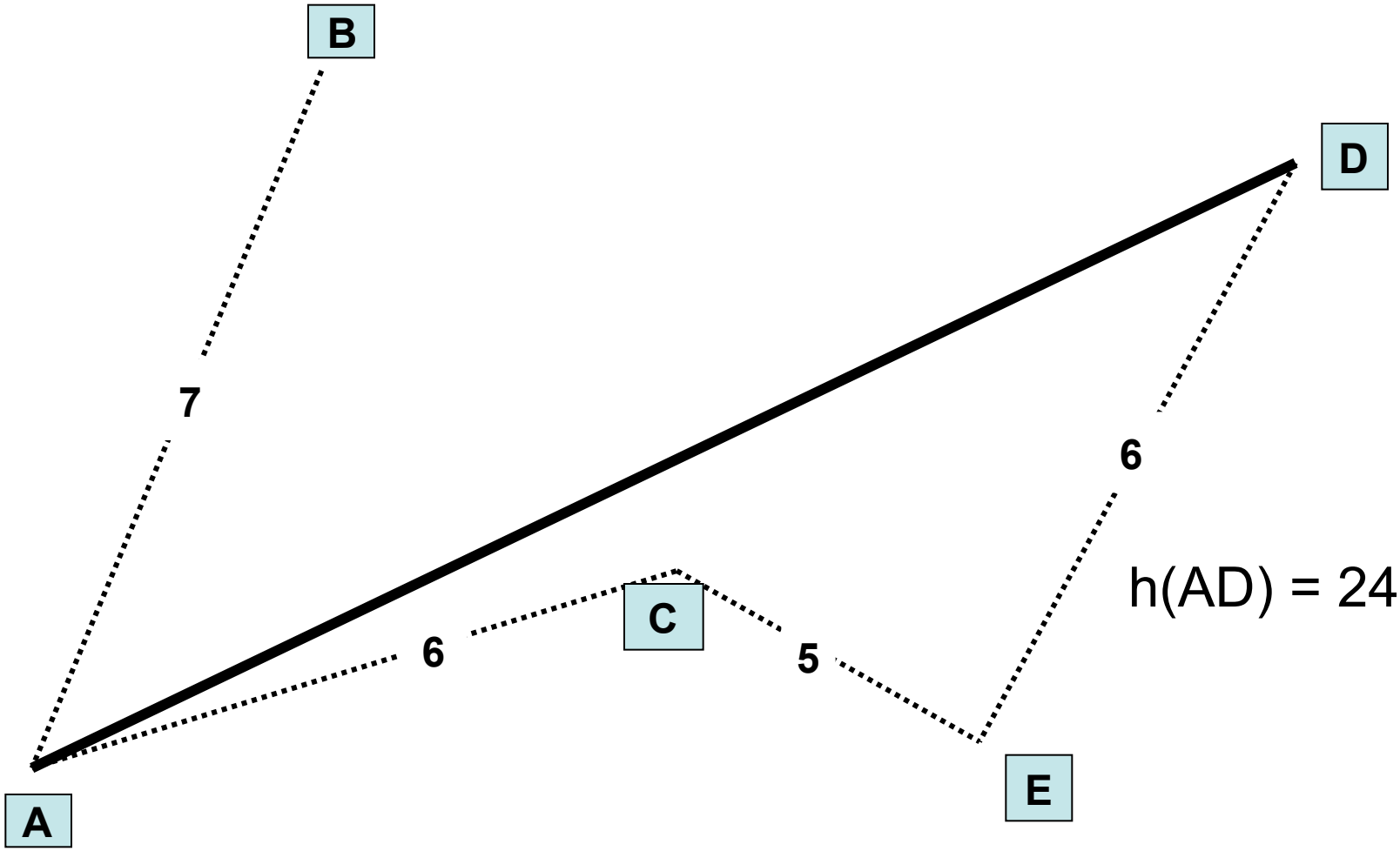
ALGORITMO A*



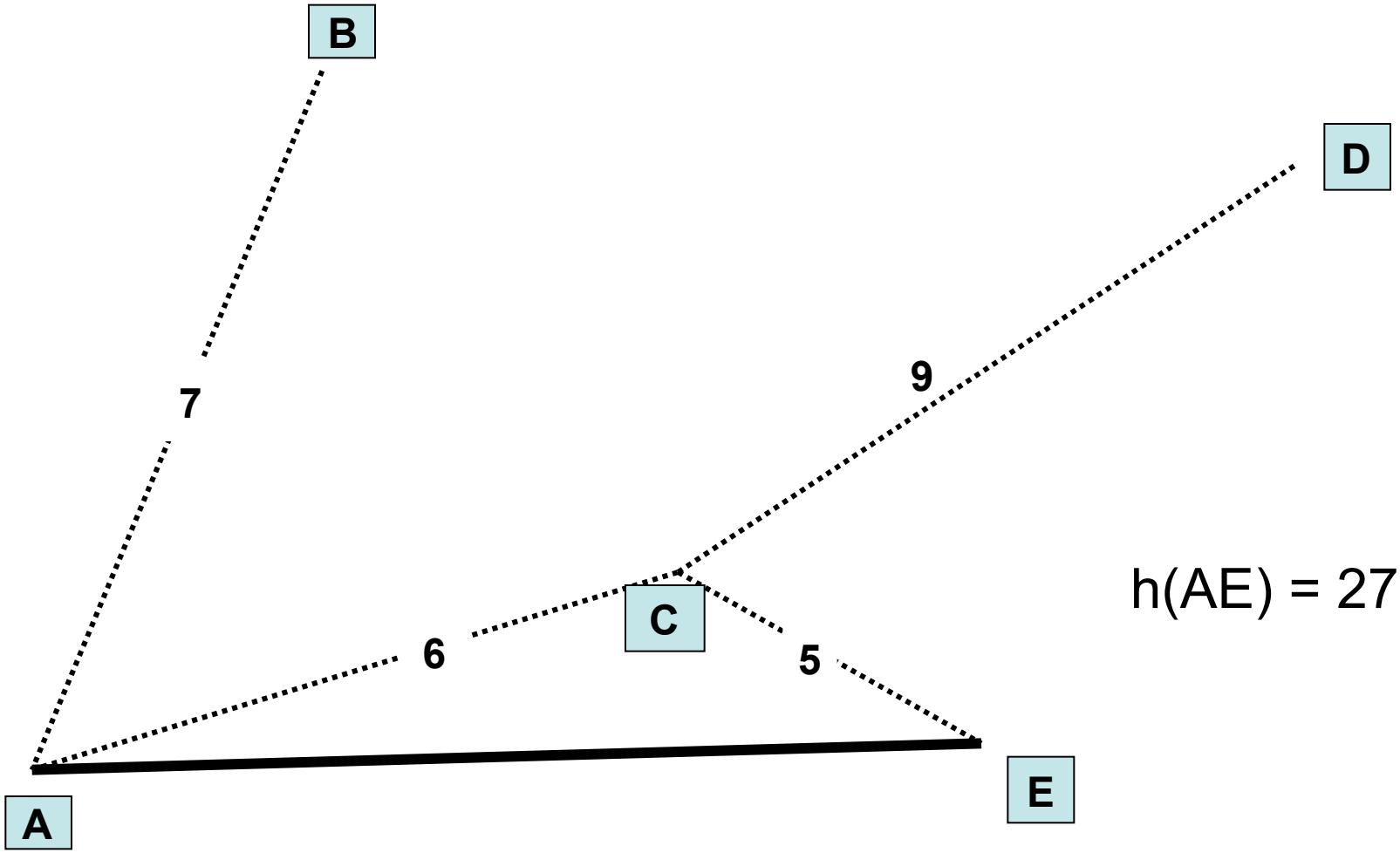
ALGORITMO A*



ALGORITMO A*



ALGORITMO A*

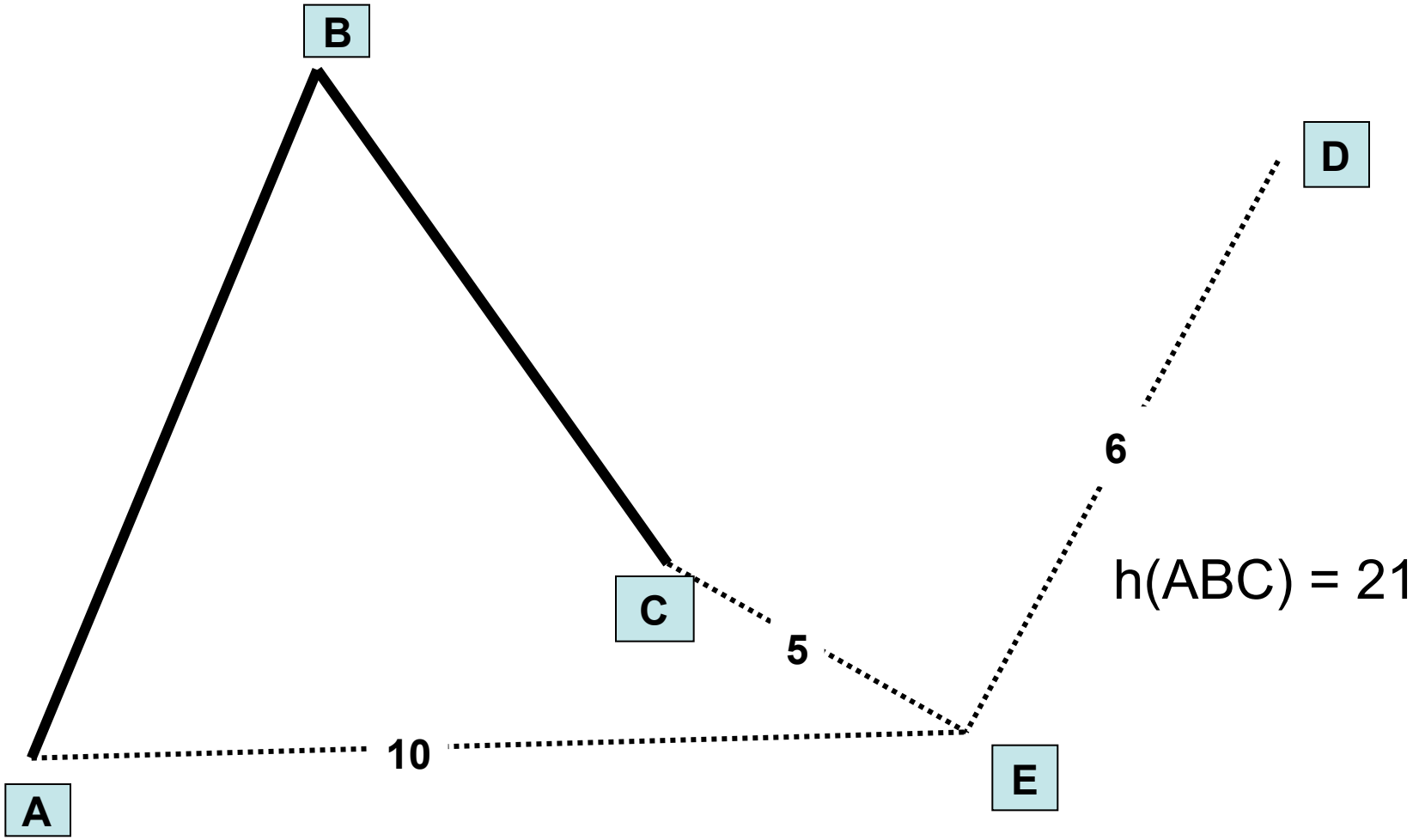


h(AE) = 27

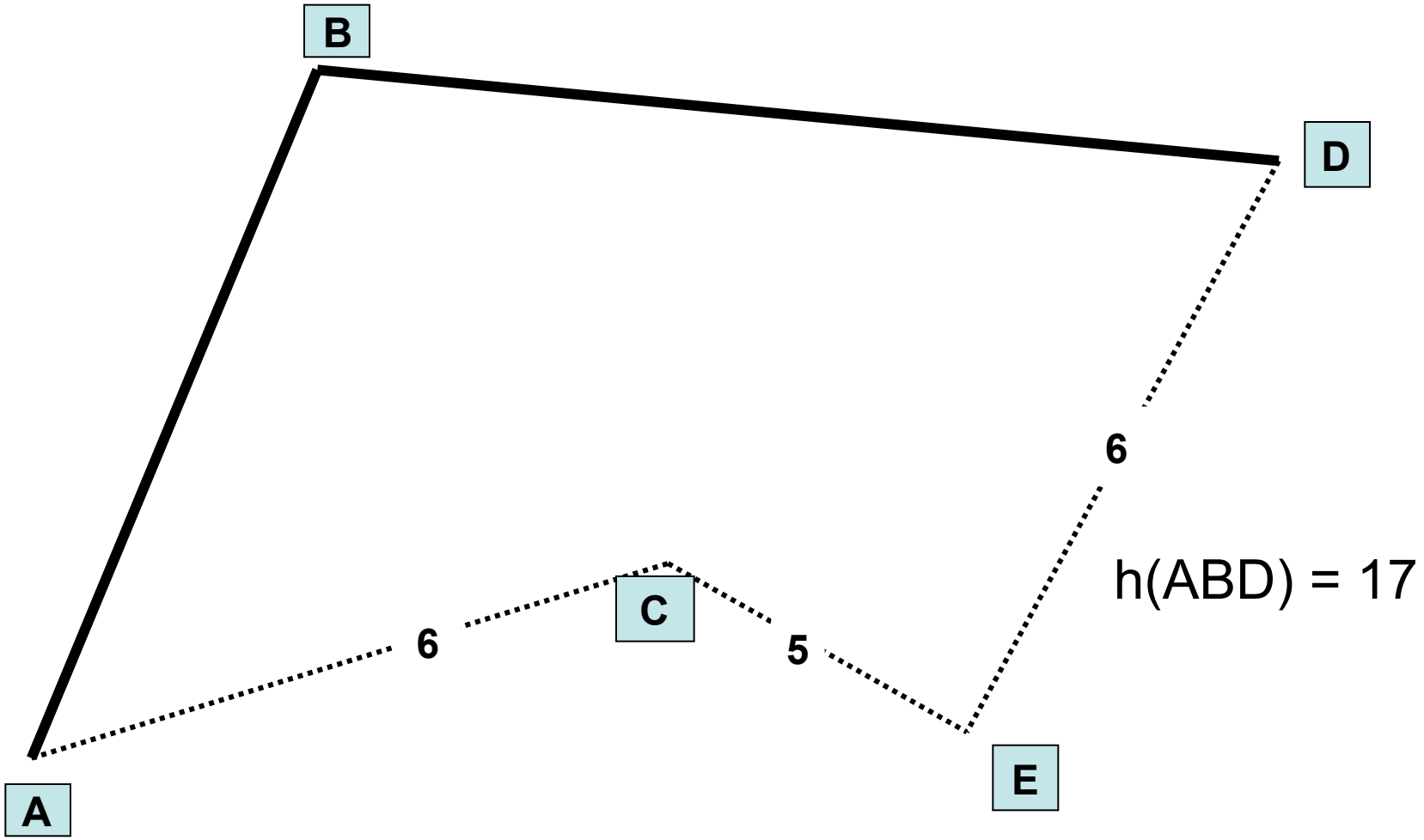
Sumando los valores encontrados (h) al coste de g, escogemos el de menor coste que, corresponde al desplazamiento **AB**

Ahora volvemos a calcular el árbol de máximo alcance para el resto de posibilidades que nos quedan

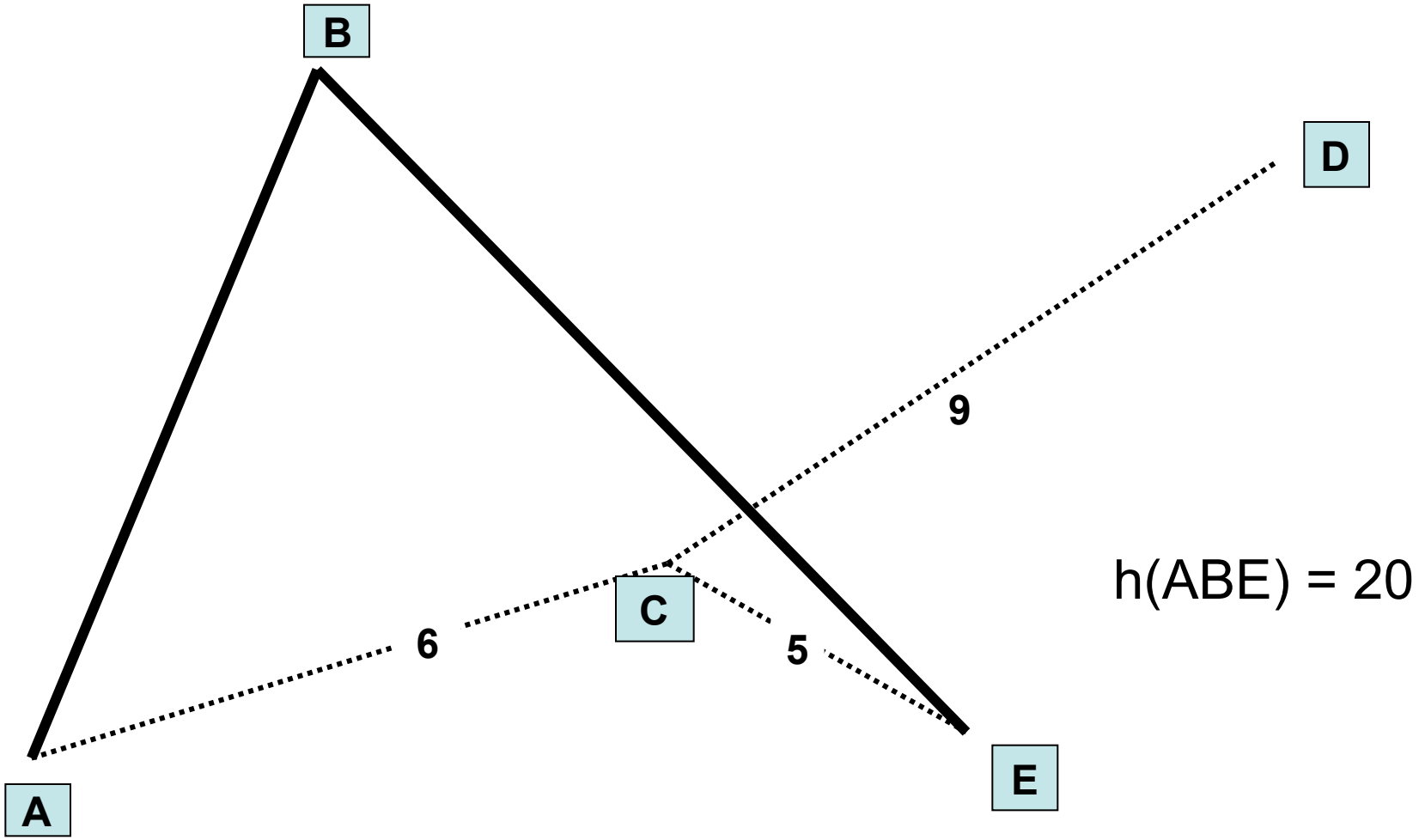
ALGORITMO A*



ALGORITMO A*



ALGORITMO A*

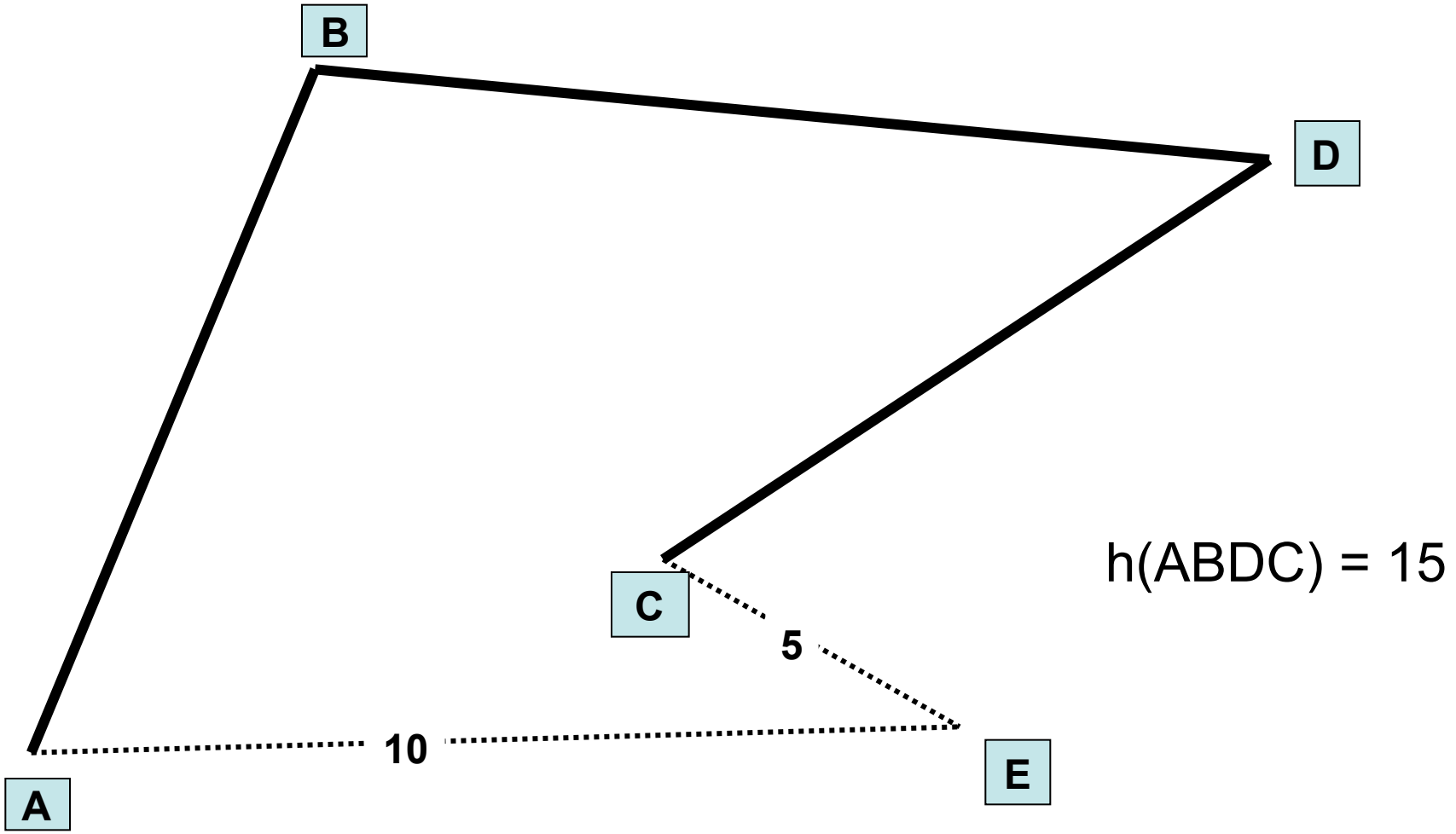


h(ABE) = 20

Sumando los valores encontrados (h) al coste de g, escogemos el de menor coste que, corresponde al desplazamiento **ABD**

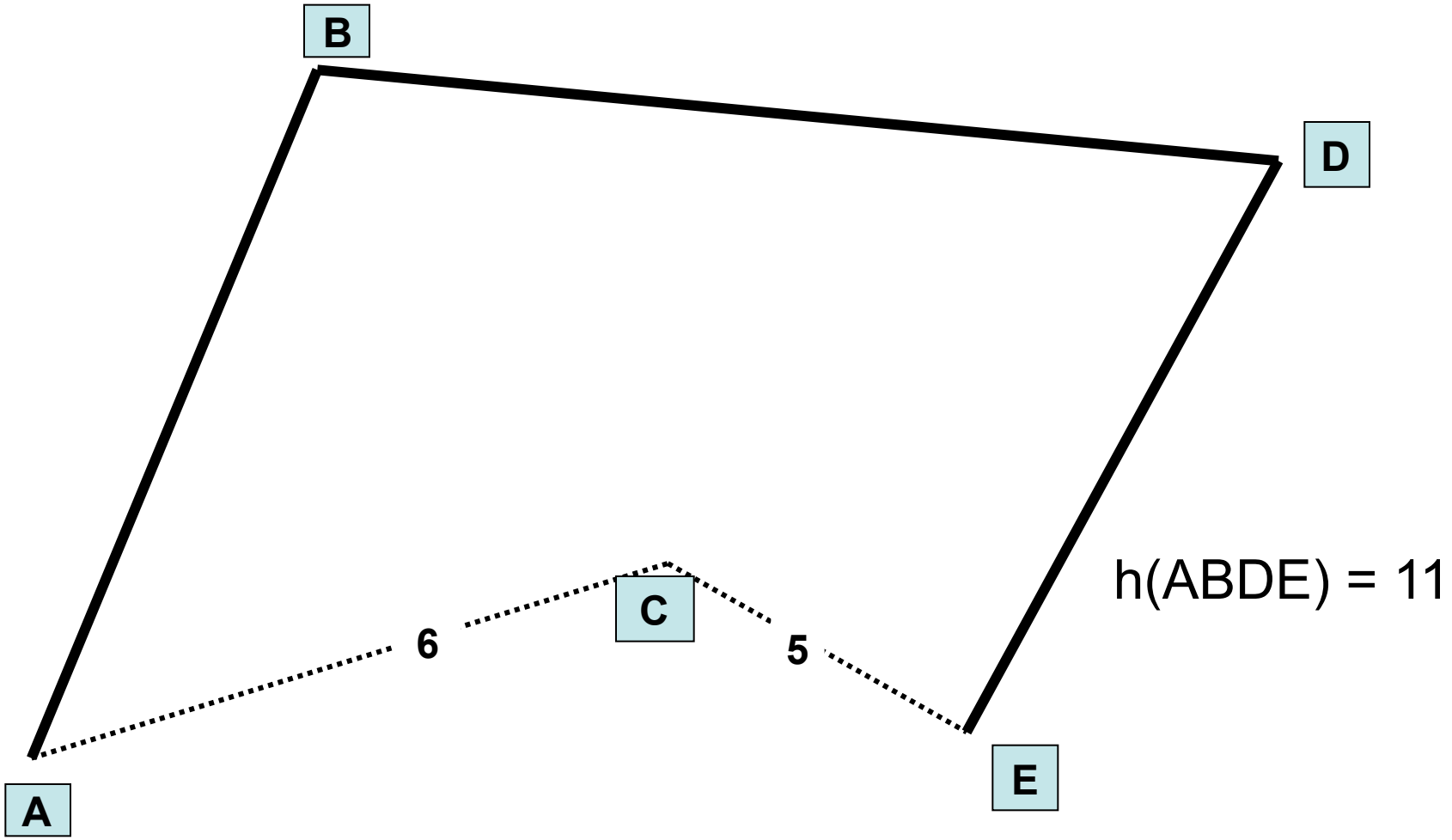
Ahora volvemos a calcular el árbol de máximo alcance para el resto de posibilidades que nos quedan

ALGORITMO A*



h(ABDC) = 15

ALGORITMO A*



Sumando los valores encontrados (h) al coste de g, escogemos el de menor coste que, corresponde al desplazamiento **ABDE**

Ya no queda otra posibilidad que visitar la ciudad **C** antes de volver a la ciudad inicial **A**

PROFUNDIZACIÓN ITERATIVA

CONSISTE EN REALIZAR SUCESIVAS BÚSQUEDAS, PRIMERO EN PROFUNDIDAD, AUMENTANDO EN CADA PASO LA PROFUNDIDAD LÍMITE, HASTA QUE SE ENCUENTRE EL NODO OBJETIVO

PROFUNDIZACIÓN ITERATIVA A* (IDA*)

CONSISTE EN UNA BÚSQUEDA EN PROFUNDIDAD MIENTRAS LA FUNCIÓN f DE LOS NODOS EXPANDIDOS SEA MENOR O IGUAL (SI SE ESTA MINIMIZANDO), O MAYOR O IGUAL (SI SE ESTA MAXIMIZANDO), DE UN UMBRAL (H)

PROFUNDIZACIÓN ITERATIVA A* (IDA*)

EN LA PRIMERA BÚSQUEDA, ESTABLECEMOS EL UMBRAL (H) IGUAL A $f(n) = g(n) + h(n) = h(n)$, DONDE n ES EL NODO INICIAL

PROFUNDIZACIÓN ITERATIVA A* (IDA*)

E X P A N D I M O S N O D O S E N PROFUNDIDAD CON VUELTA ATRÁS SIEMPRE QUE EL VALOR f DE UN SUCESOR DE UN NODO EXPANDIDO EXCEDA DEL VALOR DEL UMBRAL (H)

PROFUNDIZACIÓN ITERATIVA A* (IDA*)

SI ESTA BÚSQUEDA EN PROFUNDIDAD TERMINA EN UN NODO META, ES OBVIO QUE HEMOS ENCONTRADO UN CAMINO DE COSTE MÍNIMO A LA META. EN OTRO CASO EL COSTE DE UN CAMINO ÓPTIMO DEBE SER MAYOR QUE EL VALOR DEL UMBRAL (H)

PROFUNDIZACIÓN ITERATIVA A* (IDA*)

ASÍ, INCREMENTAMOS EL VALOR DEL UMBRAL (H) Y EMPEZAMOS DE NUEVO, ESTE VALOR (H) SERA EL MENOR DE LOS VALORES DE f PARA LOS NODOS VISITADOS (PERO NO EXPANDIDOS) EN EL PROCESO DE BÚSQUEDA, Y ASÍ SUCESIVAMENTE

BIDA*

SE REALIZA UNA BÚSQUEDA EN AMPLITUD DESDE EL NODO FINAL PARA LA CONSTITUCIÓN DE UN CONJUNTO DE NODOS DENOMINADO PERÍMETRO, DESPUES, UNA BÚSQUEDA HACIA DELANTE, A PARTIR DEL NODO INICIAL, QUE SE DETENDRÁ CUANDO SE ALCANCE ALGUNO DE LOS NODOS DEL PERÍMETRO

BIDA*

PARA LA CREACIÓN DEL PERÍMETRO SE CREA EL CONJUNTO A_d QUE CONTIENE TODOS LOS NODOS QUE ESTÁN A UNA DISTANCIA MENOR O IGUAL QUE ***d*** DEL NODO FINAL ***t***

BIDA*

EL PERÍMETRO P_d CONSISTE ENTONCES EN TODOS LOS NODOS DE A_d QUE TIENEN SUCESORES FUERA DE ÉL, ES DECIR, QUE SON DIRECTAMENTE ALCANZABLES DESDE OTROS NODOS QUE NO PERTENECEN AL CONJUNTO A_d (POR LO TANTO, $P_d \leq A_d$)

BIDA*

EVIDENTEMENTE, EL TAMAÑO DE P_d CRECE EXPONENCIALMENTE CON LA DISTANCIA ***d***.

PARA CADA UNO DE LOS NODOS m DEL PERÍMETRO, SE ALMACENA SU DISTANCIA AL NODO FINAL h*(m), Y EL RECORRIDO ÓPTIMO HASTA ESE NODO m.

BIDA*

A CONTINUACIÓN SE REALIZA LA SEGUNDA BÚSQUEDA A PARTIR DEL NODO INICIAL s.

ESTA BÚSQUEDA PODRÍA REALIZARSE CON CUALQUIER ALGORÍTMO PERO EN EL BIDA*, SERÁ EL IDA* CON LA FUNCIÓN HEURÍSTICA

$$h_d(n) = \min_{m \in P_d} \{ h(n,m) + h^*(m,t) \}$$

Así la función que se aplica es:

$$f_d(n) = g(n) + h_d(n)$$

ÁRBOLES ALTERNADOS

ESTOS ÁRBOLES SE APLICAN GENERALMENTE A JUEGOS DE ANTAGONISMO, EN LOS QUE CADA JUGADOR TIENE LA INFORMACIÓN TOTAL SOBRE EL JUEGO, NO EXISTIENDO EL AZAR

ÁRBOLES ALTERNADOS

CADA NODO DE ESTOS ÁRBOLES REPRESENTA UNA POSICIÓN. LOS CONSECUENTES DE UN NODO PROPORCIONAN LAS POSICIONES A LAS QUE SE PUEDE ACCEDER, USANDO EL CONJUNTO DE REGLAS PERMITIDO, A PARTIR DE LA POSICIÓN QUE REPRESENTA ESTE NODO

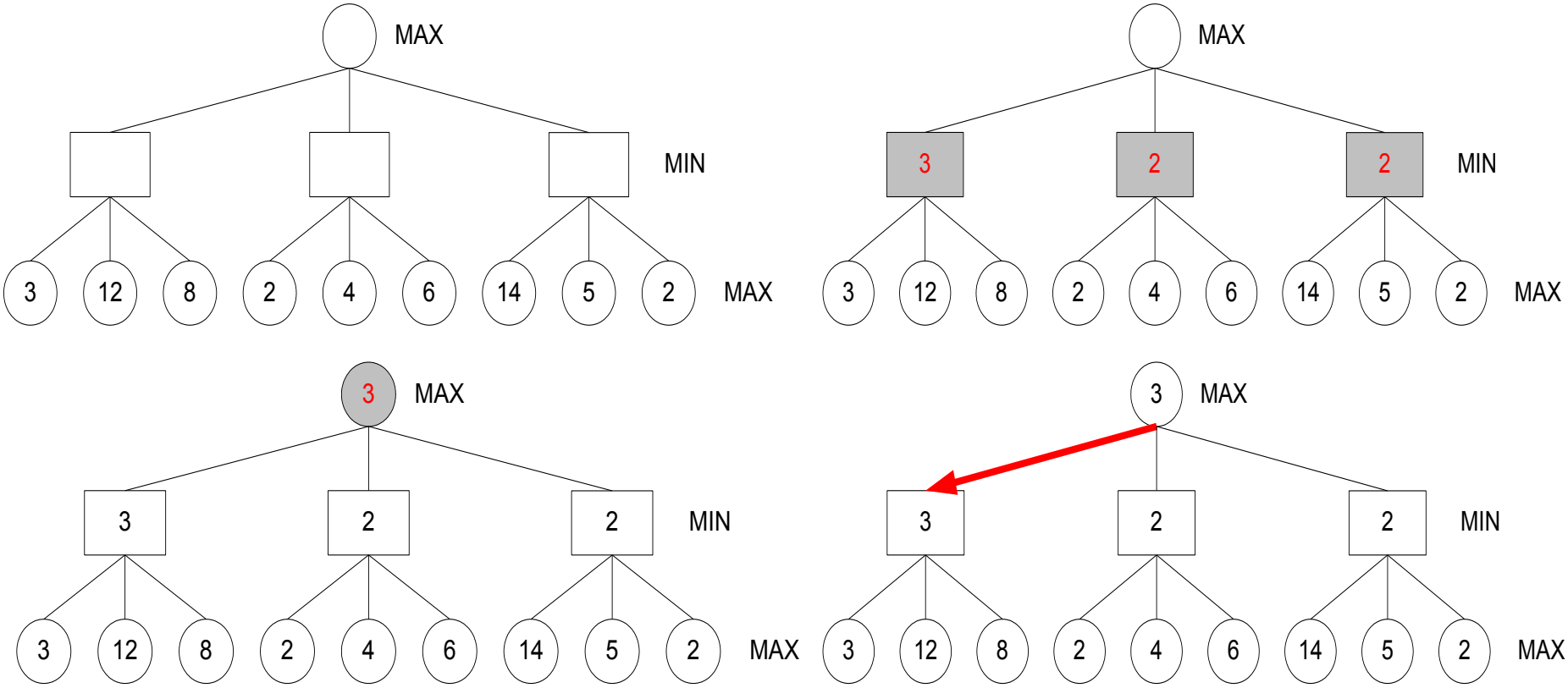
ÁRBOLES ALTERNADOS

CADA NIVEL DEL ÁRBOL CONTIENE LAS CONDICIONES POSIBLES PARA UNO DE LOS ANTAGONISTAS

MÉTODO MINIMAX

EL PROCEDIMIENTO CONSISTE EN, DEFINIENDO UNA FUNCIÓN DE EVALUACIÓN PARA LOS JUGADORES, DESCENDER POR EL ÁRBOL DE JUEGO, UN NÚMERO DE NIVELES, CALCULANDO ESTA FUNCIÓN, Y RETROCEDIENDO CON EL VALOR SUPERIOR OBTENIDO PARA EL JUGADOR QUE REALIZA EL PRIMER MOVIMIENTO, SIENDO ESTE ***MAX*** Y SU OPONENTE ***MIN***

Minimax: ejemplo



- La decisión minimax del jugador MAX es elegir la jugada correspondiente a la rama izquierda.
- Por muy bien que juegue MIN, MAX obtiene un nodo objetivo con valoración de 3.

ALGORITMO DE PODA ALFA-BETA

EN ESTE PROCEDIMIENTO LOS NODOS SE VAN EVALUANDO SEGÚN SON GENERADOS.

ALGORITMO DE PODA ALFA-BETA

LA PODA ALFA-BETA PUEDE EXPLICARSE SIMPLEMENTE COMO UNA TECNICA CONSISTENTE EN NO EXPLORAR AQUELLAS RAMAS DE UN ARBOL DE EXPLORACIÓN QUE EL ANÁLISIS, AL LLEGAR A CIERTO PUNTO, INDIQUE NO SER DE MAYOR INTERÉS PARA EL JUGADOR QUE LLEVA A CABO EL ANÁLISIS O PARA SU ADVERSARIO

ALGORITMO DE PODA ALFA-BETA

LOS VALORES DE ALFA Y BETA, SE CALCULAN COMO SIGUE:

- EL VALOR ALFA DE UN NODO ***MAX*** SE HACE IGUAL AL MÁS ALTO, HASTA EL MOMENTO, DE LOS VALORES FINALES, CALCULADOS HACIA ATRÁS, DE SUS SUCESORES.

- EL VALOR BETA DE UN NODO ***MIN*** SE HACE IGUAL AL MENOR, HASTA EL MOMENTO, DE LOS VALORES FINALES, CALCULADOS HACIA ATRÁS, PARA SUS SUCESORES.

ALGORITMO DE PODA ALFA-BETA

- LOS VALORES ALFA DE LOS NODOS ***MAX*** (INCLUYENDO EL INICIAL) NUNCA PUEDEN DECRECER.

- LOS VALORES BETA DE LOS NODOS ***MIN*** NUNCA PUEDEN CRECER.

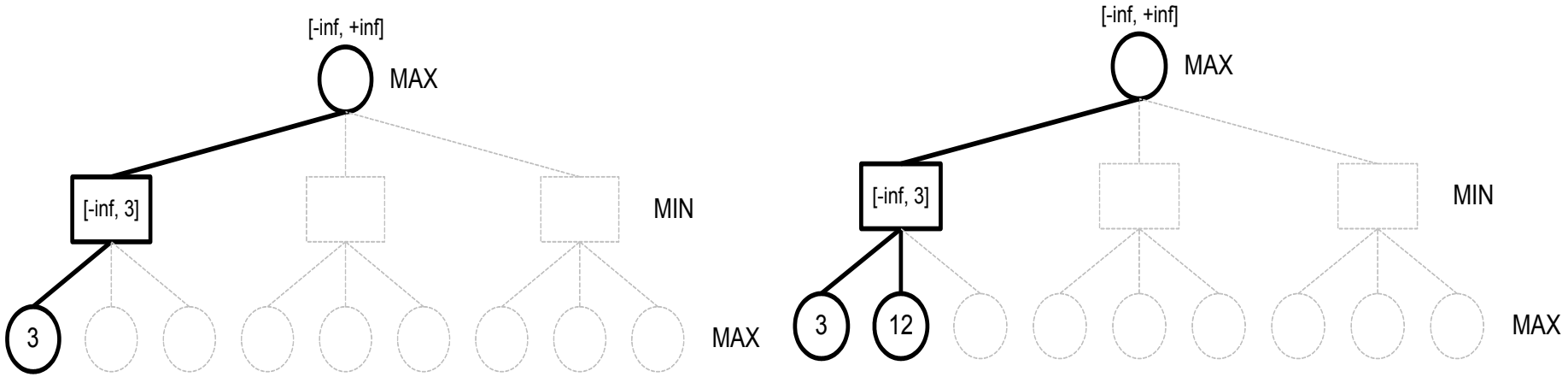
ALGORITMO DE PODA ALFA-BETA

PUEDE SUSPENDERSE LA EXPLORACIÓN POR DEBAJO DE:

- CUALQUIER NODO ***MIN*** QUE TENGA VALOR BETA MENOR O IGUAL QUE EL VALOR ALFA DE CUALQUIERA DE SUS NODOS ***MAX*** ASCENDIENTES SUYOS.

- CUALQUIER NODO ***MAX*** QUE TENGA UN VALOR ALFA MAYOR O IGUAL AL VALOR BETA DE SUS NODOS ***MIN*** ASCENDIENTES.

ALGORITMO DE PODA ALFA-BETA

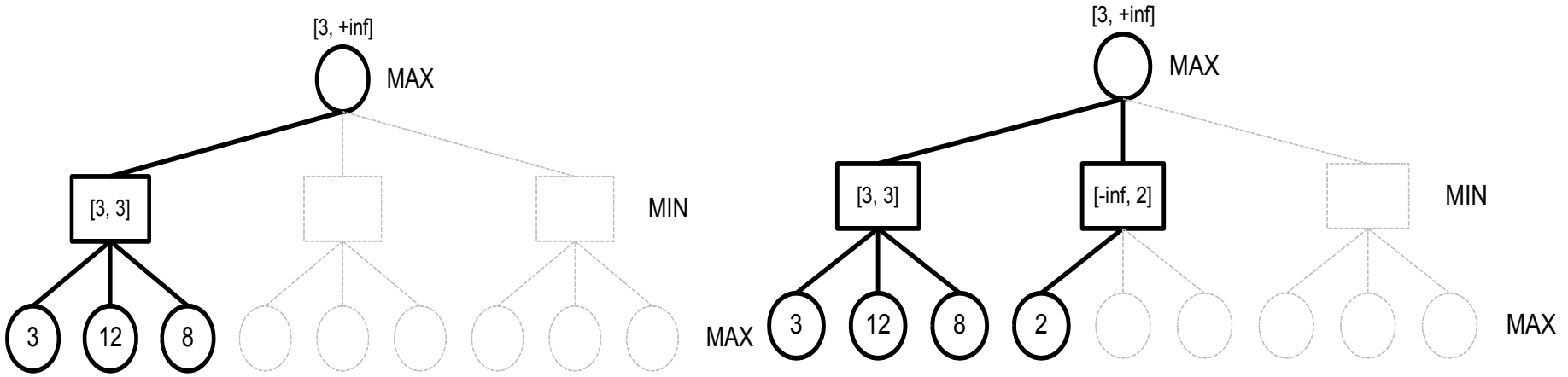


[-inf,3] es el valor que puede tomar MIN hasta el momento.

-inf: puesto que MIN elige el mínimo valor de sus hijos, le queda una alternativa para elegir un valor más bajo que 3.

3: nunca MIN elegirá un valor superior a 3, que es el más pequeño encontrado hasta el momento.

ALGORITMO DE PODA ALFA-BETA



[3,+inf] es el valor que puede tomar MAX hasta el momento.

+inf: puesto que MAX elige el máximo valor de sus hijos, le quedan dos alternativas para elegir un valor más alto que 3.

3: nunca MAX elegirá un valor inferior a 3, que es el más alto encontrado hasta el momento.

ALGORITMO DE PODA ALFA-BETA

El problema de la decisión minimax es que el número de nodos a explorar es exponencial.

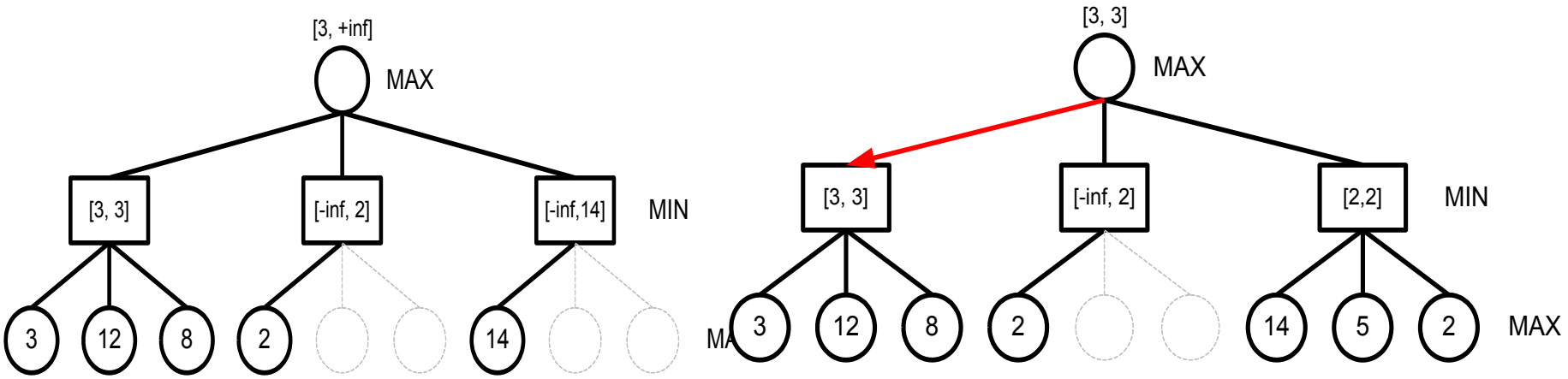
Es posible podar el árbol del juego y obtener una estrategia óptima: poda alfa-beta.

Con esta poda, la decisión minimax es independiente de los valores de las ramas podadas.

α es el valor de la mejor opción para MAX (valor más alto)

β es el valor de la mejor opción para MIN (valor más pequeño)

ALGORITMO DE PODA ALFA-BETA



Es posible podar dos nodos, debido a que el máximo valor que se puede obtener en la rama central (2), no supera al actual de MAX (3), por lo que nunca se elegirá esa opción.

La decisión minimax después de la poda es la misma.

Procedimiento de búsqueda en espacio de estados (SSS*)

Un estado del SSS* tiene la estructura (n,e,h)

n = nodo evaluado

e = estado nodo (activado/estudiado)

h = función heurística para el nodo n

Algoritmo SSS*

1. Introducir en ABIERTA el estado inicial

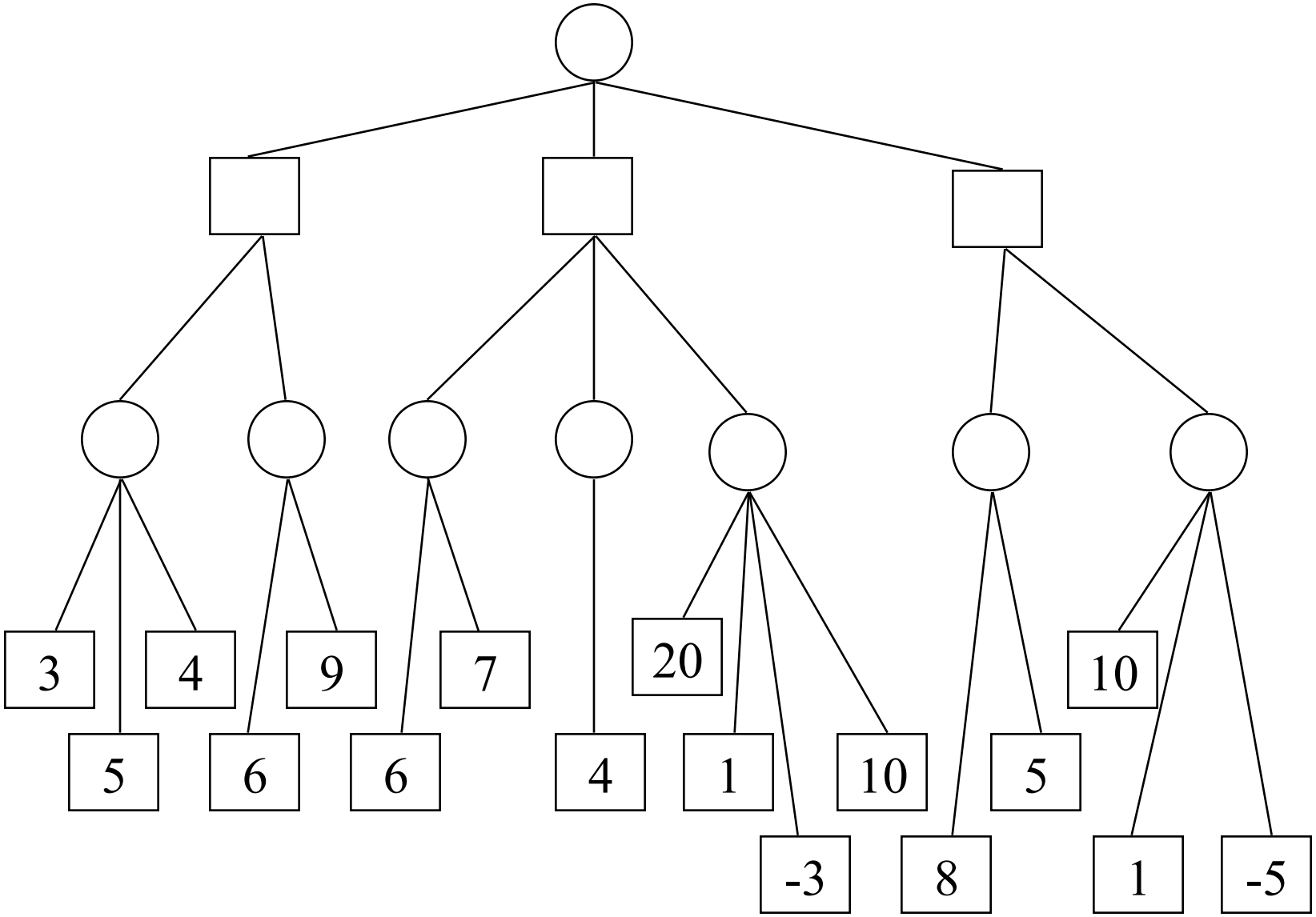
(n = i, e = activo, h = + ∞)

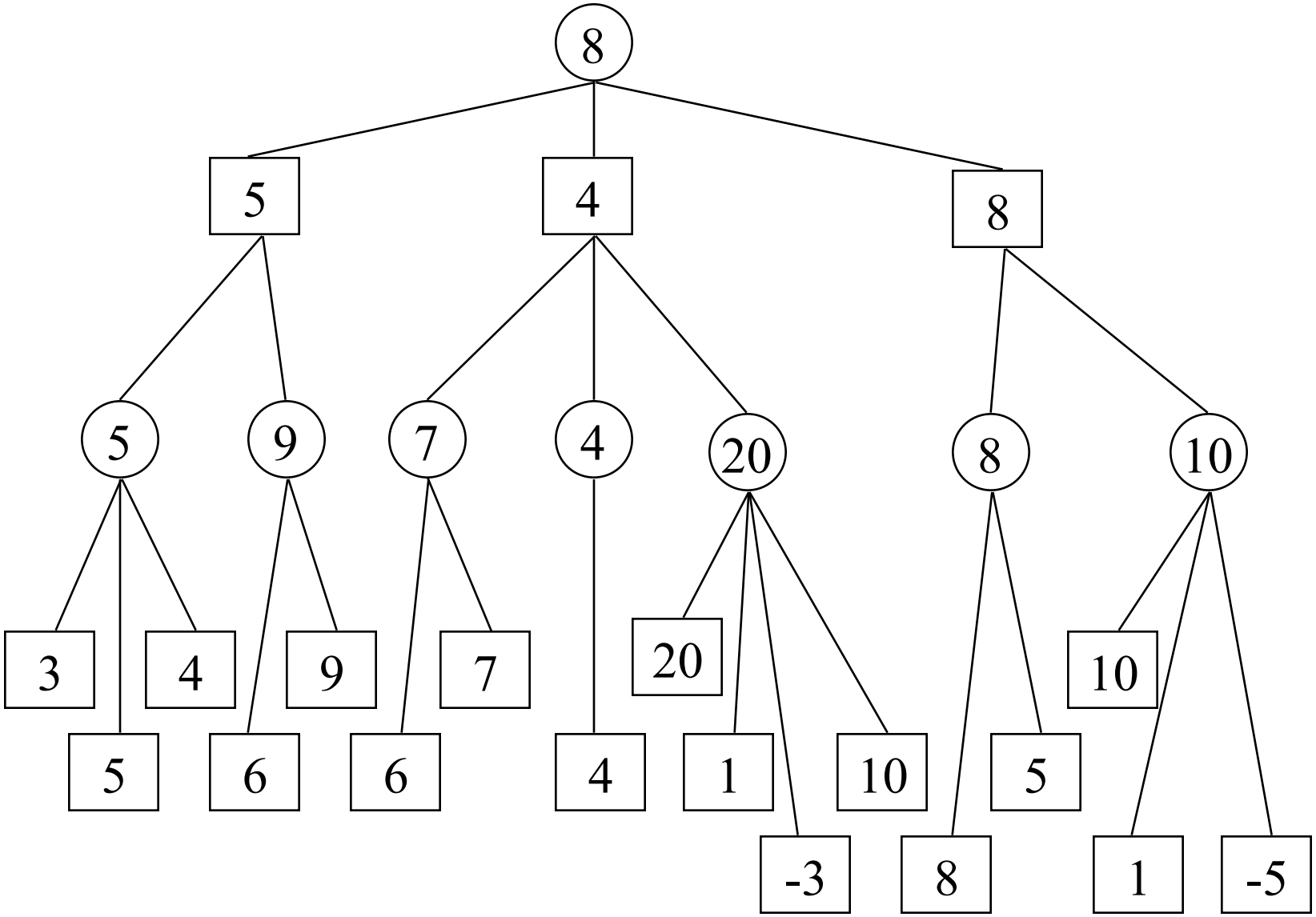
2. Eliminar el primer estado de ABIERTA (el de mayor h) (p = (n,e,h))
3. Si n = i y e = estudiado, terminar con h = valor minimax del juego
4. Sino, expandir el nodo p, aplicando un operador del espacio de estados Γ como se indica en la tabla SSS*
5. Ir a 2

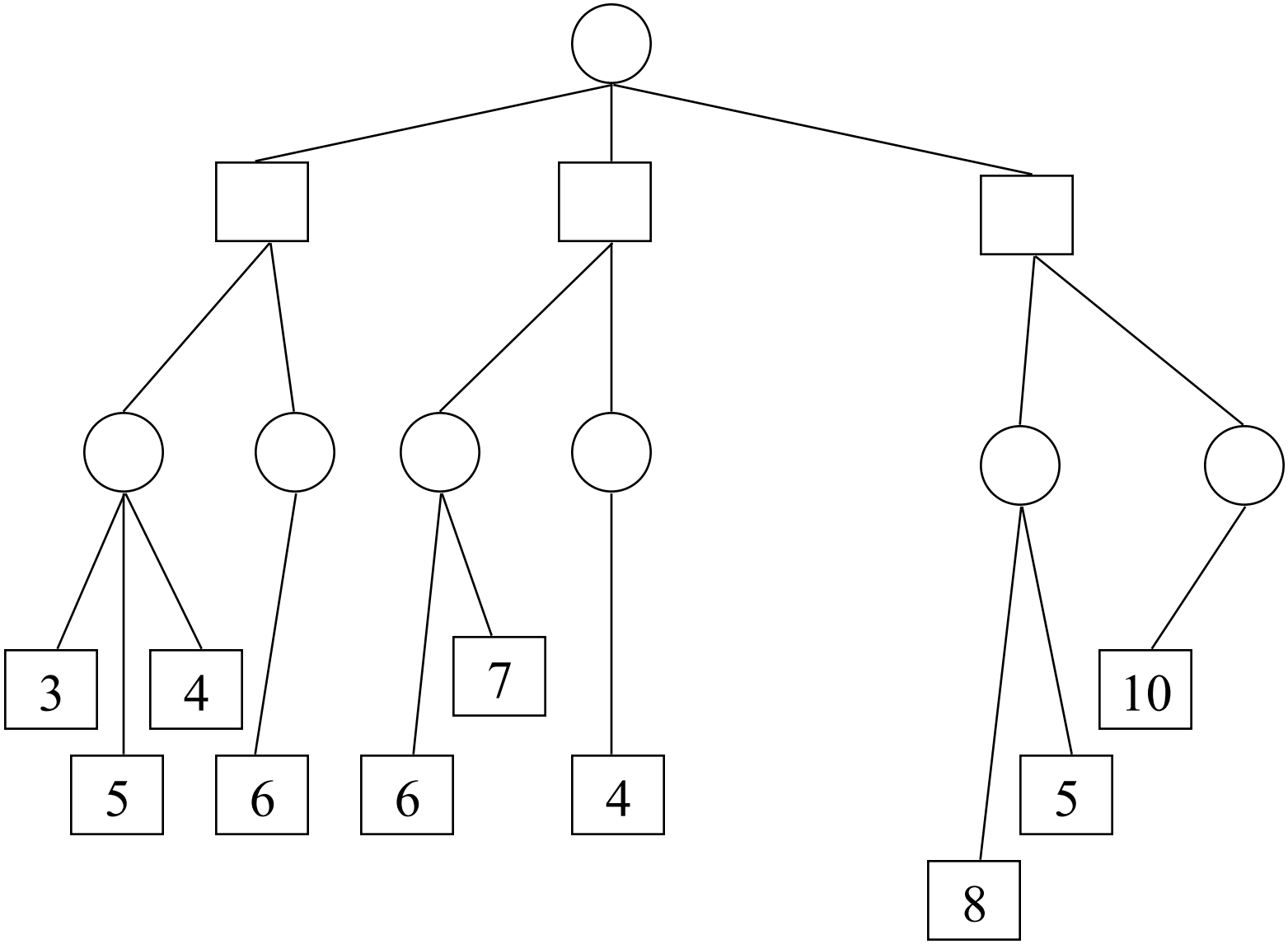
Caso de Γ	Condiciones satisfechas por el estado (n,s,h)	Acción de Γ
1	Si s = ACTIVO y n = MAX no terminal	Añadir al principio de ABIERTA los sucesores de n (n, ACTIVO, h) y BORRAR n h(padre) = h(hijos)
2	Si s = ACTIVO y n = MIN no terminal	Añadir al principio de ABIERTA el 1er. Sucesor de n (n, ACTIVADO, h) y BORRAR n h(padre) = h(hijo)

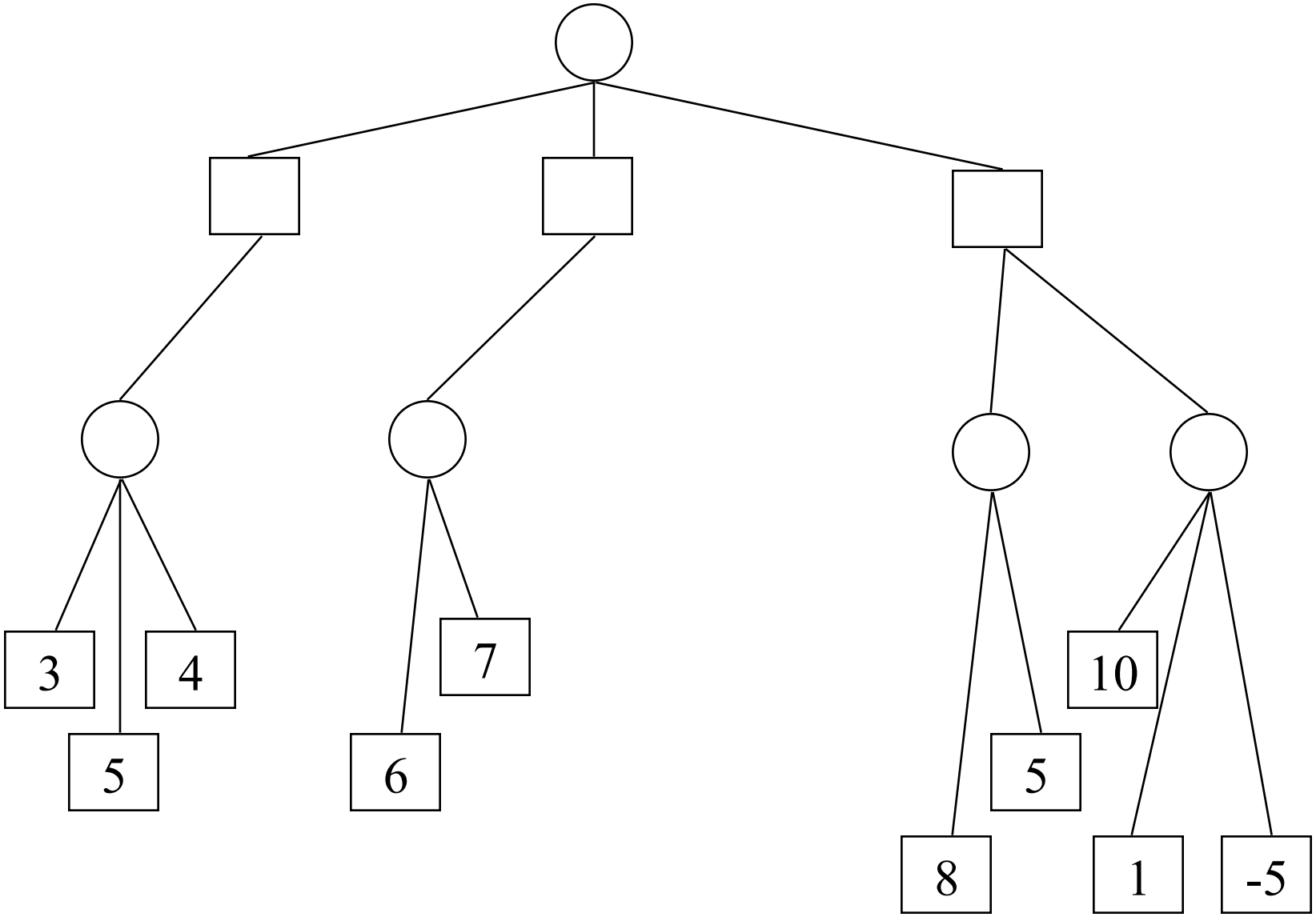
Caso de Γ	Condiciones satisfechas por el estado (n,s,h)	Acción de Γ
3	Si s = ACTIVO y n es un nodo terminal	Introducir n delante de todos los estados ESTUDIADOS de ABIERTA con menor h (n, ESTUDIADO, min{h,f(n)}). Si existen empates se ordenan de izquierda a derecha como aparecen en el árbol
4	Si s = ESTUDIADO, n = MAX y n tiene hermanos sin estudiar	Añadir al principio de ABIERTA el siguiente hermano de n (ni +1,ACTIVADO,h) con h(ni) = h(ni+1) y BORRAR n

Caso de Γ	Condiciones satisfechas por el estado (n,s,h)	Acción de Γ
5	Si s = ESTUDIADO, n = MAX y n no tiene hermanos sin estudiar	Añadir al principio de ABIERTA el padre de n (padre(n), ESTUDIADO, h) con h(hijo) = h(padre) y BORRAR n
6	Si s = ESTUDIADO, y n = MIN	Añadir al principio de ABIERTA el padre de n (padre(n), ESTUDIADO, h) con h(hijo) = h(padre), BORRAR n y todos los sucesores de padre(n)


3
4
9
7
20
10
5
6
6
4
1
10
5
-3
8
1
-5


8
5
4
8
5
9
7
4
20
8
10
3
4
9
7
20
10
5
6
6
4
1
10
5
-3
8
1
-5


3
4
5
6
6
7
4
8
5
10


3
4
5
7
6
10
5
8
1
-5