Escuela de Ingeniería Civil Informática
Facultad de Ingeniería

Estructuras de datos

Capítulo III: Árboles

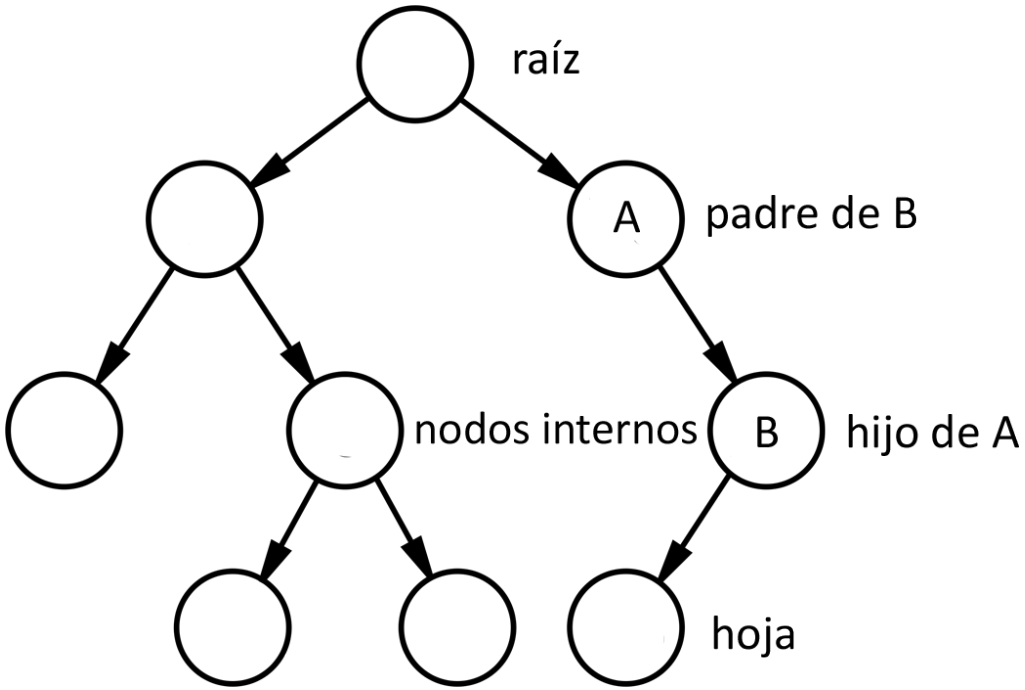
Fabián Riquelme Csori
fabian.riquelme@uv.cl

2017-II

Index

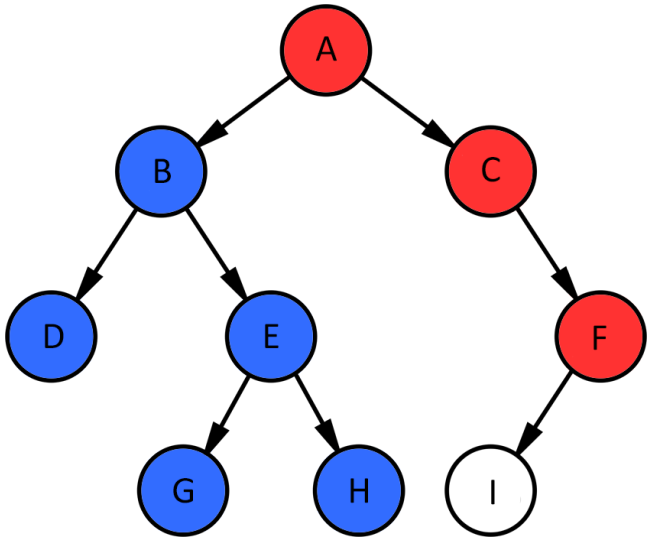
Definición general

- Un grafo es un par $G = (V, E)$ donde V es un conjunto de nodos y E un conjunto de aristas.
- Un árbol es un grafo conexo, acíclico y no dirigido (aunque se puede dibujar con aristas "hacia abajo").

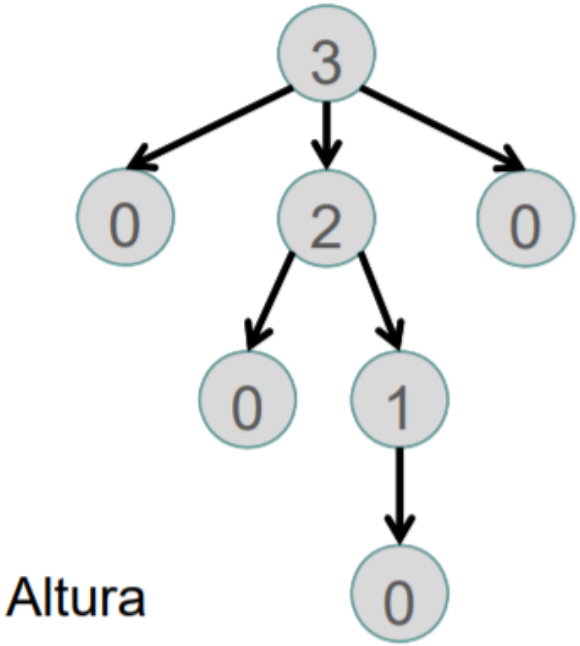


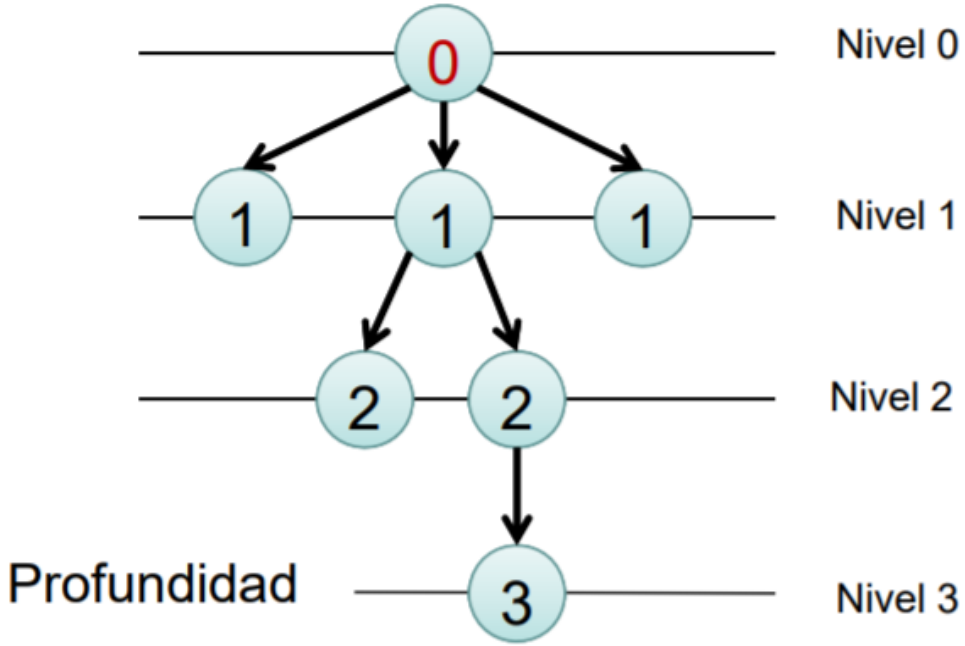
Conceptos básicos

- **Descendientes** de B: {B,D,E,G,H} (conforman un **subárbol**)
- **Ascendientes** de F: {A,C,F}
- El **camino** A-H (secuencia de nodos A-B-E-H) tiene **largo** 3.



Conceptos básicos

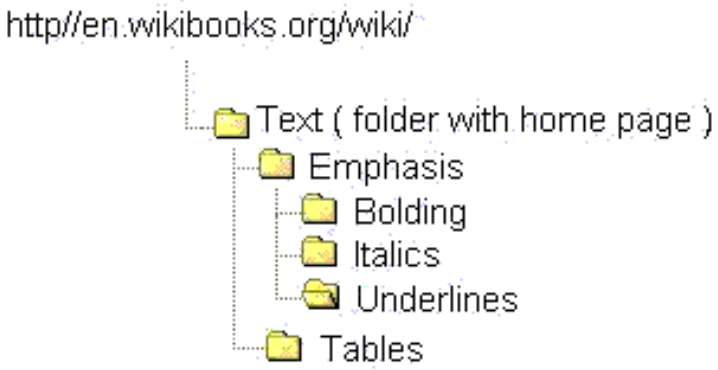




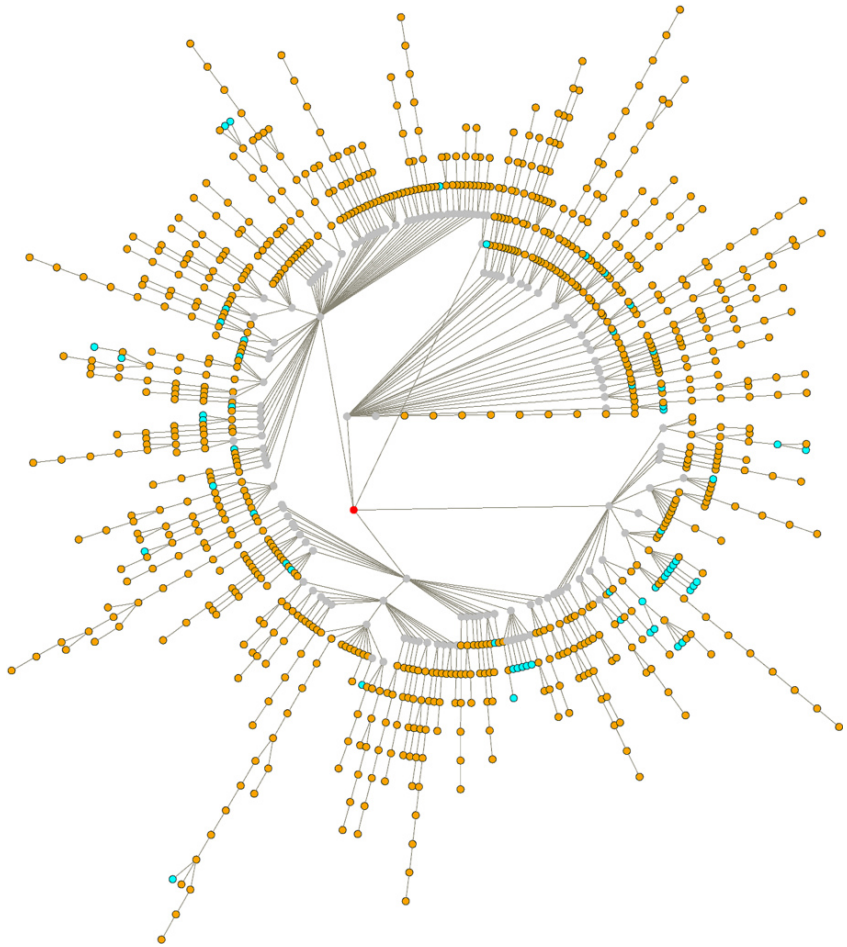
César Vaca Rodríguez, Dpto. de Informática, UValladolid

- ¿Cuál es la altura de una hoja? ¿del árbol? ¿de un nodo interno?
- ¿Cuál es la profundidad de la raíz? ¿de un nodo interno?

Las aplicaciones son muchísimas...



Sistema de organización de carpetas.

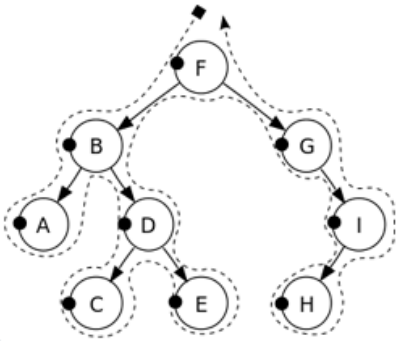


Discusión *en:Presidency of Barack Obama*, Wikipedia.

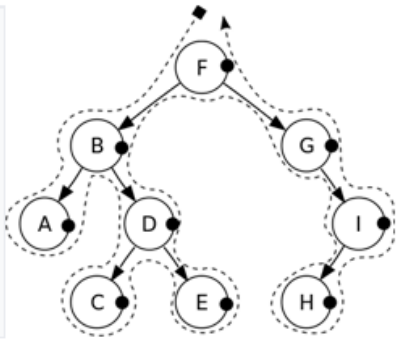
Recorrido de árboles

- Búsqueda en profundidad o Depth-first search (recursiva)
- Búsqueda en anchura o Breadth-first search (iterativa)

```
preorder(node)
  if (node = null)
    return
  visit(node)
  preorder(node.left)
  preorder(node.right)
```

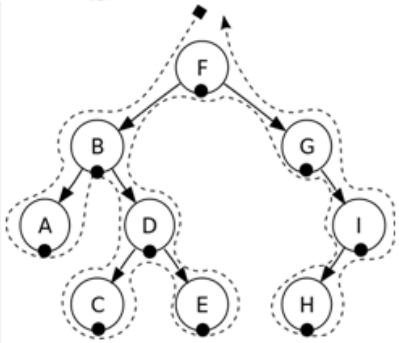


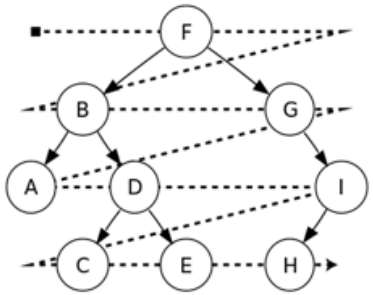

```
postorder(node)
  if (node = null)
    return
  postorder(node.left)
  postorder(node.right)
  visit(node)
```




```
inorder(node)
  if (node = null)
    return
  inorder(node.left)
  visit(node)
  inorder(node.right)
```

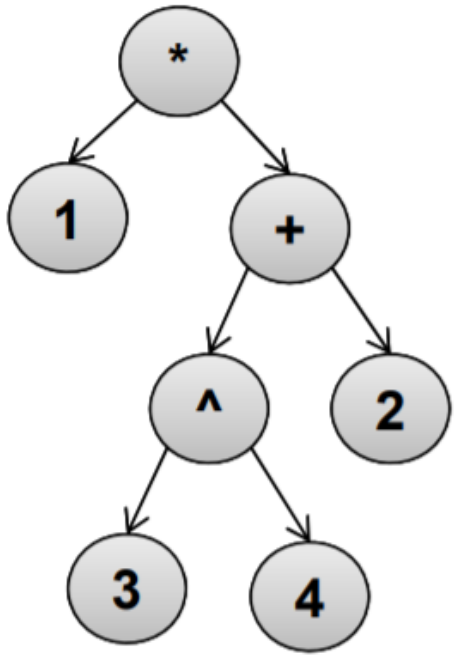
...para árboles binarios





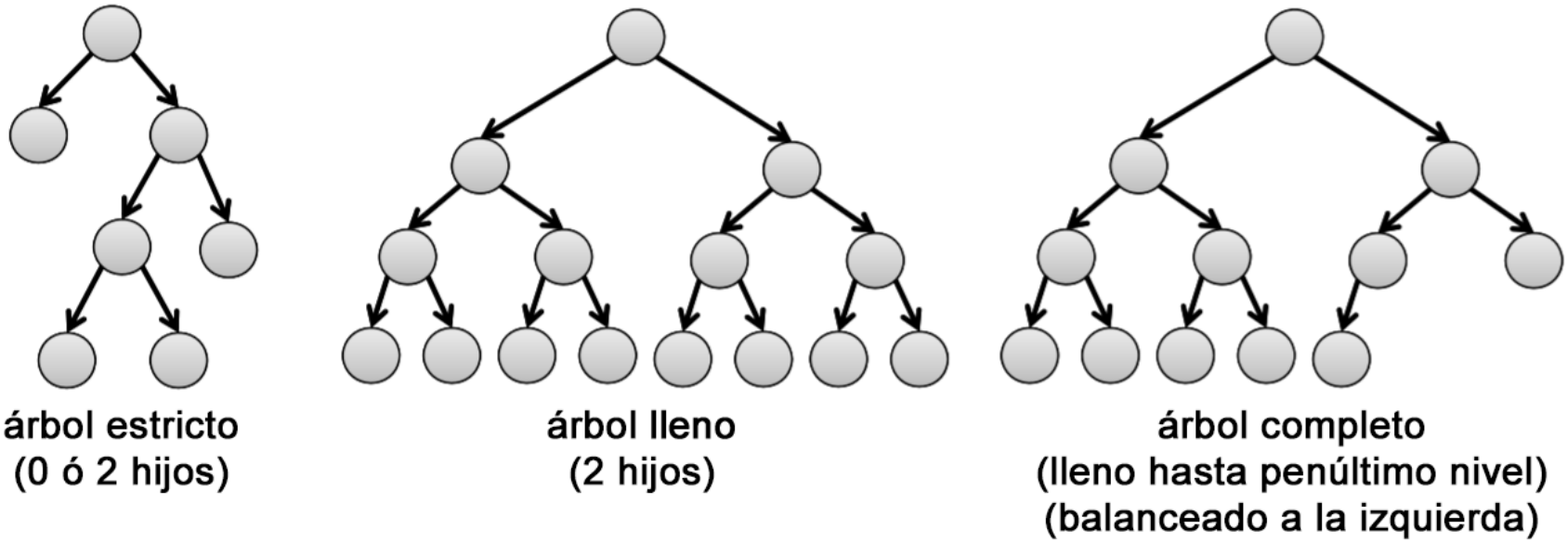
Ejemplo: Árboles sintácticos

- Preorden (notación prefija):
  * 1 + ^ 3 4 2
- Postorden (notación postfija):
  1 3 4 ^ 2 + *
- Inorden (notación habitual):
  1 * ((3 ^ 4) + 2)



Definiciones

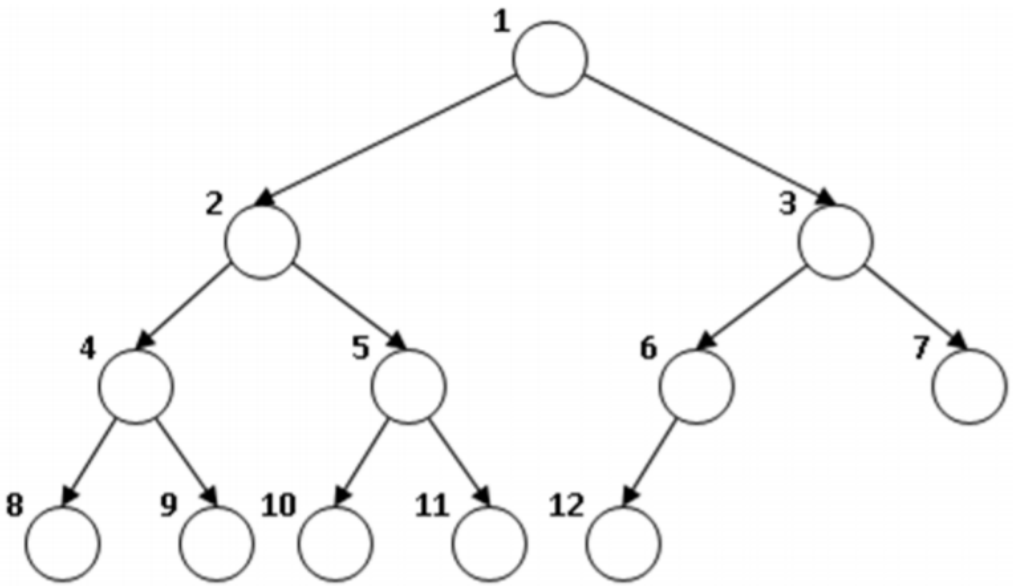
- Un árbol binario es un árbol donde cada nodo puede a lo más tener un hijo izquierdo y un hijo derecho.
- Es una estructura de datos (como lo era una lista enlazada).

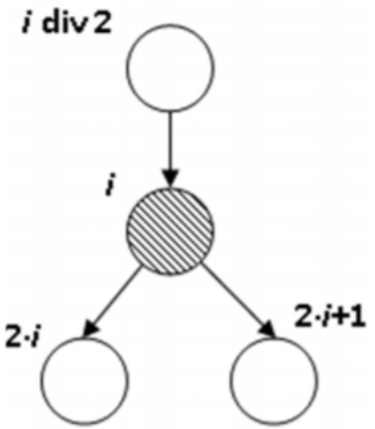


¿Cuántos nodos tiene un árbol estricto en función de su altura h?

Árboles completos

- En árboles completos, $h \in O(\log n)$
- Un árbol completo puede almacenarse en un vector dado por sus orden recorrido por niveles, de modo que para el índice de cada nodo podemos conocer su padre y sus hijos.





Ejercicio

1. Ingrese aquí:

   https://sites.google.com/site/programacioniiuno/temario/unidad-5---grafos/rboles

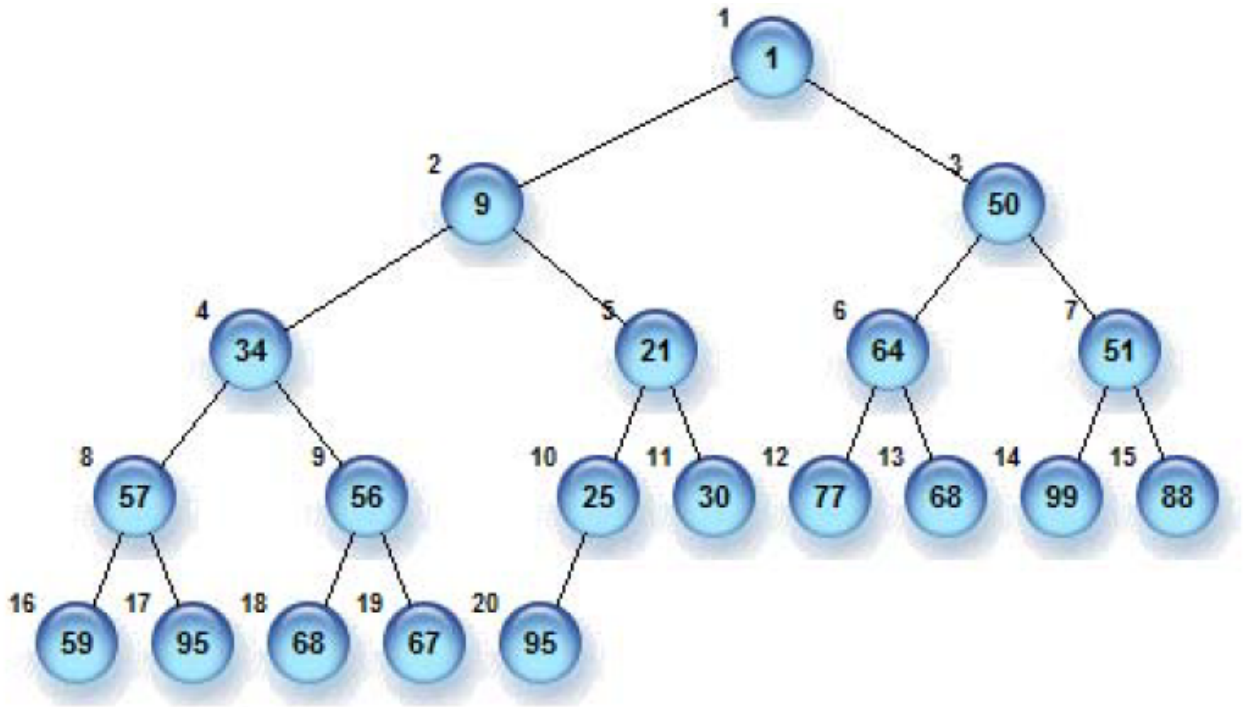
2. Inspeccione el código descrito, también descargable aquí:

   https://xp-dev.com/svn/uqbar/examples/prog2/unidad5-recursividadYArboles/arboles/

3. ¿Cuál es la estructura de datos usada para manipular árboles?
4. Testee las funciones `agregarElemento`, `buscarSubarbol`, `profundidad` y `grado`. ¿Qué es el grado en este código?
5. Testee los mecanismos de recorrido del árbol, en profundidad y en anchura. ¿Qué tipo de recorrido en profundidad se ha implementado? ¿preorder, inorder o postorder?
6. Modifique el código para recorrer el árbol con los dos mecanismos de profundidad restantes. (+5 pts)

Montículos (Heaps)

Un montículo (heap) es un árbol completo tal que el valor de cada nodo es $\leq$ que el de sus descendientes ($\leq$ puede reemplazarse por $\geq$).



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	9	50	34	21	64	51	57	56	25	30	77	68	99	88	59	95	68	67	95

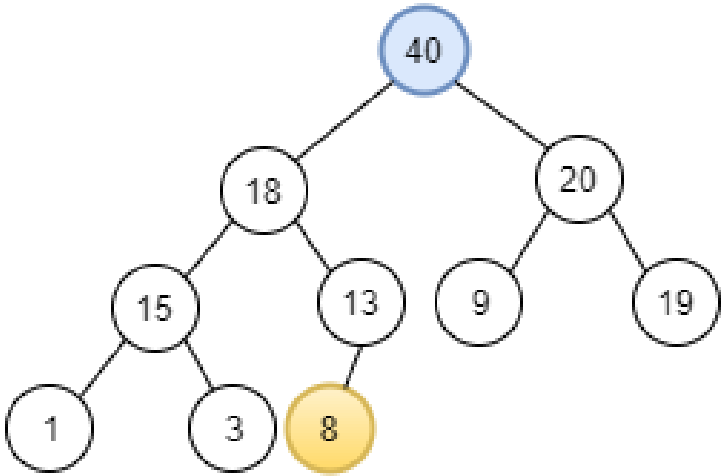
Propiedades de un montículo

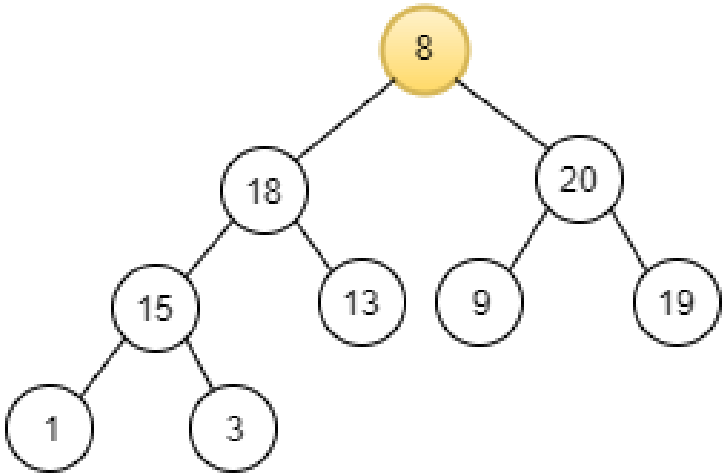
- El valor de la raíz es el mínimo.
- Sea h la altura y n el número de nodos, entonces $h \in O(\log n)$ (son árboles completos).
- Si solo un nodo está mal ubicado, puede corregirse en $O(h)$.

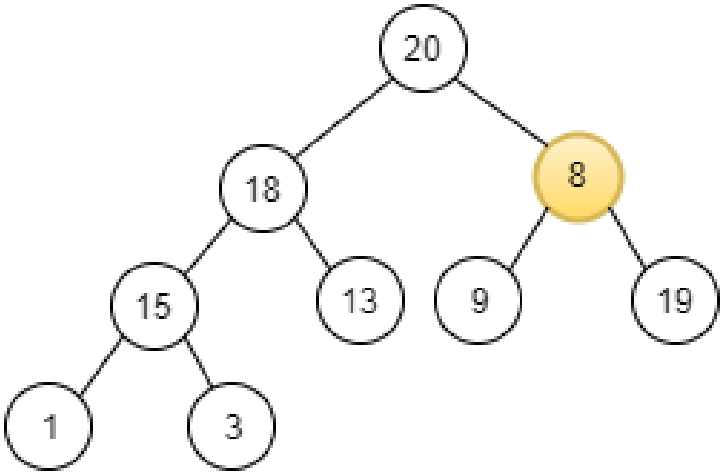
Operaciones fundamentales

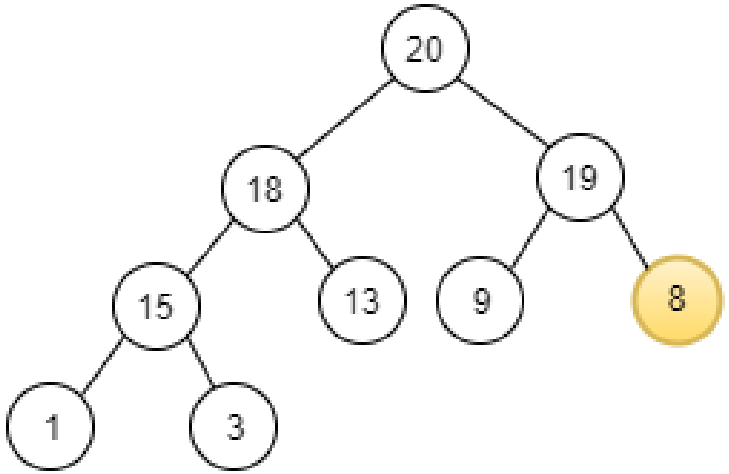
- Inserción
  1. El nuevo nodo se ubica al final del vector.
  2. Se *asciende* hasta que quede bien ubicado según su valor.
- Eliminar la raíz
  1. Se intercambia con la última hoja.
  2. Se desciende la nueva raíz hasta que quede bien ubicada según su valor.

Eliminar la raíz



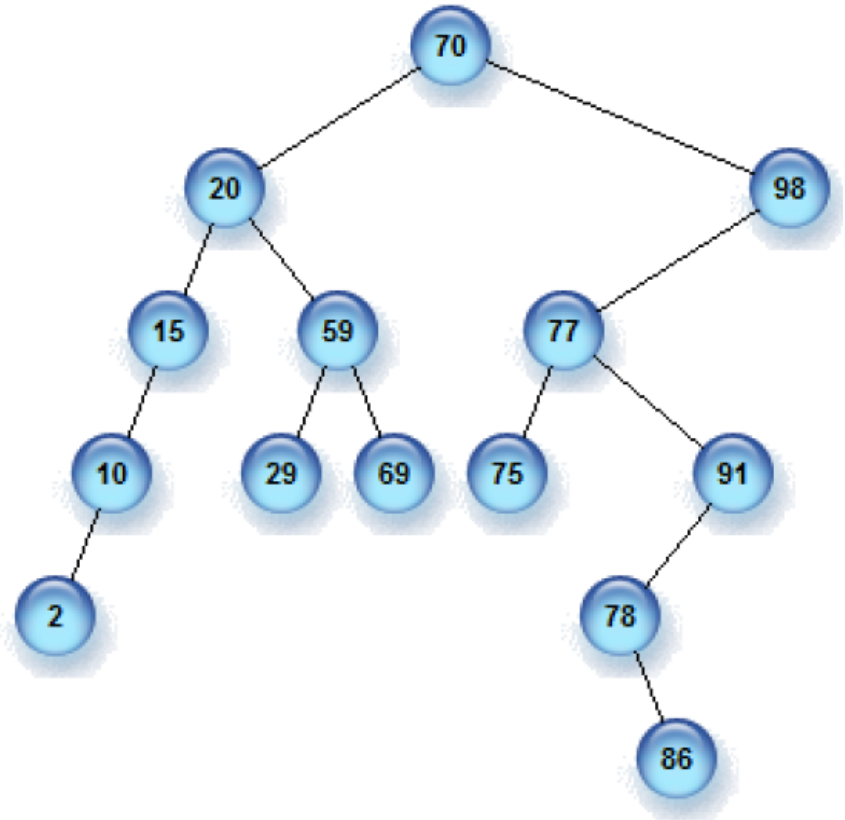






Árboles binarios de búsqueda (BST)

Un árbol binario de búsqueda (binary search tree o BST) es un árbol binario (no necesariamente completo) tal que el valor de cada nodo es $\geq$ que los de su subárbol izquierdo, y $\leq$ que los de su subárbol derecho.



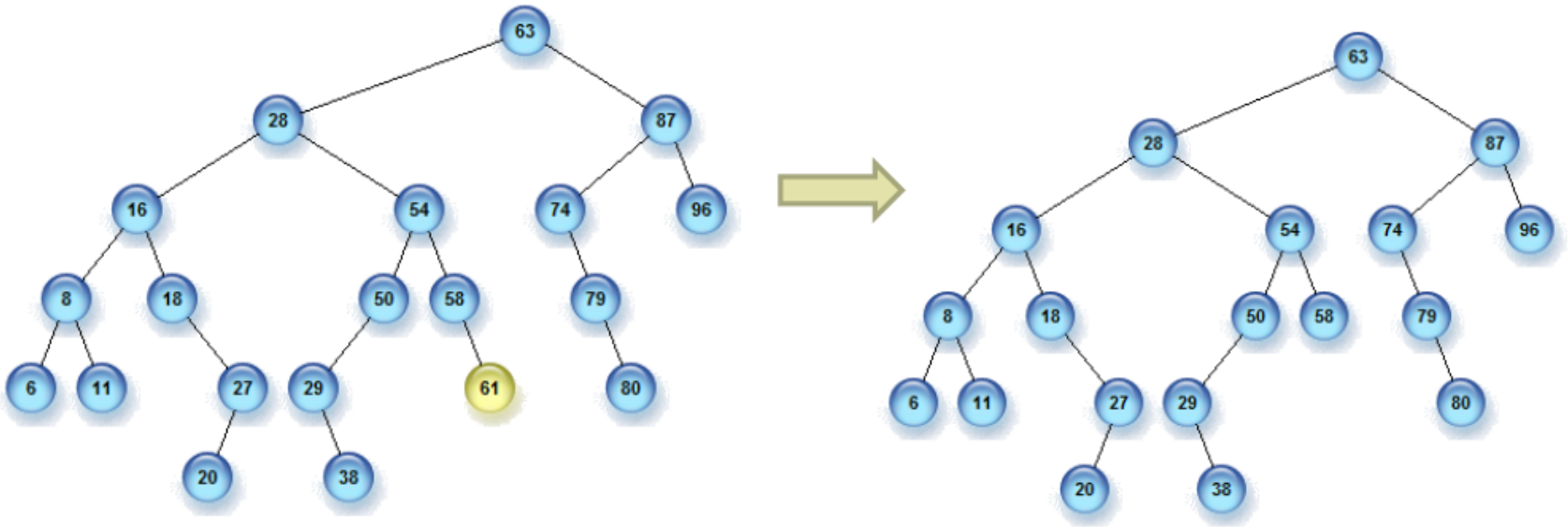
Propiedades de un árbol binario de búsqueda

- Un recorrido inorden coincide con un orden de menor a mayor.
- El mínimo es el primer nodo sin hijo izquierdo en un descenso por hijos izquierdos desde la raíz.
- El máximo es el primer nodo sin hijo derecho en un descenso por hijos derechos desde la raíz.

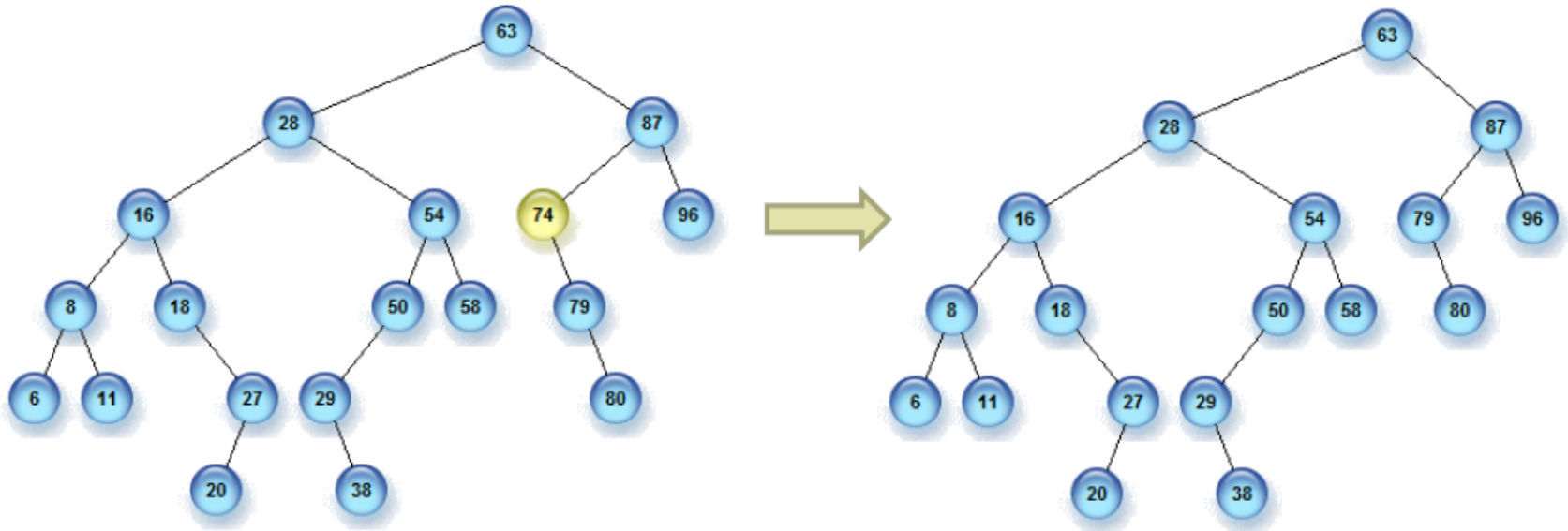
Operaciones fundamentales

- Buscar un nodo
  1. Se parte desde la raíz
  2. Se escoge el subárbol correspondiente respetando la relación $\leq$.
- Insertar un nodo
  1. Se busca el elemento en el árbol.
  2. Se inserta en el lugar donde debería haberse encontrado.
- Eliminar un nodo
  1. Si tiene 0 o 1 hijos, se elimina de manera natural.
  2. Si tiene 2 hijos, se intercambia por el máximo de su subárbol izquierdo (o por el mínimo de su subárbol derecho), y se elimina ese máximo (o ese mínimo).

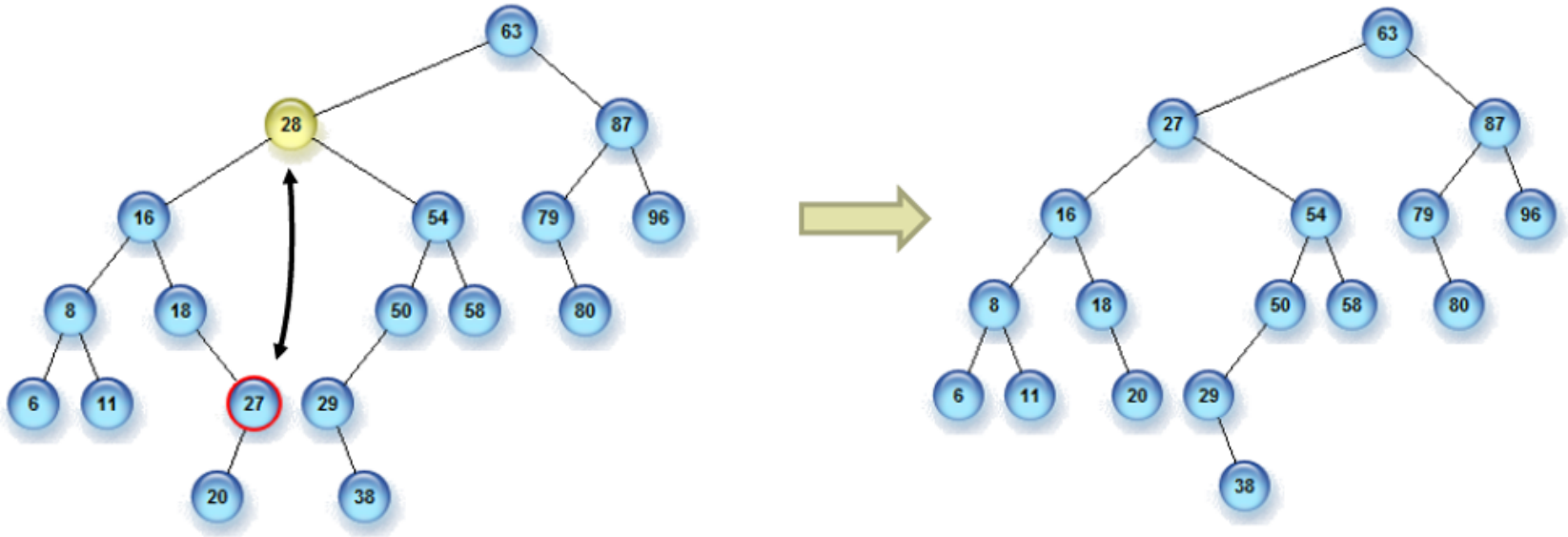
Eliminación: Nodo con 0 hijos



Eliminación: Nodo con 1 hijo



Eliminación: Nodo con 2 hijos



Árboles balanceados

- Un árbol balanceado o equilibrado es un árbol binario de búsqueda restringido para que su altura sea logarítmica.
- Para mantener un árbol balanceado, se añaden instrucciones adicionales a las operaciones de inserción y borrado.
- Ejemplos:
  - Árboles AVL: la diferencia de altura entre subárboles izquierdo y derecho ≤ 1.
  - Árboles Rojo-Negro
  - Árboles biselados (Splay trees)

Ejercicios

Utilizando el lenguaje de programación que más le acomode:

- Montículos (Heaps)
  1. Implemente un montículo utilizando un vector.
  2. Implemente las funciones de inserción y eliminación.
  3. Tip: puede inspirarse en la biblioteca estándar de C++
     `http://www.cplusplus.com/reference/algorithm/sort_heap/`
     pero la idea es que lo implemente Ud.
- Árboles binarios de búsqueda (BST)
  1. Implemente un BST utilizando un vector.
  2. Implemente las funciones de búsqueda, inserción y eliminación en los tres casos.
- Árboles AVL
  1. Investigue sobre las diferencias en las operaciones de inserción y borrado de nodos entre estos árboles y los BST.
  2. ¿Cuáles son las ventajas algorítmicas de un AVL en comparación a un BST general?

Preliminares

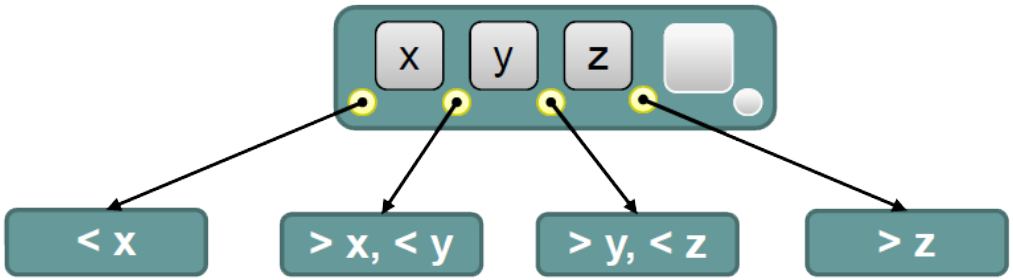
- Un árbol B es una generalización de un árbol binario de búsqueda (BST) en que cada nodo puede tener > 2 hijos.
- Es balanceado y de altura logarítmica, pero no binario, a diferencia de árboles AVL, Rojo-negro, biselados, etc.
- En cuanto a su complejidad computacional, en el "peor caso":

	BST	AVL	B
Búsqueda	$O(n)$	$O(\log n)$	$O(\log n)$
Inserción	$O(n)$	$O(\log n)$	$O(\log n)$
Borrado	$O(n)$	$O(\log n)$	$O(\log n)$

  En el "caso promedio", todos son $O(\log n)$.
- A pesar de esto, el árbol B es una estructura de datos más eficiente para gandes volúmen de datos, donde se requiere disminuir el número de accesos: bases de datos, discos duros, memorias externas, sistemas de archivos.

Árboles-(a,b)

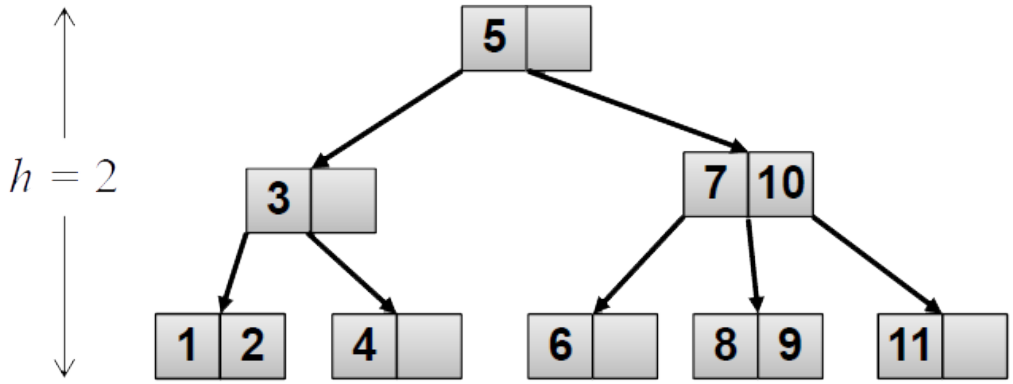
- Un árbol-(a,b) es un árbol de búsqueda balanceado, tal que:
  - Si la raíz tiene hijos, su número está en el rango $[2, b]$.
  - Los nodos internos tienen $[a, b]$ hijos, donde $2 \leq a \leq \frac{b+1}{2}$.
  - Todas las hojas están en el mismo nivel.
  - Cada nodo con k hijos tiene $k-1$ claves ordenadas ($\leq$):



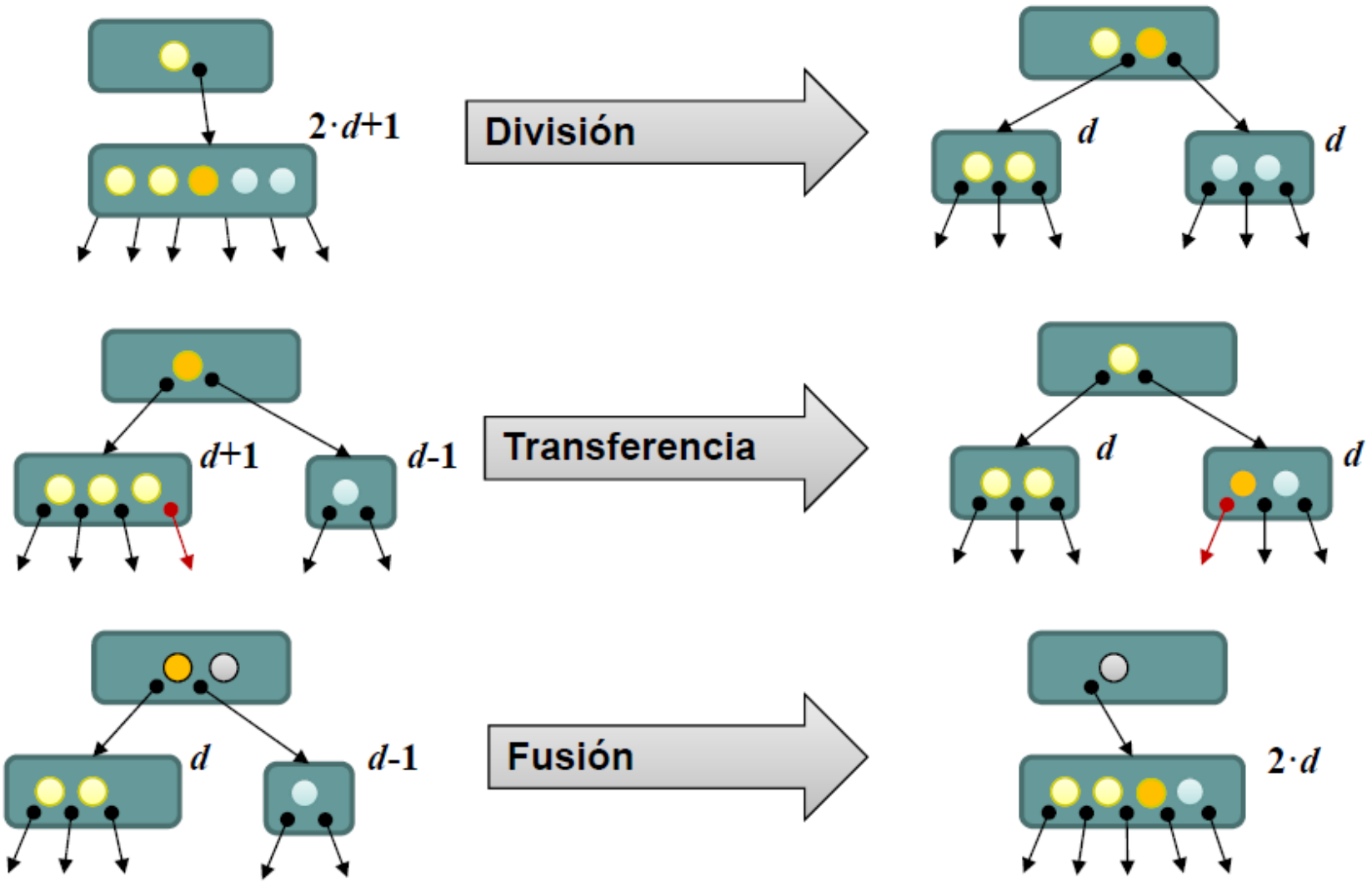
- Por ejemplo, en un árbol-(2,3):
  - La raíz tiene 0, 2 o 3 hijos.
  - Los nodos internos tienen 2 o 3 hijos.

Árboles B

- Un árbol B de orden m es un árbol-$(d+1, 2d+1)$ con $m = 2d+1$. Es decir:
  - Si la raíz tiene hijos, su número está en el rango $[2, m]$.
  - Los nodos internos tienen $[\lceil m/2 \rceil, m]$ hijos.
  - Todas las hojas están en el mismo nivel.
  - Cada nodo con k hijos tiene $k-1$ claves ordenadas ($\leq$):
- Un árbol-B de orden 3 es un árbol-(2,3).



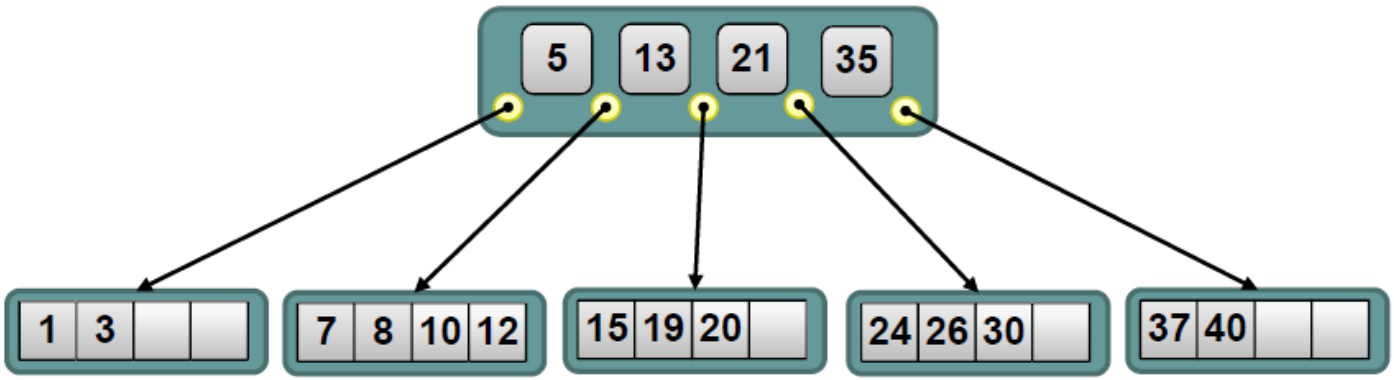
Reestructuraciones



César Vaca Rodríguez, Dpto. de Informática, UVa

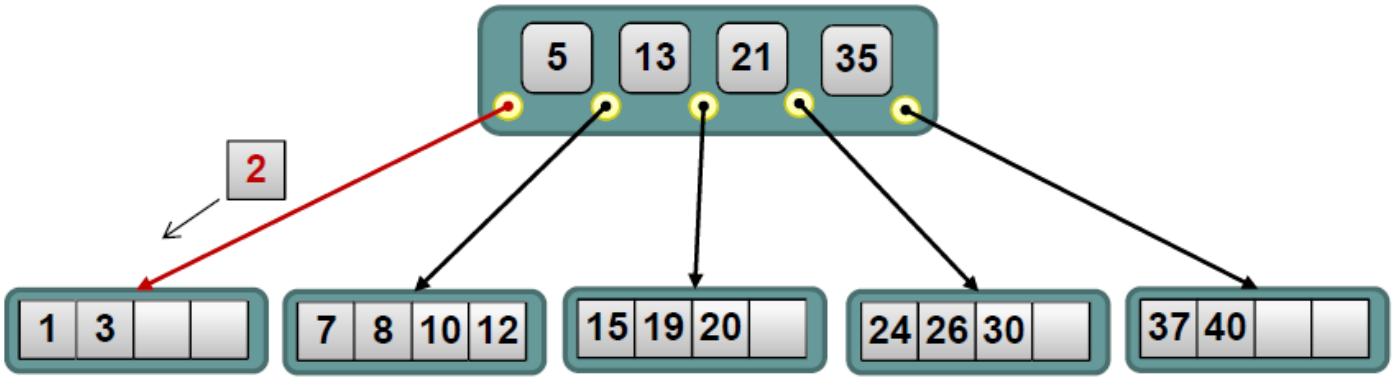
Inserción - Sin reestructuración

- Inserción del valor 2 en árbol B de orden 5 $\rightarrow$ árbol-(3,5)



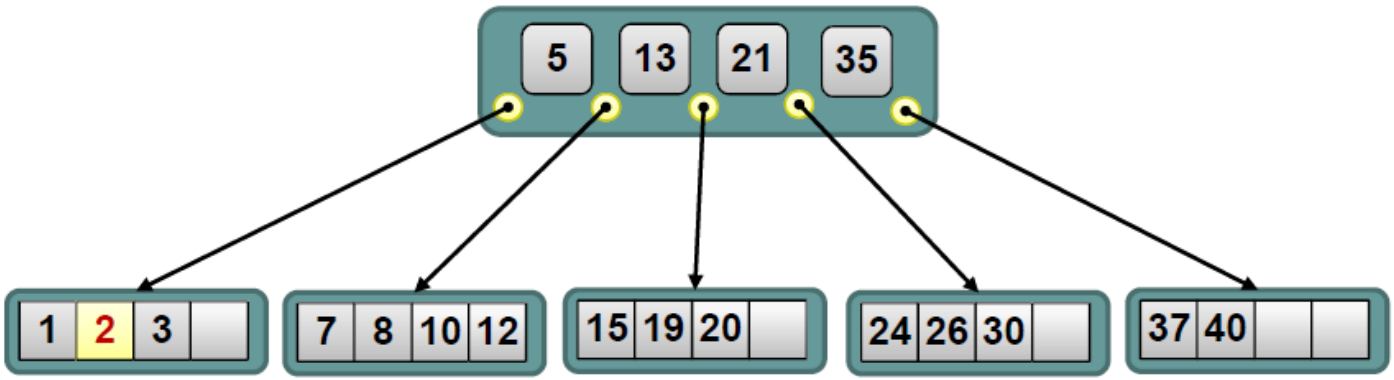
Inserción - Sin reestructuración

- Se busca el nodo hoja donde debe encontrarse el elemento



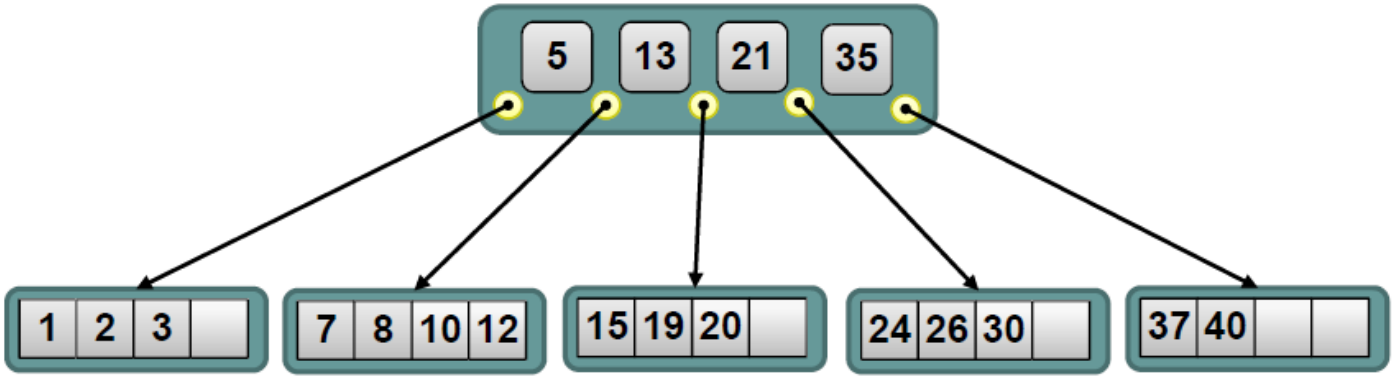
Inserción - Sin reestructuración

- Se inserta en orden en la hoja (desplazamiento). Fin



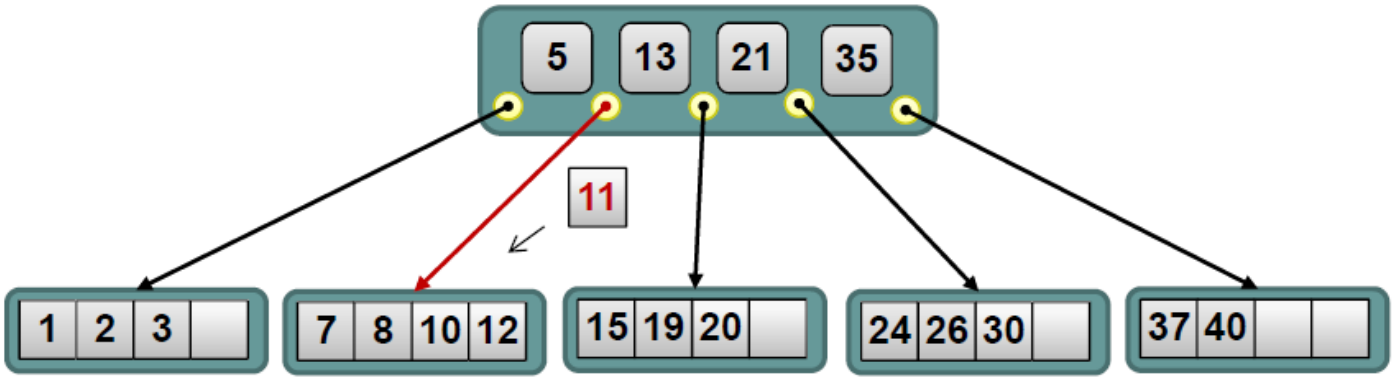
Inserción - División de nodos

- Inserción del valor 11



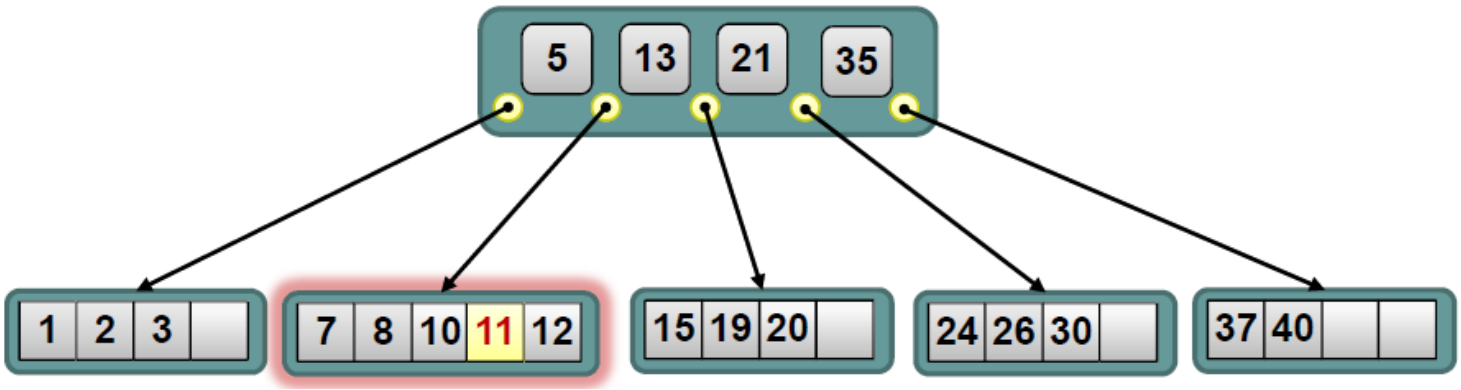
Inserción - División de nodos

- Se busca el nodo hoja donde debe encontrarse el elemento



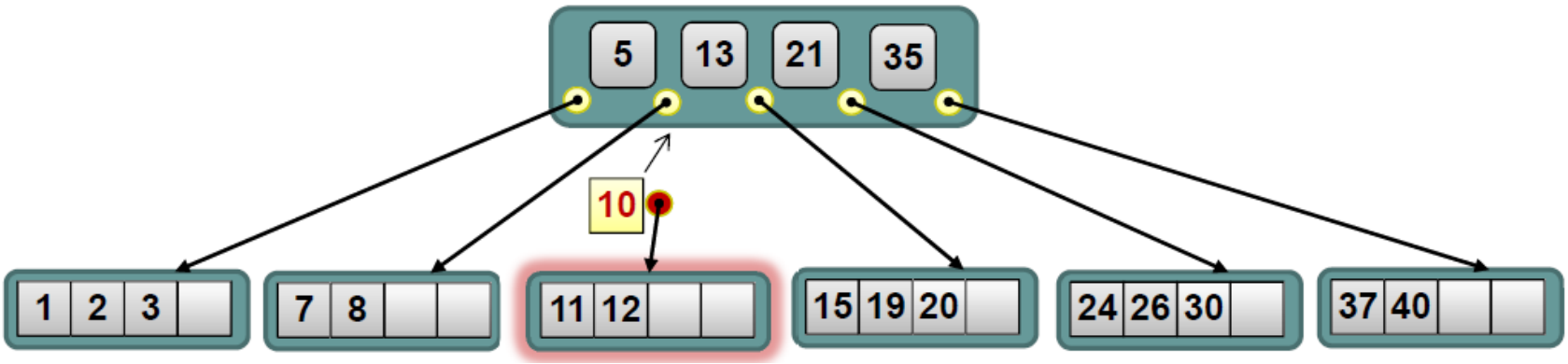
Inserción - División de nodos

- Se inserta en el nodo, pero sobrepasa el límite de claves (4)



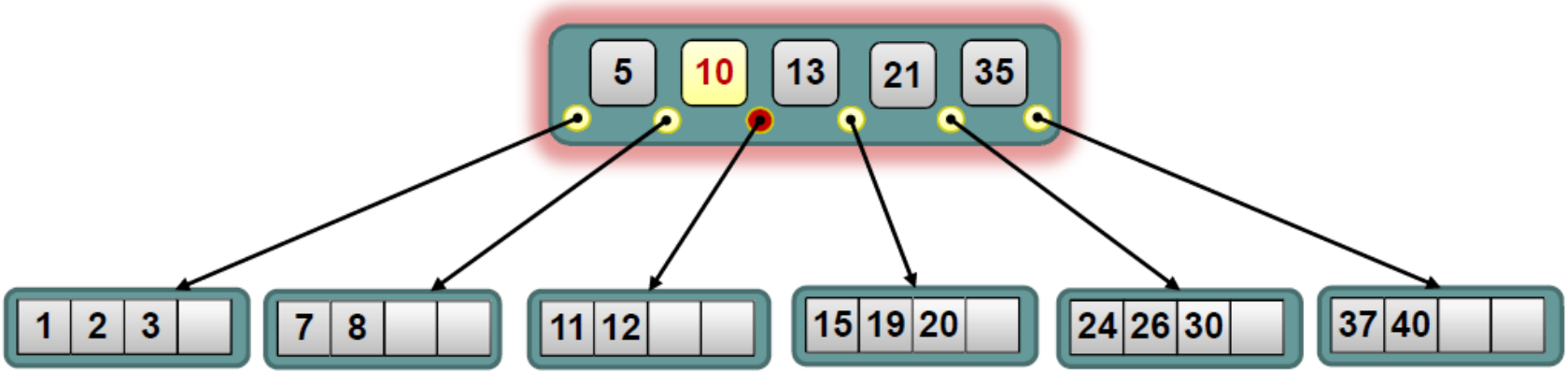
Inserción - División de nodos

- Se crea un nuevo nodo y se traslada la mitad derecha de los elementos a él.
- El elemento en posición media (10), junto con el enlace al nuevo nodo, se envía al padre para su inserción.



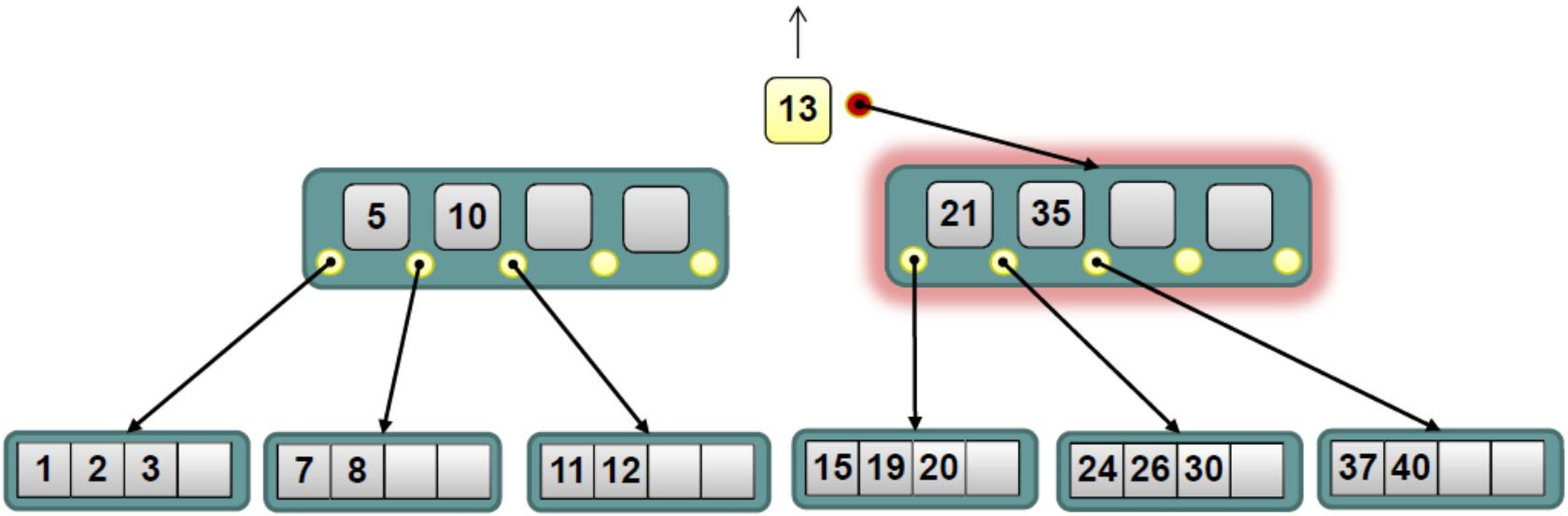
Inserción - División de nodos

- Se inserta en el padre, pero sobrepasa límite de claves (4).



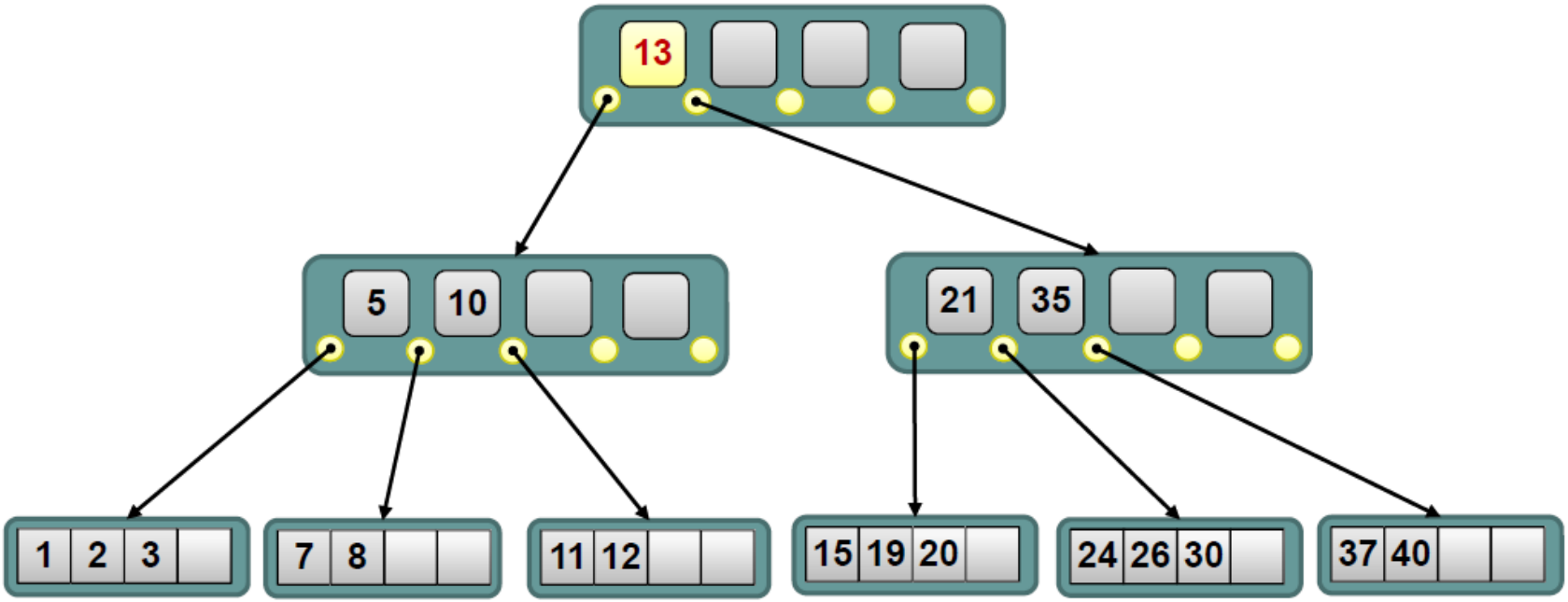
Inserción - División de nodos

- Se repite el proceso de división por la mitad.



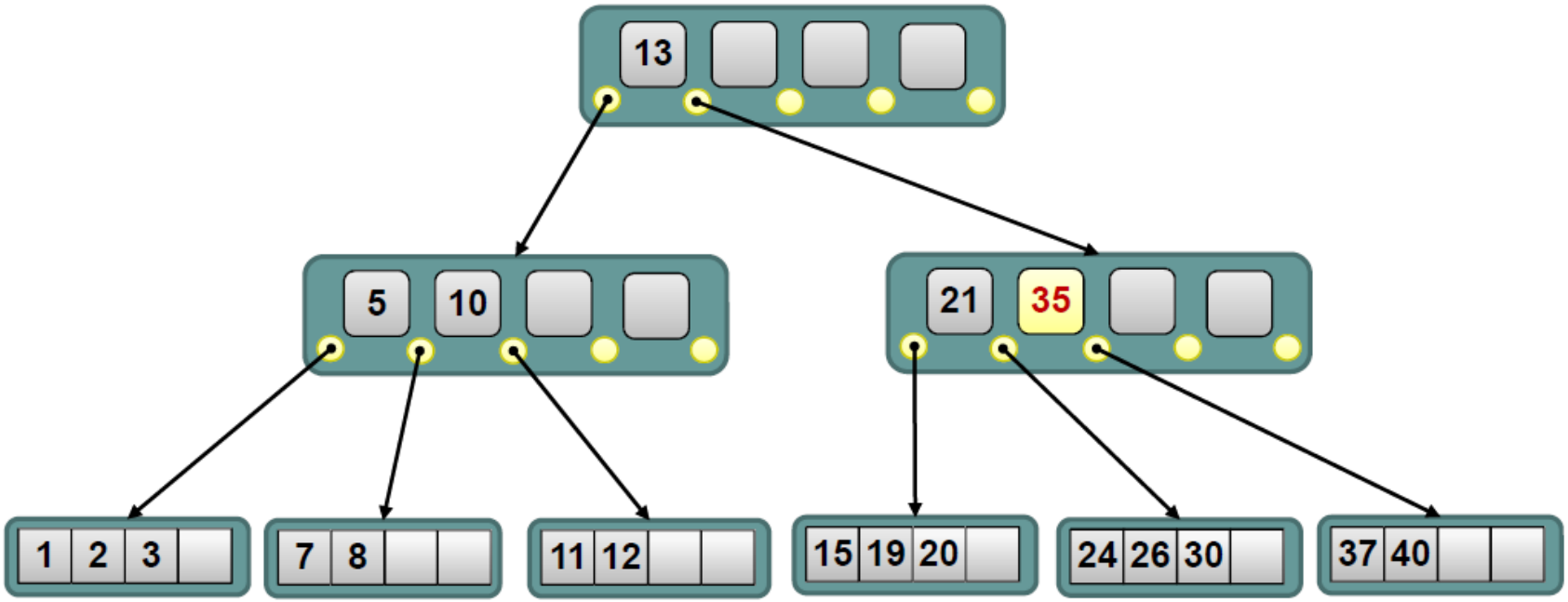
Inserción - División de nodos

- No hay padre → se crea nueva raíz con esa única clave. Fin



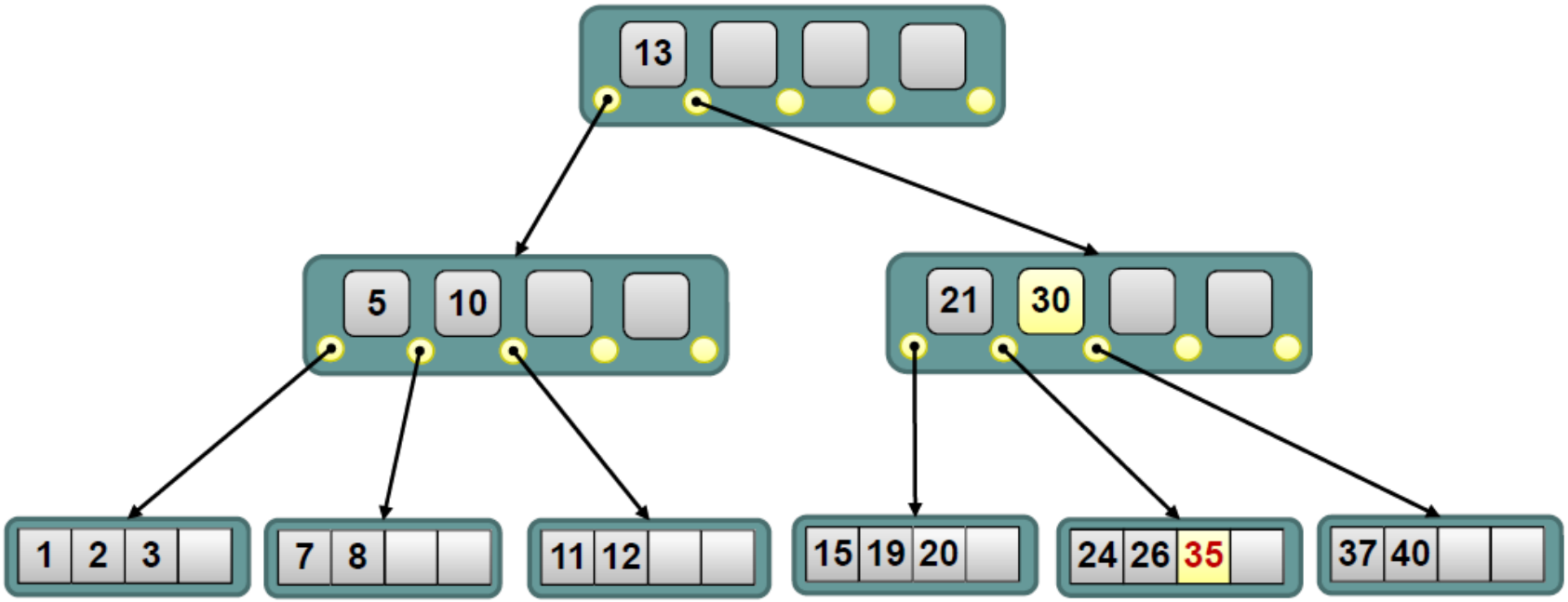
Borrado - Sin estructuración

- Borrado de clave 35. Se busca nodo donde está el elemento.



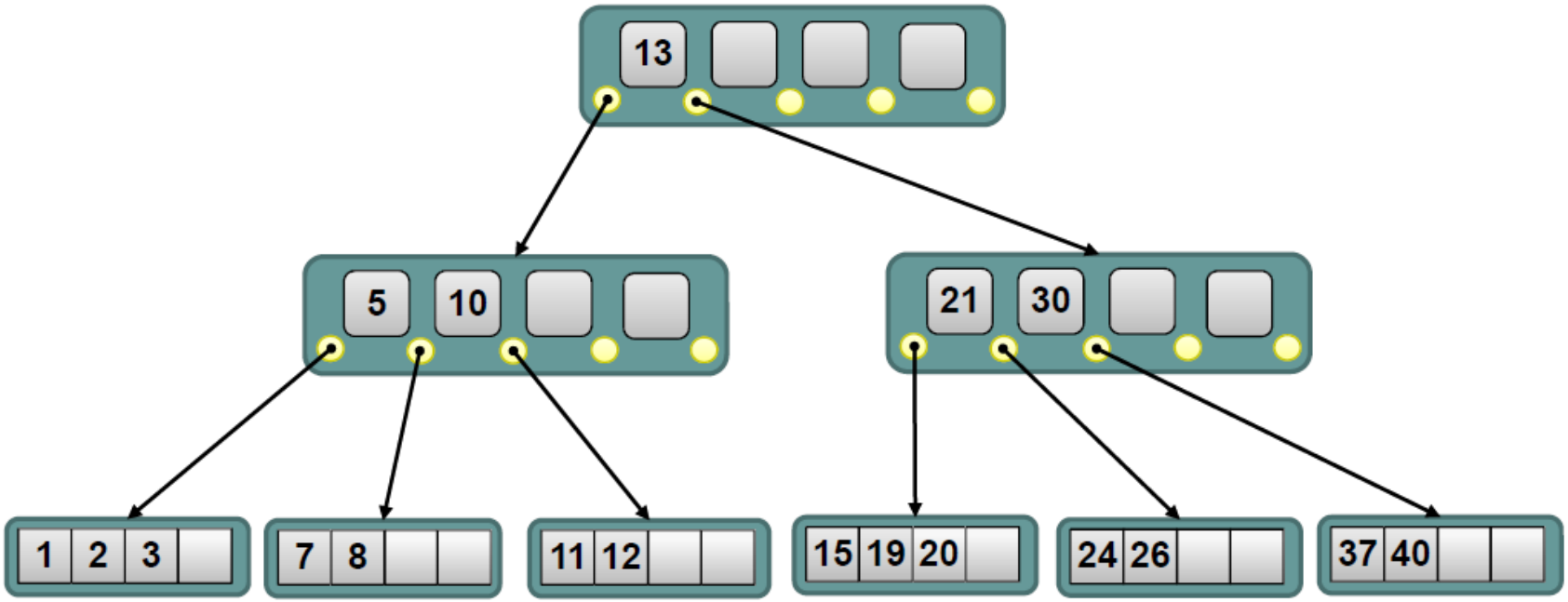
Borrado - Sin estructuración

- Es nodo interno: se intercambia con máx. de su hijo izquierdo



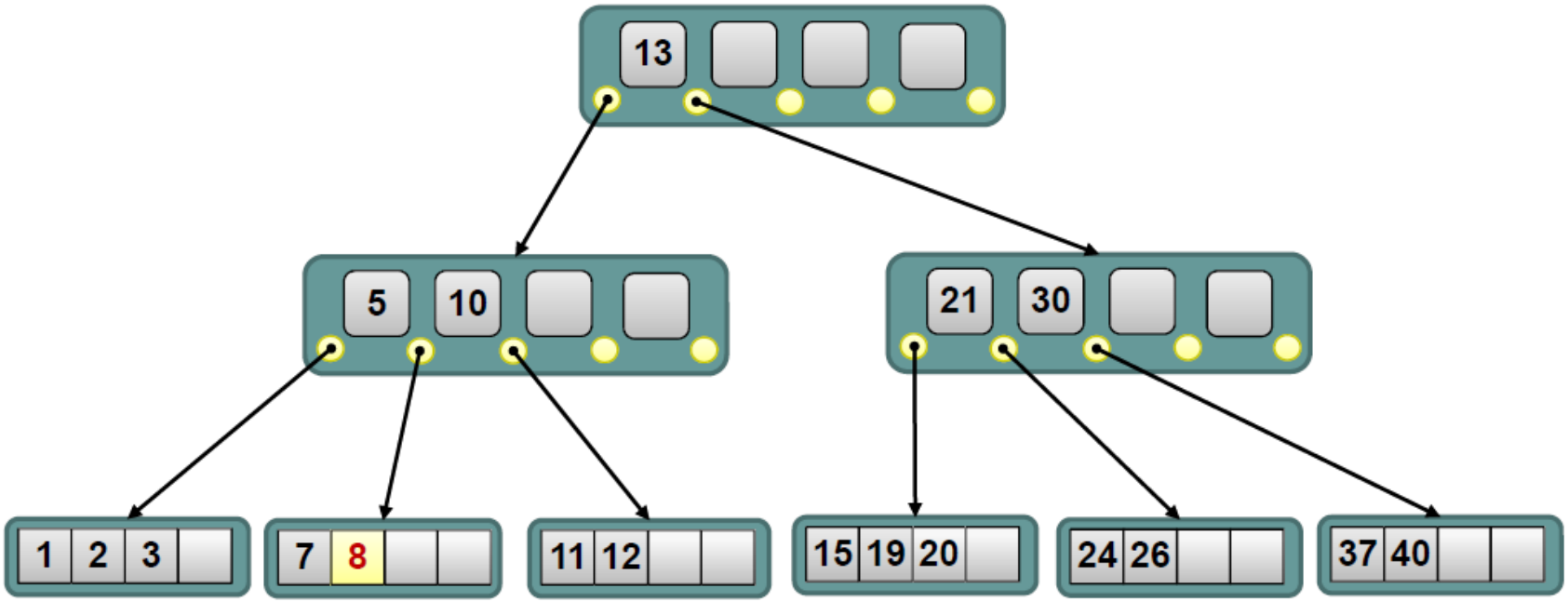
Borrado - Sin estructuración

- Se borra el elemento. Fin



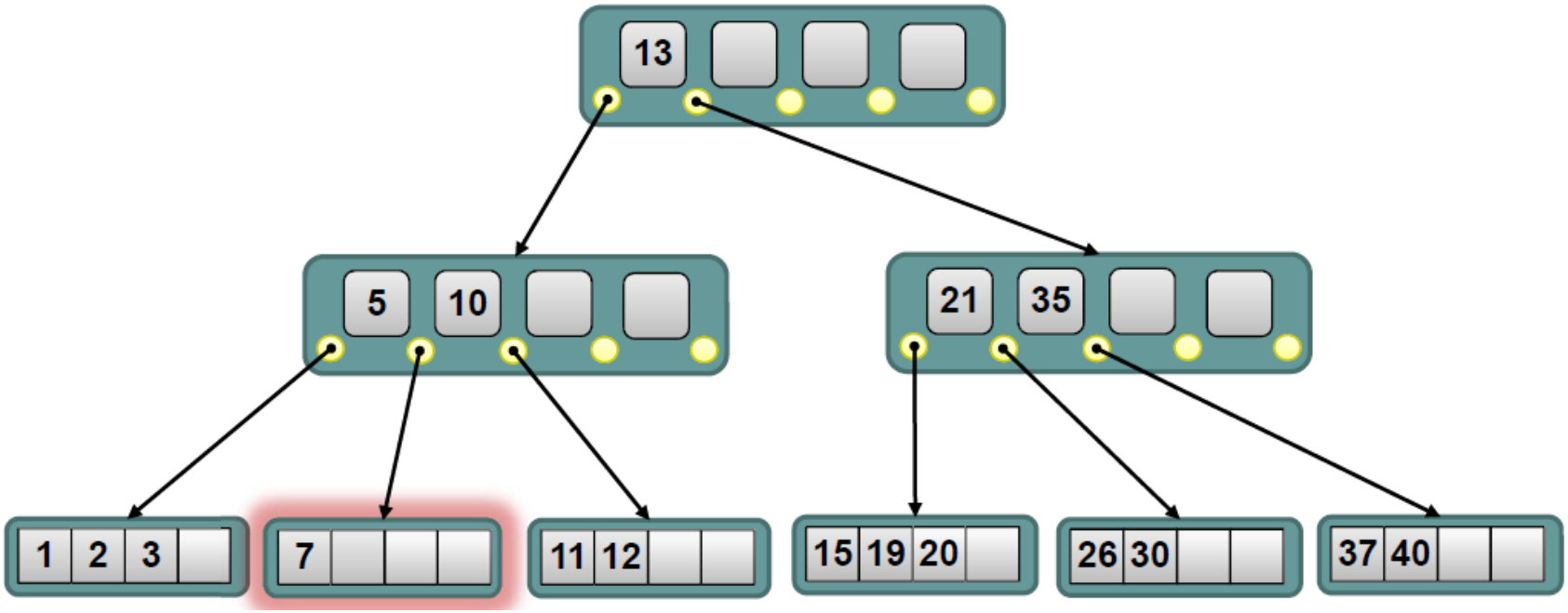
Borrado - Transferencia

- Borrado de la clave 8. Se busca el nodo.



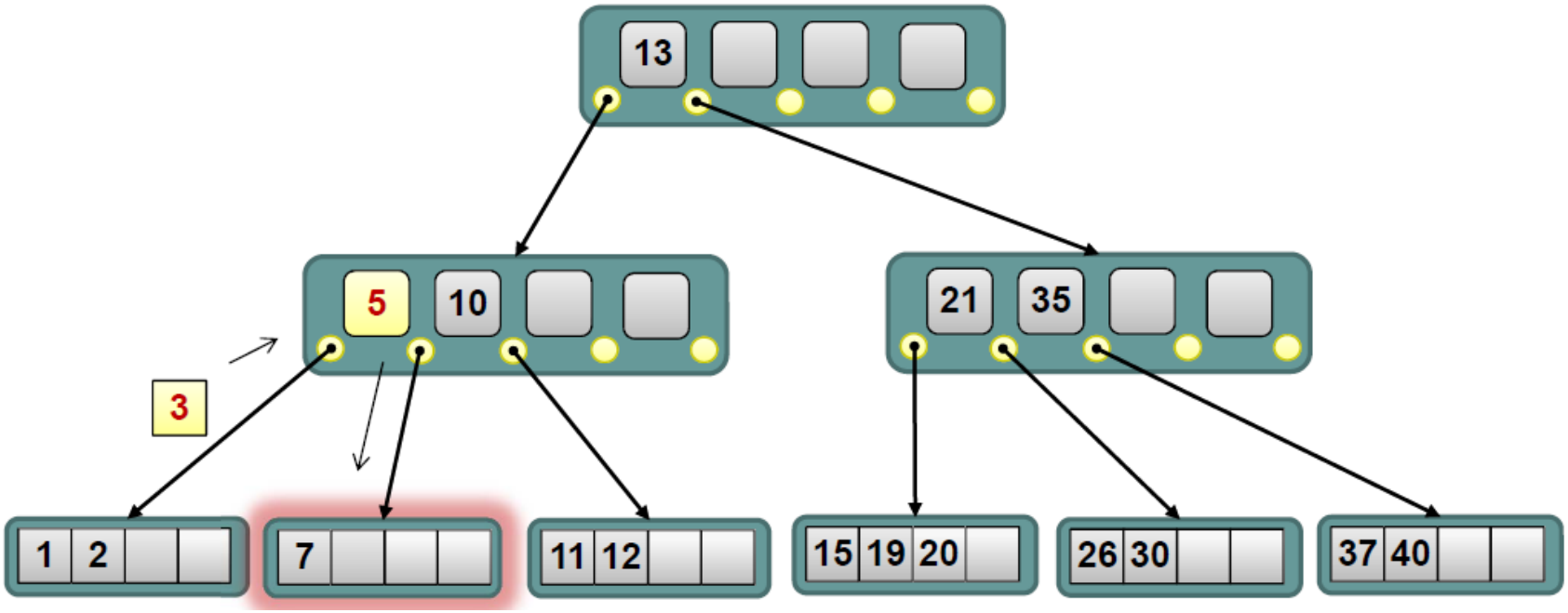
Borrado - Transferencia

- Se borra. Nodo queda con menos claves que permitidas (2).



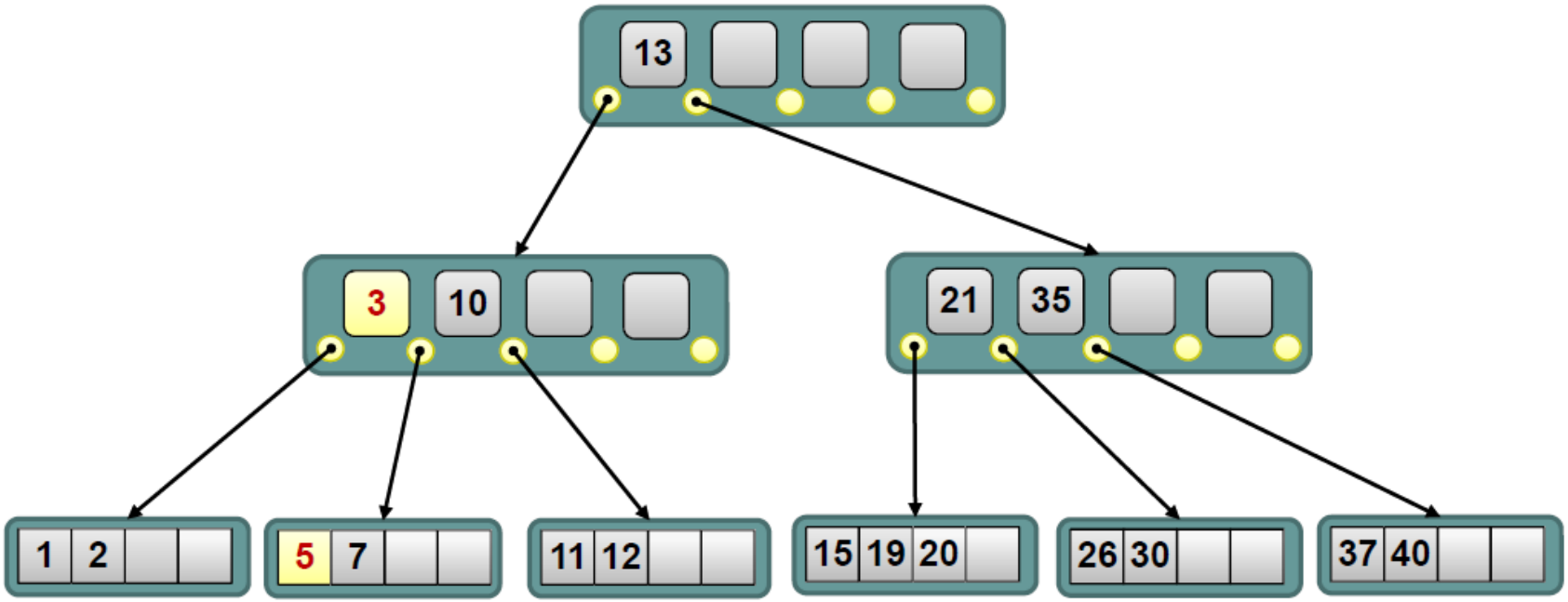
Borrado - Transferencia

- Se transfiere al padre la última clave del hermano más grande.



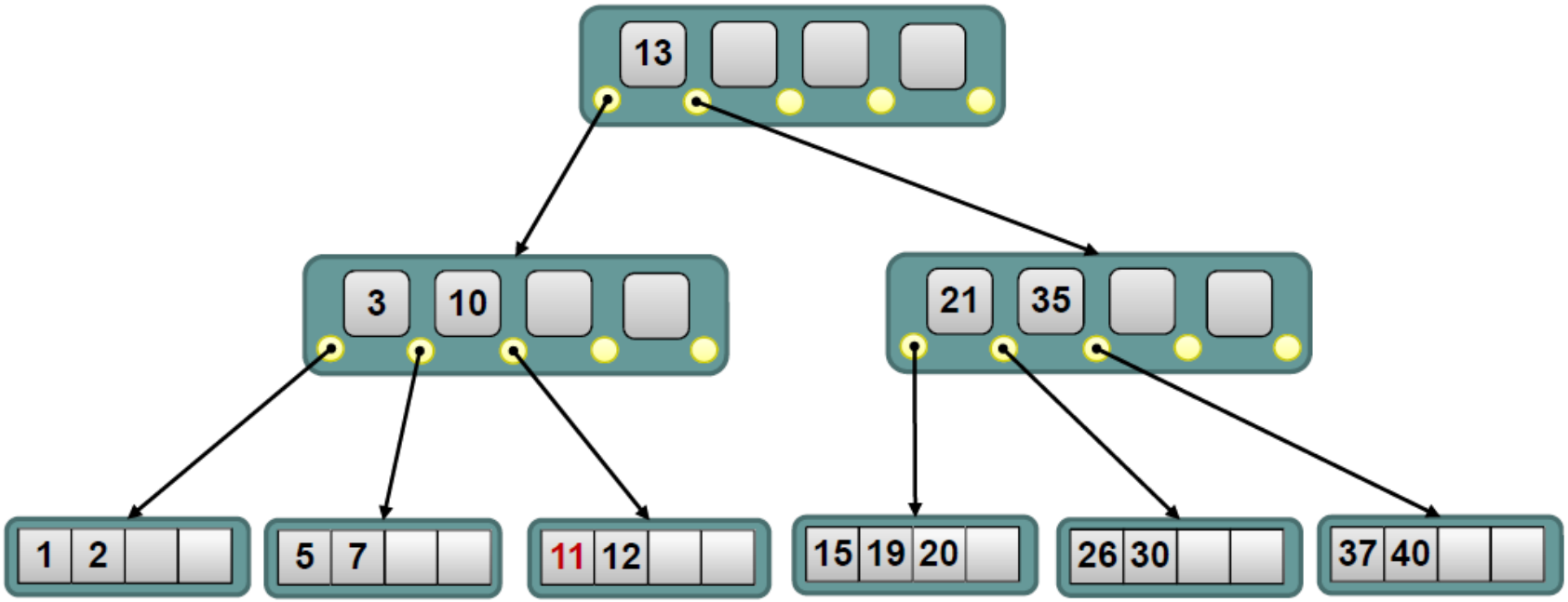
Borrado - Transferencia

- Se transfiere la antigua clave del padre al nodo. Fin



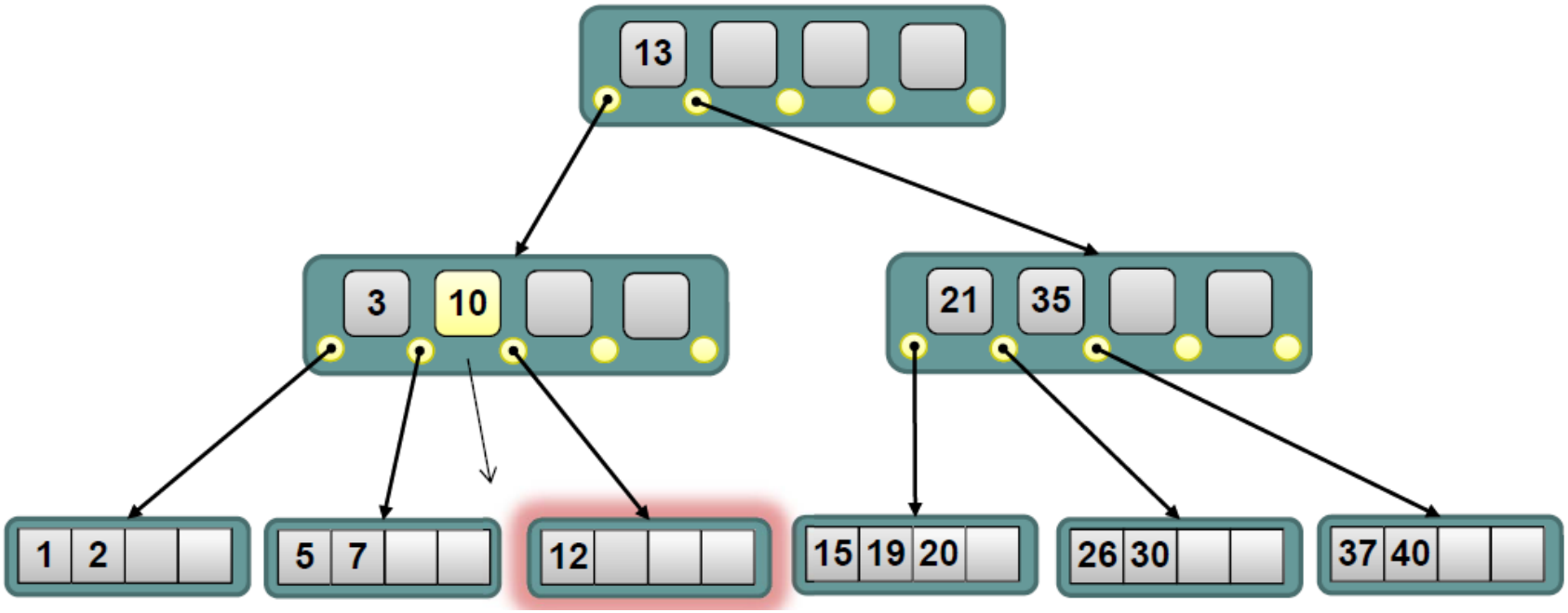
Borrado - Fusión

- Borrado de la clave 11. Se busca el nodo.



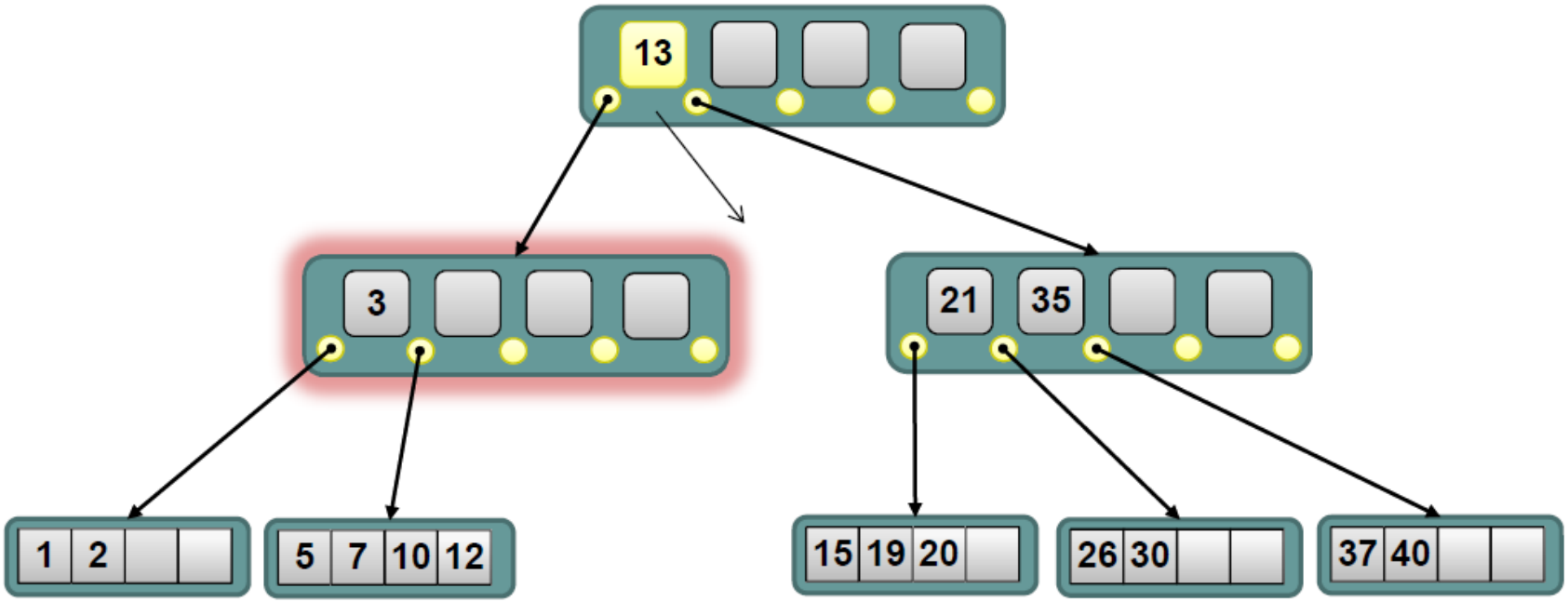
Borrado - Fusión

- Se borra. Nodo tiene 1 clave y su hermano no puede transferir.



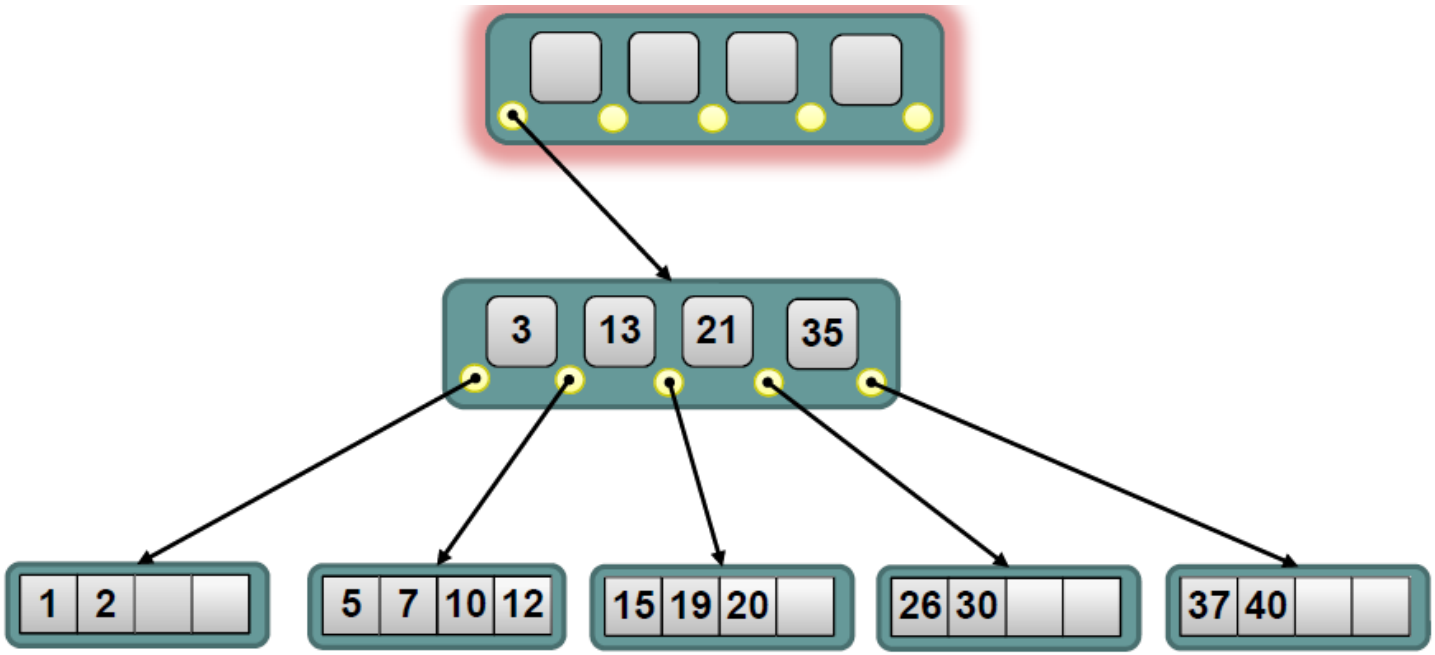
Borrado - Fusión

- Hermanos se fusionan con clave del padre.
- Padre queda con 1 clave y su hermano no puede transferir.



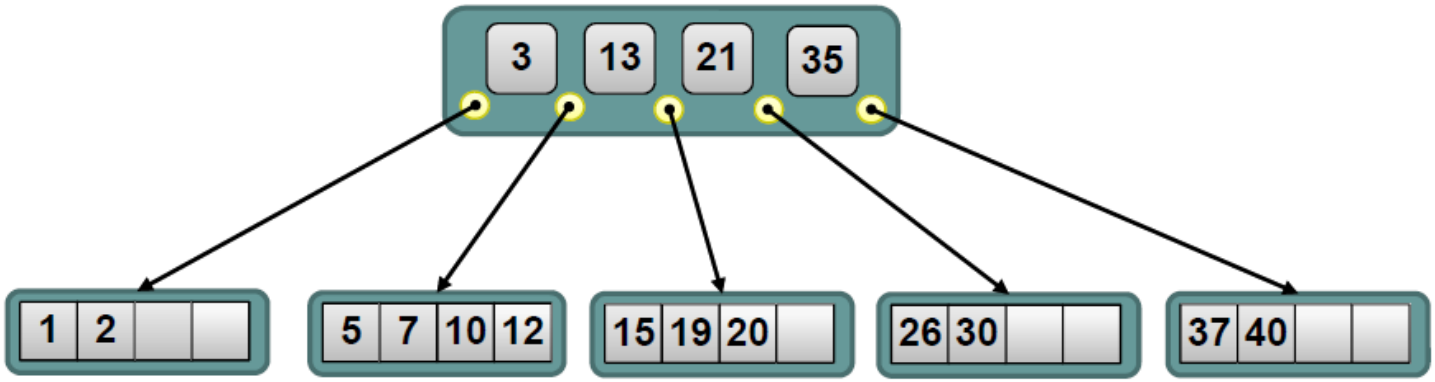
Borrado - Fusión

- Hermanos se fusionan con clave de raíz, que queda vacía.



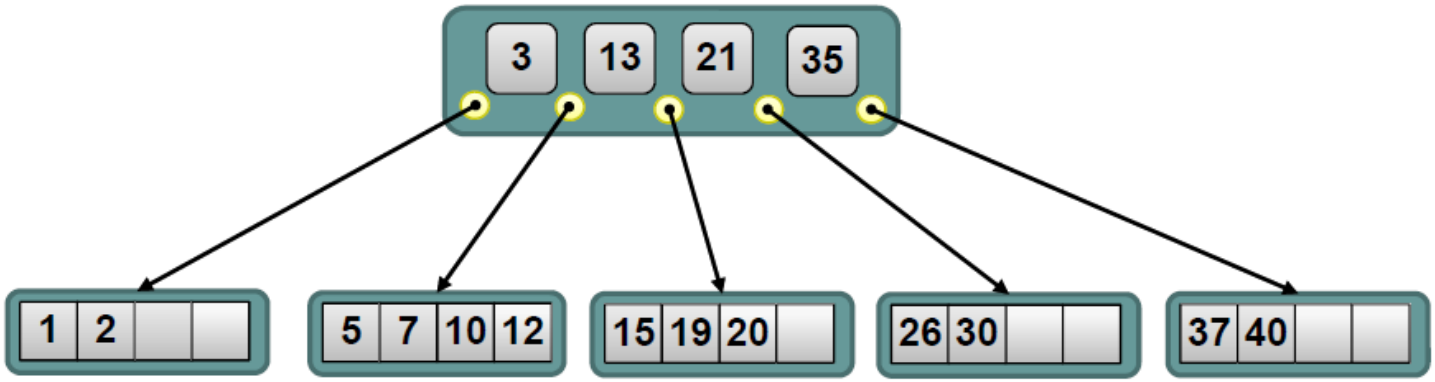
Borrado - Fusión

- Se elimina el nodo raíz. Fin



Borrado - Fusión

- Se elimina el nodo raíz. Fin



Usos y variantes

- Los árboles B y sus variantes se usan en:
  - Gestores de bases de datos
  - Sistemas de archivos: NTFS (Windows), HFS+ (Apple), btrfs, Ext4 (Linux)
- Variantes principales:
  - Árboles de prerecorrido: Antes de insertar se realiza una búsqueda que divide todos los nodos llenos. El número máximo de claves es 2d+1.
  - Árboles B+: Sólo las hojas contienen elementos, los nodos internos contienen claves para dirigir la búsqueda (esas claves se encuentran también en los nodos hoja). Los nodos hoja forman una lista doblemente enlazada.
  - Árboles B*: El número mínimo de claves es 2/3 de la capacidad. Se fusionan 3 nodos en 2, y se dividen 2 nodos en 3.

Bibliografía recomendada

- Weiss, M., Estructura de datos y algoritmos, Addison-Wesley, 1995.
- Aho, Hopcroft y Ullman, Estructuras de datos y algoritmos, Addison-Wesley, 1988.

Recursos

- Apuntes de César Vaca Rodríguez, Dpto. de Informática, Universidad de Valladolid, España, 11 Feb 2011.
- Wikimedia Commons.
- `http://www.cplusplus.com`