Programación Avanzada y Estructuras de Datos

5. Árboles

Víctor M. Sánchez Cartagena

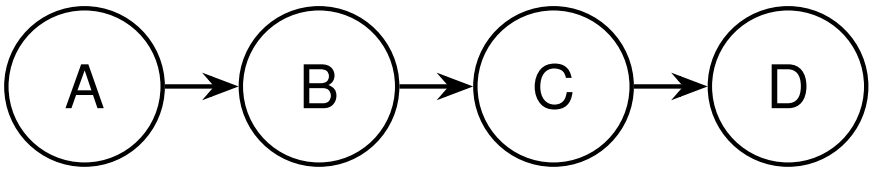
Grado en Ingeniería en Inteligencia Artificial
Dep. Lenguajes y Sistemas Informáticos
Universidad de Alicante

12 de noviembre de 2024

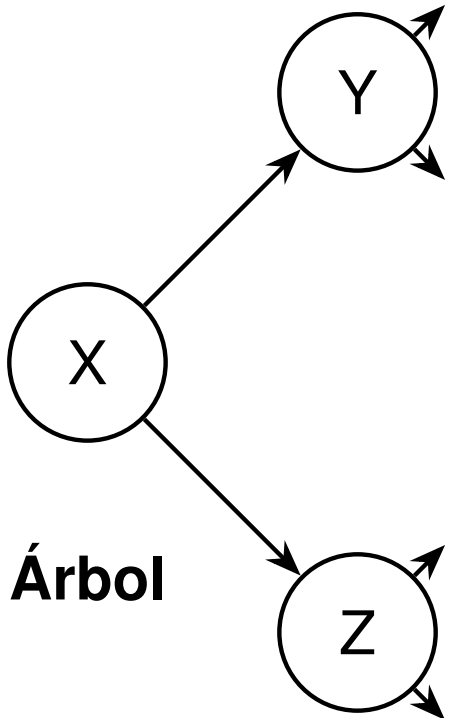
Índice

Idea intuitiva de árbol

- Para solucionar determinados problemas, es necesario organizar los datos de manera **no lineal**
- Tipos lineales (vectores, listas, pilas, colas): cada elemento tiene un sucesor y un predecesor
- Árboles: cada elemento tiene un predecesor, pero puede tener más de un sucesor

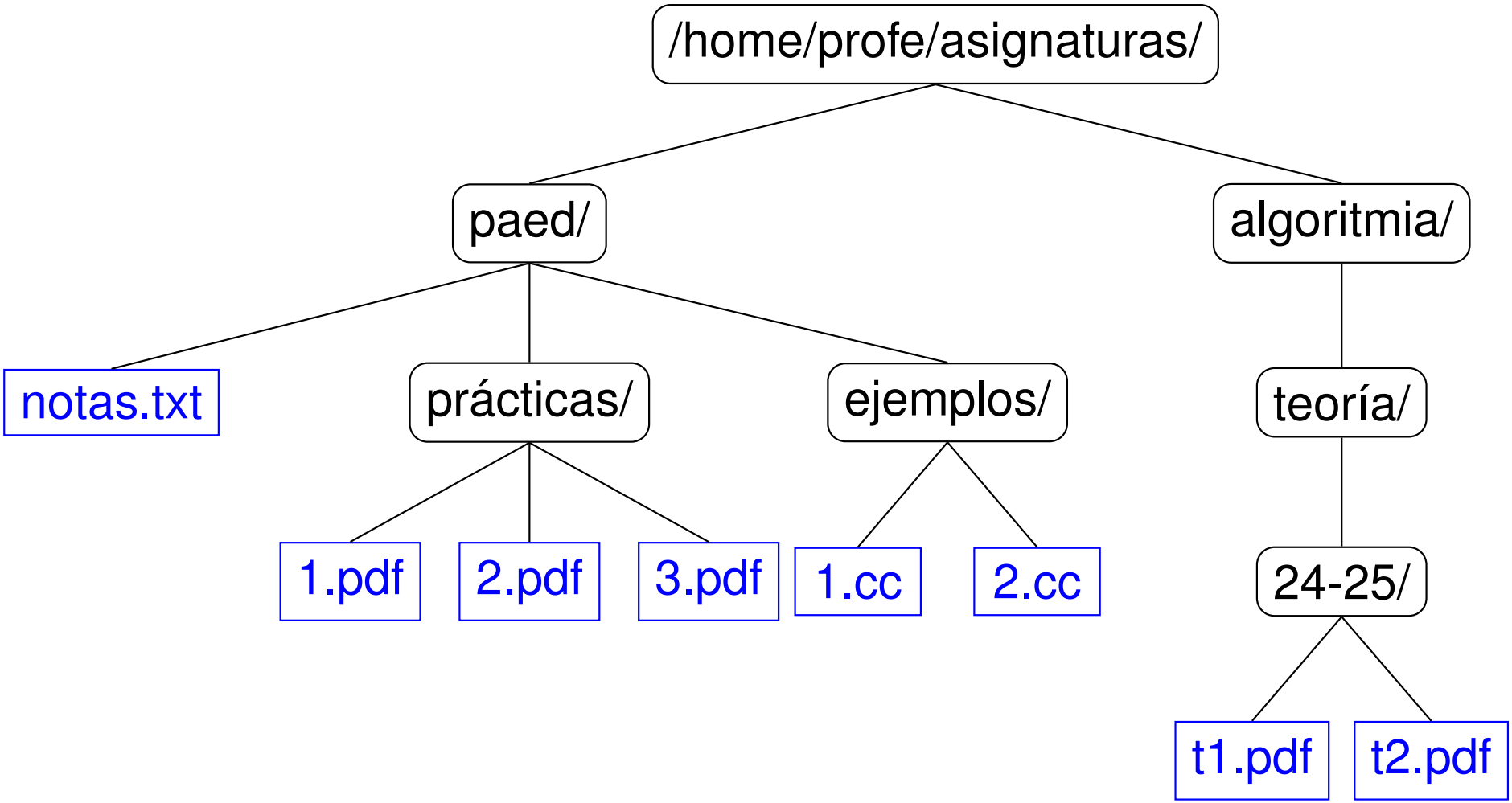


Tipo lineal



Árbol

Idea intuitiva de árbol



Definiciones

- El nodo `paed/` es el **padre** del nodo `notas.txt`
- Los **hijos** del nodo `paed/` son `notas.txt`, `prácticas/` y `ejemplos/`.
- El nodo `/home/profe/asignaturas/` es la **raíz** del árbol

Definición

Un árbol A es un conjunto de nodos que almacenan elementos y mantienen relaciones padre-hijo cumpliendo las siguientes propiedades:

- Si A no es vacío, contiene un único nodo especial, denominado **raíz**, que no tiene padre
- Cada nodo v de A que no es la raíz tiene un único **nodo padre** w; cada nodo cuyo padre es w es un **nodo hijo** de w

Definiciones

- Dos nodos con el mismo padre son **hermanos**
- Un nodo es **hoja** o **externo** si no tiene hijos
- Un nodo es **interno** si tiene uno o más hijos
- El **grado** de un nodo es el número de hijos que tiene
- El **grado** de un árbol es el máximo grado permitido para sus nodos

Definiciones

- Dos nodos con el mismo padre son **hermanos**
- Un nodo es **hoja** o **externo** si no tiene hijos
- Un nodo es **interno** si tiene uno o más hijos
- El **grado** de un nodo es el número de hijos que tiene
- El **grado** de un árbol es el máximo grado permitido para sus nodos

Preguntas

¿Cuántos nodos hoja tiene el árbol anterior?¿Cuántos nodos internos?¿Cuál es el grado del nodo `paed/`?¿Cuál es el grado del árbol?

Definiciones

- Un **camino** es una secuencia de uno o más nodos $n_1, n_2, ..., n_s$ tal que $\forall i \in 1..s-1$, n_i es padre de n_{i+1}
- La **longitud** de un camino es su número de nodos menos uno: existe un camino de longitud 0 de cada nodo hasta sí mismo

Definiciones

- Un **camino** es una secuencia de uno o más nodos $n_1, n_2, ..., n_s$ tal que $\forall i \in 1..s-1$, n_i es padre de n_{i+1}
- La **longitud** de un camino es su número de nodos menos uno: existe un camino de longitud 0 de cada nodo hasta sí mismo

Preguntas

- ¿Existe un camino de `paed/` a `24-25`? En caso afirmativo, enumera sus nodos e indica su longitud
- ¿Existe un camino de `algoritmia/` a `t1.pdf`? En caso afirmativo, enumera sus nodos e indica su longitud

Definiciones

- Un nodo w es **descendiente** de otro nodo v (y v es **ascendente** de w) si existe un camino $v, ..., w$
- Los descendientes de un nodo v, excluyendo al mismo v, se denominan **descendientes propios**
- La misma definición se aplica para los **ascendentes propios**

Definiciones

- Un nodo *w* es **descendiente** de otro nodo *v* (y *v* es **ascendente** de *w*) si existe un camino *v*, ..., *w*
- Los descendientes de un nodo *v*, excluyendo al mismo *v*, se denominan **descendientes propios**
- La misma definición se aplica para los **ascendentes propios**

Preguntas

- ¿Cuáles son los descendientes del nodo `teoría/`? ¿Cuáles de ellos son propios?
- ¿Cuáles son los ascendentes del nodo `teoría/`? ¿Cuáles de ellos son propios?

Definiciones

- La raíz de un árbol está en el nivel 1
- Los hijos de un nodo que está en el nivel i están en el nivel $i + 1$
- Dicho de otro modo: el nivel de un nodo es el resultado de sumarle 1 a la longitud del camino desde la raíz hasta ese nodo
- La **altura** de un árbol es el máximo nivel de sus nodos

Definiciones

- La raíz de un árbol está en el nivel 1
- Los hijos de un nodo que está en el nivel i están en el nivel $i + 1$
- Dicho de otro modo: el nivel de un nodo es el resultado de sumarle 1 a la longitud del camino desde la raíz hasta ese nodo
- La **altura** de un árbol es el máximo nivel de sus nodos

Preguntas

- ¿Cuál es el nivel de cada uno de estos nodos?: `/home/profe/asignaturas/`, `notas.txt`, `ejemplos/`, `24-25/`, `t1.pdf`
- ¿Cuál es la altura del árbol de ejemplo?

Especificación del TAD árbol

Operaciones:

- Devuelve el número de nodos en el árbol

```
int size() const;
```

- Comprueba si el árbol está vacío

```
bool empty() const;
```

- Devuelve un iterador que apunta a la raíz del árbol

```
Iterador root() const;
```

Especificación del TAD árbol

Operaciones:

- Asigna el valor `e` al nodo del árbol que ocupa la posición a la que apunta el iterador `it`.Devuelve `false` si el iterador no apuntaba a ninguna posición ocupada por un elemento y `true` en caso contrario

```
bool set(Iterador it, const Elem & e);
```

- Obtiene el elemento que ocupa la posición de la lista a la que apunta el iterador `it`

```
Elem get(Iterador it) const;
```

Especificación del TAD Iterador

Operaciones:

- Devuelve un iterador al nodo padre del nodo actual

```
Iterador parent() const;
```

- Devuelve un vector de iteradores a los nodos hijos del nodo actual

```
vector<Iterador> children() const;
```

- Comprueba si el nodo actual es la raíz

```
bool isRoot() const;
```

Pregunta

¿Qué imprime el siguiente fragmento de código? Asume que la variable `a` es de tipo `Arbol` y representa el árbol de ejemplo empleado anteriormente

```
Arbol a;
//....
cout << a.size() << endl;
Iterador it=a.root().children()[0].children()[2].children()[1];
cout << a.get(it) << endl;
```

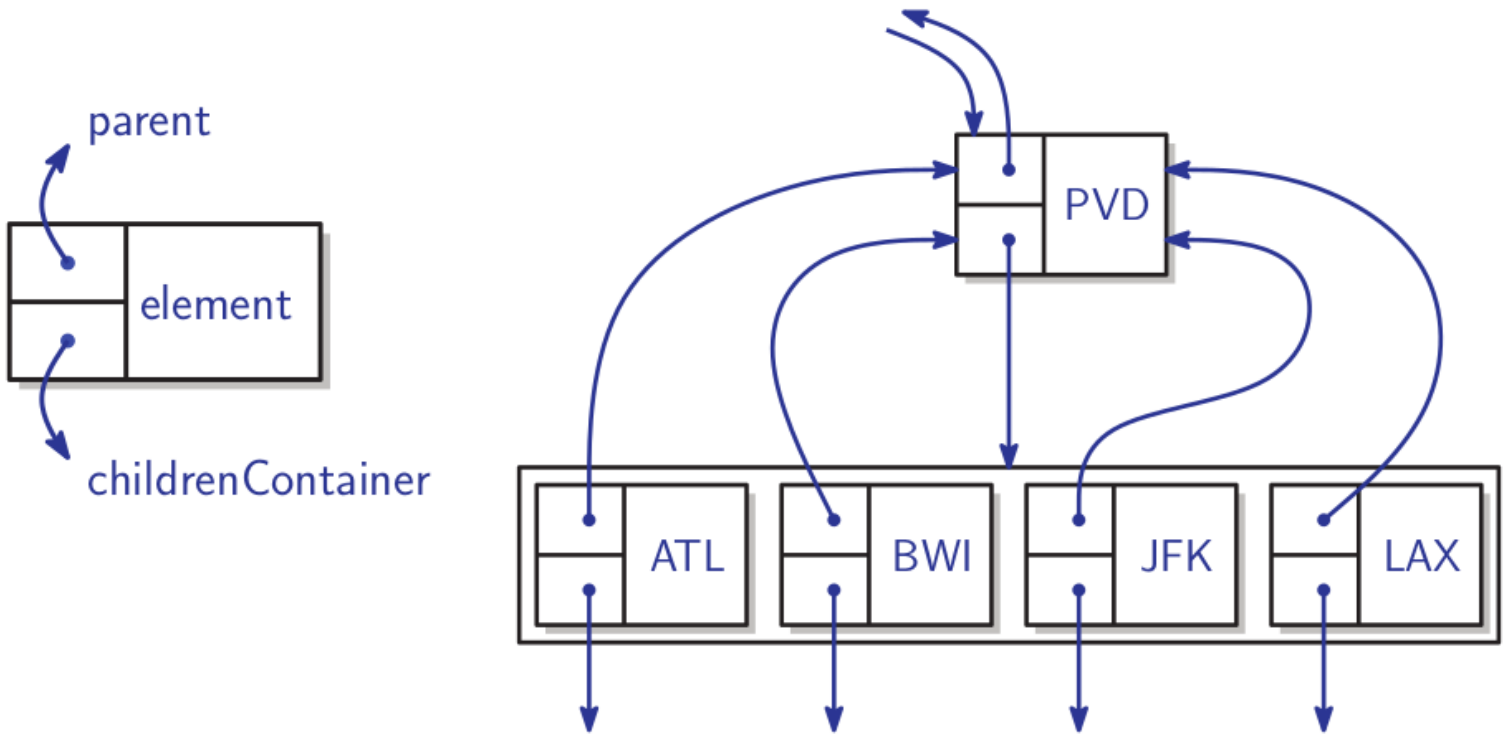
Pregunta

¿Qué imprime el siguiente fragmento de código? Asume que la variable `a` es de tipo `Arbol` y representa el árbol de ejemplo empleado anteriormente

```
Arbol a;
//....
cout << a.size() << endl;
Iterador it=a.root().children()[0].children()[2].children()[1];
cout << a.get(it) << endl;
```

```
15
2.cc
```

Implementación enlazada para árboles generales



fuente: Goodrich, M. T., Tamassia, R., & Mount, D. M. (2011). Data structures and algorithms in C++. John Wiley & Sons.

Implementación enlazada para árboles generales

```
class Arbol{
private:
Nodo* root;
....
};

class Nodo{
private:
   Elem data;
   Nodo* parent;
   vector<Nodo>* childrenContainer;
...
};
```

Implementación enlazada para árboles generales

- Complejidad asintótica respecto al número de nodos en el árbol (`size()`)

Operación	Coste
`root()`	
`empty()`	
`set()`	
`get()`	
`parent()`	
`isRoot()`	
`children()`	

$g =$ grado del árbol

Implementación enlazada para árboles generales

- Complejidad asintótica respecto al número de nodos en el árbol (`size()`)

Operación	Coste
`root()`	$\Theta(1)$
`empty()`	$\Theta(1)$
`set()`	$\Theta(1)$
`get()`	$\Theta(1)$
`parent()`	$\Theta(1)$
`isRoot()`	$\Theta(1)$
`children()`	$\Omega(1)$, $O(g)$

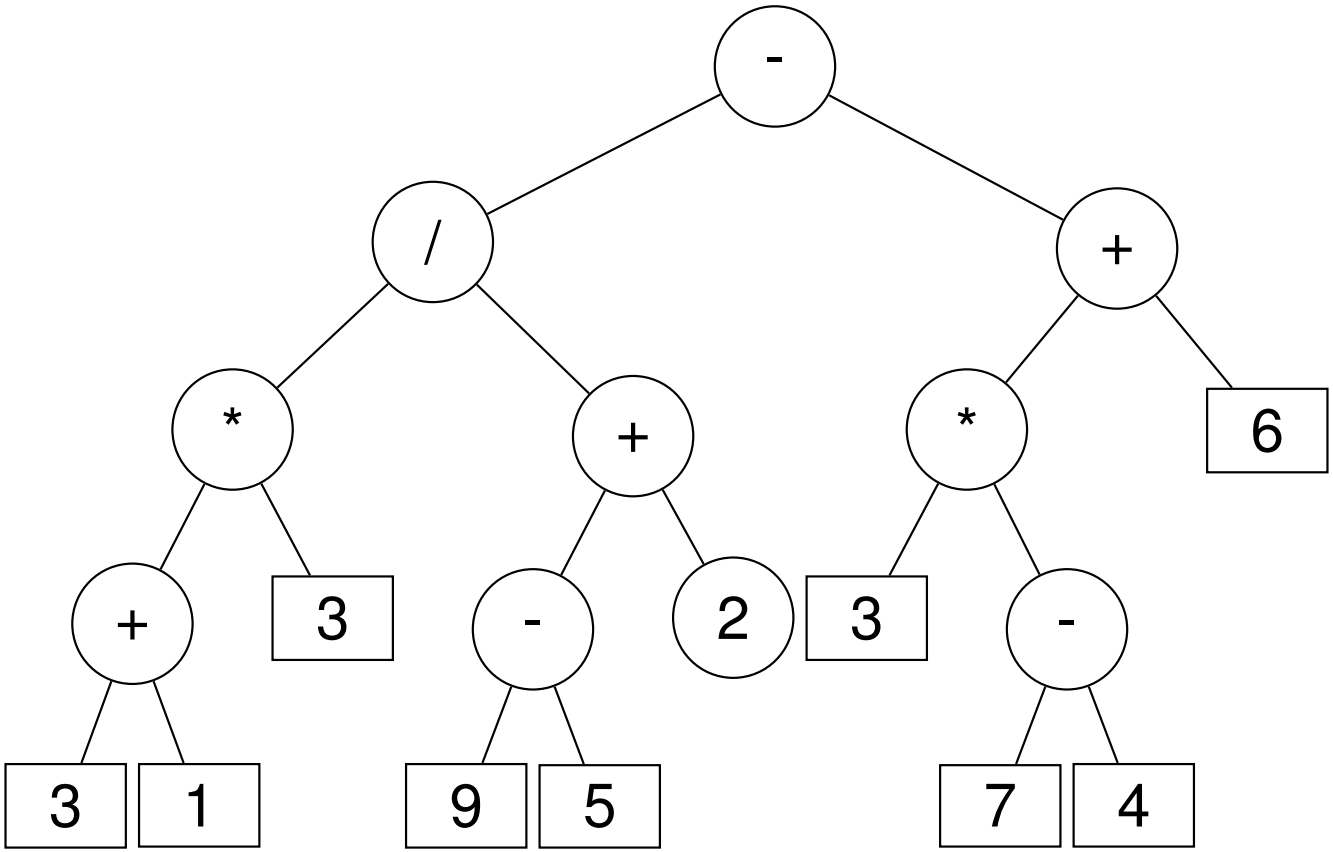
$g =$ grado del árbol

Índice

Idea intuitiva de árbol binario

- Ejemplo: compilación de expresión aritmética

```
((((3 + 1) * 3)/((9 - 5) + 2)) - ((3 * (7 - 4)) + 6))
```

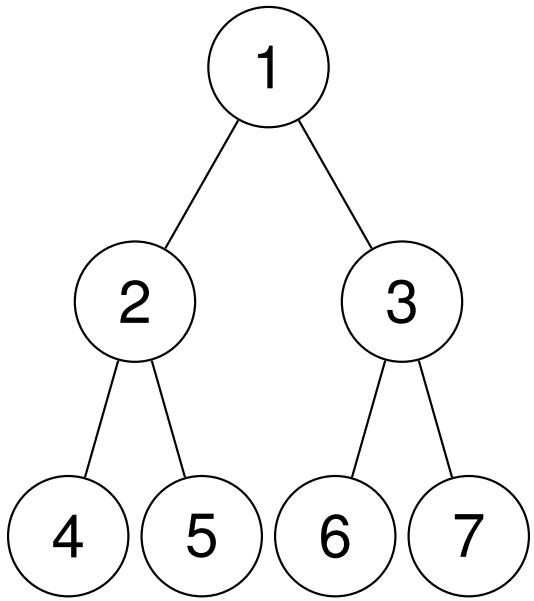


Definiciones

- Un **árbol binario** es un árbol de grado 2: cada nodo puede tener como máximo 2 hijos
- Consideraremos que cada nodo siempre tiene un hijo izquierdo y un hijo derecho
- Uno, o ambos hijos pueden estar vacíos
- Un nodo con los dos hijos vacíos es un **nodo hoja**

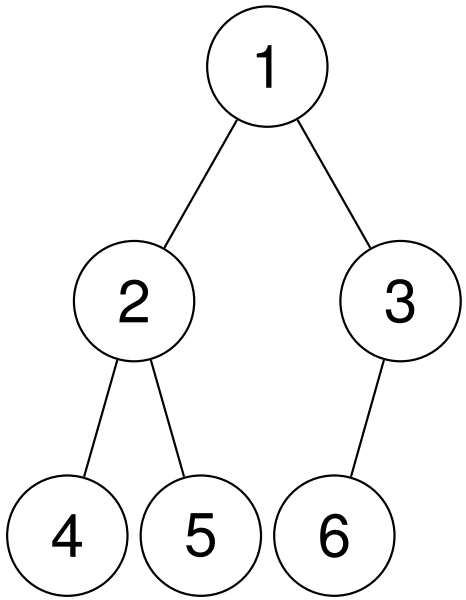
Definiciones

- Un árbol binario **perfecto** es aquel en el que todos los nodos internos tienen dos hijos y todas las hojas están en el mismo nivel



Definiciones

- Un árbol binario **completo** de altura h es aquel en el que los niveles del 1 al $h-1$ tienen el número máximo de nodos (igual que un árbol perfecto), y en el nivel h, los nodos siempre ocupan de manera continua posiciones desde la izquierda



Propiedades

- El máximo número de nodos en el nivel *i* de un árbol binario es 2^{i-1}
- Demostración informal:
  - Nivel 1 (raíz): $2^{1-1} = 2^0 = 1$
  - Nivel 2: $2^{2-1} = 2^1 = 2$
  - Nivel 3: $2^{3-1} = 2^2 = 4$
  - Y así sucesivamente, el número máximo se va duplicando en cada nivel

Propiedades

- El máximo número de nodos en un árbol binario de altura h es $2^h - 1$
- Demostración $\rightarrow$ suma del número de nodos de cada nivel i

$$\sum_{i=1}^{h} 2^{i-1} = 1 \cdot \frac{2^h - 1}{2 - 1} = 2^h - 1$$

- Suma de una progresión geométrica de razón r:

$$S = a_1 \cdot \frac{r^n - 1}{r - 1}$$

Especificación del TAD Iterador

Operaciones:

- Devuelve un iterador al nodo padre del nodo actual

```
Iterador parent() const;
```

- Devuelve un iterador al hijo izquierdo

```
Iterador left() const;
```

- Devuelve un iterador al hijo derecho

```
Iterador right() const;
```

Especificación del TAD Iterador

Operaciones:

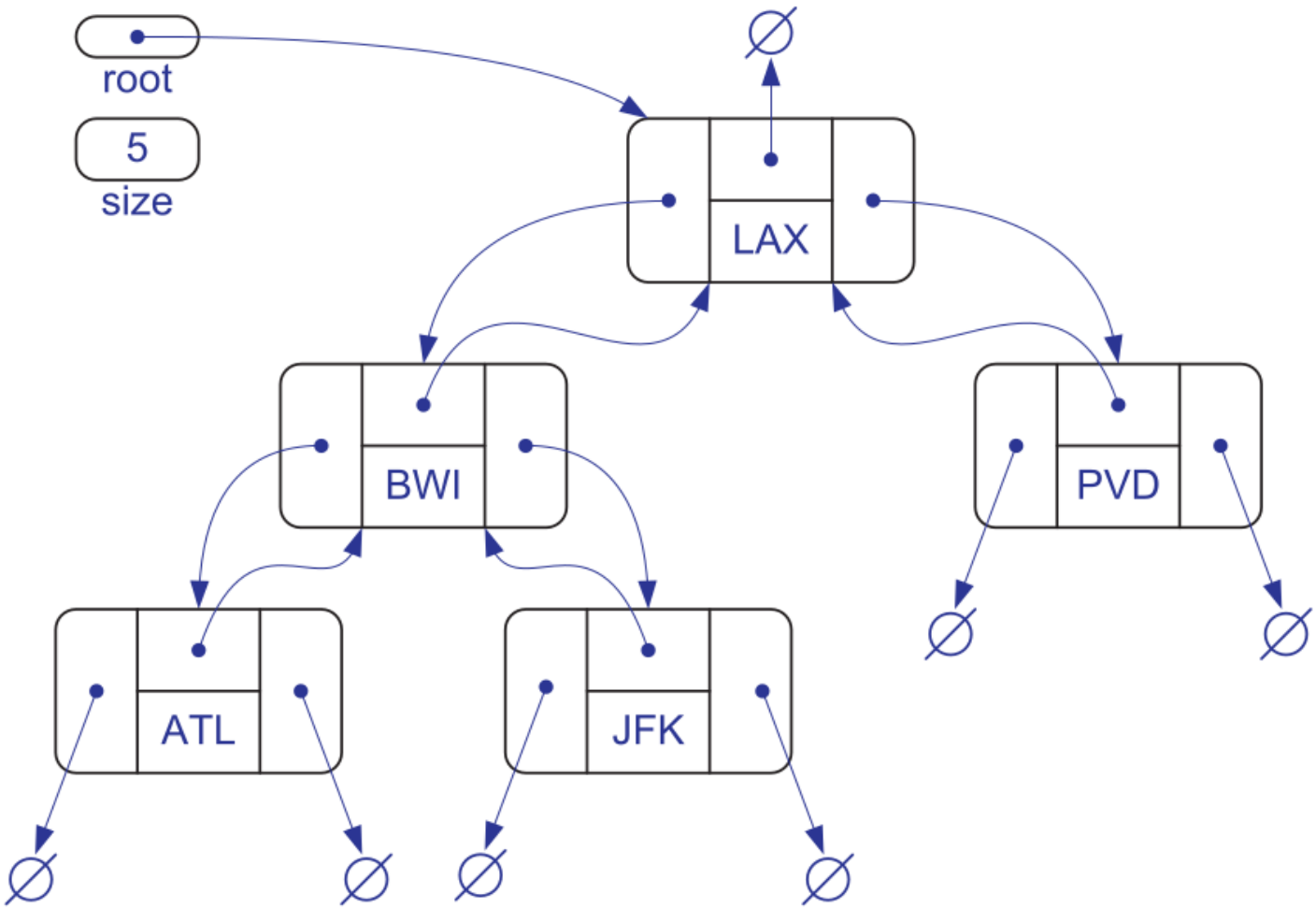
- Comprueba si el iterador apunta al nodo raíz

```
bool isRoot() const;
```

- Comprueba si el iterador apunta a un árbol vacío

```
bool isEmpty() const;
```

Implementación enlazada para árboles binarios



fuente: Goodrich, M. T., Tamassia, R., & Mount, D. M. (2011). Data structures and algorithms in C++. John Wiley & Sons.

Implementación enlazada para árboles binarios

```
class Arbol{
private:
Nodo* root;
....
};

class Nodo{
private:
    Elem data;
    Nodo* parent;
    Nodo* left;
    Nodo* right;
...
};
```

Implementación enlazada para árboles binarios

- Complejidad asintótica respecto al número de nodos en el árbol (`size()`)

Operación	Coste
`root()`	
`empty()`	
`set()`	
`get()`	
`parent()`	
`isRoot()`	
`left()`	
`right()`	

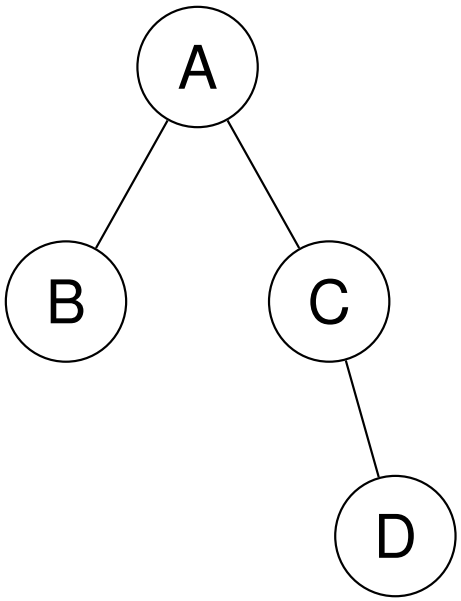
Implementación enlazada para árboles binarios

- Complejidad asintótica respecto al número de nodos en el árbol (`size()`)

Operación	Coste
`root()`	$\Theta(1)$
`empty()`	$\Theta(1)$
`set()`	$\Theta(1)$
`get()`	$\Theta(1)$
`parent()`	$\Theta(1)$
`isRoot()`	$\Theta(1)$
`left()`	$\Theta(1)$
`right()`	$\Theta(1)$

Implementación vector para árboles binarios

- Las posiciones del vector pasan a numerarse comenzando por 1
- El contenido del nodo raíz se almacena en la posición 1
- Si un nodo se almacena en la posición i, su hijo izquierdo se almacena en la posición $2i$, y su hijo derecho en la posicion $2i + 1$
- Si el árbol no es *perfecto*, puede haber posiciones "vacías", que se marcan con un valor especial



A	B	C	#	#	#	D
1	2	3	4	5	6	7

Implementación vector para árboles binarios

- La clase `Iterador` ahora contiene un índice en lugar de un puntero

Pregunta

¿Cómo implementarías el método `isRoot()` de la clase `Iterador`? ¿Y el método `parent()`? Asume que la clase `Iterador` tiene un atributo llamado `index` con el índice del vector al que apunta el iterador

Implementación vector para árboles binarios

- Las complejidades temporales son constantes al igual que en la implementación enlazada
- Si hubiésemos definido operaciones de inserción, éstas no tendrían coste constante en el peor caso (por los redimensionamientos)

Pregunta

¿Cuál es la complejidad **espacial** de la representación vectoral del árbol binario en el peor caso? (Pista: depende de la altura del árbol h y no del número de elementos que almacena)

Implementación vector para árboles binarios

- Las complejidades temporales son constantes al igual que en la implementación enlazada
- Si hubiésemos definido operaciones de inserción, éstas no tendrían coste constante en el peor caso (por los redimensionamientos)

Pregunta

¿Cuál es la complejidad **espacial** de la representación vectoral del árbol binario en el peor caso? (Pista: depende de la altura del árbol h y no del número de elementos que almacena)

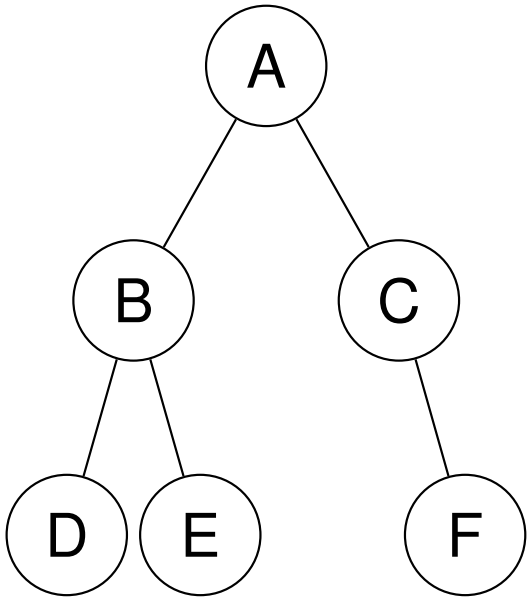
Respuesta: $O(2^h) \leftarrow$ el número de elementos de un árbol perfecto es $2^h - 1$

Índice

Recorridos en árboles binarios

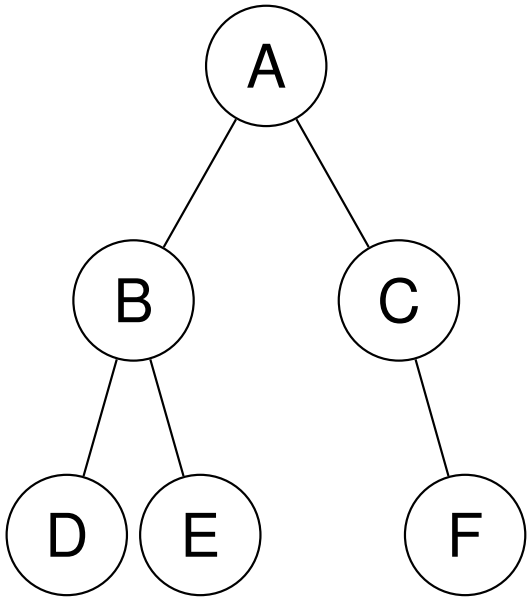
- **Recorrido**: manera sistemática de listar o recorrer todos los nodos de un árbol, sin repetir ninguno
- Dependiendo del tipo de los datos guardados en el árbol y lo que se desee hacer con ellos, elegiremos uno u otro
- Recorridos en profundidad:
  - Preorden
  - Postorden
  - Inorden
- Recorridos en anchura:
  - Niveles

Recorrido preorden




```
function PREORDEN(T,p)
    if !p.isEmpty() then
        PRINT(p)
        PREORDEN(p.left())
        PREORDEN(p.right())
```

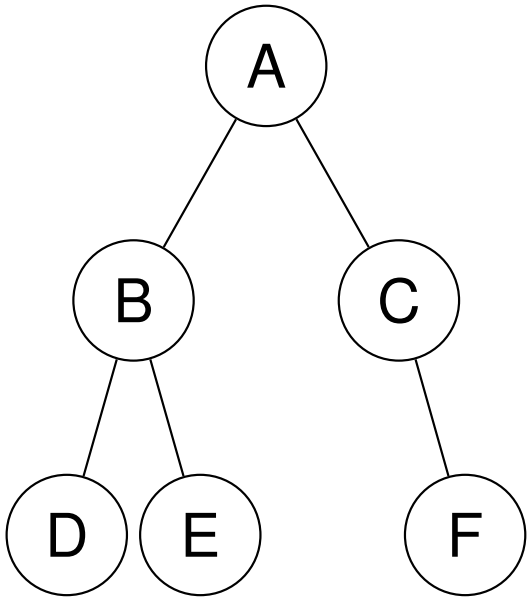
Recorrido preorden




```
function PREORDEN(T,p)
    if !p.isEmpty() then
        PRINT(p)
        PREORDEN(p.left())
        PREORDEN(p.right())
```

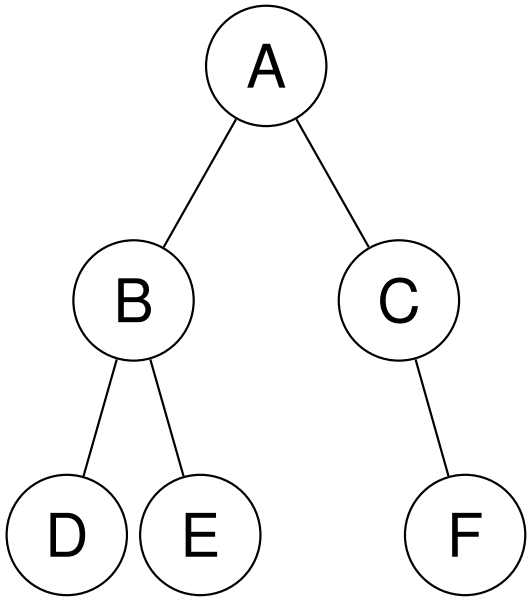
Resultado: A B D E C F

Recorrido postorden




```
function POSTORDEN(T,p)
    if !p.isEmpty() then
        POSTORDEN(p.left())
        POSTORDEN(p.right())
        PRINT(p)
```

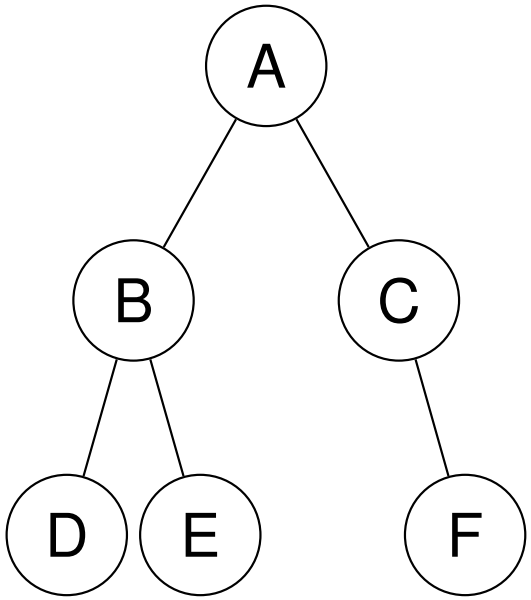
Recorrido postorden




```
function POSTORDEN(T,p)
    if !p.isEmpty() then
        POSTORDEN(p.left())
        POSTORDEN(p.right())
        PRINT(p)
```

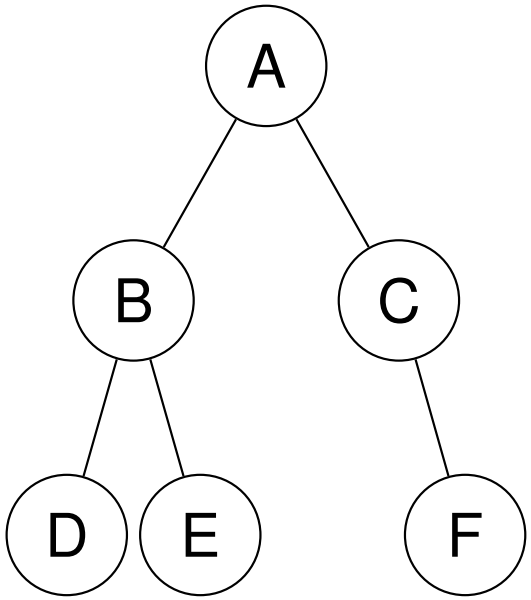
Resultado: D E B F C A

Recorrido inorden




```
function INORDEN(T,p)
    if !p.isEmpty() then
        INORDEN(p.left())
        PRINT(p)
        INORDEN(p.right())
```

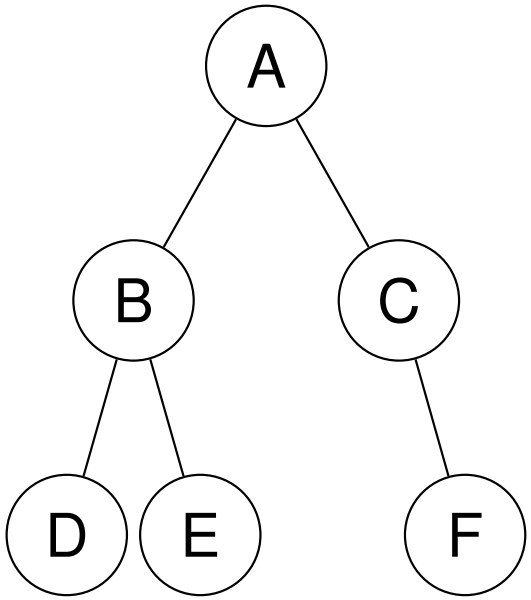
Recorrido inorden




```
function INORDEN(T,p)
    if !p.isEmpty() then
        INORDEN(p.left())
        PRINT(p)
        INORDEN(p.right())
```

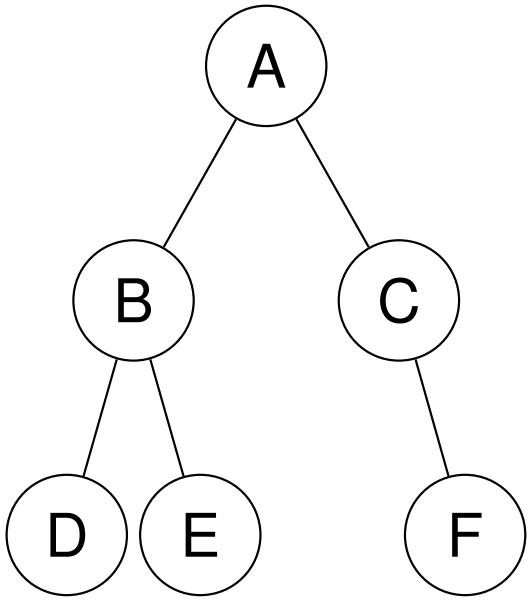
Resultado: D B E A C F

Recorrido niveles




```
function NIVELES(T,p)
    Cola c
    c.enqueue(p)
    while !c.isEmpty() do
        n=c.front()
        c.dequeue()
        PRINT(n)
        if !n.left().isEmpty()
        then
            c.enqueue(n.left())
        if !n.right().isEmpty()
        then
            c.enqueue(n.right())
```

Recorrido niveles



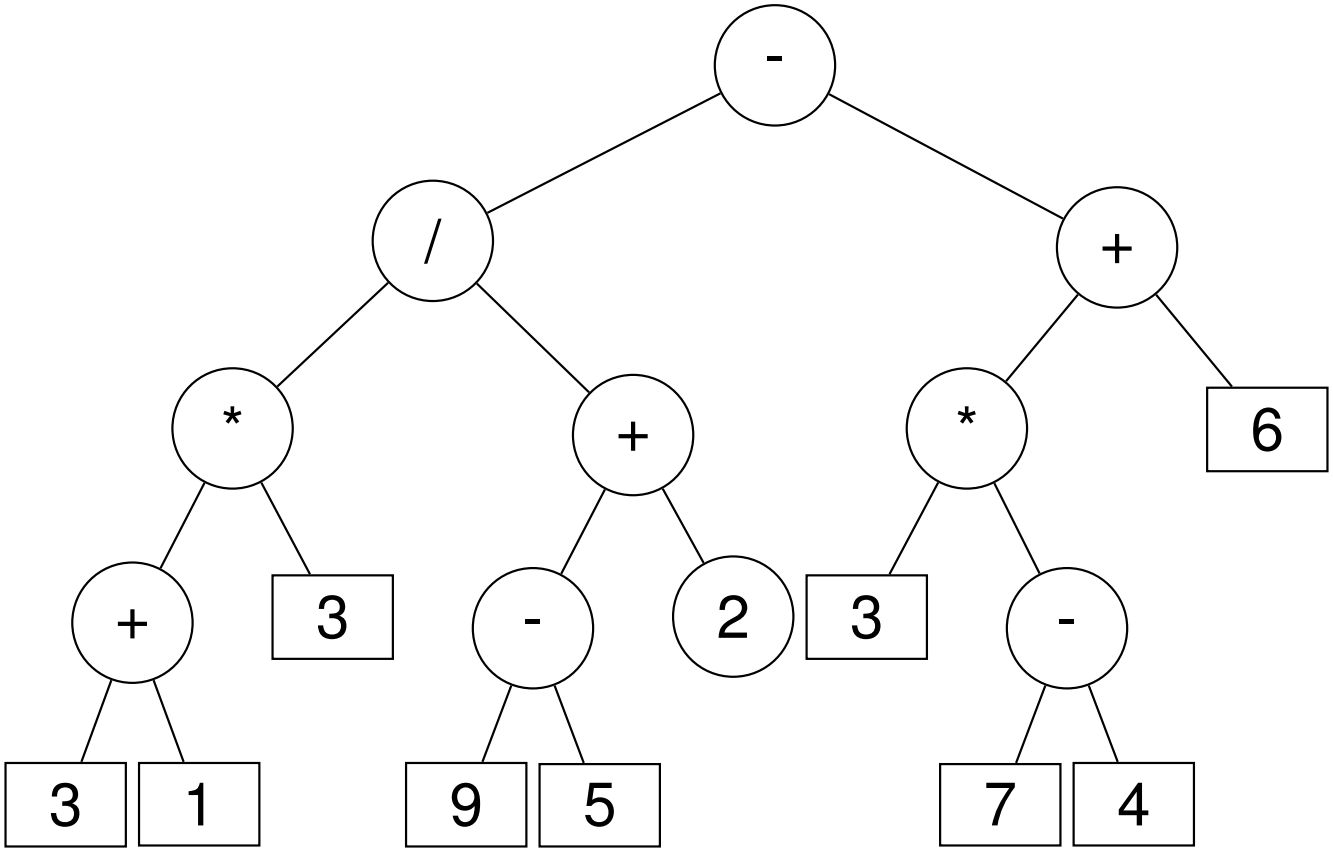

```
function NIVELES(T,p)
    Cola c
    c.enqueue(p)
    while !c.isEmpty() do
        n=c.front()
        c.dequeue()
        PRINT(n)
        if !n.left().isEmpty()
        then
            c.enqueue(n.left())
        if !n.right().isEmpty()
        then
            c.enqueue(n.right())
```

Resultado: A B C D E F

Recorridos en árboles binarios

Pregunta

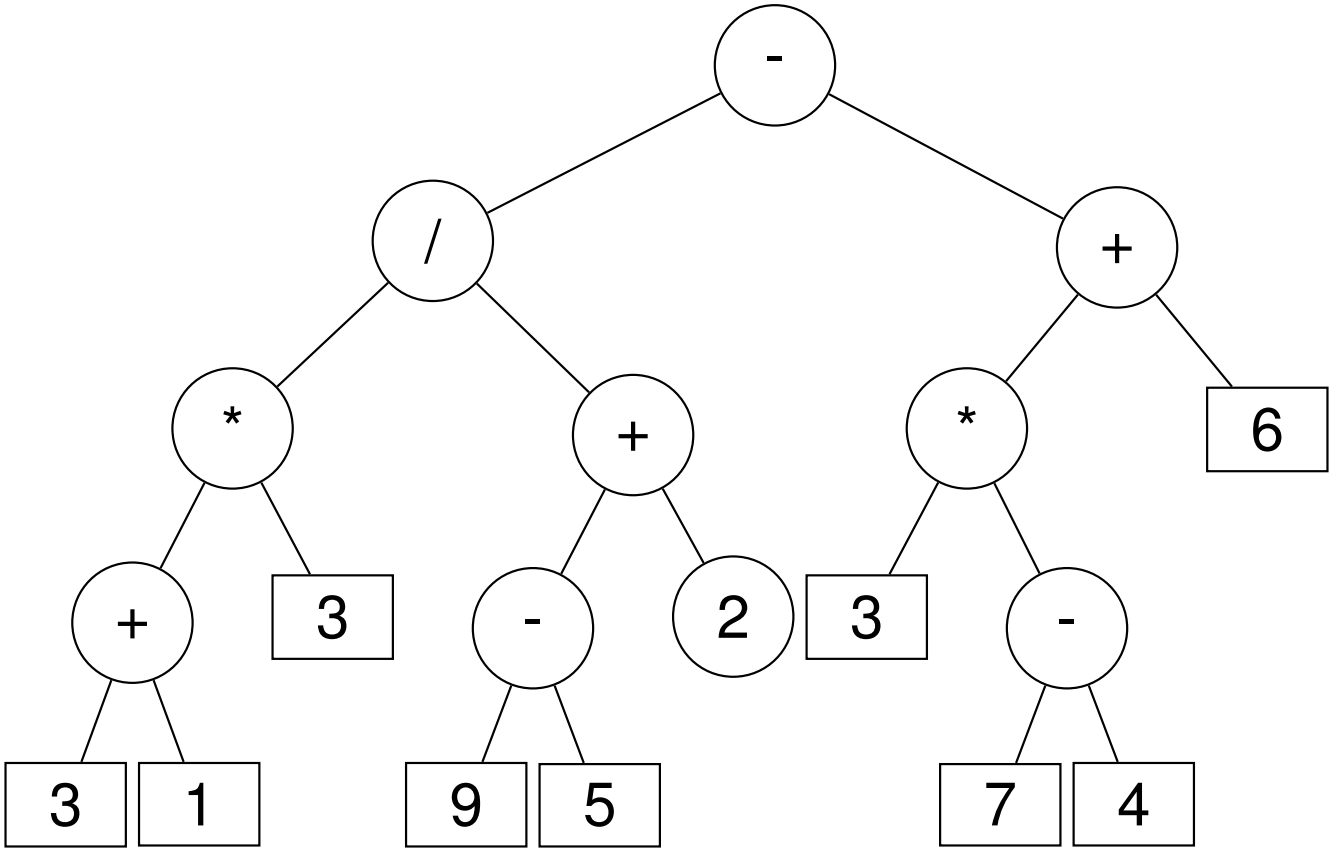
¿Qué recorrido emplearías para calcular el valor de la expresión aritmética a partir de su árbol?



Recorridos en árboles binarios

Pregunta

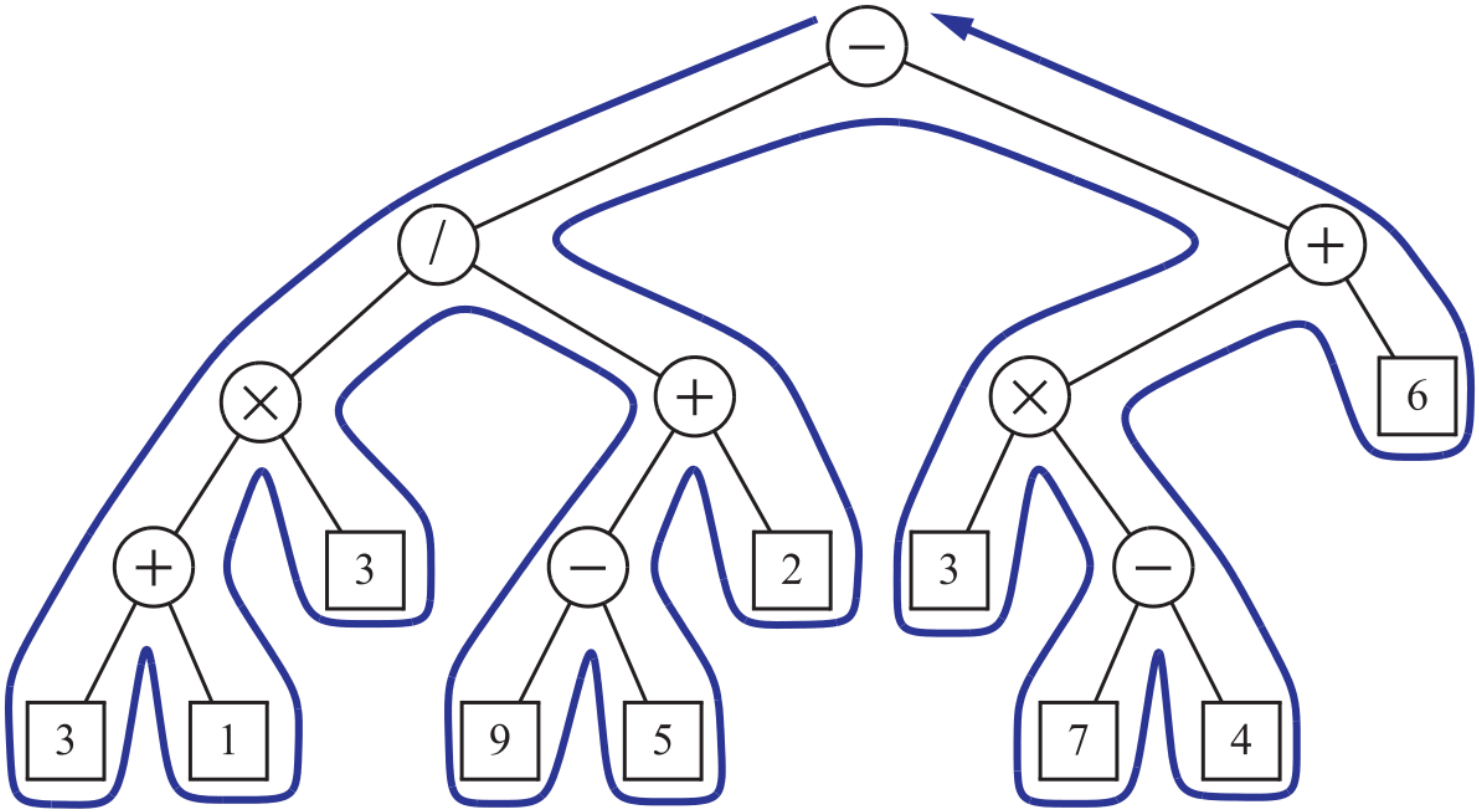
¿Qué recorrido emplearías para calcular el valor de la expresión aritmética a partir de su árbol?



Respuesta: postorden

Recorridos en árboles binarios

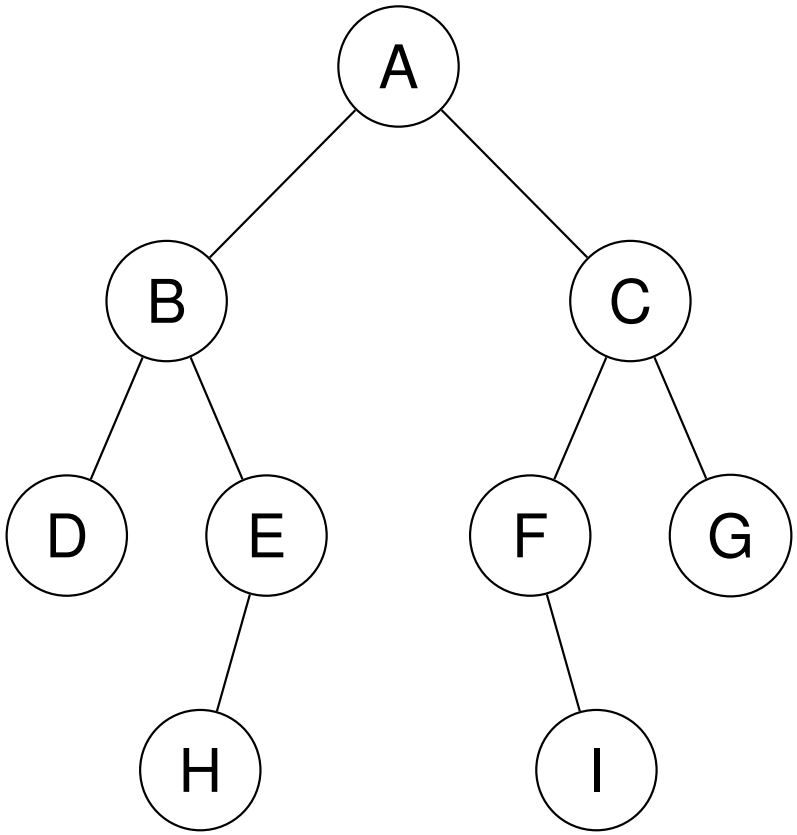
- El método del *recorrido de Euler* permite calcular los recorridos de manera gráfica



Recorridos en árboles binarios

Pregunta

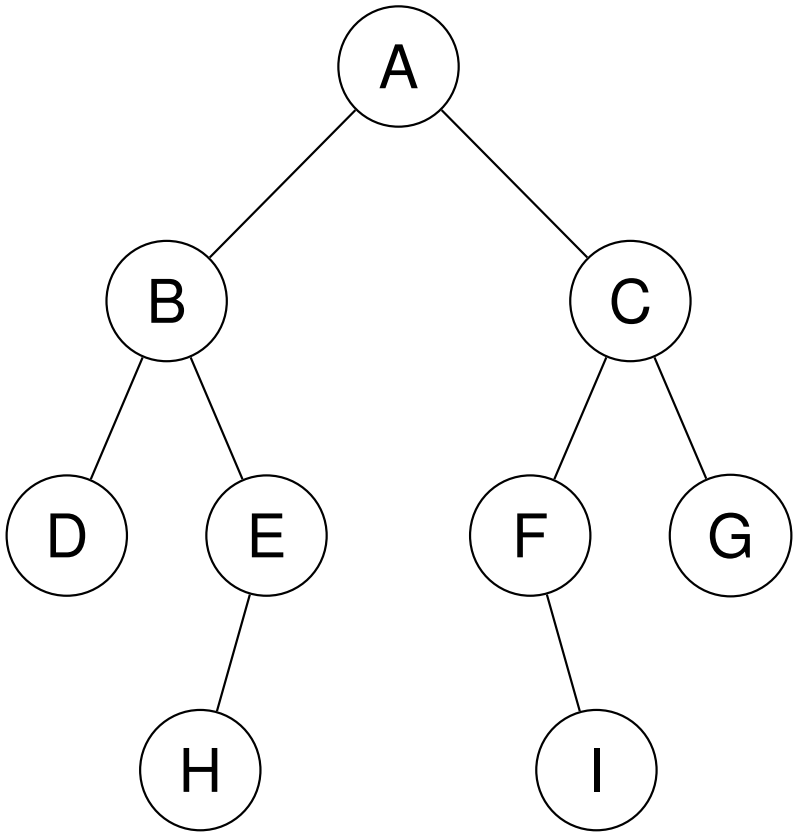
Calcula los recorridos preorden, postorden, inorden y niveles del siguiente árbol



Recorridos en árboles binarios

Pregunta

Calcula los recorridos preorden, postorden, inorden y niveles del siguiente árbol

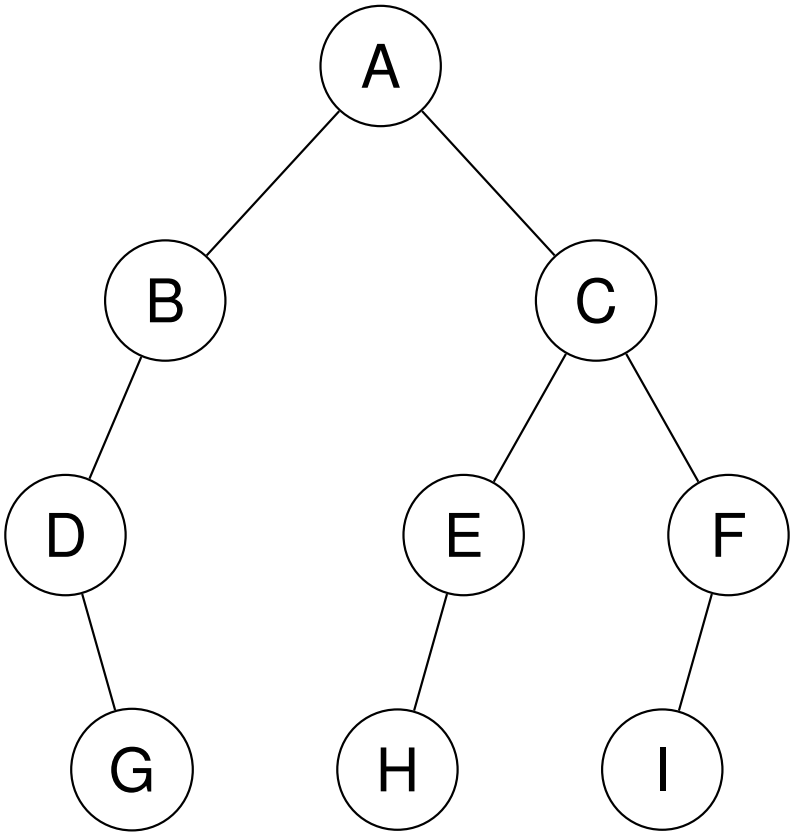


- Preorden: ABDEHCFIG
- Postorden: DHEBIFGCA
- Inorden: DBHEAFICG
- Niveles: ABCDEFGHI

Recorridos en árboles binarios

Ejercicio

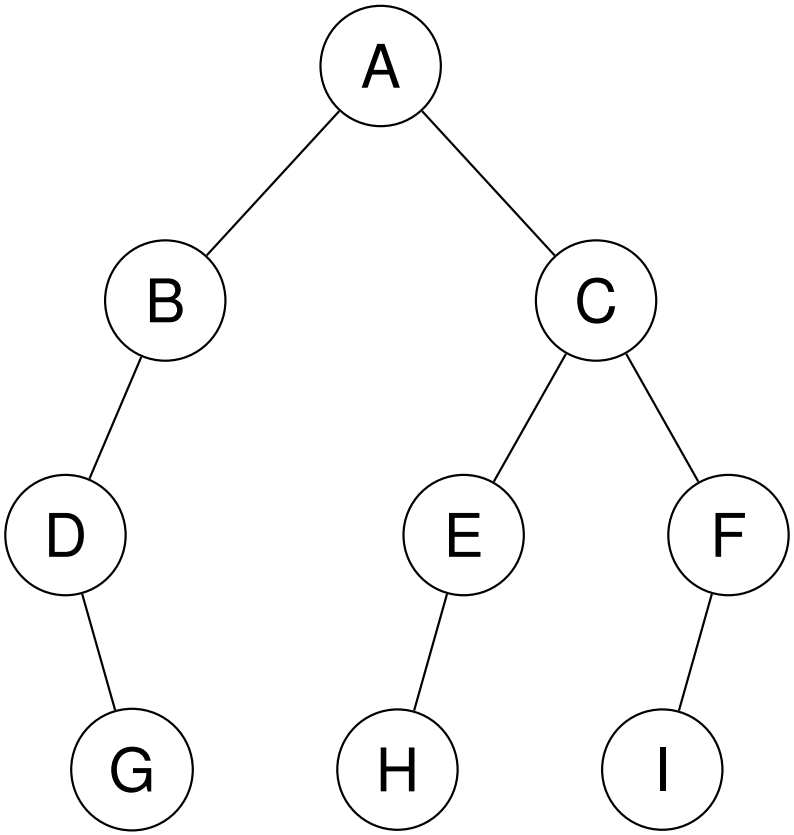
Calcula los recorridos preorden, postorden, inorden y niveles del siguiente árbol



Recorridos en árboles binarios

Ejercicio

Calcula los recorridos preorden, postorden, inorden y niveles del siguiente árbol



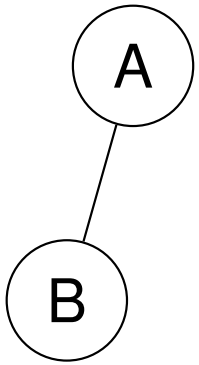
- Preorden: ABDGCEHFI
- Postorden: GDBHEIFCA
- Inorden: DGBAHECIF
- Niveles: ABCDEFGHI

Reconstrucción de árboles binarios

Pregunta

Dado un recorrido (por ejemplo, preorden), ¿crees que es posible reconstruir un único árbol a partir del mismo?

Para contestar a la pregunta, te puede ser útil calcular el recorrido preorden del siguiente árbol y luego intentar buscar otro árbol con el mismo recorrido:

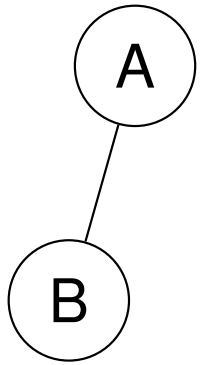


Reconstrucción de árboles binarios

Pregunta

Dado un recorrido (por ejemplo, preorden), ¿crees que es posible reconstruir un único árbol a partir del mismo?

Para contestar a la pregunta, te puede ser útil calcular el recorrido preorden del siguiente árbol y luego intentar buscar otro árbol con el mismo recorrido:



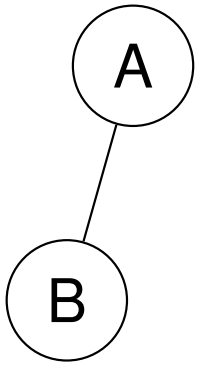
Respuesta: No es posible reconstruir un único árbol a partir de un recorrido

Reconstrucción de árboles binarios

Pregunta

¿Cuántos recorridos es necesario conocer para poder reconstruir un único árbol? ¿Cuáles?

Para contestar a la pregunta, te puede ser útil calcular los cuatro recorridos del siguiente árbol, y luego buscar pareja de recorridos casa con un único árbol

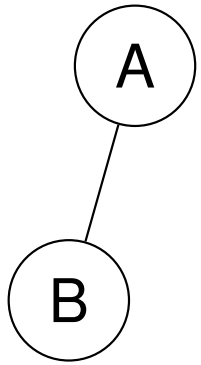


Reconstrucción de árboles binarios

Pregunta

¿Cuántos recorridos es necesario conocer para poder reconstruir un único árbol? ¿Cuáles?

Para contestar a la pregunta, te puede ser útil calcular los cuatro recorridos del siguiente árbol, y luego buscar pareja de recorridos casa con un único árbol



Respuesta: Dos. Inorden y (preorden o postorden o niveles)

Reconstrucción de árboles binarios

Algoritmo para la reconstrucción de un árbol a partir de sus recorridos preorden e inorden

1. Identifica el primer elemento del recorrido preorden: *r* es la raíz
2. Todos los elementos a la izquierda de *r* en el recorrido inorden forman parte del subárbol hijo izquierdo de *r*
3. Todos los elementos a la derecha de *r* en el recorrido inorden forman parte del subárbol hijo derecho de *r*
4. Vuelve al primer paso para cada subárbol

Reconstrucción de árboles binarios

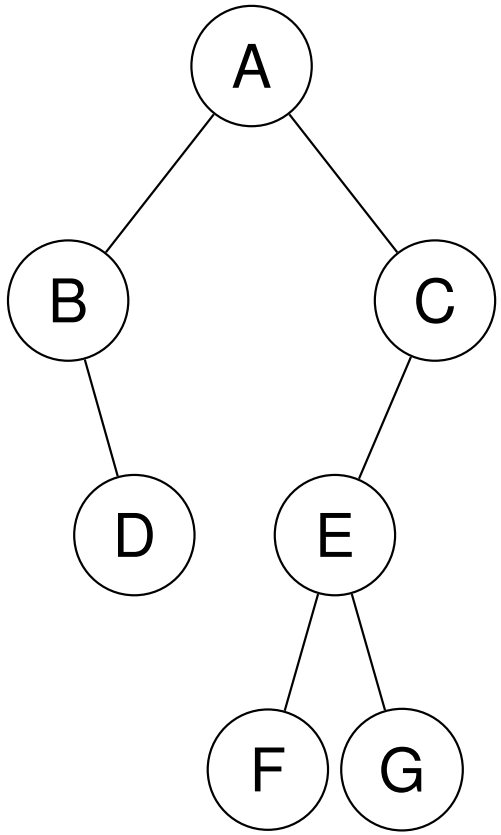
Reconstrucción de un árbol

- Preorden: ABDCEFG
- Inorden: BDAFEGC

Reconstrucción de árboles binarios

Reconstrucción de un árbol

- Preorden: ABDCEFG
- Inorden: BDAFEGC



Reconstrucción de árboles binarios

Ejercicio

Reconstruye un árbol binario a partir de los siguientes recorridos

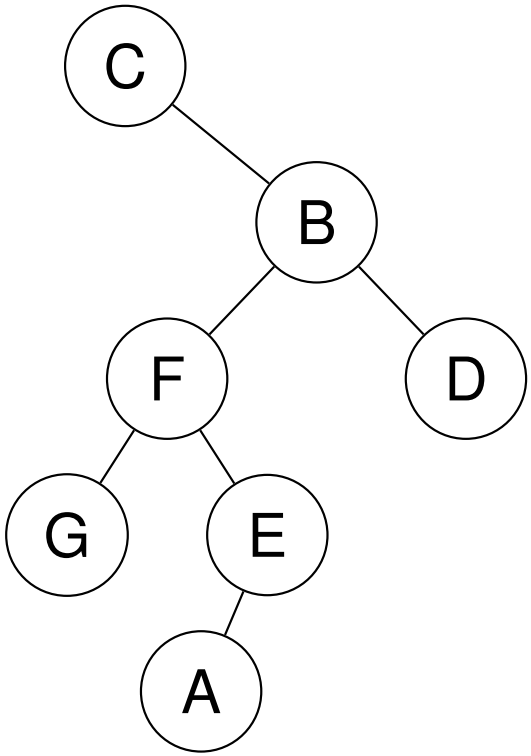
- Preorden: CBFGEAD
- Inorden: CGFAEBD

Reconstrucción de árboles binarios

Ejercicio

Reconstruye un árbol binario a partir de los siguientes recorridos

- Preorden: CBFGEAD
- Inorden: CGFAEBD



Reconstrucción de árboles binarios

Pregunta

¿Se te ocurre alguna situación bajo la que podría reconstruirse un árbol a partir de un único recorrido?

Reconstrucción de árboles binarios

Pregunta

¿Se te ocurre alguna situación bajo la que podría reconstruirse un árbol a partir de un único recorrido?

Respuesta: Si conocemos alguna otra propiedad del árbol, como su **estructura**. Se puede reconstruir un árbol conociendo su recorrido por niveles si sabemos que es perfecto o completo

Repaso

Pregunta

Contesta verdadero o falso a cada una de estas afirmaciones

- El nivel de un nodo coincide con la longitud del camino desde la raíz a ese nodo
- El grado de un árbol es la altura maxima que puede tener
- Un árbol siempre tiene más nodos internos que nodos hoja
- Existe un camino entre cualquier par de nodos de un árbol
- Un nodo hoja no tiene descendientes
- Un árbol binario con un único nodo es un árbol completo
- Todo árbol binario con dos nodos es completo
- El número máximo de nodos que se puede encontrar en el nivel 1 de un árbol binario es 2
- Un árbol binario de altura 3 obligatoriamente debe tener más de 3 nodos

Índice

Definición

Definición

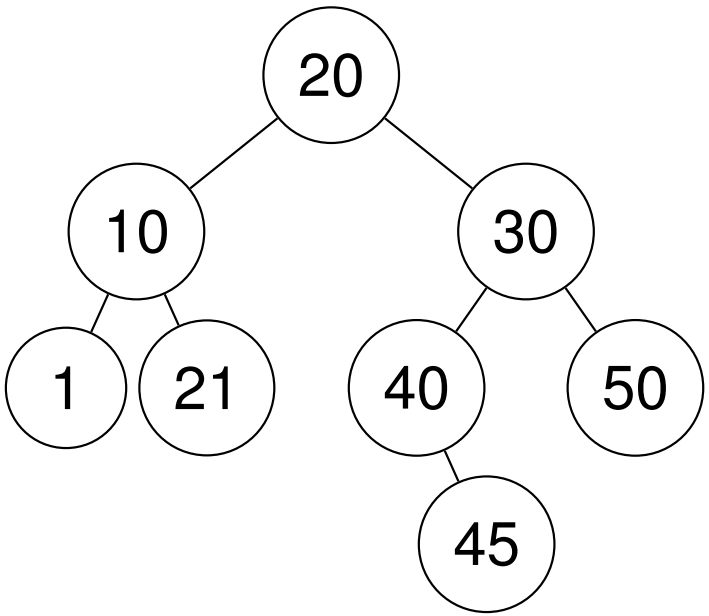
Un árbol binario de búsqueda (ABB) es un árbol binario que almacena un elemento en cada nodo. Los elementos deben ser de un tipo sobre el que esté definido una relación de orden total ($\leq$). Dado un nodo que almacena un elemento e, todos los elementos almacenados en su subárbol izquierdo deben ser $\leq e$. Todos los elementos almacenados en su subárbol derecho deben ser $> e$.

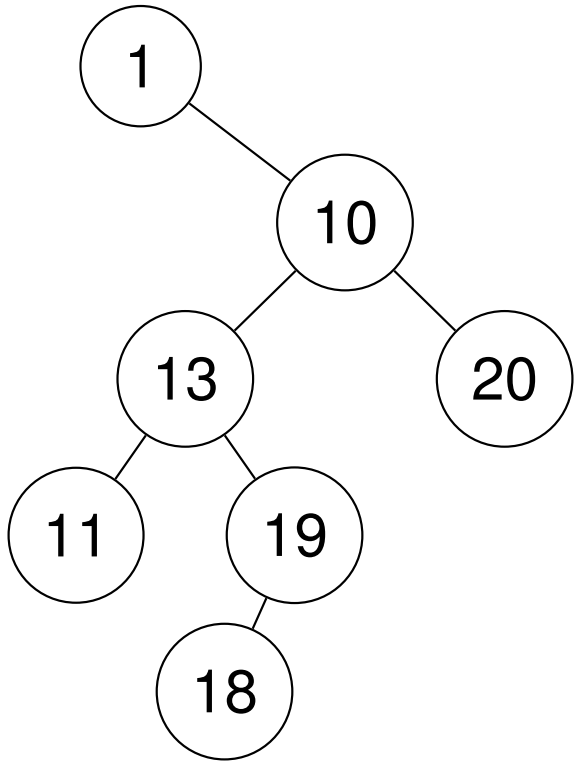
- Aunque su definición formal lo permite, asumiremos que no puede haber elementos repetidos en un árbol binario de búsqueda
- Un ABB almacena, por tanto, un **conjunto** de elementos

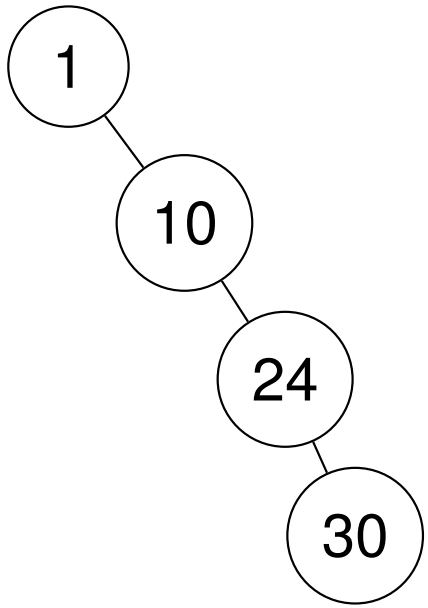
Definición

Pregunta

Indica cuáles de los siguientes árboles son árboles binarios de búsqueda







Recorridos sobre árboles binarios de búsqueda

Pregunta

¿Hay algún recorrido que nos devuelva los datos ordenados según la relación de orden parcial definida sobre los elementos almacenados?

Recorridos sobre árboles binarios de búsqueda

Pregunta

¿Hay algún recorrido que nos devuelva los datos ordenados según la relación de orden parcial definida sobre los elementos almacenados?

Respuesta: inorden

Reconstrucción de árboles binarios de búsqueda

Pregunta

¿Podemos reconstruir un árbol binario de búsqueda a partir de un único recorrido?

Reconstrucción de árboles binarios de búsqueda

Pregunta

¿Podemos reconstruir un árbol binario de búsqueda a partir de un único recorrido?

Respuesta: Sí. Preorden, postorden o niveles

Reconstrucción de árboles binarios de búsqueda

Algoritmo para la reconstrucción de un árbol binario de búsqueda a partir de sus recorrido preorden

1. Identifica el primer elemento del recorrido preorden: r es la raíz
2. Todos los elementos menores que r en el recorrido preorden forman parte del subárbol hijo izquierdo de r
3. Todos los elementos mayores que r en el recorrido preorden forman parte del subárbol hijo derecho de r
4. Vuelve al primer paso para cada subárbol

Reconstrucción de árboles binarios de búsqueda

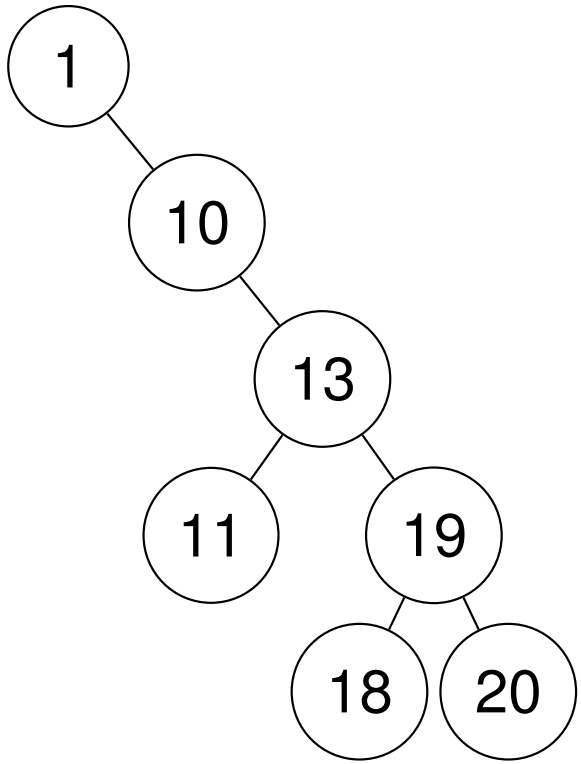
Reconstrucción de un árbol binario de búsqueda

- Preorden: 1,10,13,11,19,18,20

Reconstrucción de árboles binarios de búsqueda

Reconstrucción de un árbol binario de búsqueda

- Preorden: 1,10,13,11,19,18,20



Reconstrucción de árboles binarios de búsqueda

Ejercicio

Reconstruye un árbol binario de búsqueda a partir del siguiente recorrido

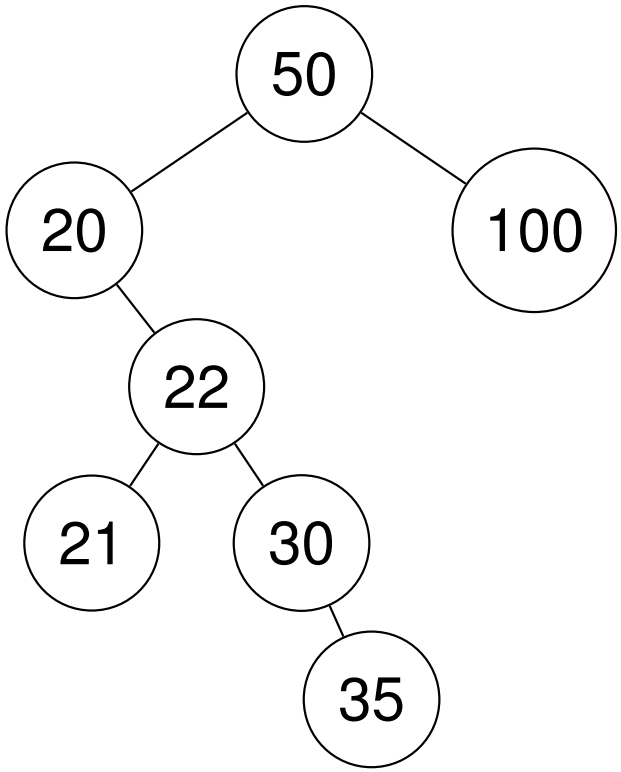
- Preorden: 50,20,22,21,30,35,100

Reconstrucción de árboles binarios de búsqueda

Ejercicio

Reconstruye un árbol binario de búsqueda a partir del siguiente recorrido

- Preorden: 50,20,22,21,30,35,100



Operaciones sobre árboles binarios de búsqueda

- Abordaremos las operaciones de búsqueda, inserción y borrado desde el punto de vista de las estructuras de datos
- Más adelante, definiremos tipos abstractos de datos (conjuntos) que pueden ser implementados mediante árboles binarios de búsqueda

Búsqueda

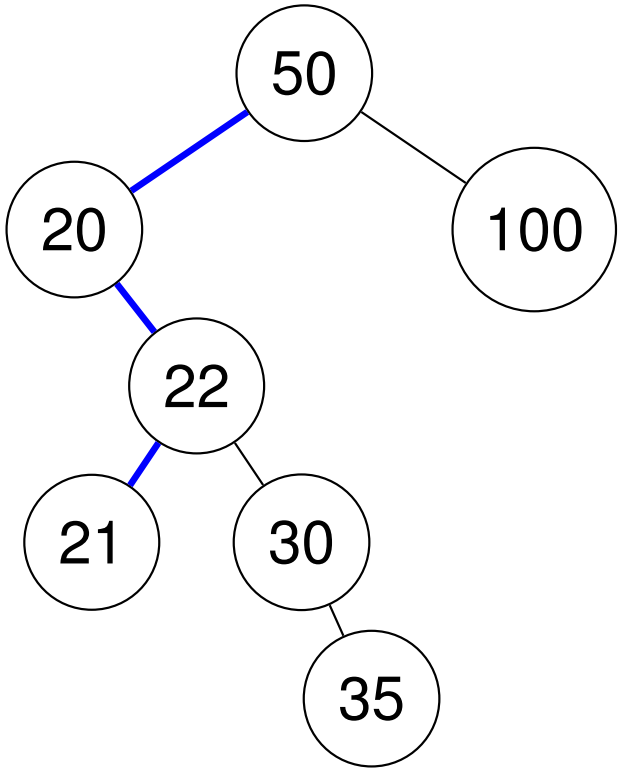
Algoritmo de búsqueda de un elemento k en un árbol binario de búsqueda:

- Si el árbol está vacío, k no está en el árbol
- En caso contrario, comparar k con el nodo raíz del árbol, al que llamaremos r:
  - Si $k = r$, el elemento está en el árbol
  - Si $k < r$, buscar k en el subárbol izquierdo
  - Si $k > r$, buscar k en el subárbol derecho

Búsqueda

Ejemplo

Búsqueda del elemento 21



Búsqueda

Pregunta

¿Cuál es la complejidad temporal de buscar en un árbol binario de búsqueda con n nodos?

Búsqueda

Pregunta

¿Cuál es la complejidad temporal de buscar en un árbol binario de búsqueda con n nodos?

- Mejor caso: el elemento es la raíz: $\Omega(1)$
- Peor caso: el elemento no está $\rightarrow$ coste proporcional a la altura $\rightarrow$

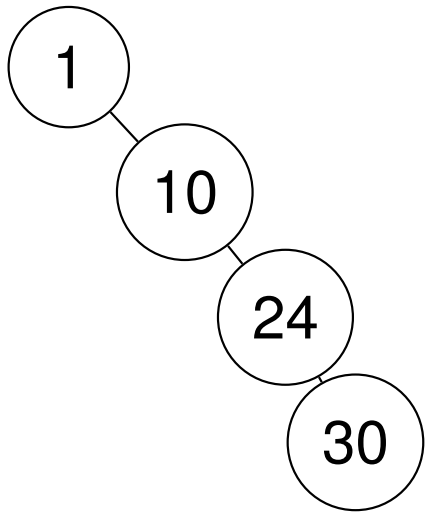
Búsqueda

Pregunta

¿Cuál es la complejidad temporal de buscar en un árbol binario de búsqueda con n nodos?

- Mejor caso: el elemento es la raíz: $\Omega(1)$
- Peor caso: el elemento no está $\rightarrow$ coste proporcional a la altura $\rightarrow O(n)$

Árbol degenerado:



Búsqueda

Pregunta

¿Cuál es la complejidad temporal de buscar en un árbol binario de búsqueda perfecto con n nodos?

Búsqueda

Pregunta

¿Cuál es la complejidad temporal de buscar en un árbol binario de búsqueda perfecto con n nodos?

- Mejor caso: el elemento es la raíz: $\Omega(1)$
- Peor caso: el elemento no está $\rightarrow$ coste proporcional a la altura $\rightarrow$

Pregunta

¿Cuál es la complejidad temporal de buscar en un árbol binario de búsqueda perfecto con n nodos?

- Mejor caso: el elemento es la raíz: $\Omega(1)$
- Peor caso: el elemento no está $\rightarrow$ coste proporcional a la altura $\rightarrow O(\log n)$

Demostración:

- Un árbol perfecto de altura h tiene $2^h - 1$ nodos
- Un árbol perfecto con n nodos tiene la siguiente altura:

$$n = 2^h - 1$$

$$n + 1 = 2^h$$

$$\log_2(n + 1) = h$$

Inserción

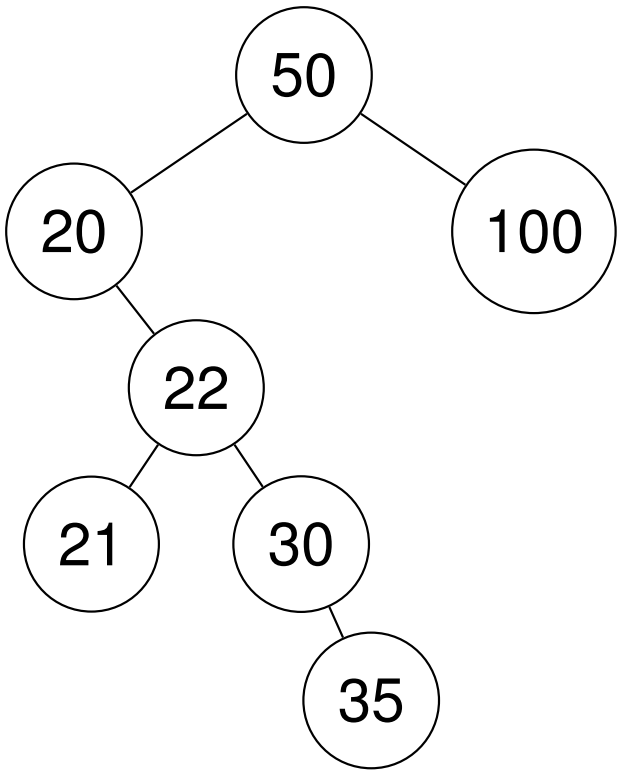
Algoritmo de inserción de un elemento k en un árbol binario de búsqueda:

- Si el árbol está vacío, k pasa a ser la raíz del árbol
- En caso contrario, comparar k con el nodo raíz del árbol, al que llamaremos r:
  - Si $k = r$, no se puede insertar
  - Si $k < r$, insertar k en el subárbol izquierdo
  - Si $k > r$, insertar k en el subárbol derecho

Inserción

Ejemplo

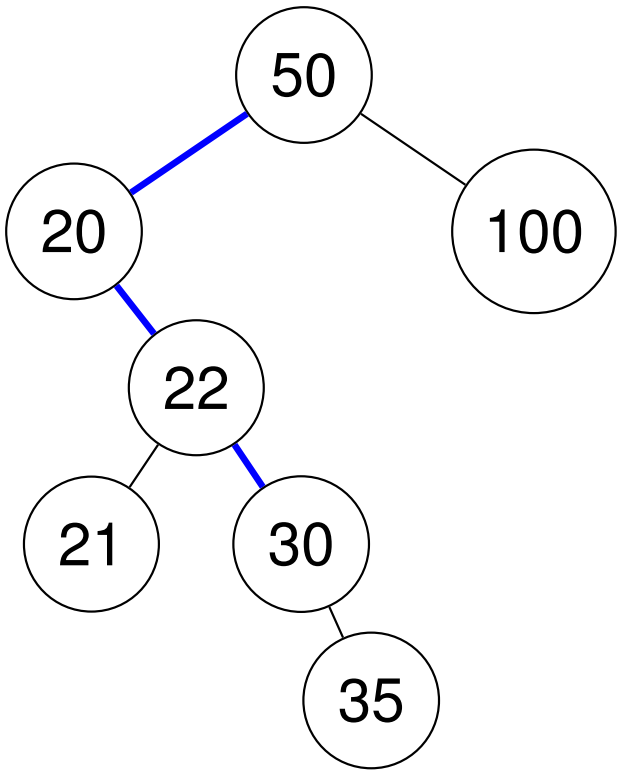
Inserción del elemento 28



Inserción

Ejemplo

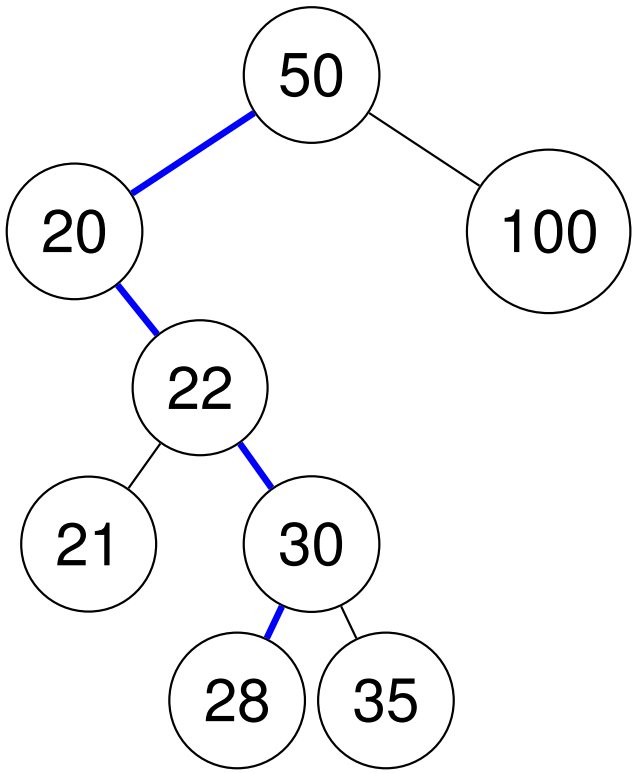
Inserción del elemento 28



Inserción

Ejemplo

Inserción del elemento 28



Inserción

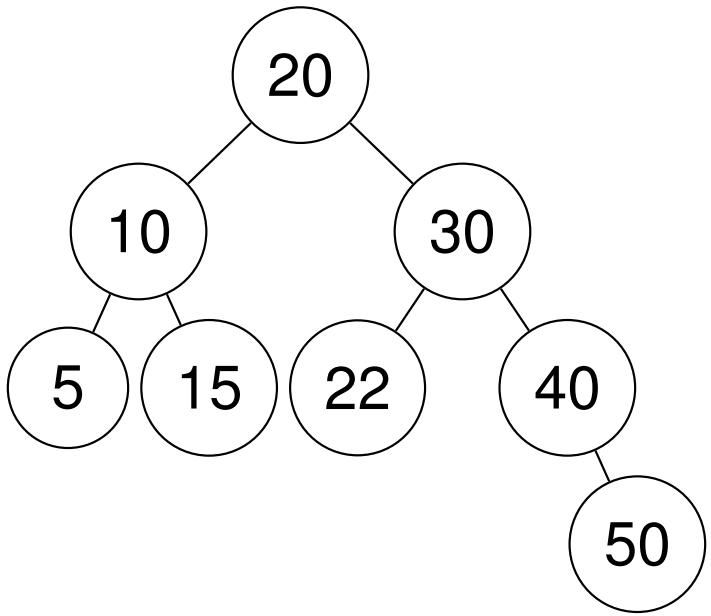
Ejercicio

Inserta los siguientes elementos en un árbol binario de búsqueda inicialmente vacío: 20, 10, 30, 40, 5, 15, 50, 22,

Inserción

Ejercicio

Inserta los siguientes elementos en un árbol binario de búsqueda inicialmente vacío: 20, 10, 30, 40, 5, 15, 50, 22,



Inserción

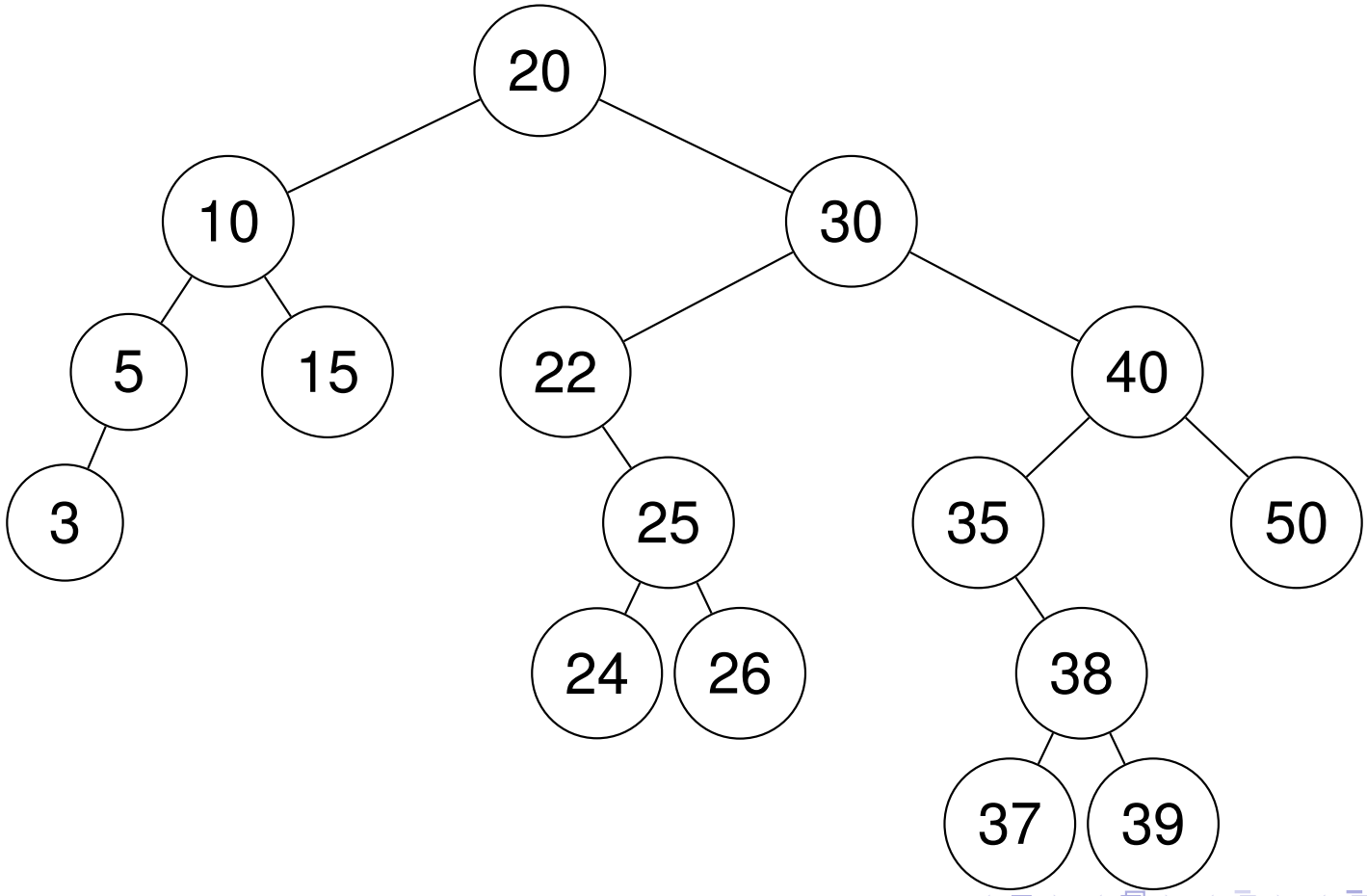
Ejercicio

Inserta los siguientes elementos en el árbol obtenido en el ejercicio anterior: 25, 24, 26, 3, 35, 38, 39, 37

Inserción

Ejercicio

Inserta los siguientes elementos en el árbol obtenido en el ejercicio anterior: 25, 24, 26, 3, 35, 38, 39, 37



Borrado

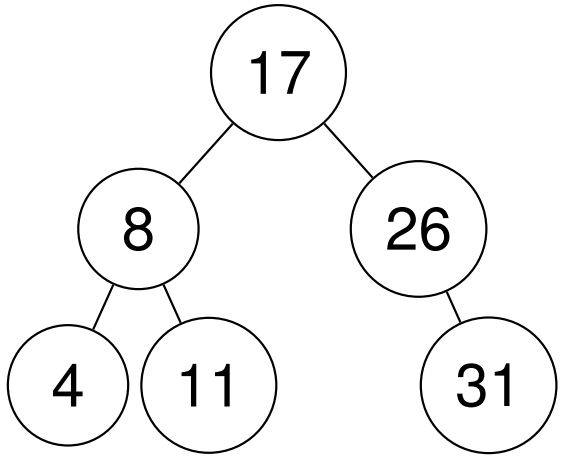
Algoritmo de borrado de un elemento k en un árbol binario de búsqueda:

Se aplica el algoritmo de búsqueda hasta encontrar k. Se pueden dar tres situaciones:

- k está en un nodo hoja $\rightarrow$ se elimina el nodo directamente

Ejemplo

Borrado del 11



Borrado

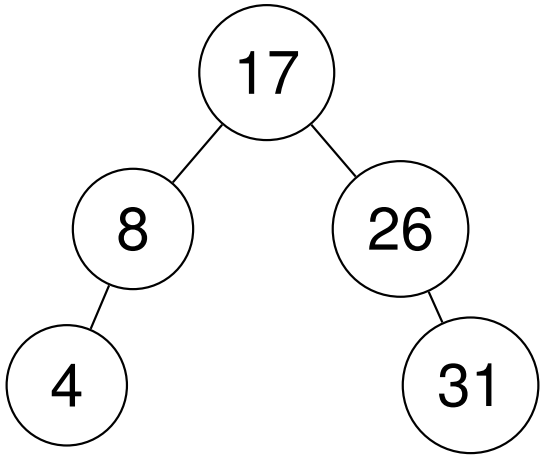
Algoritmo de borrado de un elemento k en un árbol binario de búsqueda:

Se aplica el algoritmo de búsqueda hasta encontrar k. Se pueden dar tres situaciones:

- k está en un nodo hoja $\rightarrow$ se elimina el nodo directamente

Ejemplo

Borrado del 11



Borrado

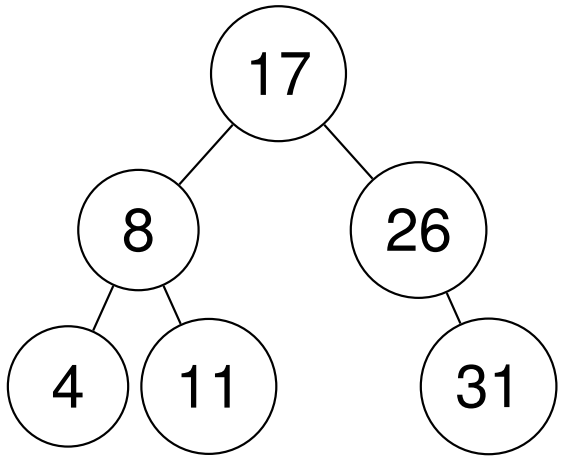
Algoritmo de borrado de un elemento k en un árbol binario de búsqueda:

Se aplica el algoritmo de búsqueda hasta encontrar k. Se pueden dar tres situaciones:

- k está en un nodo que tiene un solo hijo $\rightarrow$ el nodo se sustituye por su único hijo

Ejemplo

Borrado del 26



Borrado

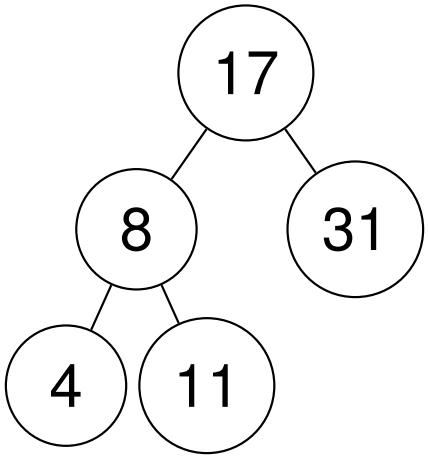
Algoritmo de borrado de un elemento k en un árbol binario de búsqueda:

Se aplica el algoritmo de búsqueda hasta encontrar k. Se pueden dar tres situaciones:

- k está en un nodo que tiene un solo hijo $\rightarrow$ el nodo se sustituye por su único hijo

Ejemplo

Borrado del 26



Borrado

Algoritmo de borrado de un elemento k en un árbol binario de búsqueda:

Se aplica el algoritmo de búsqueda hasta encontrar k. Se pueden dar tres situaciones:

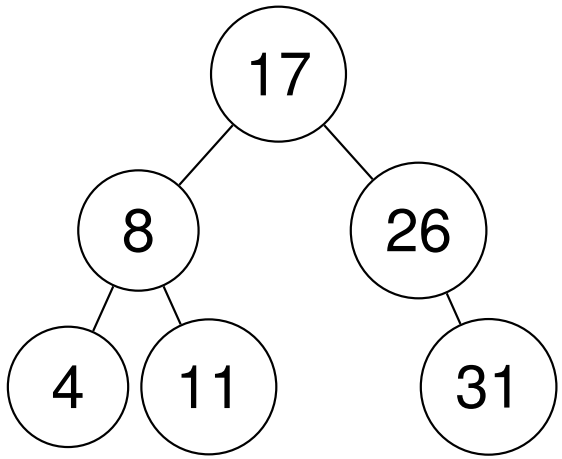
- k está en un nodo con dos hijos $\rightarrow$ k se sustiye por el mayor elemento de su subárbol izquierdo (M_i) o por el menor elemento de su subárbol derecho (m_d); a continuación, se vuelve a aplicar el algoritmo para borrar M_i o m_d

Borrado

Algoritmo de borrado de un elemento k en un árbol binario de búsqueda:

Ejemplo

Borrado del 17. Estrategia: sustitución por M_i

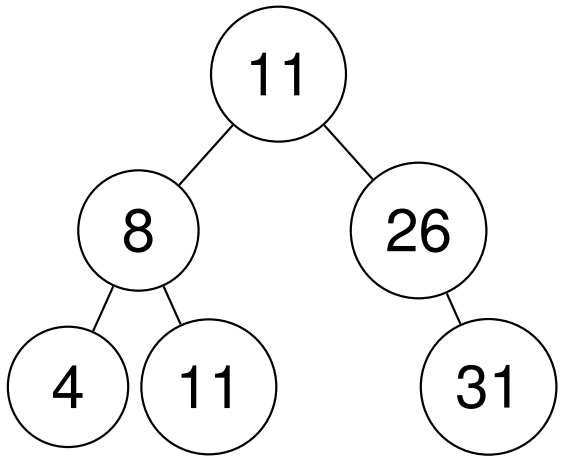


Borrado

Algoritmo de borrado de un elemento k en un árbol binario de búsqueda:

Ejemplo

Borrado del 17. Estrategia: sustitución por M_i

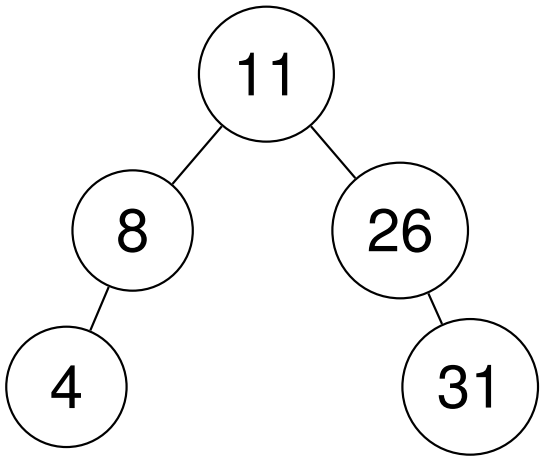


Borrado

Algoritmo de borrado de un elemento k en un árbol binario de búsqueda:

Ejemplo

Borrado del 17. Estrategia: sustitución por M_i

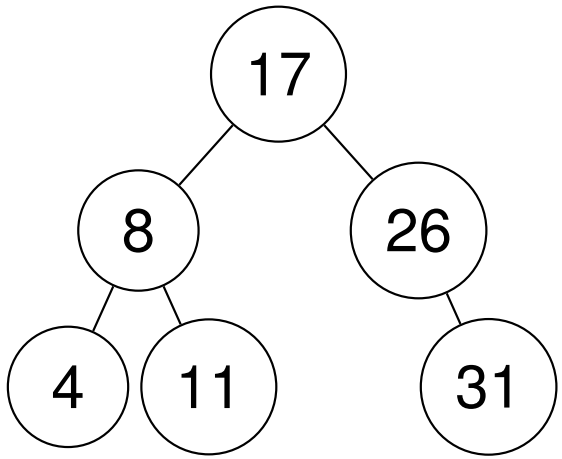


Borrado

Algoritmo de borrado de un elemento k en un árbol binario de búsqueda:

Ejemplo

Borrado del 17. Estrategia: sustitución por m_d

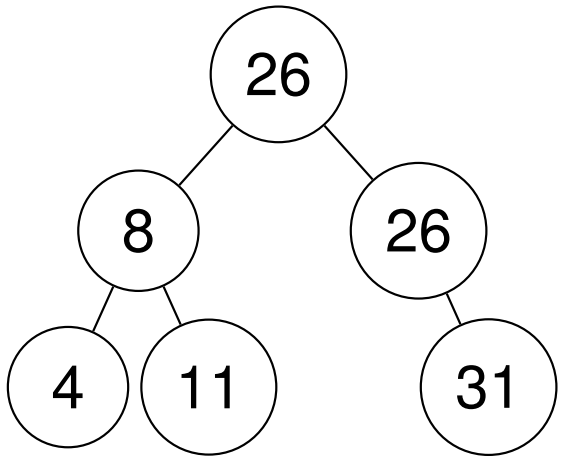


Borrado

Algoritmo de borrado de un elemento k en un árbol binario de búsqueda:

Ejemplo

Borrado del 17. Estrategia: sustitución por m_d

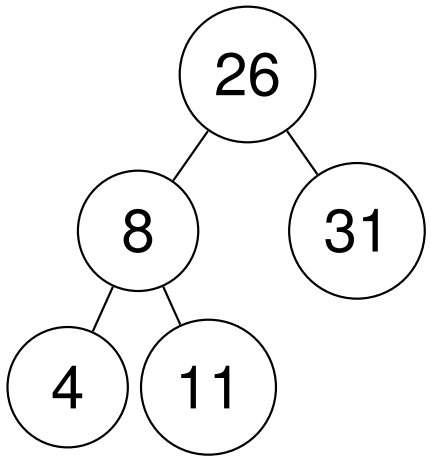


Borrado

Algoritmo de borrado de un elemento k en un árbol binario de búsqueda:

Ejemplo

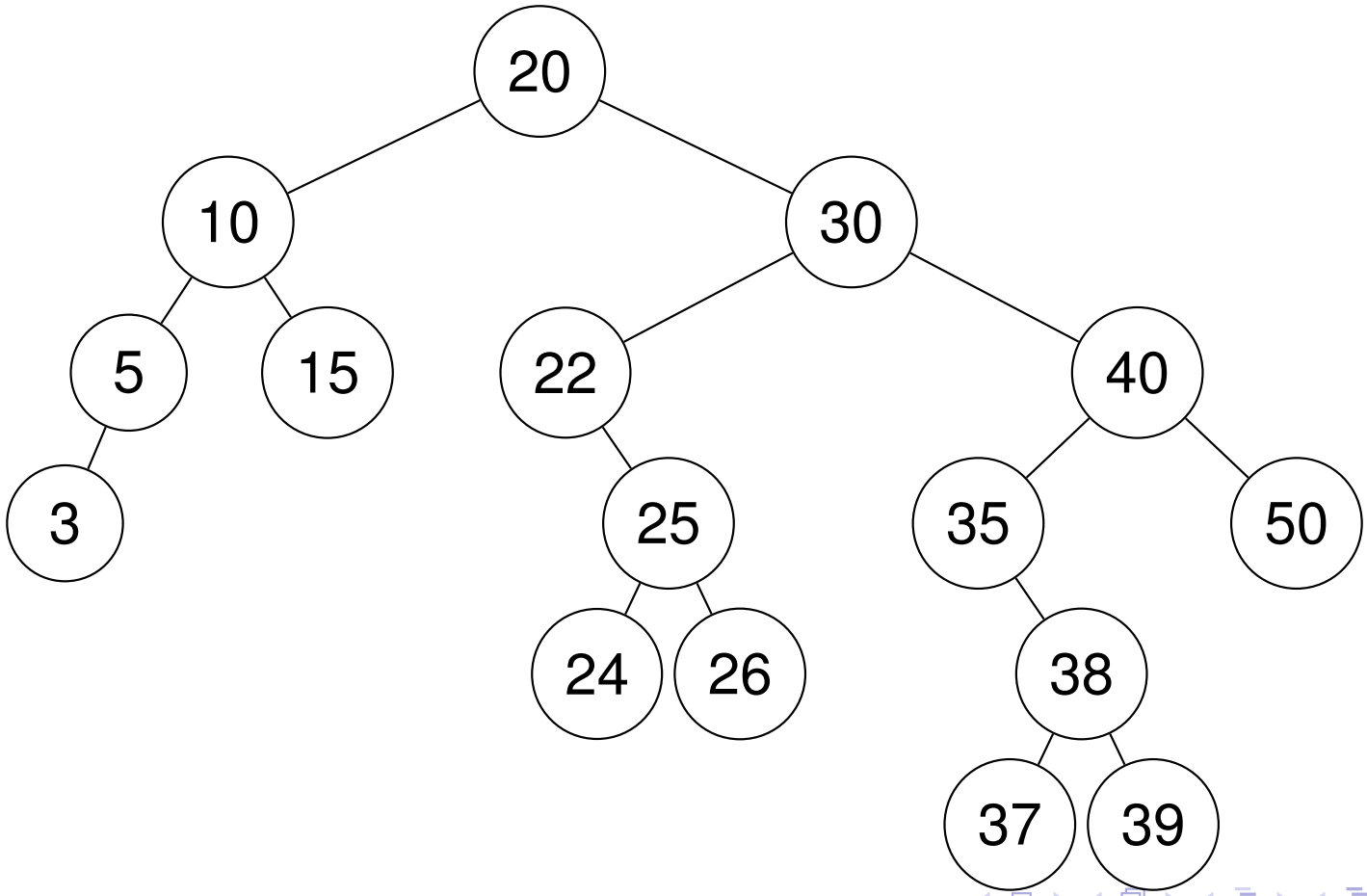
Borrado del 17. Estrategia: sustitución por m_d



Borrado

Ejercicio

Borra los siguientes elementos del árbol que se muestra a continuación: 5, 3, 30, 22, 39. Criterio: sustitución por m_d

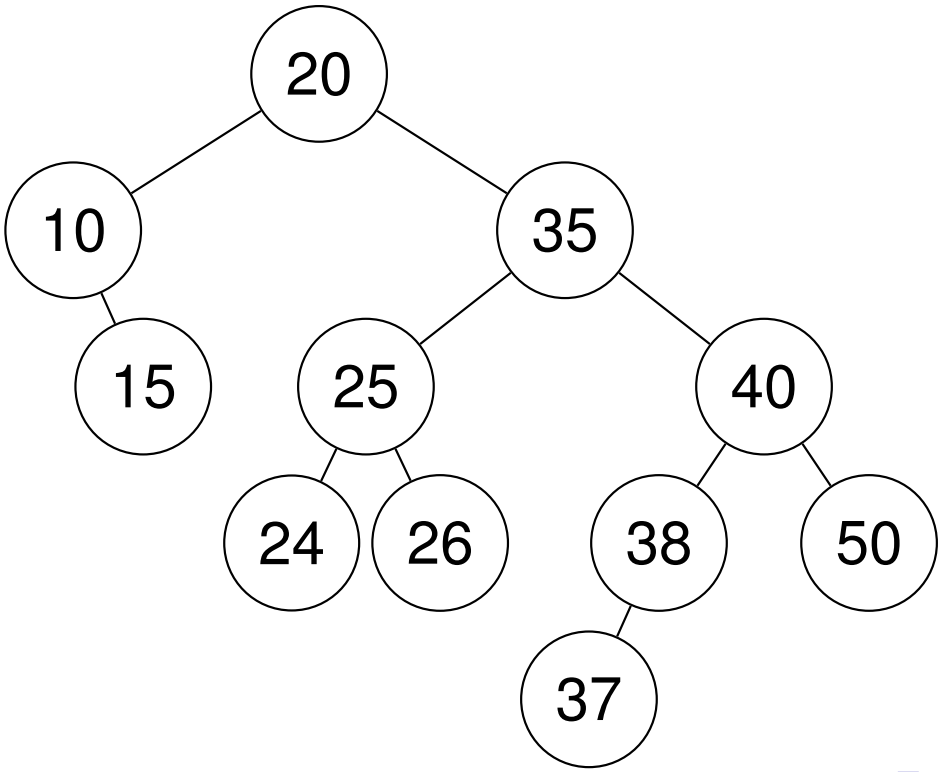


Borrado

Ejercicio

Borra los siguientes elementos del árbol que se muestra a continuación: 5, 3, 30, 22, 39. Criterio: sustitución por m_d

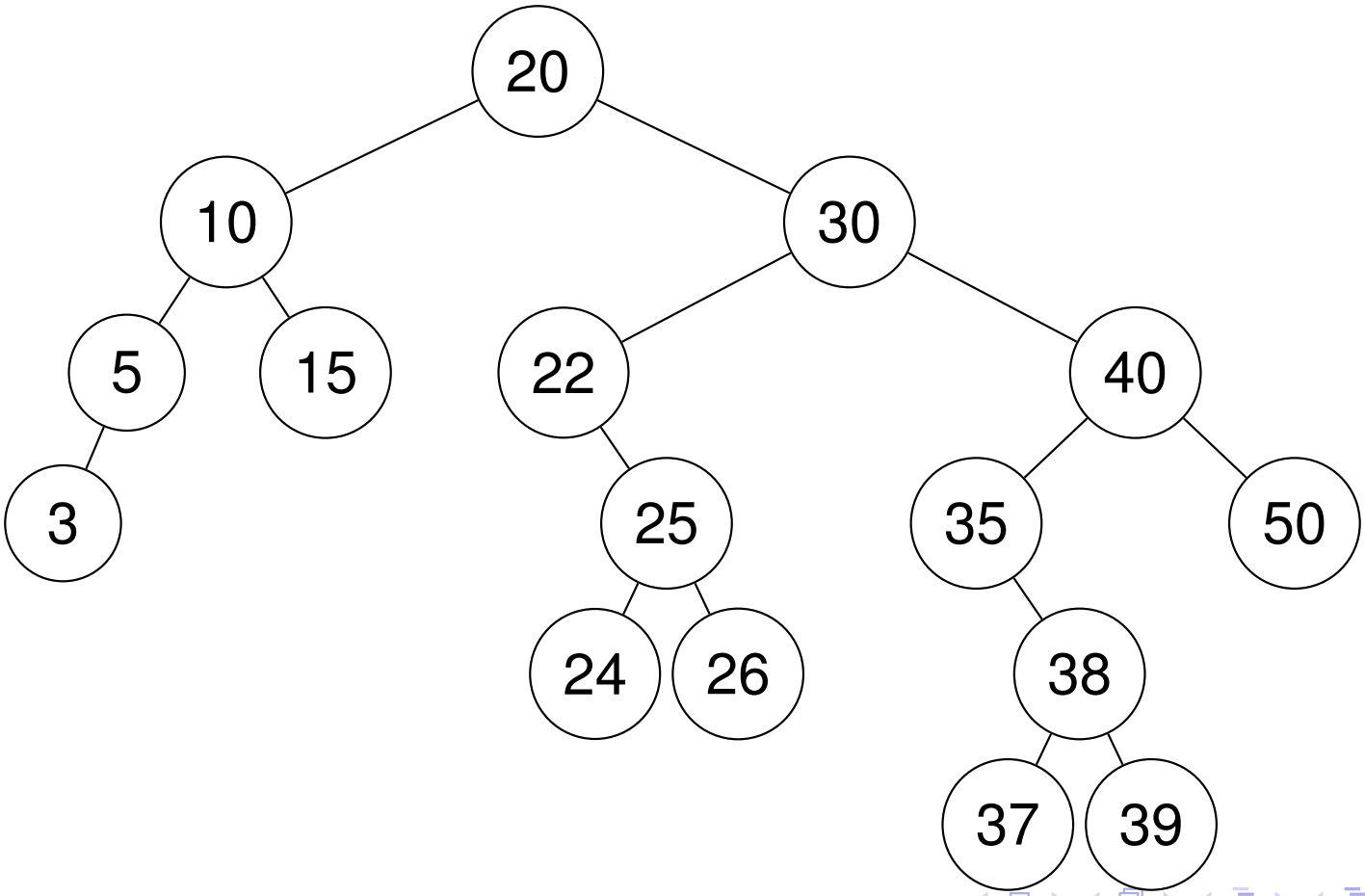
Solución:



Borrado

Ejercicio

Borra los siguientes elementos del árbol que se muestra a continuación: 5, 3, 30, 22, 39. Criterio: sustitución por M_i

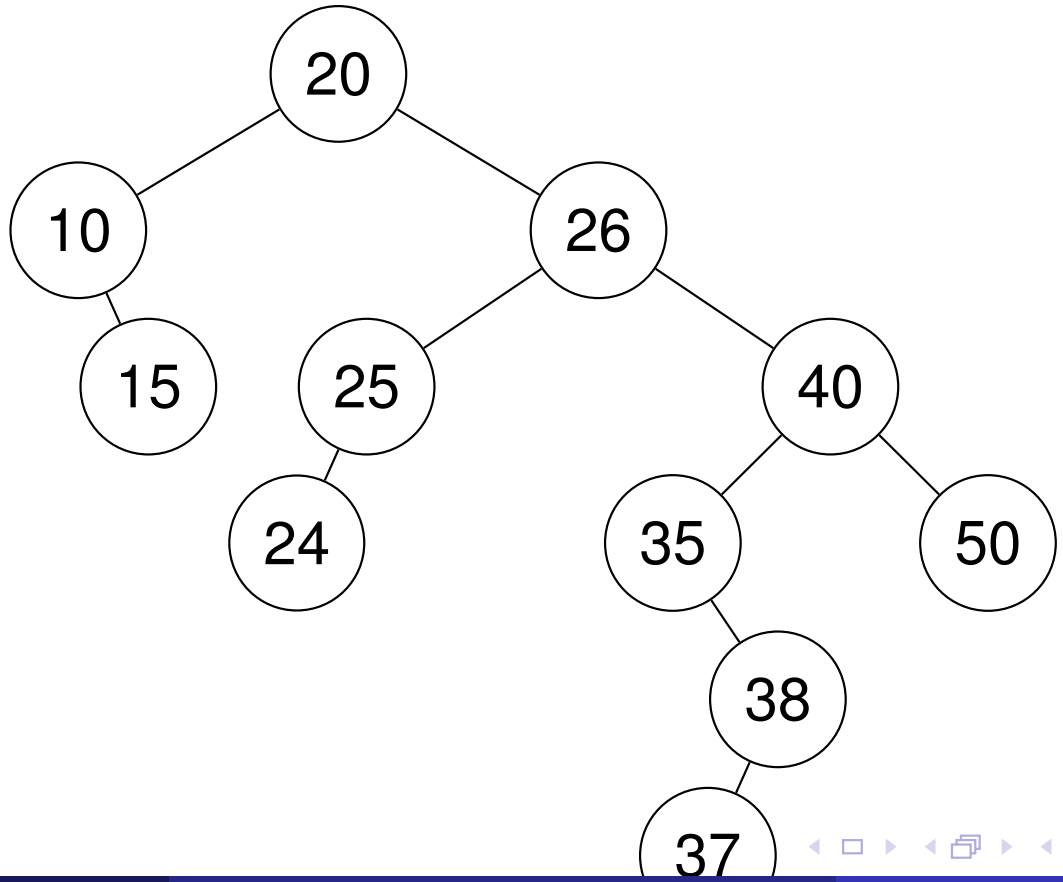


Borrado

Ejercicio

Borra los siguientes elementos del árbol que se muestra a continuación: 5, 3, 30, 22, 39. Criterio: sustitución por M_i

Solución:



Búsqueda

Pregunta

¿Cuál es la complejidad temporal de insertar un elemento en un árbol binario de búsqueda con n nodos?¿Y de borrar un elemento? (con implementación enlazada)

Búsqueda

Pregunta

¿Cuál es la complejidad temporal de insertar un elemento en un árbol binario de búsqueda con n nodos?¿Y de borrar un elemento? (con implementación enlazada)

- Inserción: $\Omega(1)$; $O(n)$
- Borrado: $\Omega(1)$; $O(n)$

Pregunta

¿Podrías dibujar un ejemplo de árbol para cada caso?

Para practicar

`https://www.cs.usfca.edu/~galles/visualization/BST.html`

Criterio: M_i

Para practicar

- Inserta los elementos 1,2,3,4,5,6,7,8,9 en un árbol binario de búsqueda inicialmente vacío. ¿Cuál es la complejidad de insertar n elementos ordenados ascendentemente?
- Borra los elementos en el árbol anterior en el mismo orden. ¿Cuál es la complejidad de cada borrado? (con implementación enlazada)

Para practicar

- Inserta los elementos 16, 6, 27, 3, 8, 23, 15, 25, 5, 19, 12, 7, 17, 24 (generados aleatoriamente) en un árbol binario de búsqueda inicialmente vacío. ¿Cuál piensas que es la altura promedio de un árbol binario de búsqueda tras n inserciones?
- Borra (M_i) elementos del árbol obtenido en el apartado anterior, en el siguiente orden: 25, 17, 23, 24, 6, 3, 16

Repaso

Pregunta

Contesta verdadero o falso a cada una de estas afirmaciones

- El menor elemento en un árbol binario de búsqueda siempre se encuentra en un nodo hoja
- Al borrar un elemento que se encuentra en un nodo con dos hijos, sustituir el valor a borrar por el menor elemento del hijo izquierdo es una estrategia válida
- El coste temporal, en el peor caso, de insertar un elemento en un árbol binario de búsqueda perfecto (con implementación enlazada) es lineal

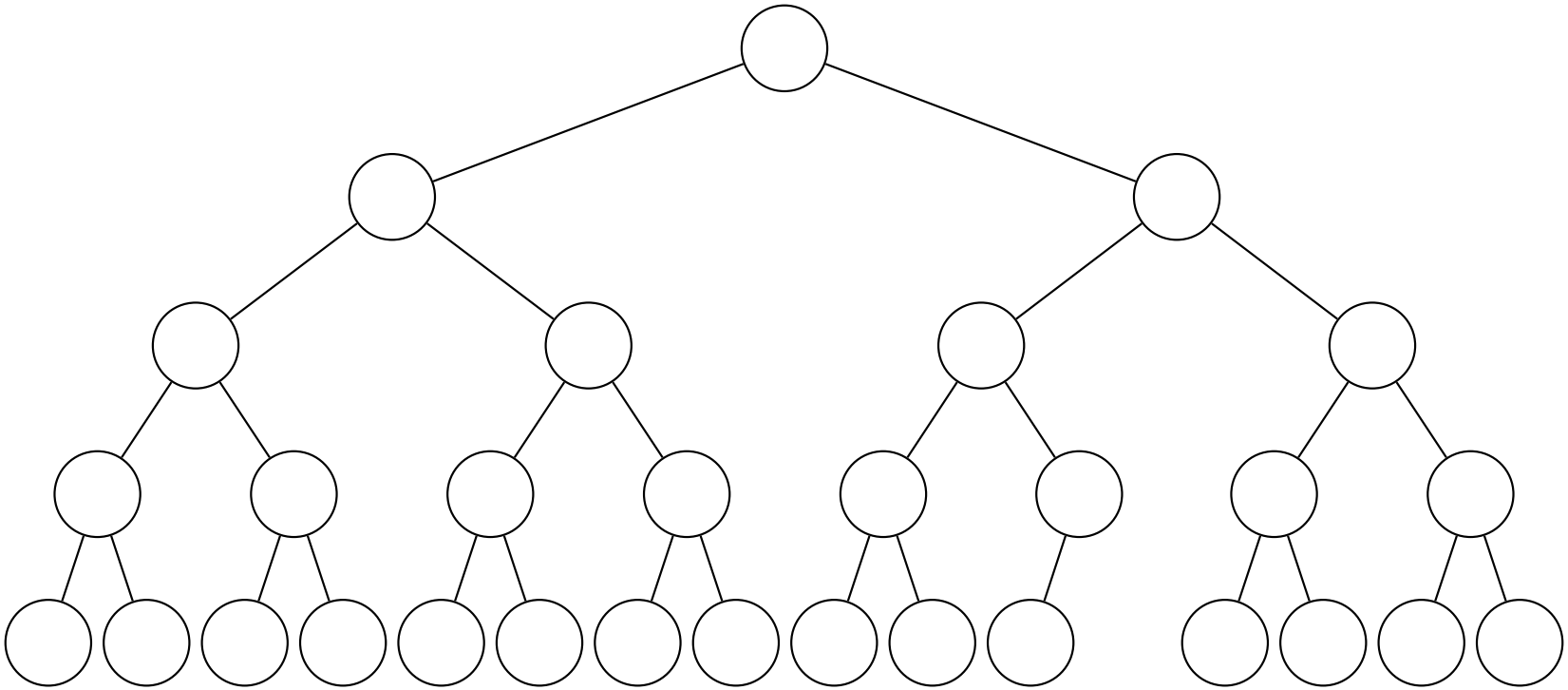
Índice

Árboles AVL: motivación

- El coste de buscar un elemento en un árbol binario de búsqueda en el peor caso es $O(n)$ (árbol degenerado)
- En un árbol de búsqueda perfecto, dicho coste es $O(\log(n))$
- **Idea**: ¿Podemos modificar un ABB tras cada inserción o borrado para que siempre sea lo más parecido posible a un árbol perfecto?
- **Requisito:** Realizar esas modificaciones en tiempo constante respecto al número de elementos almacenados en el árbol
- **Resultado:** Inserciones, búsquedas y borrados en $O(\log(n))$

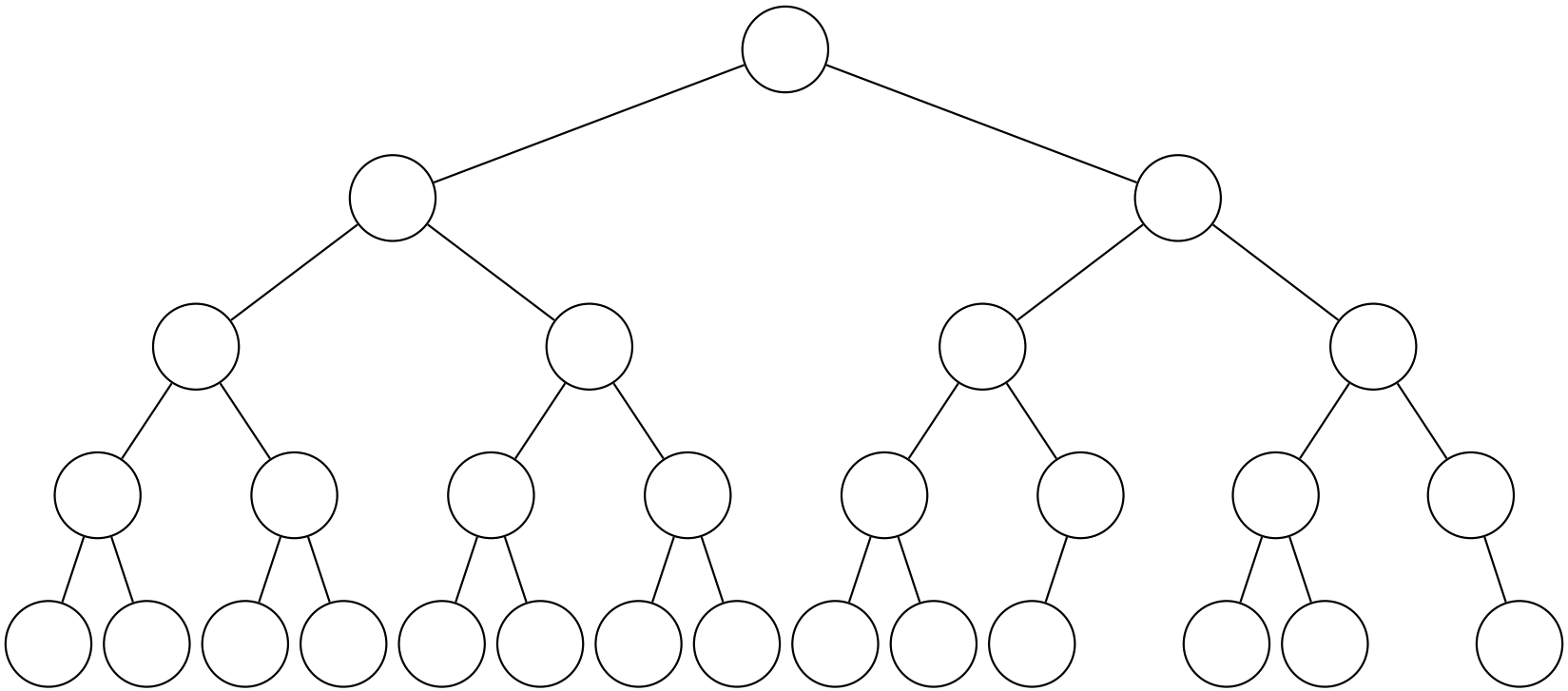
Árboles completamente equilibrados

- Un árbol perfecto, por definición, tiene un número impar de elementos
- Un árbol **completamente equilibrado** es aquel en el que, para todo nodo, el número de nodos de su subárbol izquierdo y de su subárbol derecho difiere como mucho en uno:



Árboles completamente equilibrados

- El siguiente árbol no es completamente equilibrado:



Árboles AVL

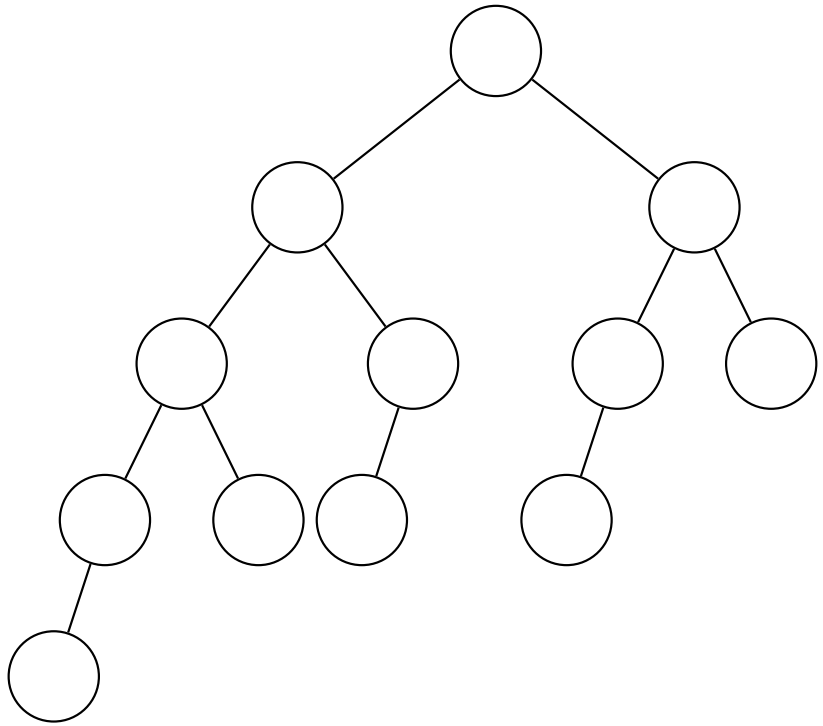
- Los árboles completamente equilibrados requerirían la modificación del árbol tras prácticamente cada inserción o borrado
- **Árboles AVL**: árboles equilibrados respecto a la altura, desarrollados por Adelson-Velskii y Landis en 1962

Definición

Un árbol está equilibrado respecto a la altura si y sólo si, para cada uno de sus nodos, las **alturas** de sus subárboles izquierdo y derecho difieren como máximo en 1

Pregunta

¿El siguiente árbol es completamente equilibrado? ¿Y equilibrado respecto a la altura?



Árboles AVL

- Un árbol equilibrado respecto a la altura de altura h puede tener menos de $2^h - 1$ nodos
- No obstante, se ha demostrado que un árbol equilibrado respecto a la altura con n nodos tiene como máximo una altura de $2\log(n) + 2$
- La complejidad de la búsqueda en el peor caso es $O(\log(n))$

Factor de equilibrio

Definición

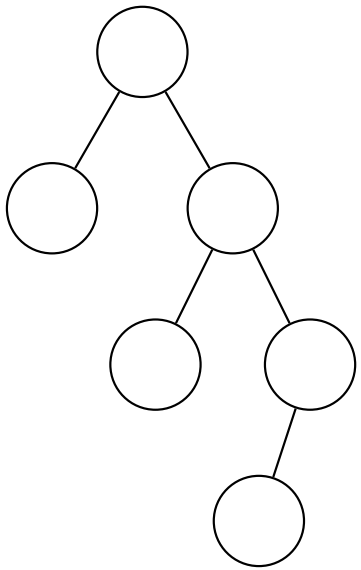
El factor de equilibrio (f_e) de un nodo se define como $h_r - h_l$, siendo h_r la altura de su subárbol derecho y h_l la altura de su subárbol izquierdo

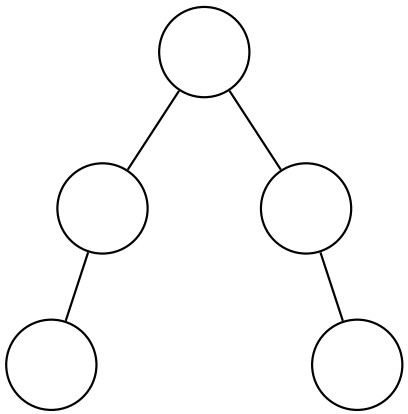
- Un árbol es equilibrado respecto a la altura si y sólo si todos sus nodos se cumple que $f_e \in \{-1, 0, 1\}$

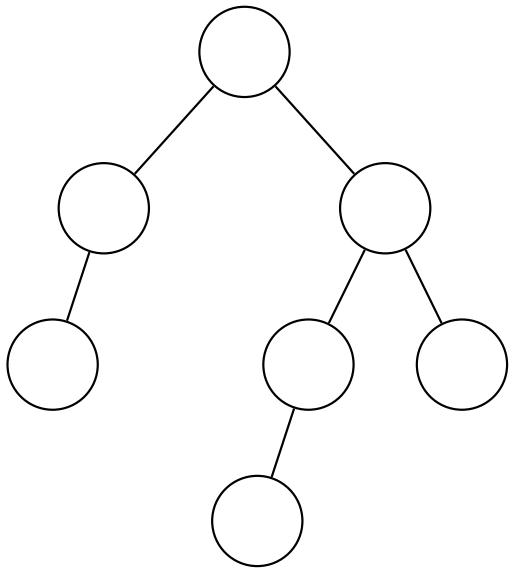
Factor de equilibrio

Pregunta

Calcula el factor de equilibrio de cada nodo para cada uno de estos árboles e indica si son equilibrados respecto a la altura







Implementación enlazada para árboles AVL

```
class Arbol{
private:
Nodo* root;
....
};

class Nodo{
private:
    Elem data;
    int balanceFactor; //Factor de equilibrio
    Nodo* parent;
    Nodo* left;
    Nodo* right;
...
};
```

Pregunta

¿Piensas que la implementación mediante vector es adecuada para un árbol binario de búsqueda o AVL?

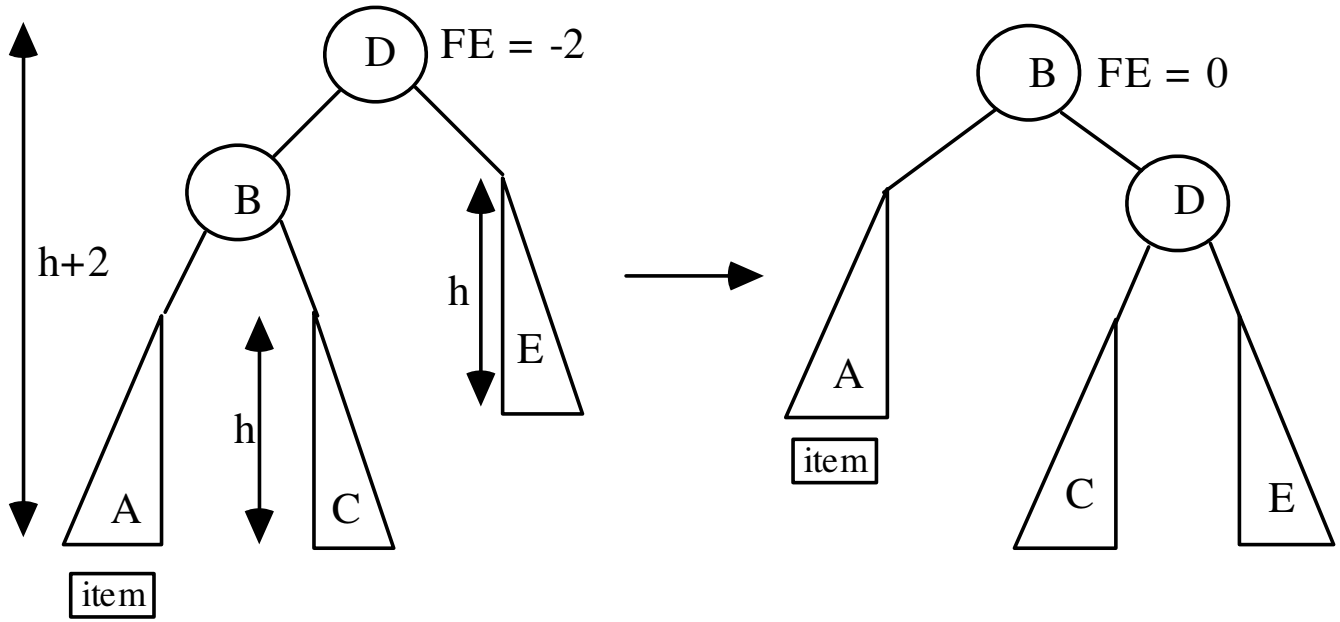
Inserción en árboles AVL

Algoritmo de inserción en árboles AVL:

1. Insertar el elemento siguiendo el algoritmo general de inserción en árboles binarios de búsqueda
2. Actualizar el factor de equilibrio de cada nodo en el camino del nodo insertado a la raíz
3. Tan pronto aparezca un factor de equilibrio 2 o -2, realizar una **rotación**

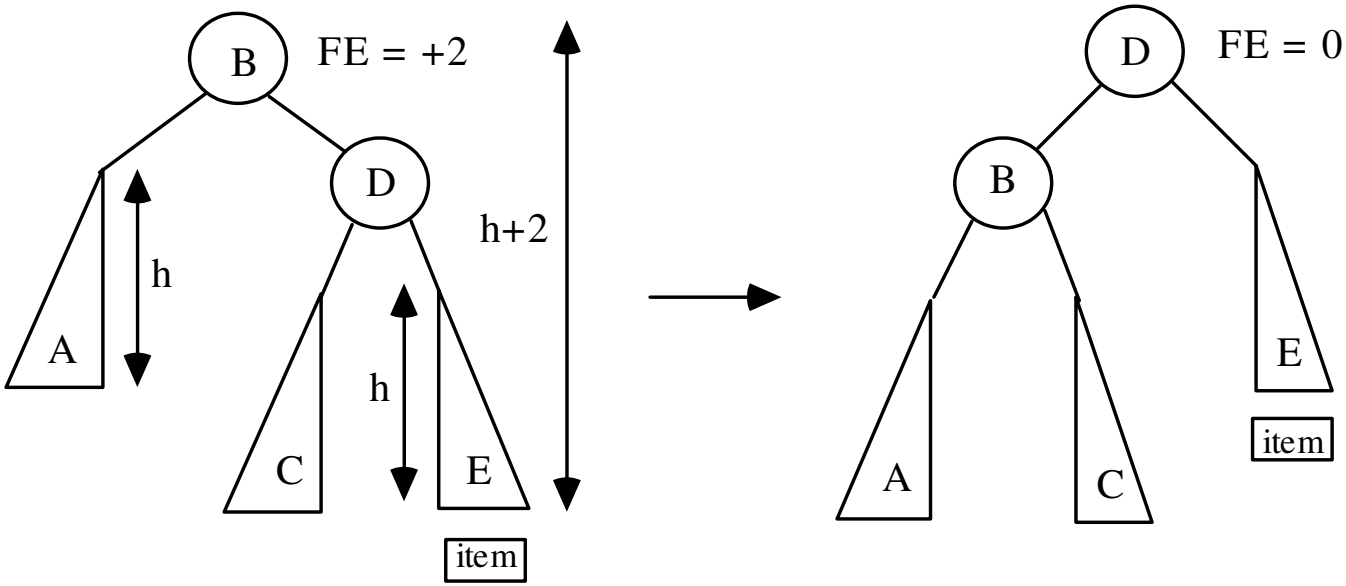
Inserción en árboles AVL

Rotación II: $f_e = -2$; hijo izquierdo con $f_e = -1$



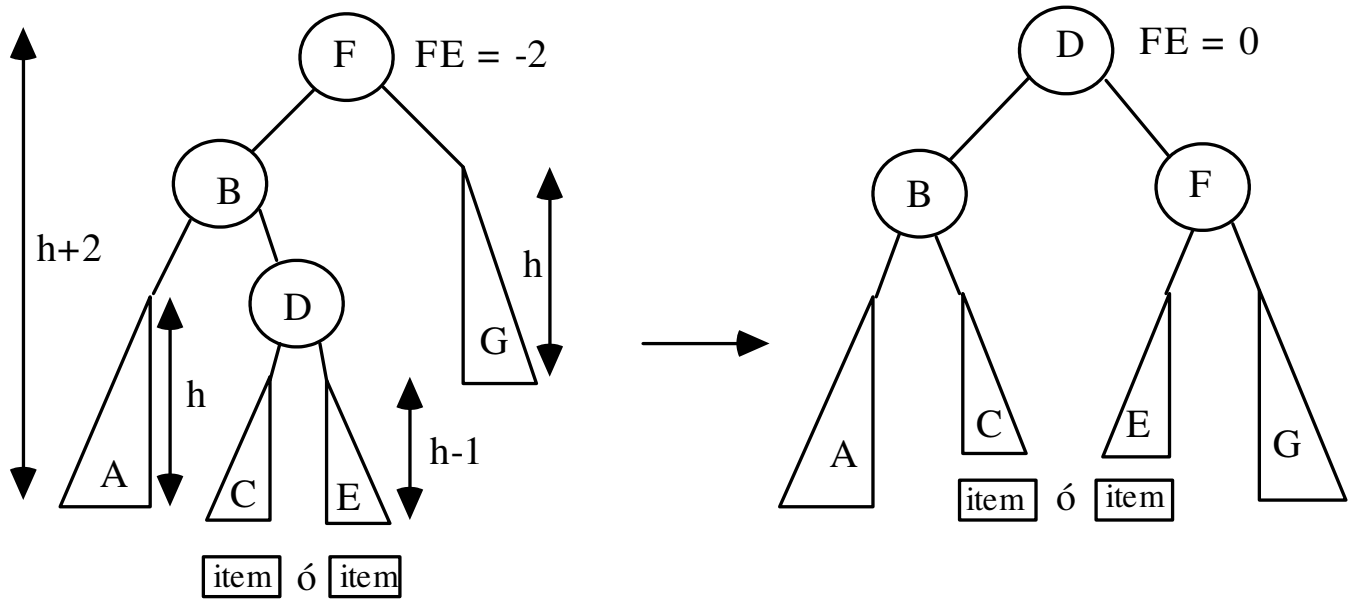
Inserción en árboles AVL

Rotación DD: $f_e = 2$; hijo derecho con $f_e = 1$



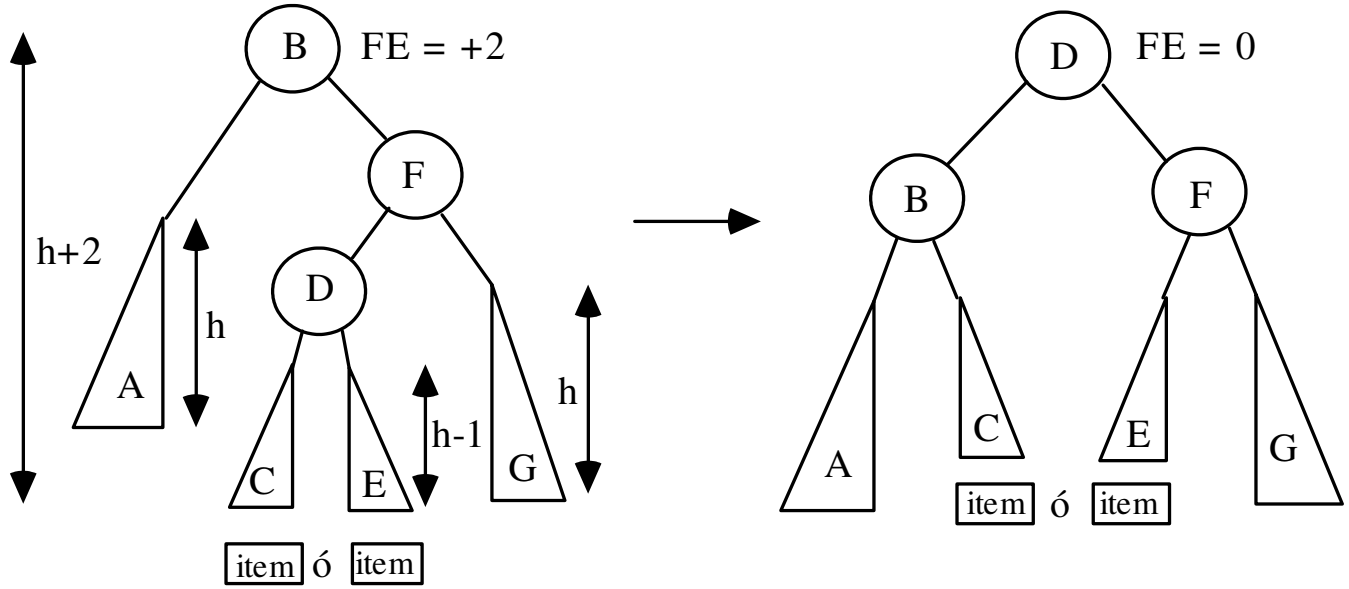
Inserción en árboles AVL

Rotación ID: $f_e = -2$; hijo izquierdo con $f_e = 1$



Inserción en árboles AVL

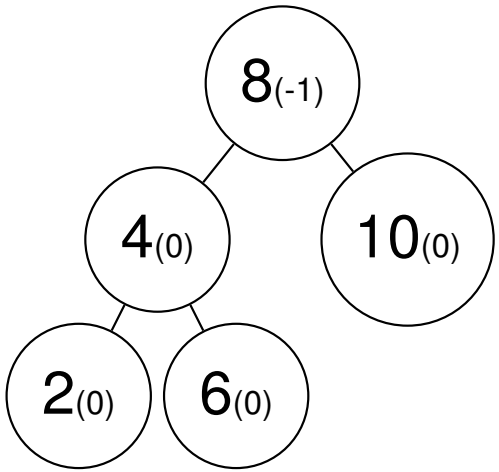
Rotación DI: $f_e = 2$; hijo derecho con $f_e = -1$



Inserción en árboles AVL

Ejemplo

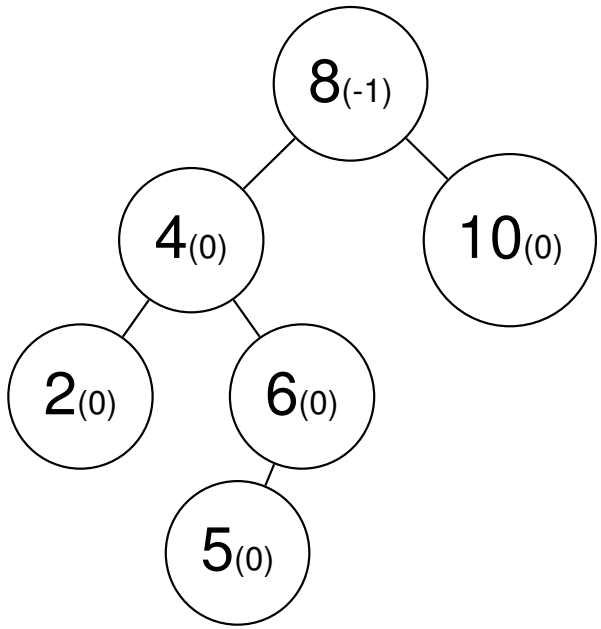
Inserta en el siguiente árbol AVL los elementos 5 y 12



Inserción en árboles AVL

Ejemplo

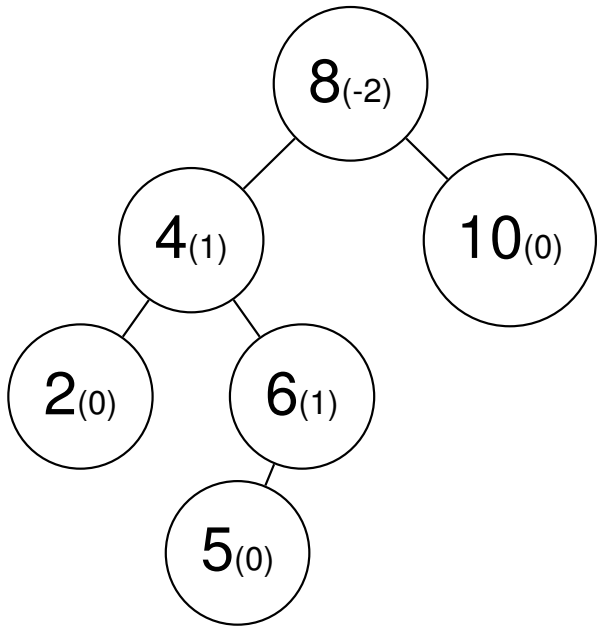
Inserta en el siguiente árbol AVL los elementos 5 y 12



Inserción en árboles AVL

Ejemplo

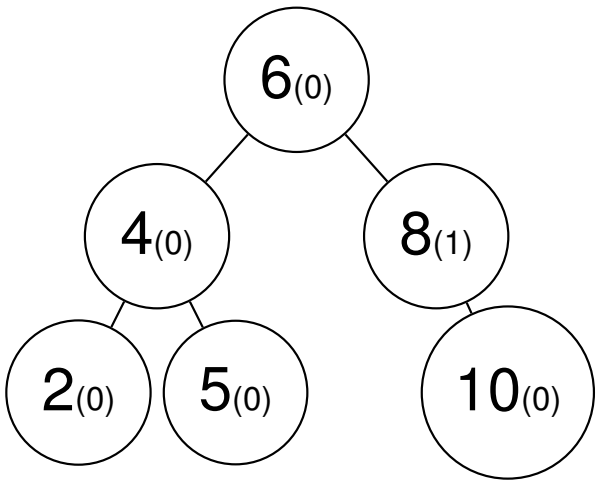
Inserta en el siguiente árbol AVL los elementos 5 y 12



Inserción en árboles AVL

Ejemplo

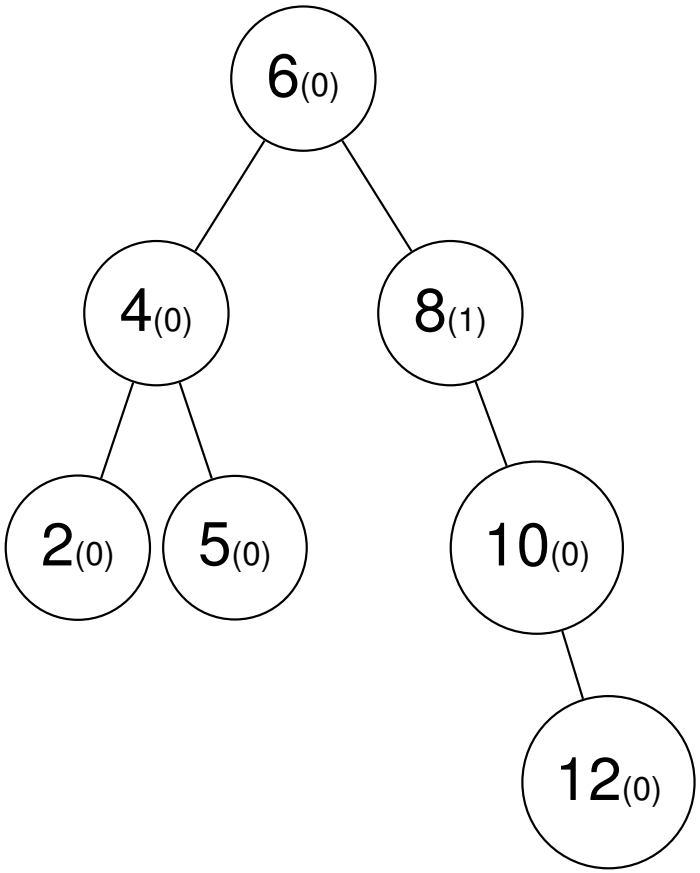
Inserta en el siguiente árbol AVL los elementos 5 y 12



Inserción en árboles AVL

Ejemplo

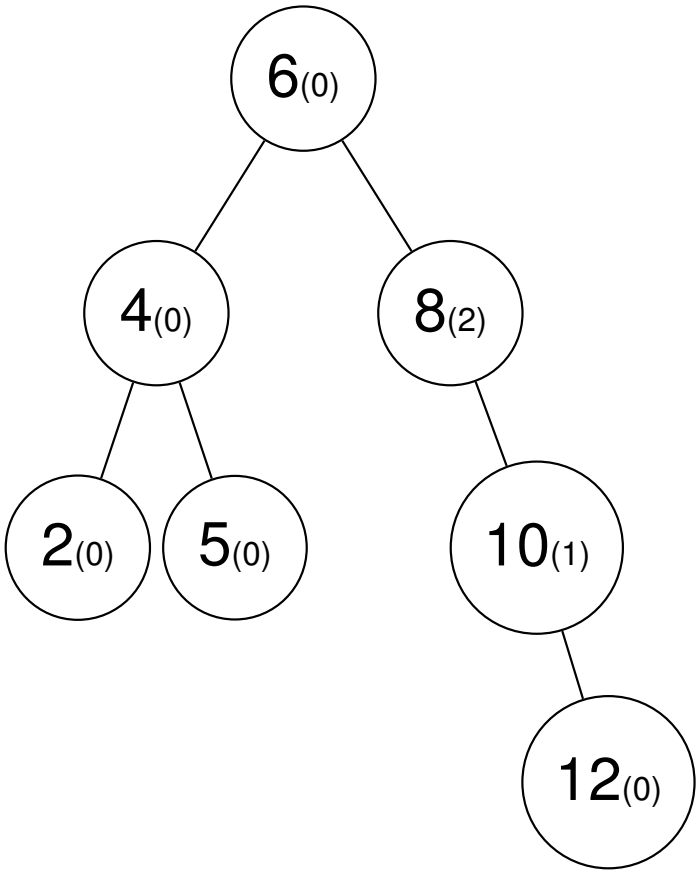
Inserta en el siguiente árbol AVL los elementos 5 y 12



Inserción en árboles AVL

Ejemplo

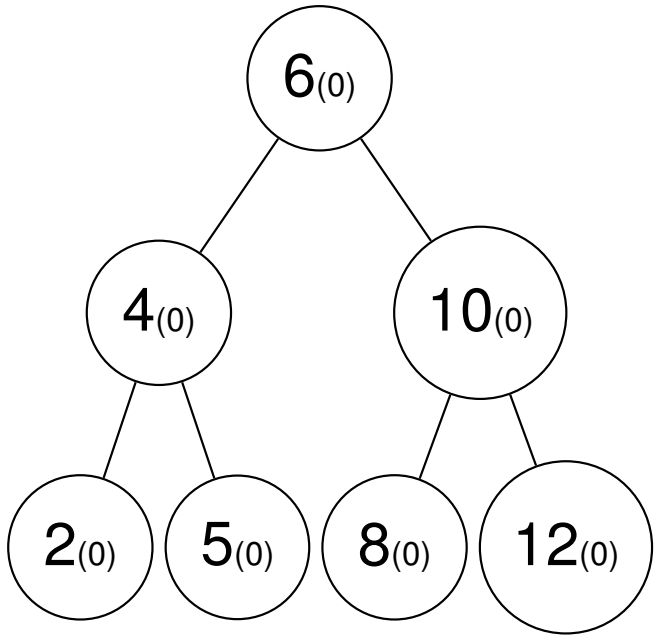
Inserta en el siguiente árbol AVL los elementos 5 y 12



Inserción en árboles AVL

Ejemplo

Inserta en el siguiente árbol AVL los elementos 5 y 12



Inserción en árboles AVL

Ejercicio

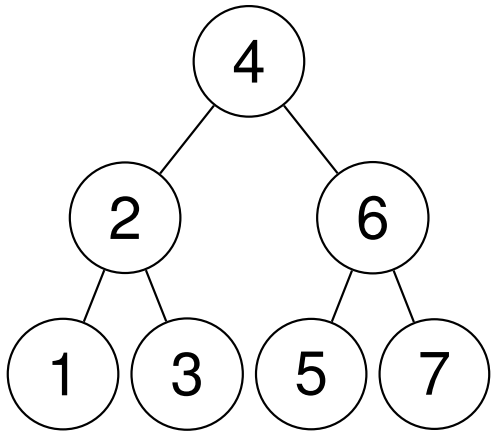
Inserta los siguientes elementos en un árbol AVL inicialmente vacío: 4, 5, 7, 2, 1, 3, 6. Indica los tipos de rotaciones realizadas.

Inserción en árboles AVL

Ejercicio

Inserta los siguientes elementos en un árbol AVL inicialmente vacío: 4, 5, 7, 2, 1, 3, 6. Indica los tipos de rotaciones realizadas.

Solución: DD, II, ID, DI



Inserción en árboles AVL

Ejercicio

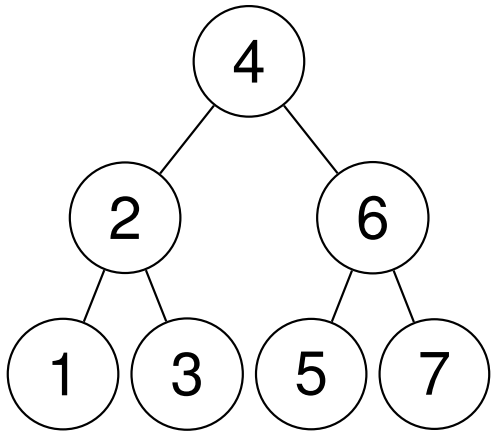
Inserta los siguientes elementos en un árbol AVL inicialmente vacío: 1, 2, 3, 4, 5, 6, 7. Indica los tipos de rotaciones realizadas.

Inserción en árboles AVL

Ejercicio

Inserta los siguientes elementos en un árbol AVL inicialmente vacío: 1, 2, 3, 4, 5, 6, 7. Indica los tipos de rotaciones realizadas.

Solución: DD, DD, DD, DD



Inserción en árboles AVL

Pregunta

¿Cuántas rotaciones como máximo se producen tras una inserción?¿Cuál es la complejidad temporal asintótica de una inserción en un árbol AVL respecto al número de nodos n del árbol?

Inserción en árboles AVL

Pregunta

¿Cuántas rotaciones como máximo se producen tras una inserción?¿Cuál es la complejidad temporal asintótica de una inserción en un árbol AVL respecto al número de nodos n del árbol?

- Una rotación como máximo
- $\Omega(1)$; $O(\log(n))$

Borrado en árboles AVL

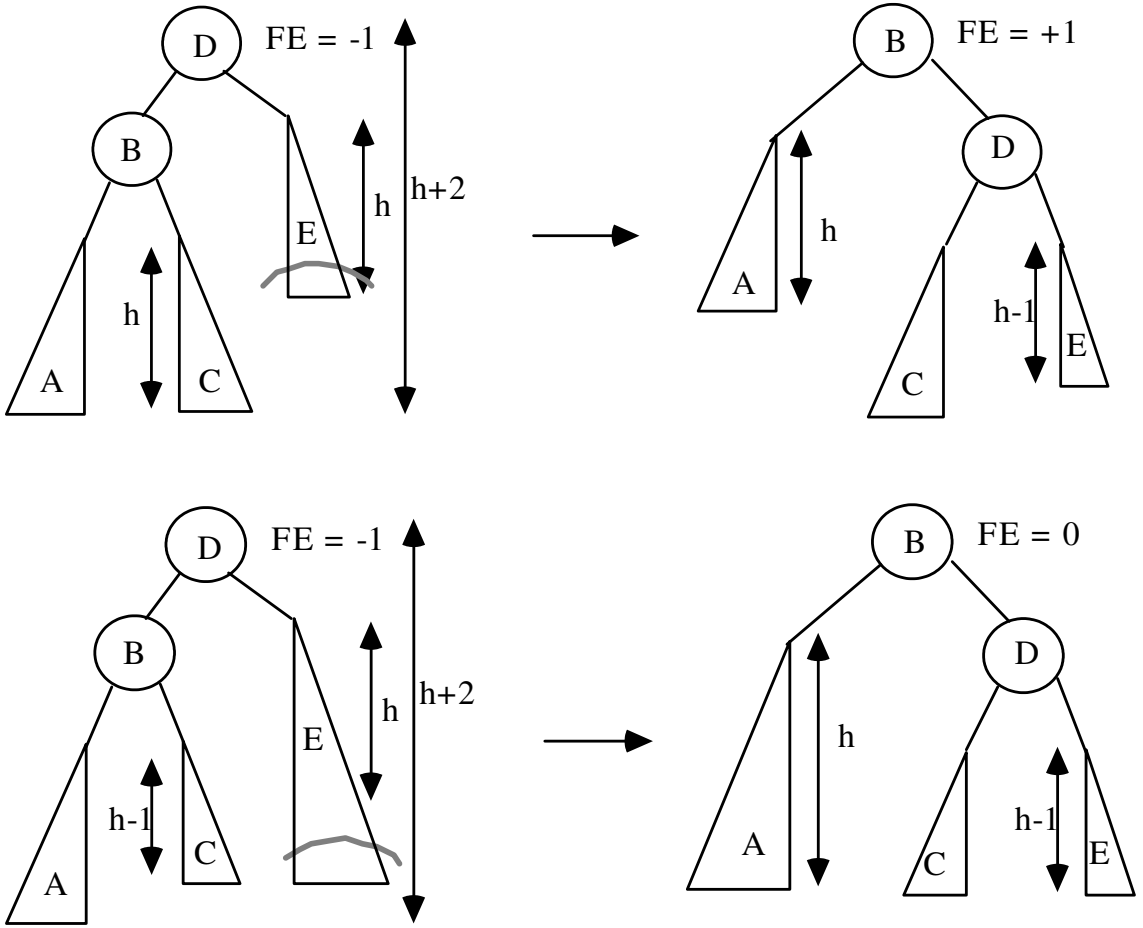
Algoritmo de borrado en árboles AVL:

1. Borrar el elemento siguiendo el algoritmo general de borrado en árboles binarios de búsqueda
2. Actualizar el factor de equilibrio de cada nodo en el camino del nodo borrado[1] a la raíz
3. Tan pronto aparezca un factor de equilibrio 2 o -2, realizar una **rotación**
4. Tras la rotación, continuar actualizando el factor de equilibria hasta la raíz, y repetir el paso 3 tantas veces como sea necesario

[1] Si se ha borrado un elemento en un nodo con dos hijos, hay que empezar a actualizar los factores de equilibrio desde el lugar donde estaban M_i o m_d

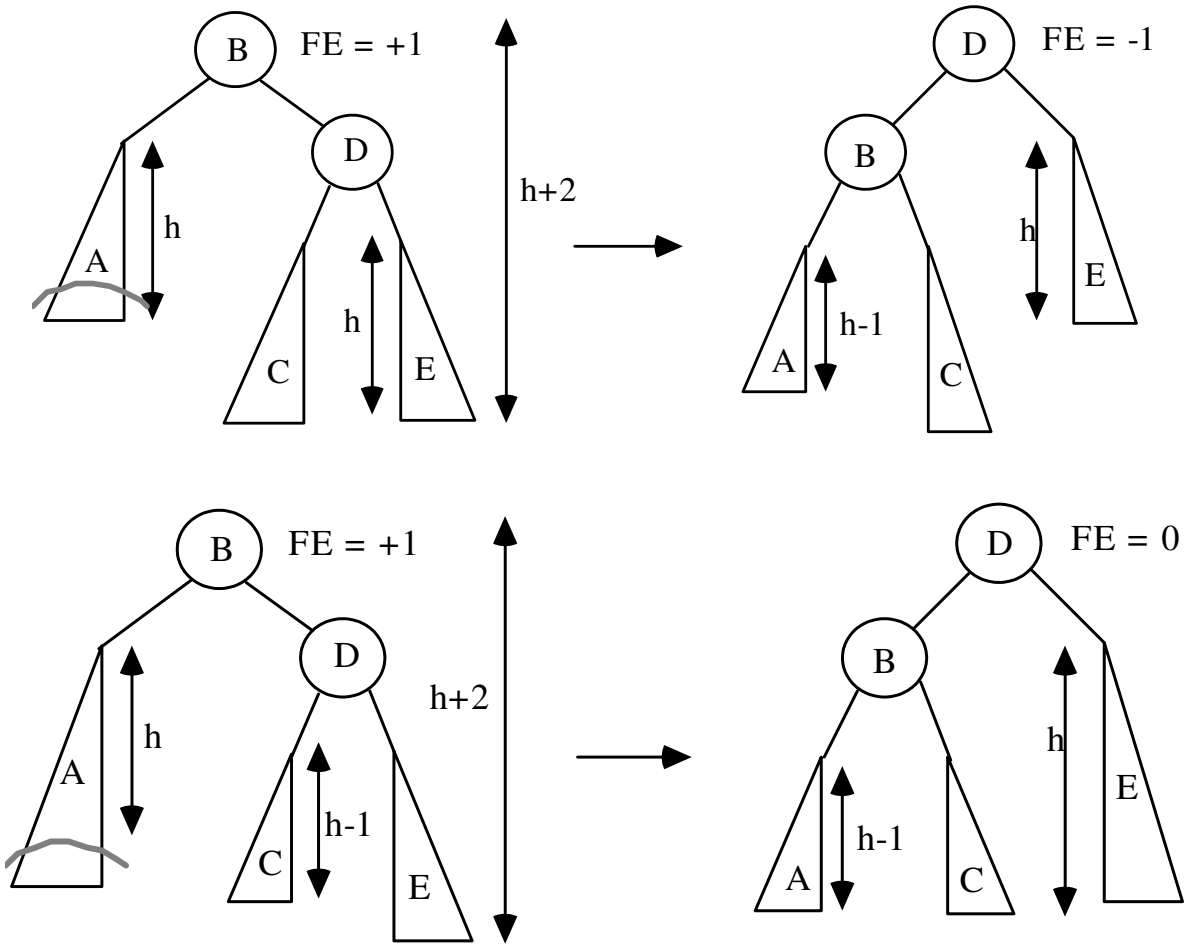
Borrado en árboles AVL

Rotación II: $f_e = -2$; hijo izquierdo con $f_e = 0$ o $f_e = -1$



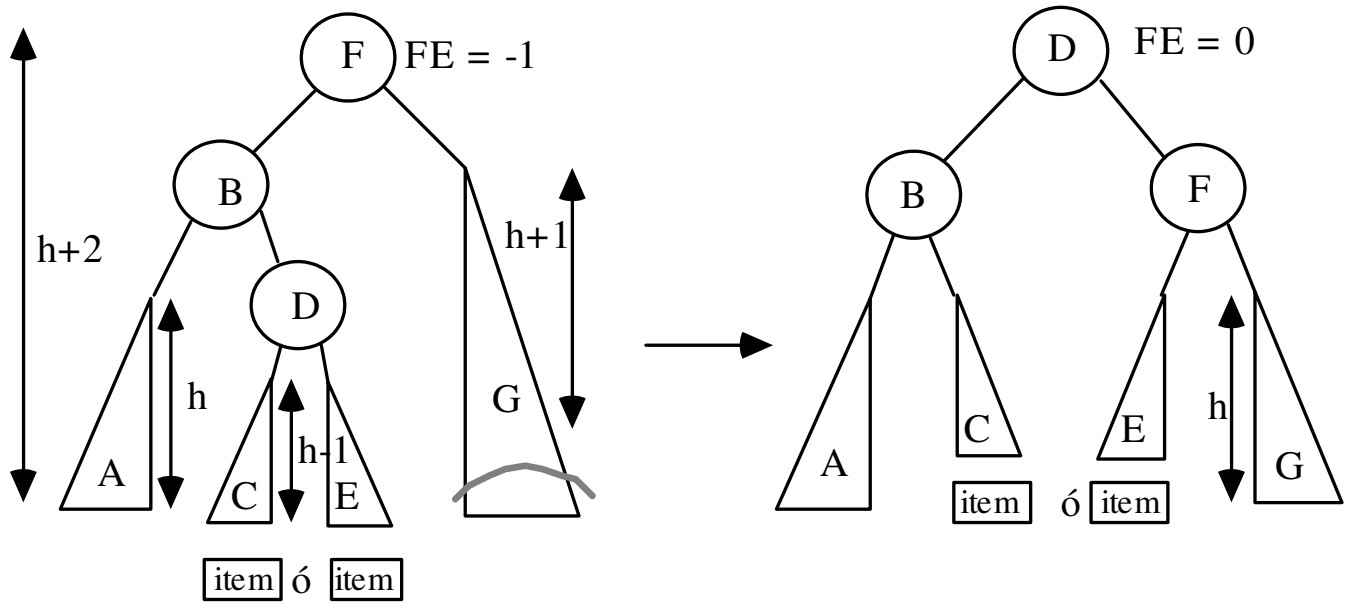
Borrado en árboles AVL

Rotación DD: $f_e = 2$; hijo izquierdo con $f_e = 0$ o $f_e = 1$



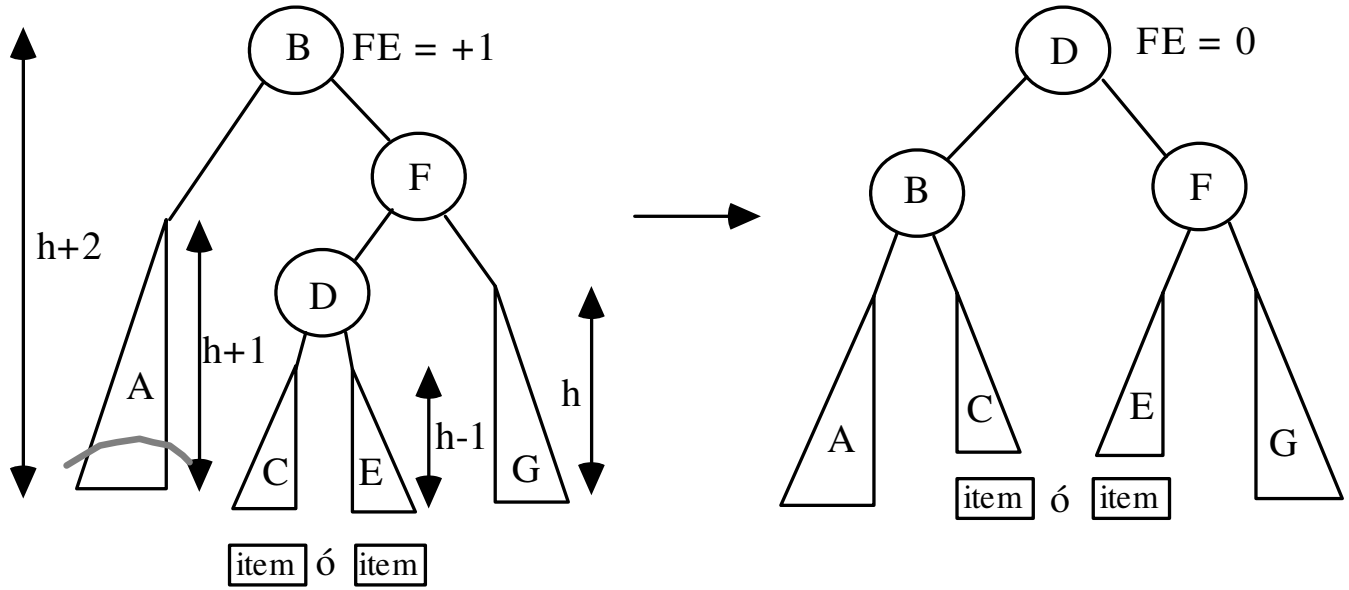
Borrado en árboles AVL

Rotación ID: $f_e = -2$; hijo izquierdo con $f_e = 1$



Borrado en árboles AVL

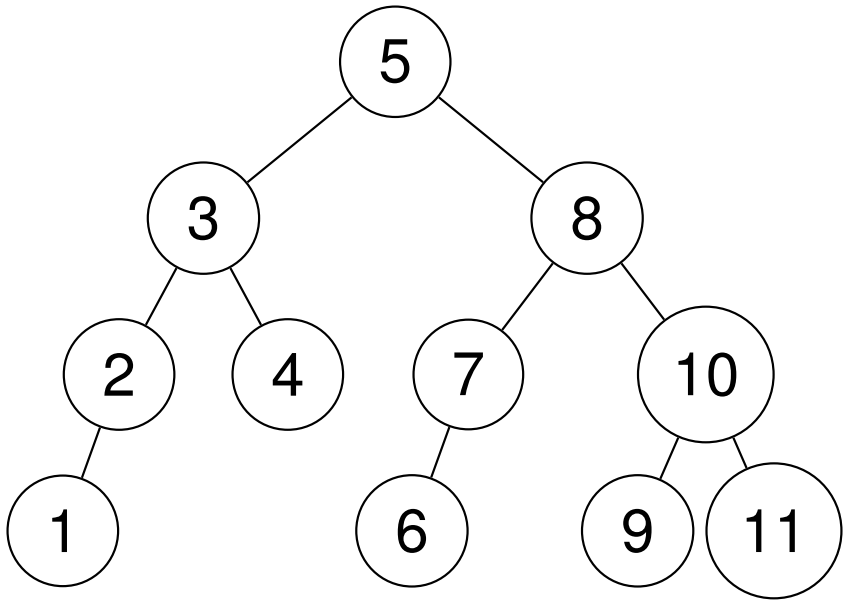
Rotación DI: $f_e = 2$; hijo derecho con $f_e = -1$



Borrado en árboles AVL

Ejercicio

Dado el siguiente árbol AVL, borra los siguientes elementos: 4, 8, 6, 5, 2, 1, 7. Criterio: M_i. Indica los tipos de rotaciones realizadas.

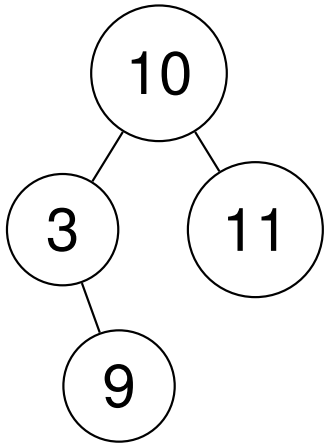


Borrado en árboles AVL

Ejercicio

Dado el siguiente árbol AVL, borra los siguientes elementos: 4, 8, 6, 5, 2, 1, 7. Criterio: M_i. Indica los tipos de rotaciones realizadas.

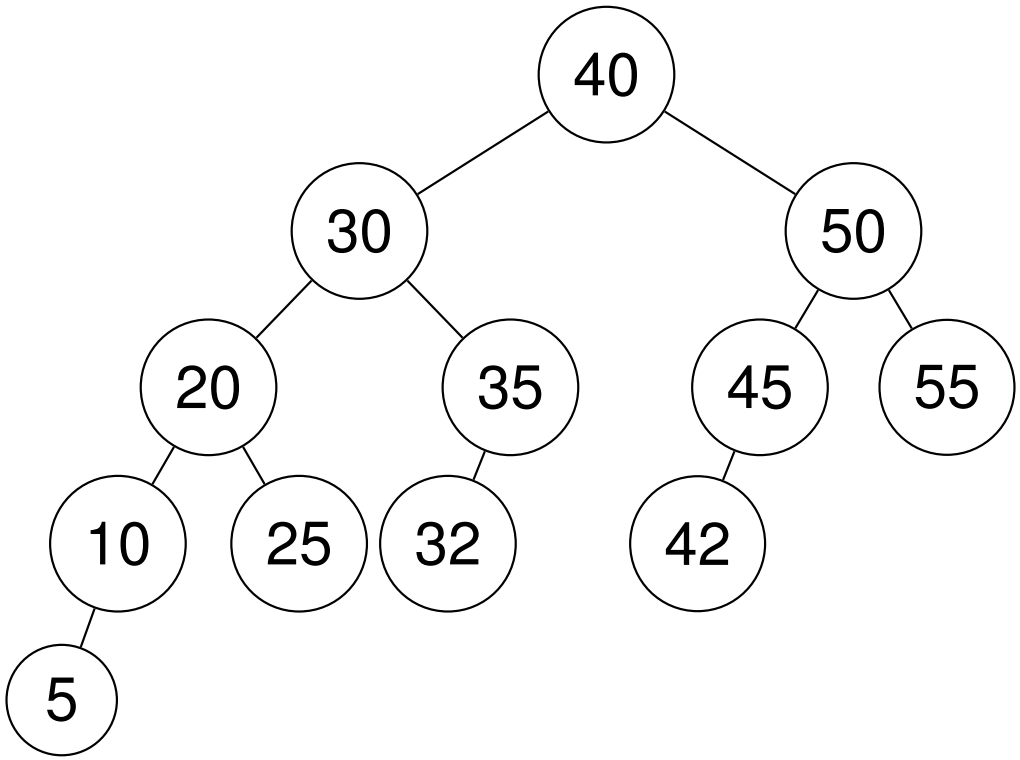
Solución: II, DD, DI, DD



Borrado en árboles AVL

Ejercicio

Dado el siguiente árbol AVL, borra los siguientes elementos: 55, 32, 40, 30. Criterio: M_i. Indica los tipos de rotaciones realizadas.

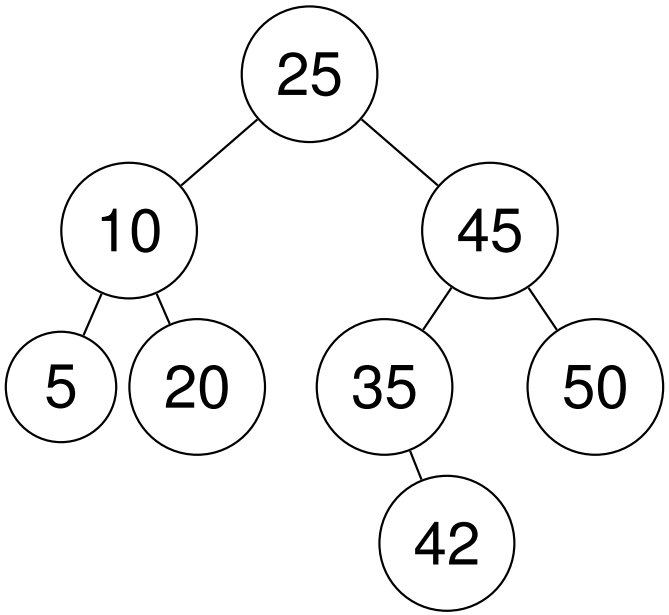


Borrado en árboles AVL

Ejercicio

Dado el siguiente árbol AVL, borra los siguientes elementos: 55, 32, 40, 30. Criterio: M_i. Indica los tipos de rotaciones realizadas.

Solución: II, II, DD, II



Borrado en árboles AVL

Pregunta

¿Cuántas rotaciones como máximo se producen tras un borrado?¿Cuál es la complejidad temporal asintótica de una borrado en un árbol AVL respecto al número de nodos n del árbol?

Borrado en árboles AVL

Pregunta

¿Cuántas rotaciones como máximo se producen tras un borrado?¿Cuál es la complejidad temporal asintótica de una borrado en un árbol AVL respecto al número de nodos n del árbol?

- Como máximo tantas rotaciones como nodos en el camino de búsqueda hasta el elemento a ser borrado
- $\Omega(1)$; $O(\log(n))$