

Árboles binarios de búsqueda.

Guillermo Palma

Universidad Simón Bolívar
Departamento de Computación y T.I.

CI-2612: Algoritmos y Estructuras II



Plan

- 1 Representación de árboles enraizados
- 2 Árboles binarios de búsqueda (ABB)
 - Características de los ABB
 - Consultando los ABB
 - Inserción en ABB
- 3 Eliminación en un ABB



Árboles enraizados

- Estructura de datos enlazadas donde cada uno de los nodos es un objeto enlazado a otro objeto
- El objeto de nodo se representa con varios campos:
 - ▶ Un campo para la clave
 - ▶ Un campo para el valor que se quiere almacenar
 - ▶ Un apuntador al padre
 - ▶ Un apuntador al hijo izquierdo
 - ▶ Un apuntador al hijo derecho
 - ▶ Posiblemente apuntadores a otros nodos en caso de no ser árboles binarios
- El número de ramificaciones puede ser variable



Ejemplo de un nodo de los árboles enraizados

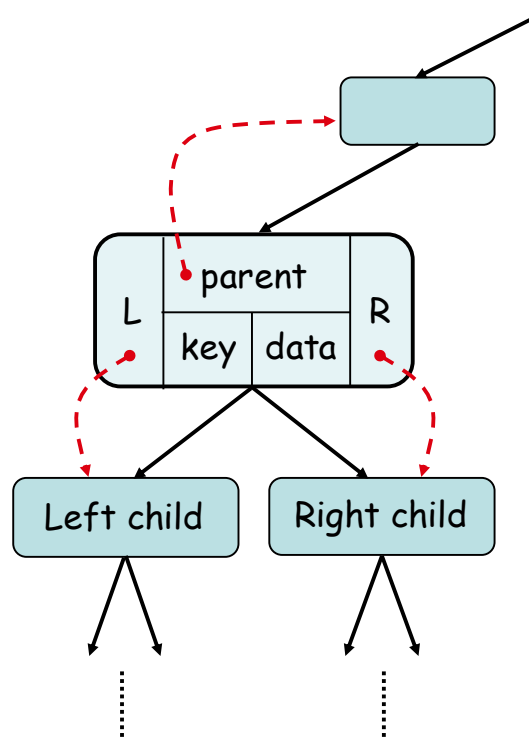


Figura: Fuente [1]



Ejemplo de un árbol binario

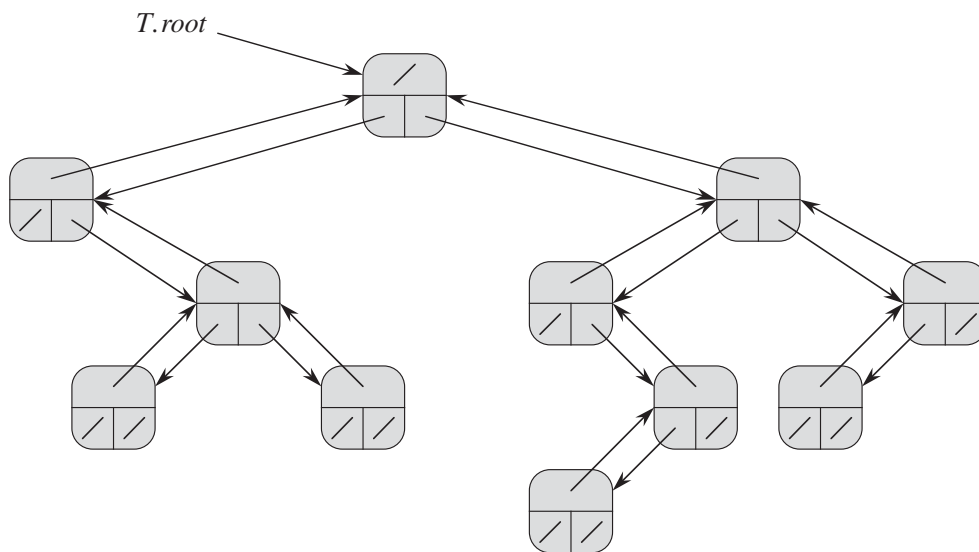


Figura: Fuente [1]



Ejemplo de un árbol sin restricciones en el número de ramificaciones

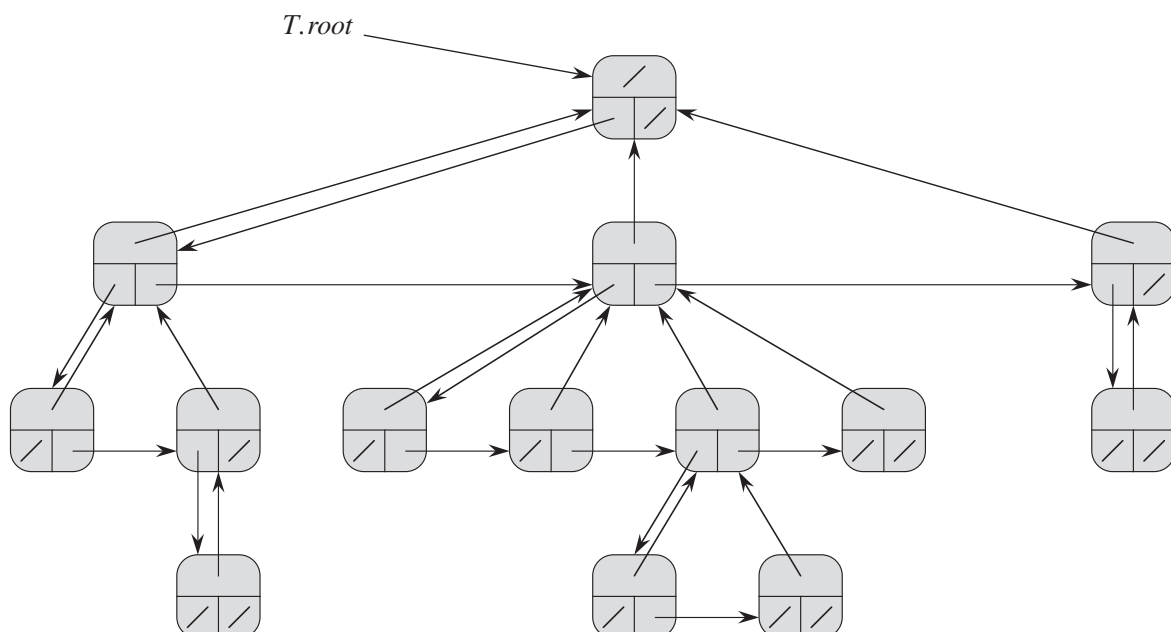


Figura: Fuente [1]



Propiedad de los ABB

Sea x un nodo cualquiera del ABB, se tiene que:

- Si y es un subárbol izquierdo de x , entonces $y.key \leq x.key$
- Si y es un subárbol derecho de x , entonces $y.key \geq x.key$



Ejemplo de un ABB

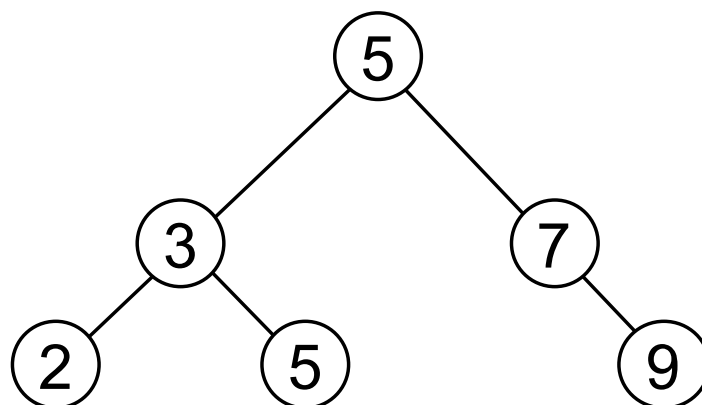


Figura: Fuente [1]



Sobre los ABB

- Soportan varias operaciones del TAD Conjunto, tales como Búsqueda, Mínimo, Máximo, Sucesor, predecesor, insertar, eliminar
- La altura esperada de un ABB con n elementos es $\log n$
- En promedio las operaciones de los ABB se ejecutan en $\Theta(\log n)$
- El peor caso de las operaciones de los ABB ocurre cuando el árbol está totalmente desbalanceado, es decir, es una lista enlazada .
- El tiempo en el peor caso de las operaciones de los ABB es $O(n)$



Recorriendo los ABB

- Recorrido Inorder: el valor de la raíz se muestra entre los valores de los valores del hijo izquierdo y derecho, es decir, se imprime **izquierdo - raíz - derecho**
- Recorrido Preorder: se imprime **raíz - izquierdo - derecho**
- Recorrido Postorder: se imprime **izquierdo - derecho - raíz**



Ejemplo de los recorridos de los ABB

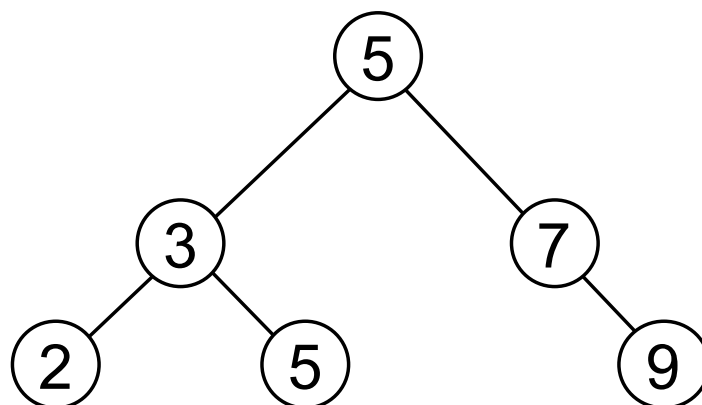


Figura: Fuente [1]

- Recorrido Inorder: 2 3 5 5 7 9
- Recorrido Preorder: 5 3 2 5 7 9
- Recorrido Postorder: 2 5 3 9 7 5



Recorriendo los ABB

Procedimiento Inorder-Tree-Walk(x)

inicio

si $x \neq NIL$ **entonces**

 Inorder-Tree-Walk($x.left$);

 imprimir(x);

 Inorder-Tree-Walk($x.right$);



Recorriendo los ABB

Teorema

Si un árbol con raíz en x tiene n elementos, entonces el recorrido en inorder tiene un tiempo de $\Theta(n)$.



Búsqueda en ABB

- Dada una clave, se retorna el apuntador al node que contiene la clave, o NIL en caso de que la clave no exista
- La idea es comenzar desde la raíz del árbol y se compara la clave del nodo actual.
- Si la clave es menor que la del nodo actual entonces, se busca en subárbol izquierdo, y si es mayor en subárbol derecho.



Ejemplo de búsqueda en un ABB

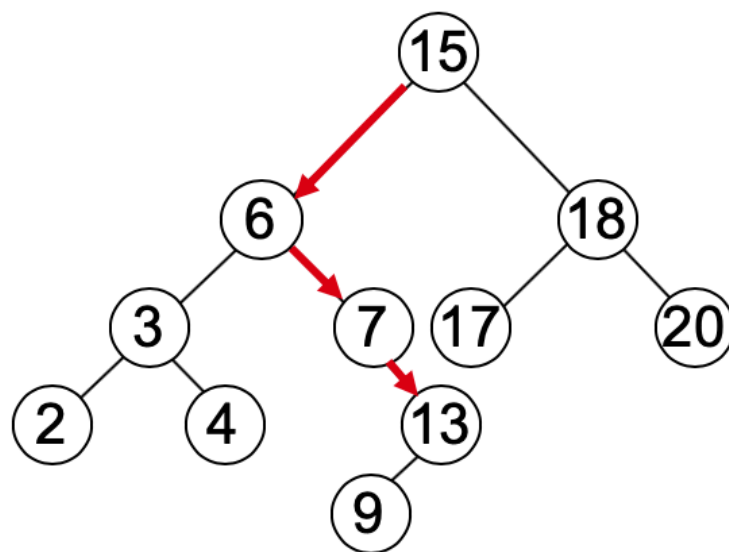


Figura: Ejemplo búsqueda del nodo con clave 13. Fuente [1]



Búsqueda en un ABB, versión recursiva

Función $\text{Tree-Search}(x, k)$

inicio

```

si  $x = NIL \vee k = x.key$  entonces
  | devolver  $x$  ;
si  $k < x.key$  entonces
  | devolver  $\text{Tree-Search}(x.left, k)$  ;
en otro caso
  | devolver  $\text{Tree-Search}(x.right, k)$  ;

```

- Si la altura del ABB es h , el tiempo de la búsqueda es $O(h)$



Búsqueda en un ABB, versión iterativa

Función Iterative-Tree-Search(x, k)

inicio

mientras $x \neq NIL \wedge k \neq x.key$ **hacer**

si $k < x.key$ **entonces**

$x \leftarrow x.left$;

en otro caso

$x \leftarrow x.right$;

devolver x ;



Encontrando la clave mínima de un ABB

Función Tree-Minimum(x)

inicio

mientras $x.left \neq NIL$ **hacer**

$x \leftarrow x.left$;

devolver x ;

- Si la altura del ABB es h , el tiempo de encontrar la clave mínima es $O(h)$



Encontrando la clave mínima de un ABB

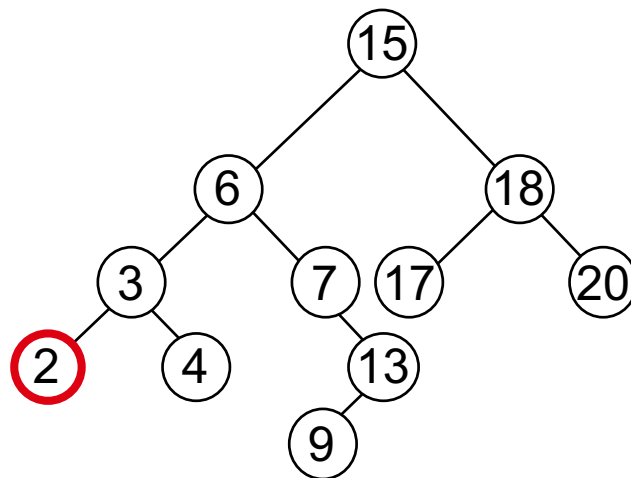


Figura: La clave mínima del árbol es 2. Fuente [1]



Encontrando la clave máxima de un ABB

Función Tree-Maximum(x)

inicio

```
mientras  $x.right \neq NIL$  hacer  
   $x \leftarrow x.right$ ;  
devolver  $x$  ;
```

- Si la altura del ABB es h , el tiempo de encontrar la clave máxima es $O(h)$



Encontrando la clave máxima de un ABB

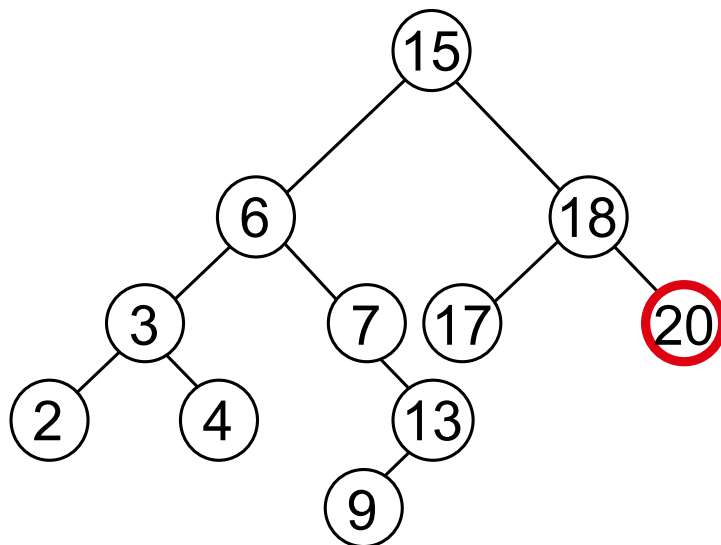


Figura: La clave máxima del árbol es 20. Fuente [1].



Obteniendo el sucesor en un ABB

Definición de sucesor

Una clave y es sucesor de la clave x , si y es la clave más pequeña que es mayor que la clave x .



Obteniendo el sucesor en un ABB

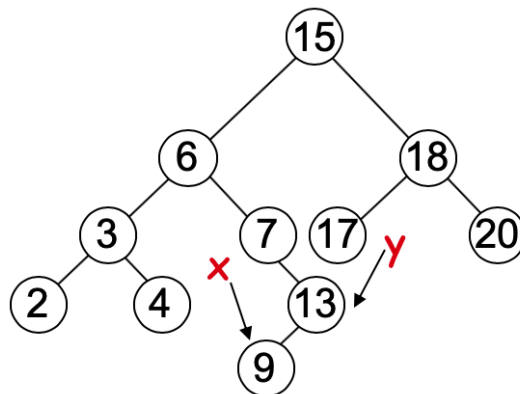


Figura: Se tiene que el sucesor de 15 es 17, que el sucesor de 13 es 15, y que el sucesor de 9 es 13. Fuente [1].



Obteniendo el sucesor en un ABB

Función Tree-Successor(x)

inicio

```

si  $x.right \neq NIL$  entonces
  | devolver Tree-Search ( $x.right$ ) ;
 $y \leftarrow x.p$  ;
mientras  $y \neq NIL \wedge x = y.right$  hacer
  |  $x \leftarrow y$  ;
  |  $y \leftarrow y.p$  ;
devolver  $y$  ;
  
```



Análisis de las operaciones sobre el ABB

Teorema

Si la altura del ABB es h , el tiempo de las operaciones de BÚSQUEDA, MÍNIMO, MÁXIMO, SUCESOR y PREDECESOR son $O(h)$



Inserción en ABB

- Se quiere insertar un nodo con clave k en el ABB
- Se comienza desde la raíz
- Si la clave a insertar es menor al nodo actual se recorre el subárbol izquierdo, de lo contrario el derecho
- Se inserta el nodo cuando se encuentra el elemento NIL



Ejemplo de inserción en un ABB

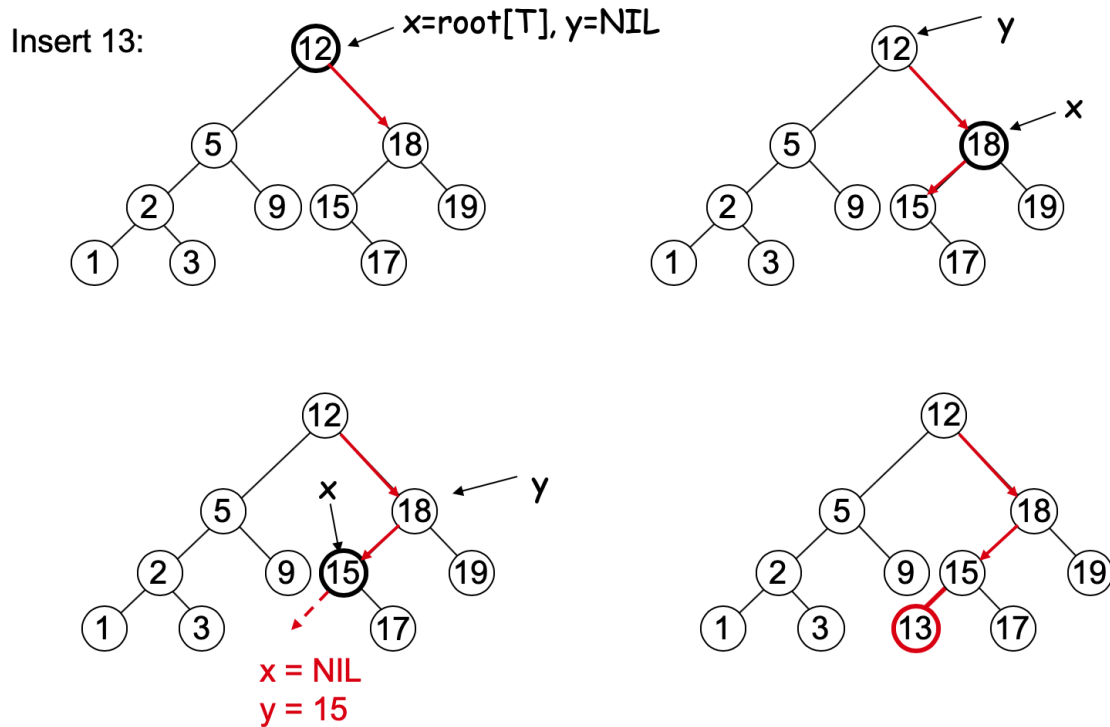


Figura: Fuente [1].



Inserción en un ABB

Procedimiento Tree-Insert(T, z)

inicio

```

 $y \leftarrow \text{NIL};$ 
 $x \leftarrow T.\text{root};$ 
mientras  $x \neq \text{NIL}$  hacer
     $y \leftarrow x;$ 
    si  $z.\text{key} < x.\text{key}$  entonces
         $x \leftarrow x.\text{left};$ 
    en otro caso
         $x \leftarrow x.\text{right};$ 
 $z.p \leftarrow y;$ 
si  $y = \text{NIL}$  entonces  $T.\text{root} \leftarrow z;$ 
si no, si  $z.\text{key} < y.\text{key}$  entonces  $y.\text{left} \leftarrow z;$ 
en otro caso  $y.\text{right} \leftarrow z;$ 
  
```



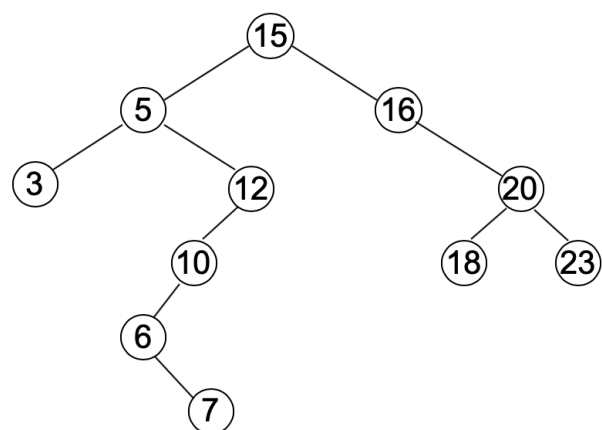
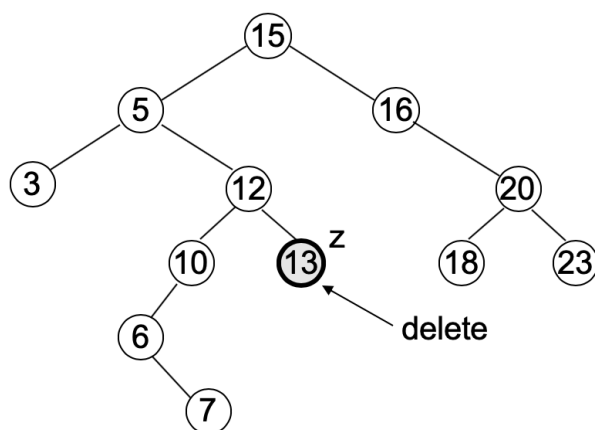
Eliminación en un ABB

- Se quiere eliminar un nodo con clave z en el ABB
- Suponiendo que la clave z está en el ABB, hay tres casos posibles
 - ▶ Caso 1: z no tiene hijos
 - ▶ Caso 2: z tiene un solo hijo izquierdo o derecho
 - ▶ Caso 3: z tiene dos hijos



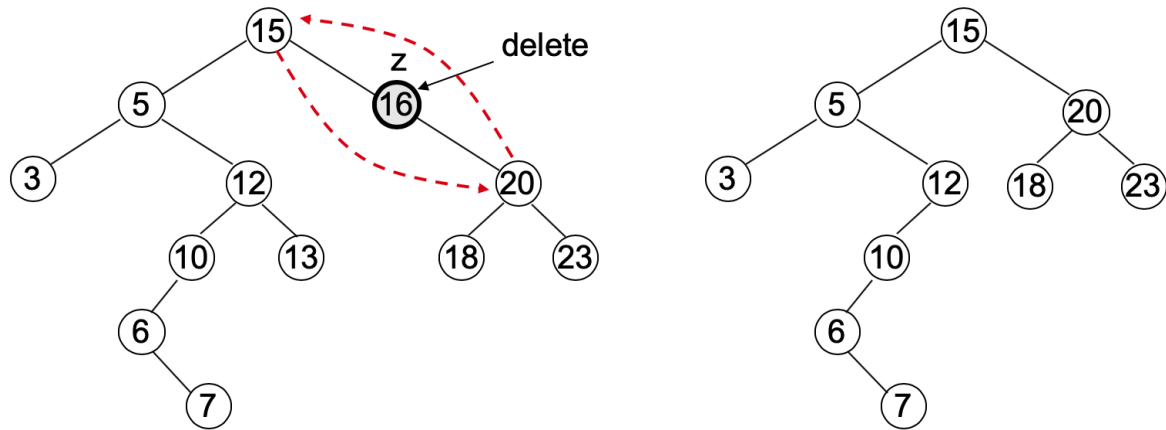
Eliminación en un ABB

- Caso 1: z no tiene hijos
 - ▶ Se elimina z haciendo que el padre de z apunte a NIL.



Eliminación en un ABB

- Caso 2: z tiene un solo hijo izquierdo o derecho
 - ▶ Se elimina z haciendo que el padre de z apunte al hijo de z.

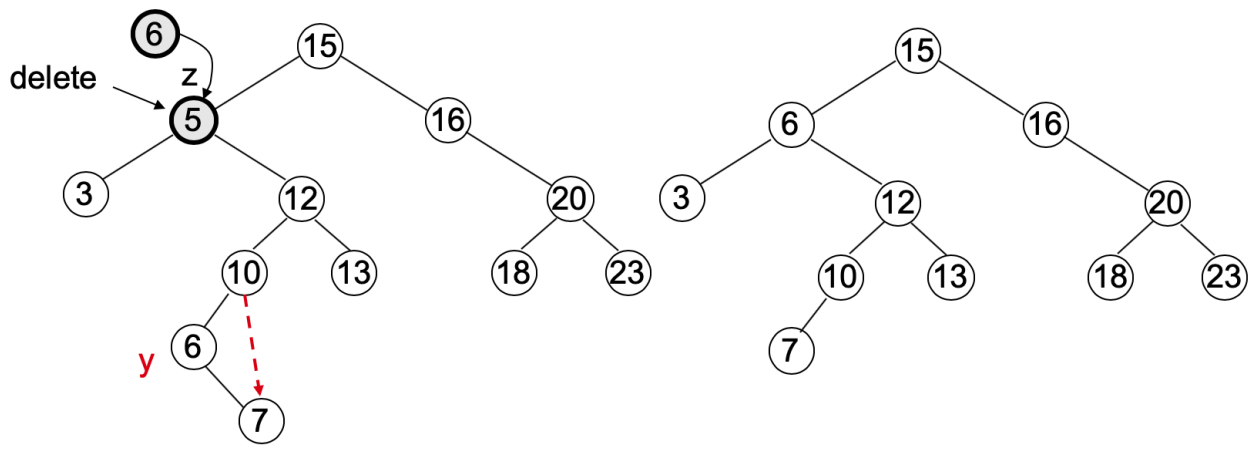


Eliminación en un ABB

- Caso 3: z tiene dos hijos
 - ▶ Se encuentra el sucesor y de z en el subárbol derecho
 - ▶ Se tiene que y clave mínima en el subárbol derecho
 - ▶ Se tiene que y o no tiene hijos o tiene un hijo derecho. No tienen hijo izquierdo
 - ▶ Se elimina y del árbol aplicando el caso 1 o el caso 2
 - ▶ Se reemplaza la clave y valor del nodo con clave z por la clave y el valor del nodo con clave y



Eliminación en un ABB, caso 3



Eliminación en un ABB

Procedimiento Tree-Delete(T, z)

inicio

```

si  $z.left = NIL \vee z.right = NIL$  entonces  $y \leftarrow z$  ;
en otro caso  $y \leftarrow \text{Tree-successor}(z)$  ;
si  $y.left \neq NIL$  entonces  $x \leftarrow y.left$ ;
en otro caso  $x \leftarrow y.right$ ;
si  $x \neq NIL$  entonces  $x.p \leftarrow y.p$  ;
si  $y.p = NIL$  entonces  $T.root \leftarrow x$  ;
en otro caso
    si  $y = y.p.left$  entonces  $y.p.left \leftarrow x$  ;
    en otro caso  $y.p.right \leftarrow x$  ;
si  $y \neq z$  entonces
     $z.key \leftarrow y.key$  ;
     $z.data \leftarrow y.data$ 
  
```



Eliminación en un ABB

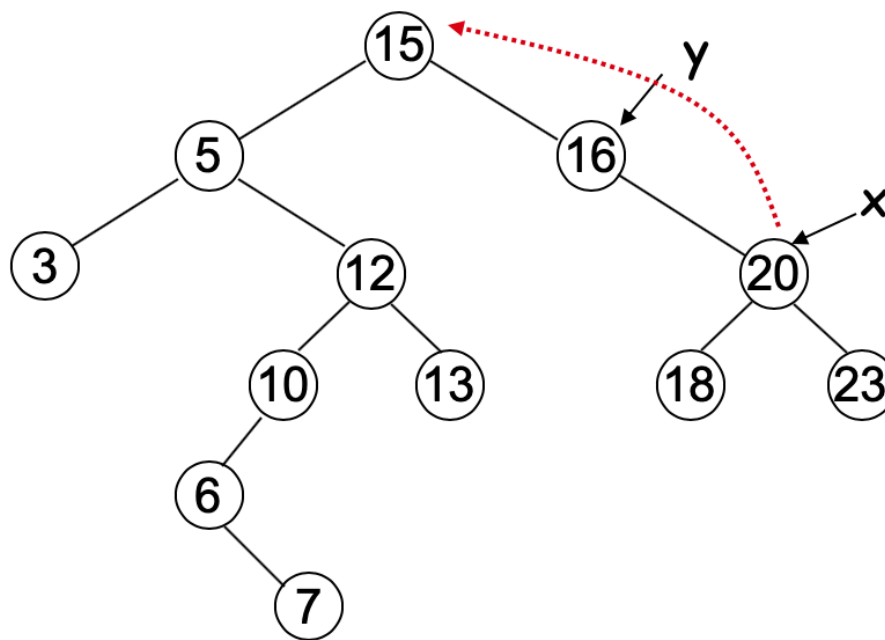


Figura: Resultado de la operación $x.p \leftarrow y.p$



Eliminación en un ABB

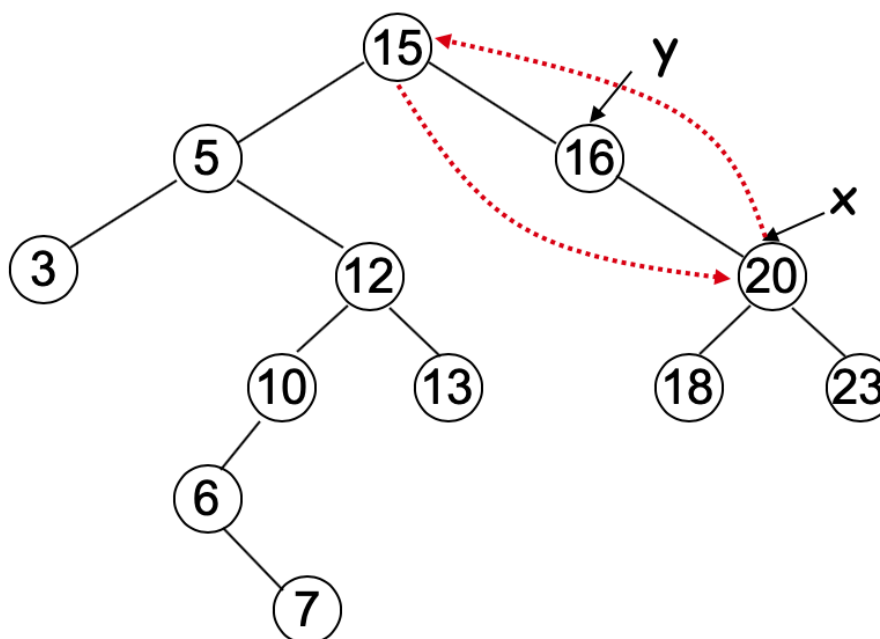


Figura: Resultado de la operación $y.p.right \leftarrow x$



Análisis de las operaciones sobre el ABB

Teorema

Si la altura del ABB es h , el tiempo de las operaciones de INSERCIÓN y ELIMINACIÓN son $O(h)$



Referencias



T. Cormen, C. Leirserson, R. Rivest, and C. Stein.
Introduction to Algorithms.
McGraw Hill, 3ra edition, 2009.

