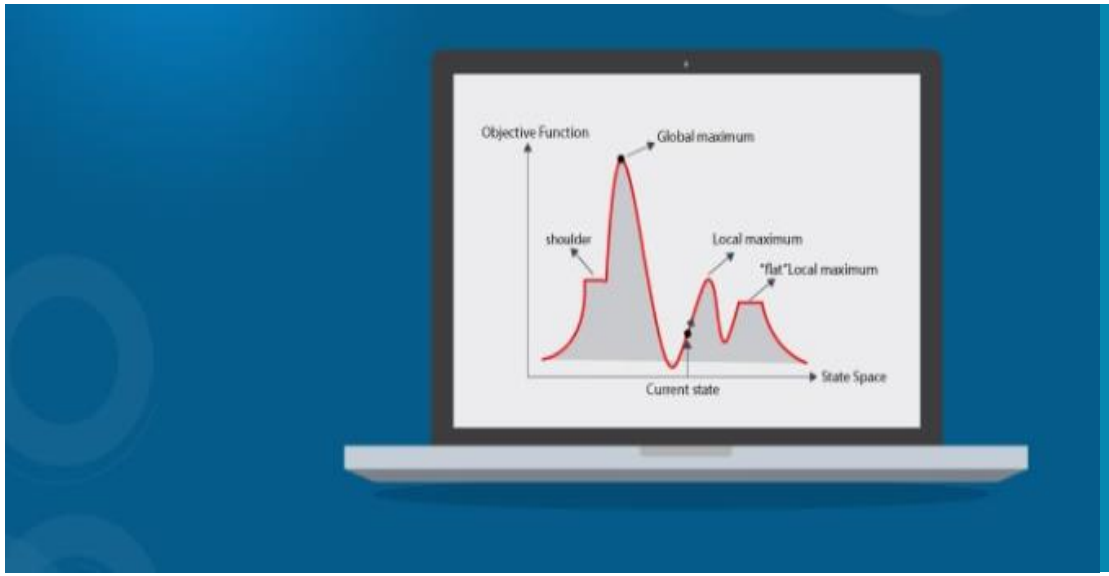


Inteligencia Artificial

Unidad 1: Representación Avanzada del Conocimiento

TEMA 2: Algoritmos de la IA Clásica

Módulo 2: Algoritmo de Búsqueda **Hill-Climbing**

Unidad 1

Representación Avanzada del Conocimiento

TEMA 2: Algoritmos de la IA Clásica

Sesión 6

MÓDULO 2: Algoritmo de Búsqueda Hill-Climbing

Contenido

1. Búsqueda local y optimización
2. ¿Cómo funciona el Algoritmo Hill-Climbing?
3. Terminología del Algoritmo Hill-Climbing
4. Ventajas y desventajas del Algoritmo Hill-Climbing
5. Tipos de Algoritmos Hill-Climbing
6. Ejemplos de aplicación

Preguntas

1. Búsqueda local y optimización

Algunos problemas requieren encontrar un resultado que maximice o minimice un estado o valor.

Lo que hace una búsqueda local óptima es encontrar los extremos (mínimo o máximo) de una región especifica (local)

- La **optimización** se refiere a encontrar el conjunto de entradas (path o ruta) a una función objetivo que da como resultado la salida máxima o mínima de la función objetivo.

- La **búsqueda local o los algoritmos de búsqueda local** funcionan con un solo estado: “el actual”
  - Usan poca memoria
  - Pueden encontrar soluciones razonables en espacios de estados grandes

1. Búsqueda local y optimización

Características de los Algoritmos de búsqueda local

- Se inicia con una configuración inicial (generalmente aleatoria).
- A través de operadores se realizan cambios hasta alcanzar un estado desde el cual no se puede alcanzar ningún estado mejor.
- Por lo general **se encuentran óptimos locales.**
- Existe una **función heurística** que evalúa la calidad de la solución, mas no está necesariamente ligada a un costo.
- La **función heurística** se usará para podar el espacio de búsqueda (soluciones que no merecen la pena explorar).
- No se suele guardar historia del camino recorrido.
- La falta de memoria podría suponer problemas de bucles.

1. Búsqueda local y optimización

Algoritmos de búsqueda local vs. Algoritmos de búsqueda global (completa)

¿Si existen Algoritmos de búsqueda local también los hay de búsqueda global?

Algoritmos de búsqueda local

- **NO** encuentran siempre la solución correcta u óptima, si existe.
- Se utilizan cuando solo nos preocupamos por una solución, (no es importante el camino recorrido hacia una solución).
- Algunos algoritmos de búsqueda local son:
  - La escalada de colinas (Hill Climbing)
  - El recocido simulado (Simulated annealing)
  - La búsqueda tabú (Tabu search)

Algoritmos de búsqueda global (completa)

- **SI** encuentran siempre la solución correcta u óptima si es que existe, dado el tiempo suficiente.
- Como nos interesa la mejor solución, por ello se busca y recorre el camino necesario hacia ella.
- Algunos algoritmos de búsqueda global son:
  - Algoritmo Estrella o A * (A-star)
  - Algoritmo genético (Genetic Algorithm)
  - Optimización de Enjambre de partículas (Particle Swarm Optimization)

2. ¿Cómo funciona el Algoritmo Hill-Climbing?

Algoritmo de búsqueda Hill-Climbing (método de ascenso de colinas)

Objetivo: Encontrar buenas soluciones, pero **no la mejor solución**, puesto que **no es exhaustivo**.

Características

- **Informado:** Utiliza información del estado para elegir un nodo u otro y usa heurísticas (funciones o reglas) para encontrar la solución.
- **No exhaustivo:** No explora todo el espacio de estados. Como máximo, sólo encuentra una solución.
- **Es eficiente:** porque evita la exploración de una parte del espacio de estados.
- **Enfoque codicioso:** el algoritmo se mueve en la dirección de optimizar el costo, es decir, encontrar máximos / mínimos locales.
- **Sin retroceso:** no puede recordar el estado anterior del sistema, por lo que no es posible retroceder al estado anterior.
- **Mecanismo de retroalimentación:** la retroalimentación del cálculo anterior ayuda a decidir el siguiente curso de acción, es decir, si subir o bajar la pendiente.

2. ¿Cómo funciona el Algoritmo Hill-Climbing?

Hill-Climbing (también es conocido como el método de ascenso de colinas) es uno de los **algoritmos de búsqueda informado** que permite solo ir mejorando una solución.

- Es un algoritmo de búsqueda local que se mueve continuamente en la dirección de aumentar la elevación / valor para encontrar el pico de la montaña o la mejor solución al problema.
- Termina cuando alcanza un valor pico donde ningún vecino tiene un valor más alto.
- También se le llama búsqueda local codiciosa, ya que solo mira a su buen estado vecino inmediato y no más allá de eso.
- Un nodo del algoritmo de escalada tiene dos componentes que son el estado y el valor.
- **Hill-Climbing** se usa principalmente cuando se dispone de una buena heurística.
- En este algoritmo, no necesitamos mantener y manejar el árbol o gráfico de búsqueda, ya que solo mantiene un único estado actual.

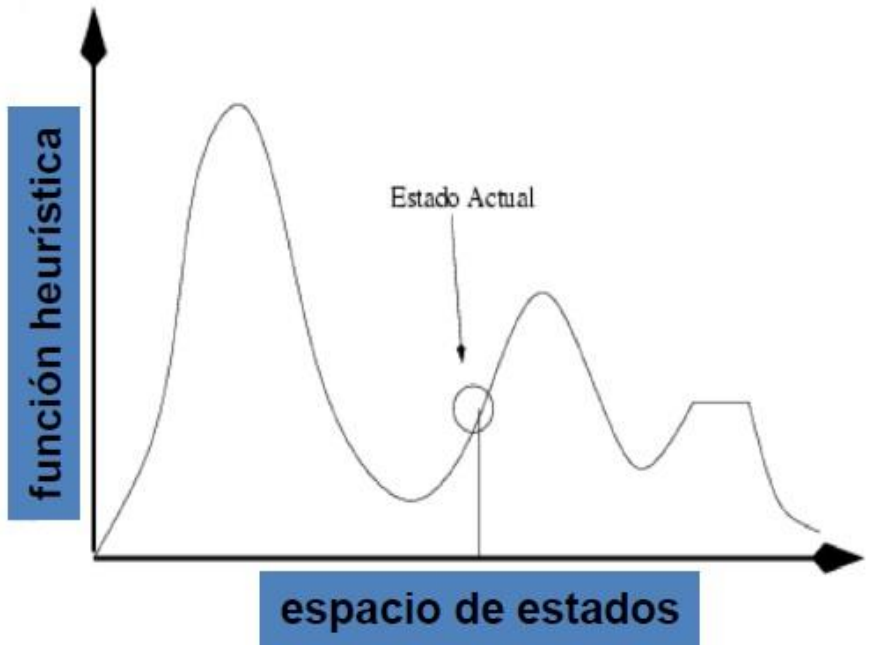
2. ¿Cómo funciona el Algoritmo Hill-Climbing?

Este algoritmo de búsqueda usa una técnica de **mejoramiento iterativo** que **permite solo ir mejorando una solución**.

PASO #1: Comienza a partir de un punto (punto actual) en el espacio de búsqueda.

PASO #2: Si el nuevo punto es mejor, se transforma en el punto actual, si no, otro punto vecino es seleccionado y evaluado.

PASO #3: El método termina cuando no hay mejorías, o cuando se alcanza un número predefinido de iteraciones (etapas).



3. Terminología del Algoritmo Hill-Climbing

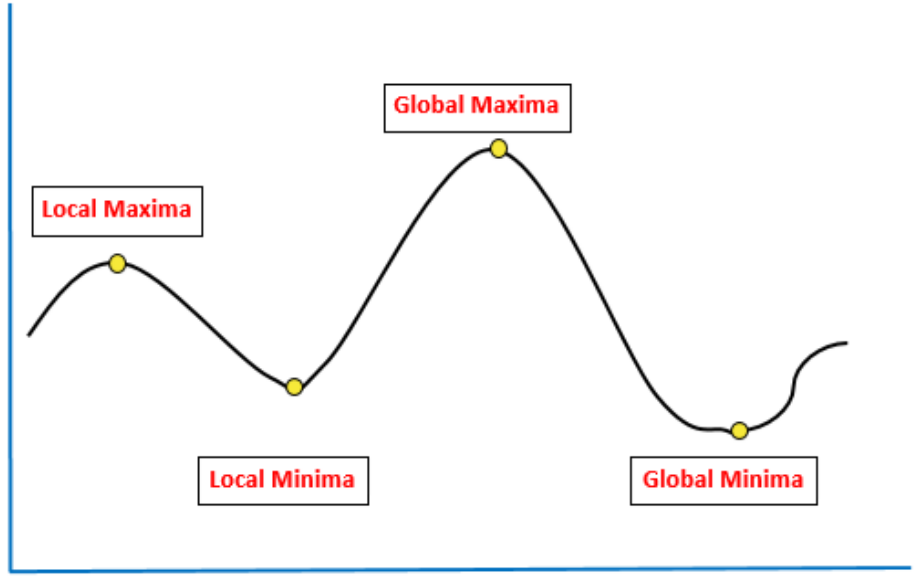
<u>Diagrama del espacio de estado:</u> escalada en la inteligencia artificial

Máximos / mínimos locales: Los mínimos locales es un estado que es mejor que su estado vecino, sin embargo, no es el mejor estado posible ya que existe un estado en el que el valor de la función objetivo es mayor.

Máximos / Mínimos Globales: Es el mejor estado posible en el diagrama de estados. Aquí el valor de la función objetivo es el más alto.

Estado actual: El estado actual es el estado en el que el agente está presente actualmente.

Máximo local plano: Esta región se representa mediante una línea recta en la que todos los estados vecinos tienen el mismo valor, por lo que cada nodo es el máximo local en la región.



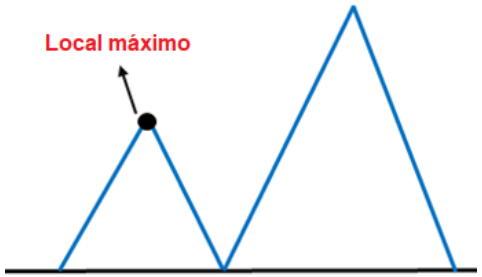
4. Ventajas y desventajas del Algoritmo Hill-Climbing

Los algoritmos de ascenso a colina **puede que nunca lleguen a encontrar una solución**, si son atrapados en estados que no son el objetivo, desde donde no se puede hallar mejores estados, por ejemplo:

1. Problema de Máximo local
2. Problema de Cresta (o risco)
3. Problema de Meseta

¿Cuáles son las **desventajas** que tiene el Algoritmo de Escalada?

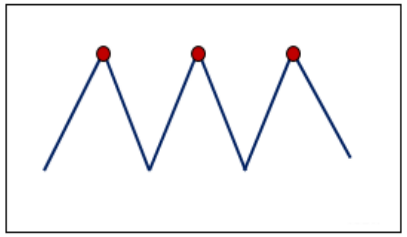
4. Ventajas y desventajas del Algoritmo Hill-Climbing



1. Máximo local

El algoritmo termina cuando el nodo actual es local máximo, ya que es mejor que sus vecinos. Sin embargo, existe un máximo global donde el valor de la función objetivo es mayor.

Solución: regresar a un estado anterior y explorar en una dirección diferente, ya que comienza a explorar rutas alternativas cuando se encuentra con el máximo local.



2. Cresta

La cresta o risco ocurre cuando hay múltiples picos y todos tienen el mismo valor o, en otras palabras, hay múltiples máximos locales que son iguales a los máximos globales.

Es parecido al máximo local, imposible de atravesar con movimientos simples.

Solución: el obstáculo de la cresta/risco se puede resolver moviéndose en varias direcciones al mismo tiempo (aplicar dos o mas reglas, antes de realizar una prueba del nuevo estado).

3. Meseta

La meseta es la región donde todos los nodos vecinos tienen el mismo valor de función objetivo, por lo que al algoritmo le resulta difícil seleccionar una dirección adecuada.

Solución: el problema de la meseta se puede resolver dando un gran salto desde el estado actual que lo llevará a la región que no es la meseta (y encontrar una nueva sección de estados).

4. Ventajas y desventajas del Algoritmo Hill-Climbing

¿Cuáles son las **ventajas** que tiene el Algoritmo de Escalada?

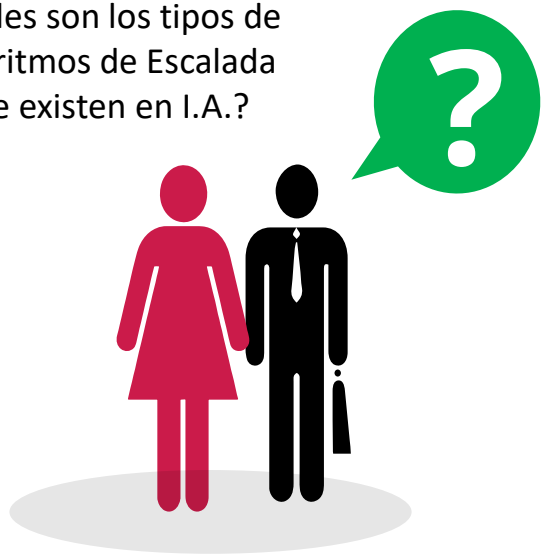
Hill Climbing es muy útil en problemas relacionados con:

- La optimización de problemas matemáticos. Uno de los ejemplos ampliamente discutidos del algoritmo de escalada de colinas es el Problema del vendedor ambulante en el que necesitamos minimizar la distancia recorrida por el vendedor.
- La resolución de problemas de optimización utilizando solo una potencia de cálculo limitada: Reduce el número de nodos a analizar.
- La programación de trabajos el diseño de chips.

En resumen, es útil para problemas relacionados con la optimización, porque brinda soluciones decentes a problemas computacionalmente desafiantes.

5. Tipos de algoritmos de escalada en I.A.

¿Cuáles son los tipos de Algoritmos de Escalada que existen en I.A.?



A continuación, analizamos los **tipos de algoritmo de escalada**:

1. Escalada simple
2. Escalada de la colina más empinada (por máxima pendiente)
3. Escalada estocástica

5. Tipos de algoritmos de escalada en I.A.

1. Escalada simple

- Es la forma más simple del algoritmo de escalada de colinas.
- Solo tiene en cuenta el nodo vecino para su funcionamiento.
- Si el nodo vecino es mejor que el nodo actual, establece el nodo vecino como nodo actual.
- El algoritmo comprueba solo un vecino a la vez.

2. Escalada por máxima pendiente

- Es una forma avanzada de algoritmo de escalada simple.
- Busca no solamente un estado mejor que el actual, sino el mejor de todos los estados posible.
- Se ejecuta a través de todos los nodos vecinos más cercanos y selecciona el nodo más cercano al estado objetivo.
- El algoritmo requiere más potencia de cálculo que el algoritmo simple de escalada de colinas, ya que busca en varios vecinos a la vez.

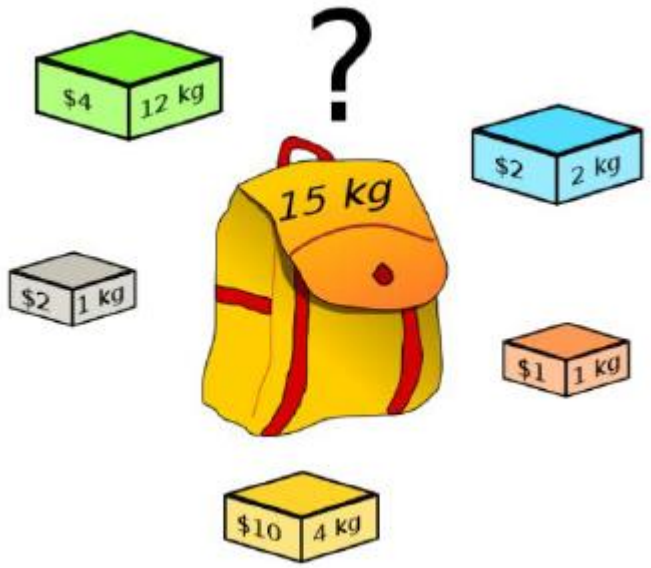
3. Escalada estocástica

- Este tipo de algoritmo de escalada no mira todos sus nodos vecinos para comprobar si es mejor que el nodo actual.
- Selecciona aleatoriamente un nodo vecino y, según los criterios predefinidos, decide si ir al nodo vecino o seleccionar un nodo alternativo.

Heurística = de proximidad

6. Ejemplos aplicando el Algoritmo Hill Climbing

EJEMPLO #1: El problema de la Mochila



Estado Inicial: mochila vacía

Estado Final: cualquier combinación de objetos en la mochila

Operadores: meter o sacar objetos de la mochila

Heurística: $max \sum_i Valor_i$ o $max \sum_i \frac{Valor_i}{Peso_i}$

6. Ejemplos aplicando el Algoritmo Hill Climbing

EJEMPLO #1: El problema de la Mochila

Capacidad de la mochila = 8kg

Artículos = 6

1kg - 2€
2kg - 5€
4kg - 6€
5kg - 10€
7kg - 13€
8kg - 16€

				GANANCIAS EN DIMENSION (Kg.)								
Etapa	**Artículo**	**Dimensión (Kg)**	**Ganancia (€)**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
0	**Estado inicial**			0	0	0	0	0	0	0	0	0
1	A	1	2									
2	B	2	5									
3	C	4	6									
4	D	5	10									
5	E	7	13									
6	F	8	16									

0 – 8 kg. (máximo)

Estado inicial: Mochila vacía

Etapa #0: El estado de la mochila esta vacía y la ganancia es 0.
Evaluamos 6 etapas porque solo hay 6 objetos (de A-F, existe solo 1 und. x cada objeto).
Los colocamos en la tabla en orden creciente en dimensión con su respectiva ganancia.

6. Ejemplos aplicando el Algoritmo Hill Climbing

EJEMPLO #1: El problema de la Mochila

Capacidad de la mochila = 8kg

Artículos = 6

1kg - 2€
2kg - 5€
4kg - 6€
5kg - 10€
7kg - 13€
8kg - 16€

				GANANCIAS EN DIMENSION (Kg.)								
Etapa	**Artículo**	**Dimensión (Kg)**	**Ganancia (€)**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
0	**Estado inicial**			0	0	0	0	0	0	0	0	0
1	A	1	2	0	2 A	2 A	2 A	2 A	2 A	2 A	2 A	2 A
2	B	2	5	0	2 A	5 B	7 A+B	7 A+B	7 A+B	7 A+B	7 A+B	7 A+B

0 – 8 kg. (máximo)

Estado inicial: Mochila vacía

Etapa #1: Introducimos el objeto A en el casillero de 1kg con la ganancia asociada igual a 2.
Como no hay otro objeto a introducir perteneciente a etapas previas, copiamos esta casilla en las restantes (de la casilla 2 a la de 8kg).

Etapa #2: Introducimos el objeto B en el casillero de 2kg con la ganancia asociada igual a 5, manteniendo los valores de las casilla 0-1 de la etapa #1.
En el siguiente casillero, verificamos si podemos introducir uno o mas objetos que totalicen 3kg => A + B cumplen la condición con una ganancia de 2+5 = 7.
Colocamos 7 en el casillero 3(kg) haciendo referencia a los objetos A+B y copiamos este valor al resto de casillas porque ya no hay mas objetos que introducir.

6. Ejemplos aplicando el Algoritmo Hill Climbing

EJEMPLO #1: El problema de la Mochila

Capacidad de la mochila = 8kg

Artículos = 6

1kg - 2€
2kg - 5€
4kg - 6€
5kg - 10€
7kg - 13€
8kg - 16€

				GANANCIAS EN DIMENSION (Kg.)								
Etapa	**Artículo**	**Dimensión (Kg)**	**Ganancia (€)**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
0	**Estado inicial**			0	0	0	0	0	0	0	0	0
1	A	1	2	0	2 A	2 A	2 A	2 A	2 A	2 A	2 A	2 A
2	B	2	5	0	2 A	5 B	7 A+B	7 A+B	7 A+B	7 A+B	7 A+B	7 A+B
3	C	4	6	0	2 A	5 B	7 A+B	7 A+B	8 A+C	11 B+C	13 A+B+C	13 A+B+C

0 – 8 kg. (máximo)

Estado inicial: Mochila vacía

Etapa #3: No introducimos el objeto C en el casillero de 4kg con la ganancia asociada igual a 6 porque la ganancia de la etapa #2 para el casillero 4 fue de 7 (mayor a 6, por tanto, lo dejamos en 7).

Evaluamos que otros objetos podemos introducir en los siguientes casilleros mayores a 4kg:

- En la casilla de 5kg podemos colocar los objetos A y C que totalizan una ganancia 8 (mayor a 7 de la etapa anterior)
- En la casilla de 6kg podemos colocar los objetos B y C que totalizan una ganancia de 11 (mayor a 7 de la etapa anterior)
- En la casilla de 7kg podemos colocar los objetos A, B y C que totalizan una ganancia de 13 (mayor a 7 de la etapa anterior)
- En la casilla de 8kg ya no podemos colocar mas objetos, por tanto copiamos el resultado de la casilla de 7kg.

6. Ejemplos aplicando el Algoritmo Hill Climbing

EJEMPLO #1: El problema de la Mochila

Capacidad de la mochila = 8kg

Artículos = 6

1kg - 2€
2kg - 5€
4kg - 6€
5kg - 10€
7kg - 13€
8kg - 16€

				GANANCIAS EN DIMENSION (Kg.)								
Etapa	**Artículo**	**Dimensión (Kg)**	**Ganancia (€)**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
0	**Estado inicial**			0	0	0	0	0	0	0	0	0
1	A	1	2	0	2 A	2 A	2 A	2 A	2 A	2 A	2 A	2 A
2	B	2	5	0	2 A	5 B	7 A+B	7 A+B	7 A+B	7 A+B	7 A+B	7 A+B
3	C	4	6	0	2 A	5 B	7 A+B	7 A+B	8 A+C	11 B+C	13 A+B+C	13 A+B+C
4	D	5	10	0	2 A	5 B	7 A+B	7 A+B	10 D	12 A+D	15 B+D	17 A+B+D

0 – 8 kg. (máximo)

Estado inicial: Mochila vacía

Etapa #4: Introducimos el objeto D en el casillero de 5kg con la ganancia asociada igual a 10, manteniendo los valores de las casilla 0-4 de la etapa #3.

Evaluamos que otros objetos podemos introducir en los siguientes casilleros mayores a 5kg:

- En la casilla de 6kg podemos colocar los objetos A y D que totalizan una ganancia 12 (mayor a 11 de la etapa anterior)
- En la casilla de 7kg podemos colocar los objetos B y D que totalizan una ganancia de 15 (mayor a 13 de la etapa anterior)
- En la casilla de 8kg podemos colocar los objetos A, B y D que totalizan una ganancia de 17 (mayor a 13 de la etapa anterior)

6. Ejemplos aplicando el Algoritmo Hill Climbing

EJEMPLO #1: El problema de la Mochila

Capacidad de la mochila = 8kg

Artículos = 6

1kg - 2€
2kg - 5€
4kg - 6€
5kg - 10€
7kg - 13€
8kg - 16€

				GANANCIAS EN DIMENSION (Kg.)								
Etapa	**Artículo**	**Dimensión (Kg)**	**Ganancia (€)**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
0	**Estado inicial**			0	0	0	0	0	0	0	0	0
1	A	1	2	0	2 A	2 A	2 A	2 A	2 A	2 A	2 A	2 A
2	B	2	5	0	2 A	5 B	7 A+B	7 A+B	7 A+B	7 A+B	7 A+B	7 A+B
3	C	4	6	0	2 A	5 B	7 A+B	7 A+B	8 A+C	11 B+C	13 A+B+C	13 A+B+C
4	D	5	10	0	2 A	5 B	7 A+B	7 A+B	10 D	12 A+D	15 B+D	17 A+B+D
5	E	7	13	0	2 A	5 B	7 A+B	7 A+B	10 D	12 A+D	15 B+D	17 A+B+D

0 – 8 kg. (máximo)

Estado inicial: Mochila vacía

Etapa #5: No introducimos el objeto E en el casillero de 7kg con la ganancia asociada igual a 13 porque la ganancia de la etapa #4 para el casillero 7 fue de 15 (mayor a 13, por tanto, lo dejamos en 15).
Evaluamos que otros objetos podemos introducir en los siguientes casilleros mayores a 7kg:

- En la casilla de 8kg podemos colocar los objetos A y E que totalizan una ganancia de 15 (menor a 17 de la etapa anterior). Lo dejamos con 17.

6. Ejemplos aplicando el Algoritmo Hill Climbing

EJEMPLO #1: El problema de la Mochila

Capacidad de la mochila = 8kg

Artículos = 6

1kg - 2€
2kg - 5€
4kg - 6€
5kg - 10€
7kg - 13€
8kg - 16€

				GANANCIAS EN DIMENSION (Kg.)								
Etapa	**Artículo**	**Dimensión (Kg)**	**Ganancia (€)**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
0	**Estado inicial**			0	0	0	0	0	0	0	0	0
1	A	1	2	0	2 A	2 A	2 A	2 A	2 A	2 A	2 A	2 A
2	B	2	5	0	2 A	5 B	7 A+B	7 A+B	7 A+B	7 A+B	7 A+B	7 A+B
3	C	4	6	0	2 A	5 B	7 A+B	7 A+B	8 A+C	11 B+C	13 A+B+C	13 A+B+C
4	D	5	10	0	2 A	5 B	7 A+B	7 A+B	10 D	12 A+D	15 B+D	17 A+B+D
5	E	7	13	0	2 A	5 B	7 A+B	7 A+B	10 D	12 A+D	15 B+D	17 A+B+D
6	F	8	16	0	2 A	5 B	7 A+B	7 A+B	10 D	12 A+D	15 B+D	17 A+B+D

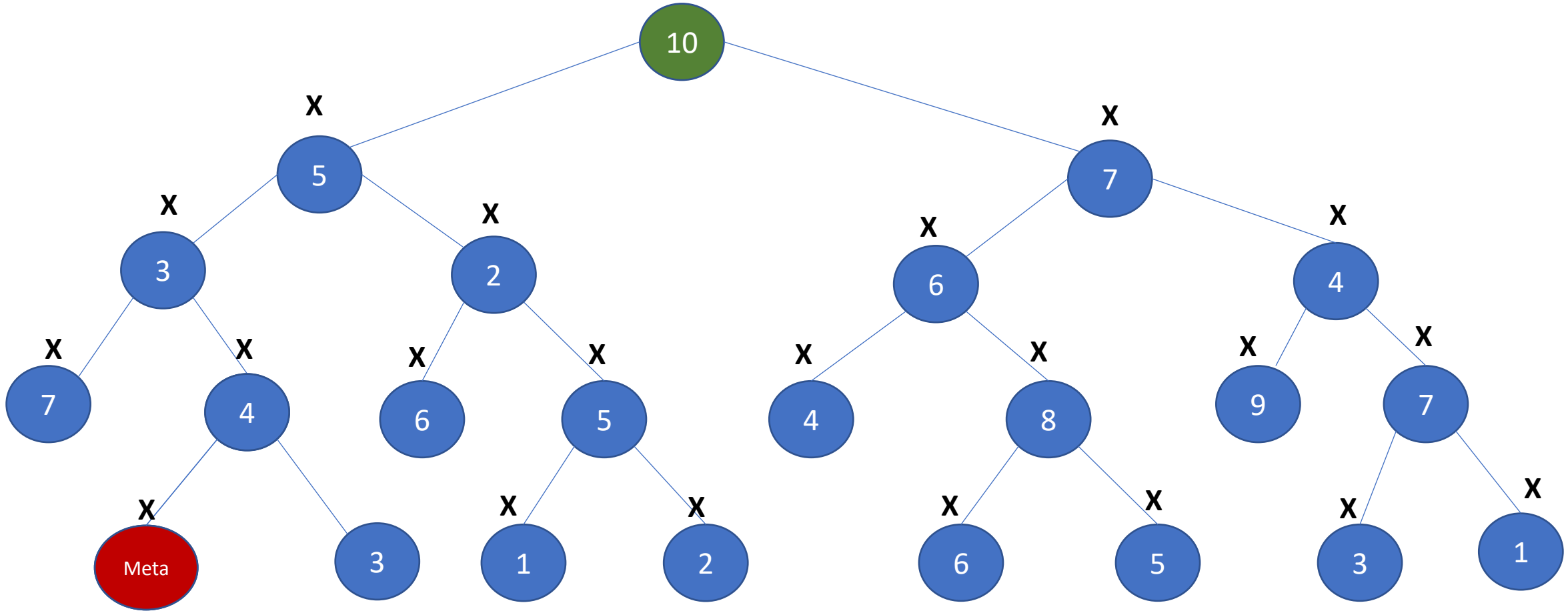
0 – 8 kg. (máximo)

Estado inicial: Mochila vacía

Etapa #6: No introducimos el objeto F en el casillero de 8kg con la ganancia asociada igual a 16 porque la ganancia de la etapa #5 para el casillero 8 fue de 17 (mayor a 16, por tanto, lo dejamos en 17).
Hemos llegado al **Estado final = Mochila llena = 8kg => Maximizando las ganancias a 17 €**

6. Ejemplos de aplicación

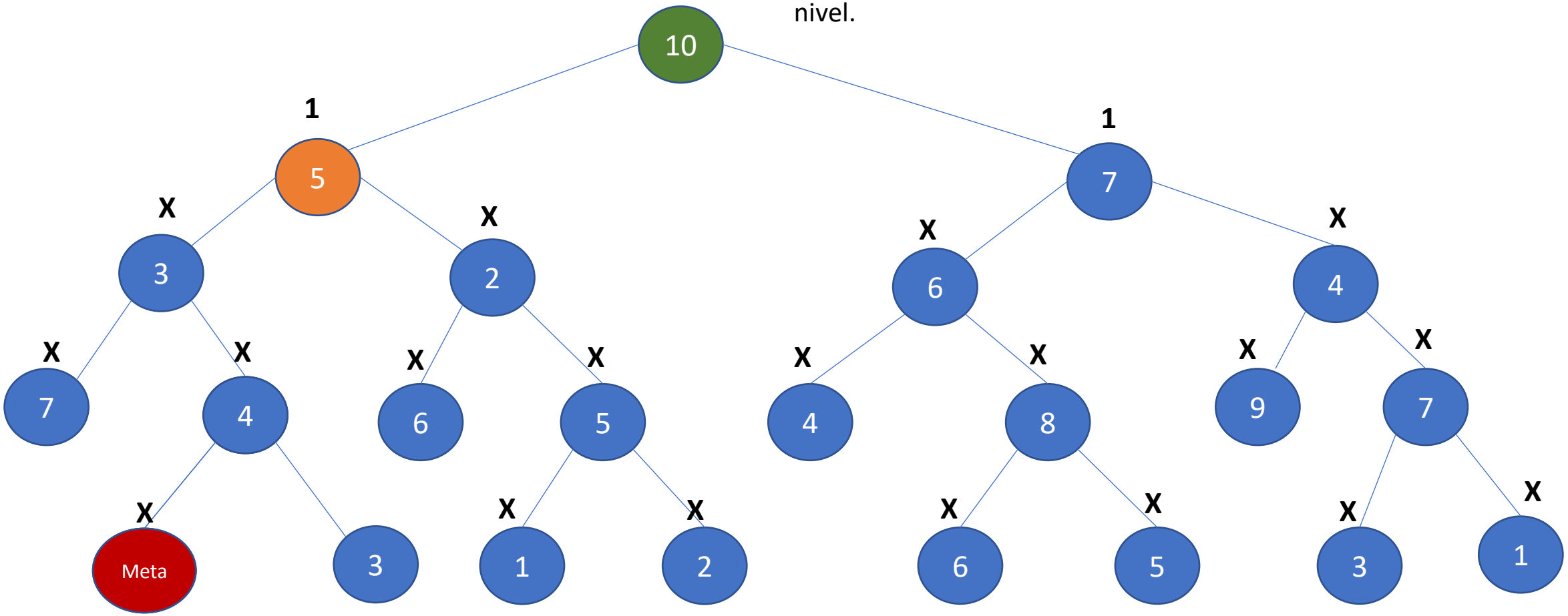
EJEMPLO #2: Aplicado al recorrido de un árbol (es una representación del espacio de Estados o Nodos)



6. Ejemplos de aplicación

EJEMPLO #2: Aplicado al recorrido de un árbol.

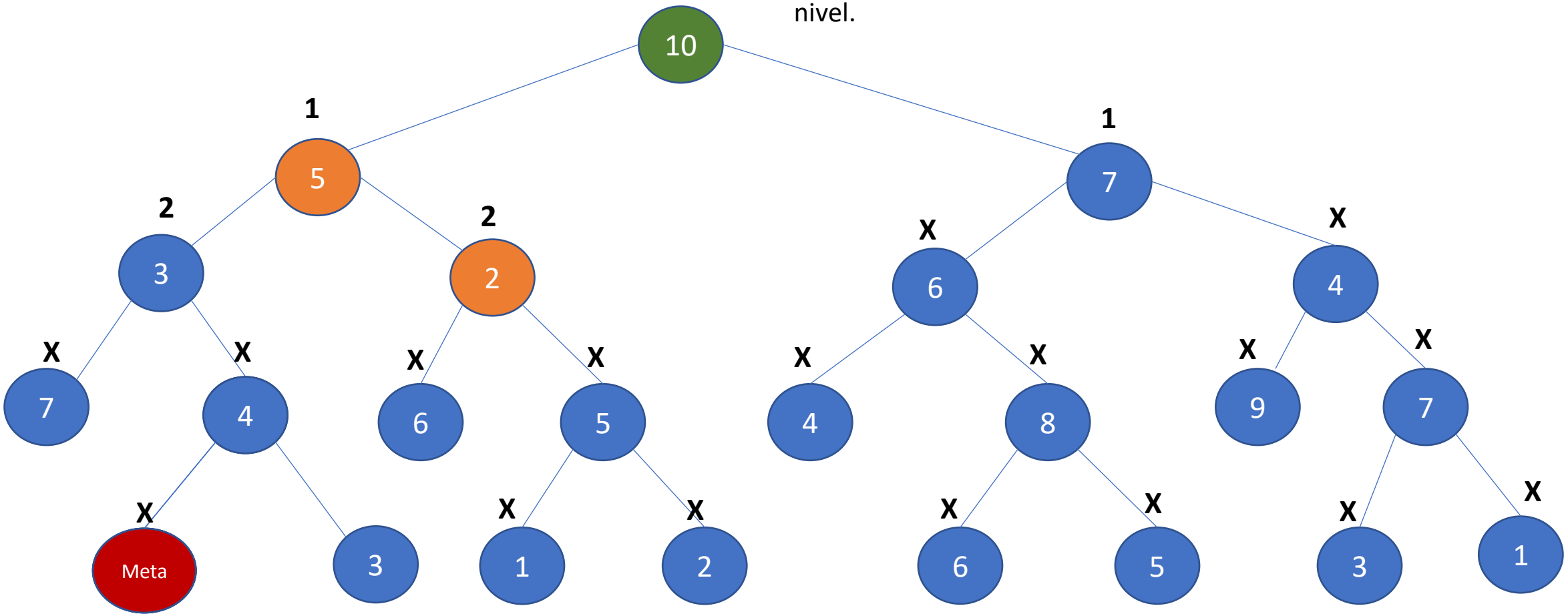
PASO #1: Desde el nodo inicial #10 elegimos el # de nodo con menor valor (7 > 5 => elegimos 5 y asignamos 1 a este nivel.



6. Ejemplos de aplicación

EJEMPLO #2: Aplicado al recorrido de un árbol.

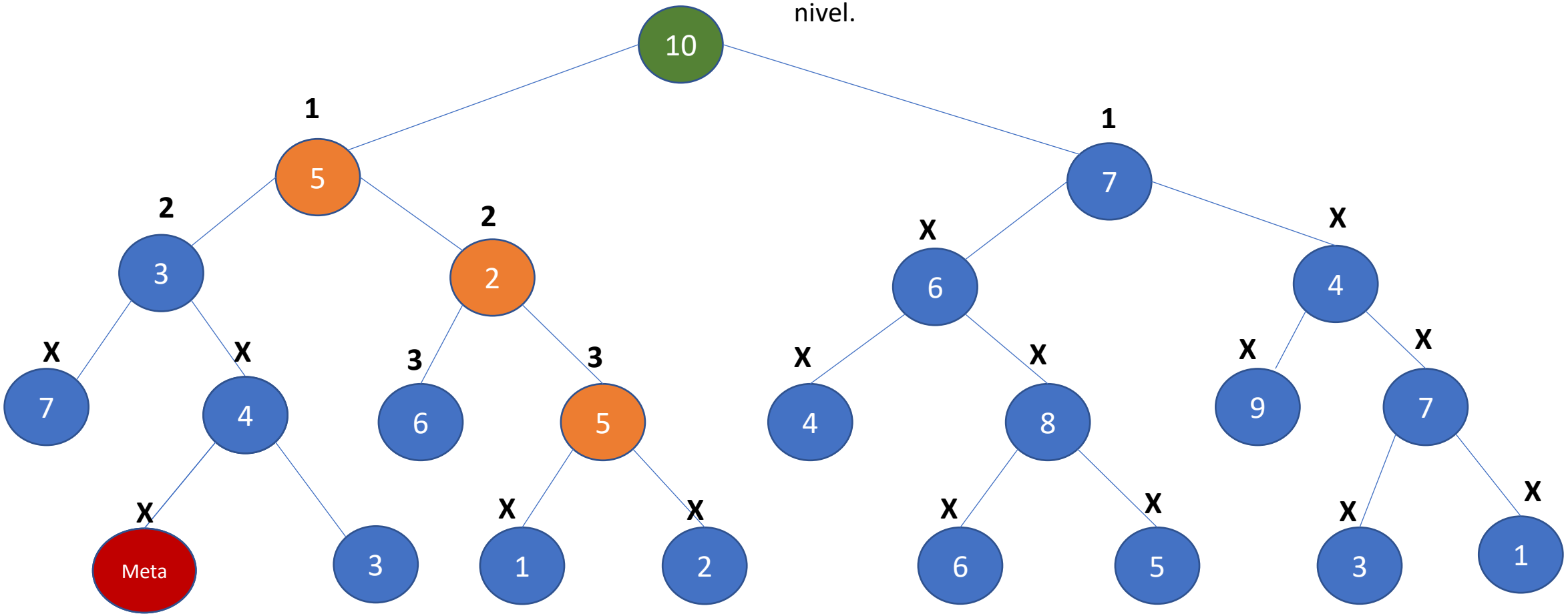
PASO #2: el nodo #5 tiene 2 hijos, elegimos el # de nodo con menor valor (3 > 2 => elegimos 2 y asignamos 2 a este nivel.



6. Ejemplos de aplicación

EJEMPLO #2: Aplicado al recorrido de un árbol.

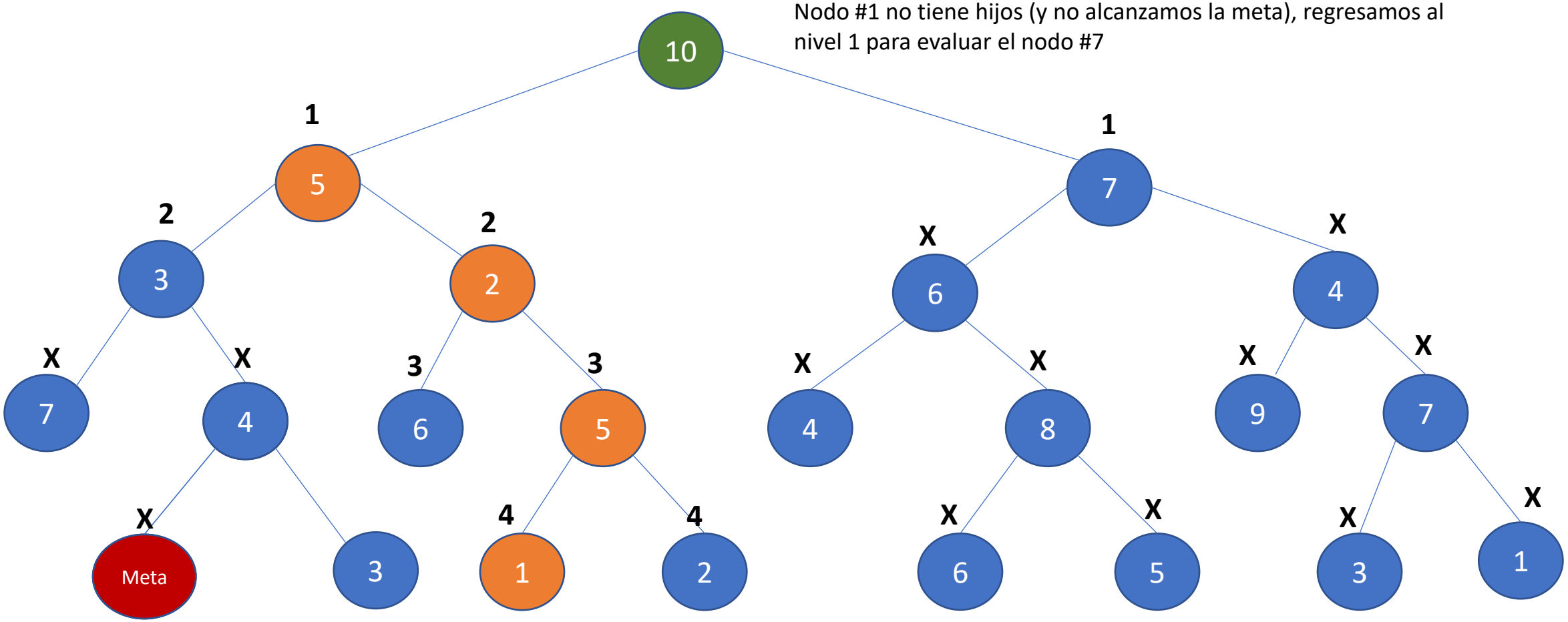
PASO #3: el nodo #2 tiene 2 hijos, elegimos el # de nodo con menor valor (6 > 5 => elegimos 5 y asignamos 3 a este nivel.



6. Ejemplos de aplicación

EJEMPLO #2: Aplicado al recorrido de un árbol.

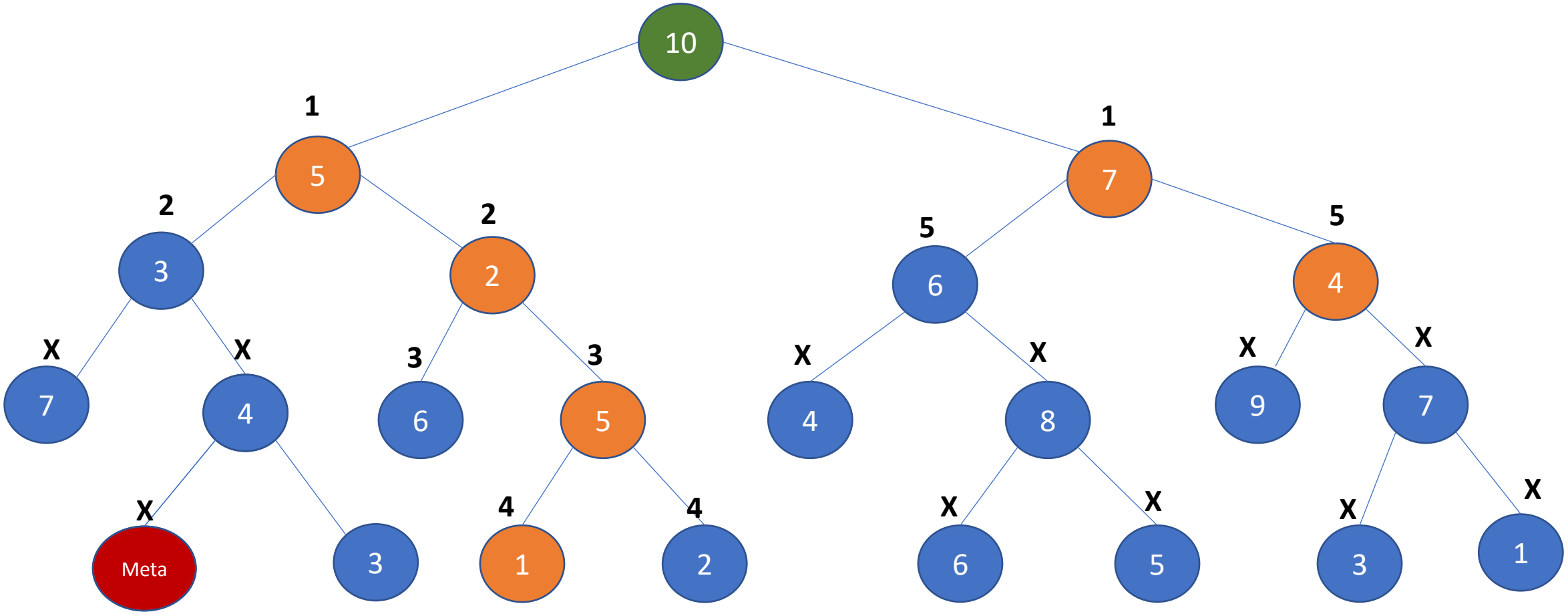
PASO #4: el nodo #5 tiene 2 hijos, elegimos el # de nodo con menor valor (2 > 1 => elegimos 1 y asignamos 4 a este nivel. Como el Nodo #1 no tiene hijos (y no alcanzamos la meta), regresamos al nivel 1 para evaluar el nodo #7



6. Ejemplos de aplicación

EJEMPLO #2: Aplicado al recorrido de un árbol.

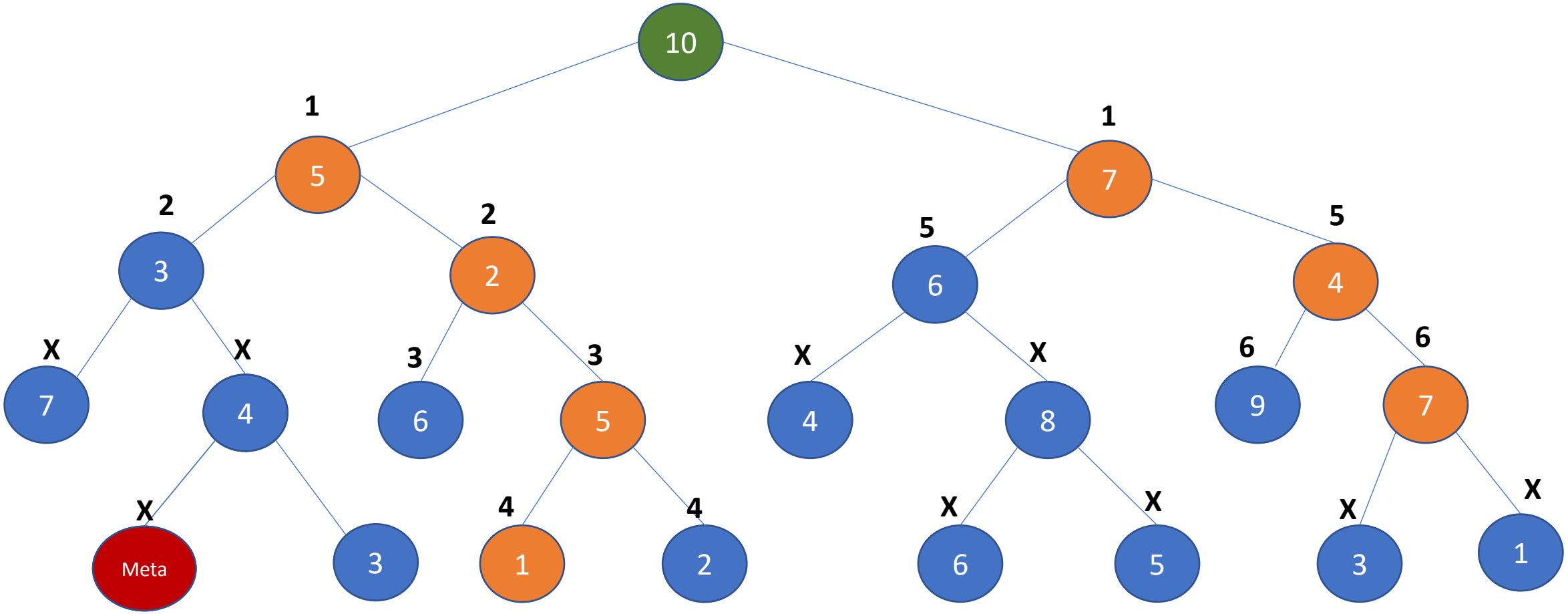
PASO #5: el nodo #7 tiene 2 hijos, elegimos el # de nodo con menor valor (6 > 4 => elegimos 4 y asignamos 5 a este nivel.



6. Ejemplos de aplicación

EJEMPLO #2: Aplicado al recorrido de un árbol.

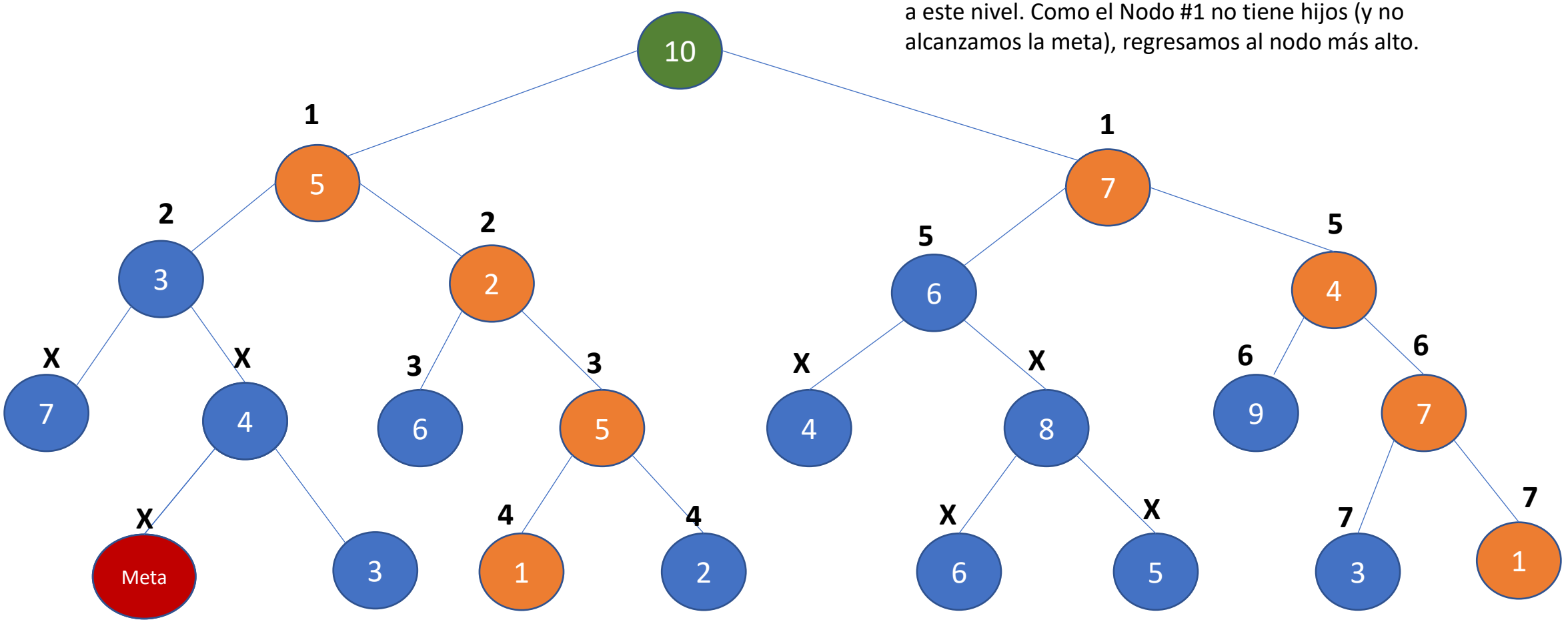
PASO #6: el nodo #4 tiene 2 hijos, elegimos el # de nodo con menor valor (9 > 7 => elegimos 7 y asignamos 6 a este nivel.



6. Ejemplos de aplicación

EJEMPLO #2: Aplicado al recorrido de un árbol.

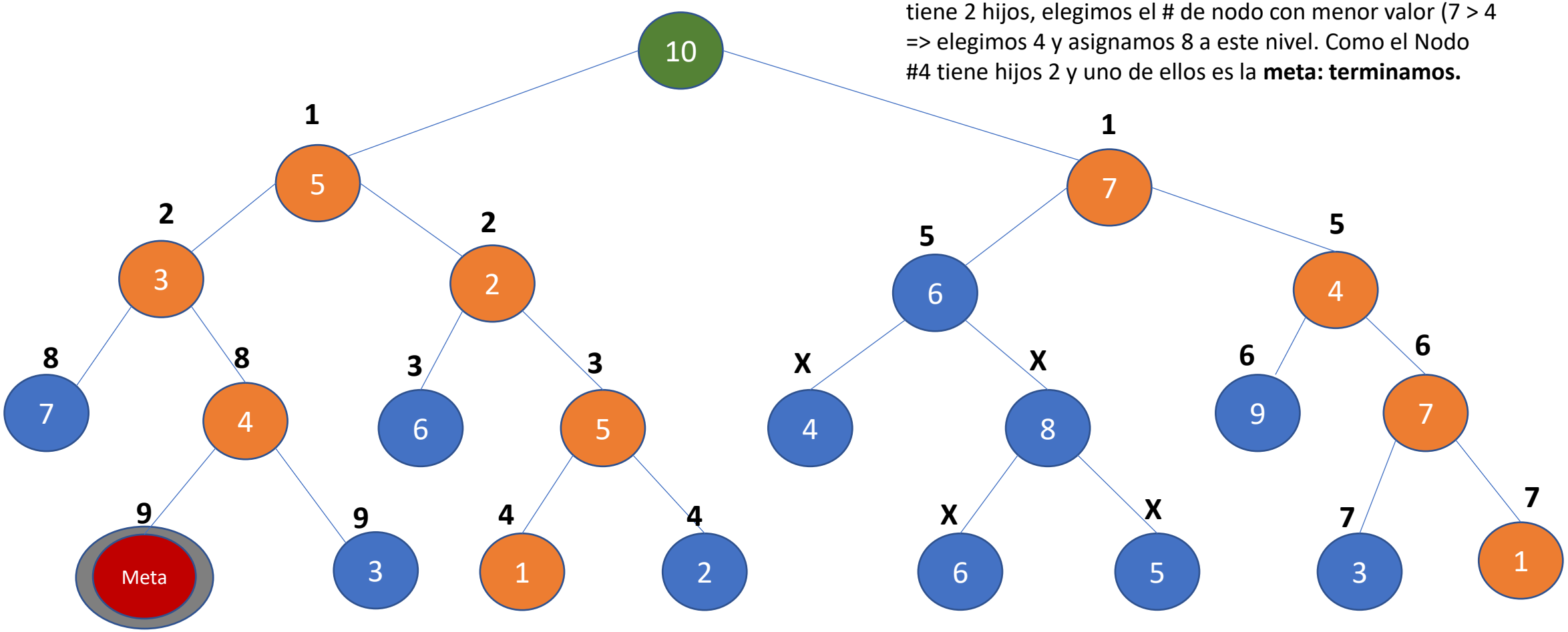
PASO #7: el nodo #7 tiene 2 hijos, elegimos el # de nodo con menor valor (3 > 1 => elegimos 1 y asignamos 7 a este nivel. Como el Nodo #1 no tiene hijos (y no alcanzamos la meta), regresamos al nodo más alto.



6. Ejemplos de aplicación

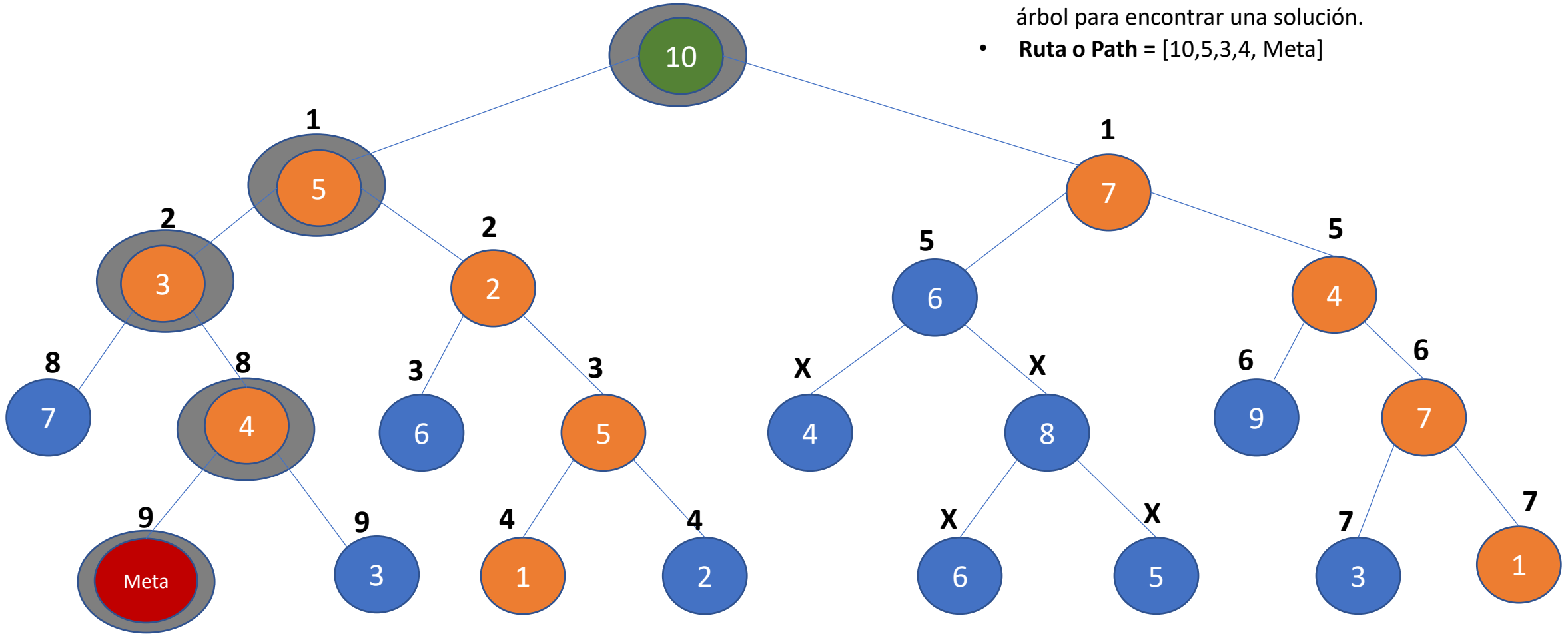
EJEMPLO #2: Aplicado al recorrido de un árbol.

PASO #8: el nivel mas alto es el 1 pero ya fue evaluado, el siguiente nivel, el 2, queda por evaluar el nodo #3 que tiene 2 hijos, elegimos el # de nodo con menor valor (7 > 4 => elegimos 4 y asignamos 8 a este nivel. Como el Nodo #4 tiene hijos 2 y uno de ellos es la **meta: terminamos.**



6. Ejemplos de aplicación

EJEMPLO #2: Aplicado al recorrido de un árbol.



Conclusión:

- Observamos que no tuvimos que recorrer todo el árbol para encontrar una solución.
- **Ruta o Path =** [10,5,3,4, Meta]

PREGUNTAS