

ÁRBOLES MONOCRITERIO: ÁRBOLES B

Marco A. Camejo M.

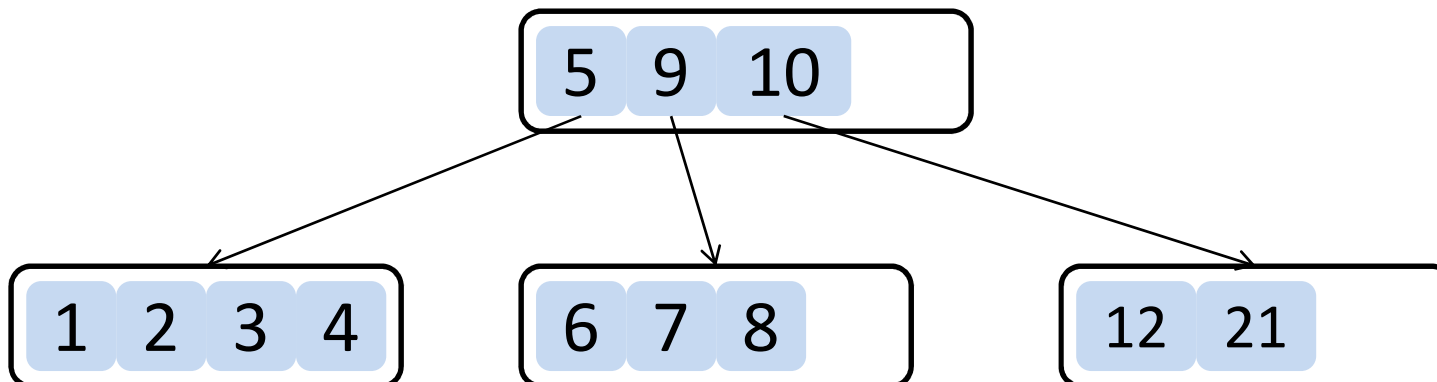
Árboles B. Introducción



- Estructura de datos dinámica orientada al manejo de conjuntos grandes de datos
- Inventados por Bayer y McCreight (1971)
- Utilizados en sistemas de archivos y para mantener índices en bases de datos relacionales

Árboles B. Introducción

- Estructura y Comportamiento similar a la de los Árboles Binarios de Búsqueda



Árboles B. Definición



Un árbol B de orden t , es un árbol que satisface:

1. Todas las hojas están en el mismo nivel y contienen datos
2. Todo nodo a excepción de la raíz debe tener al menos $(t-1)$ claves
3. Todo nodo puede tener como máximo $(2t-1)$ claves
4. Sea n el número de claves que posee un nodo, entonces ese nodo tiene $(n+1)$ hijos

Árboles B. Altura



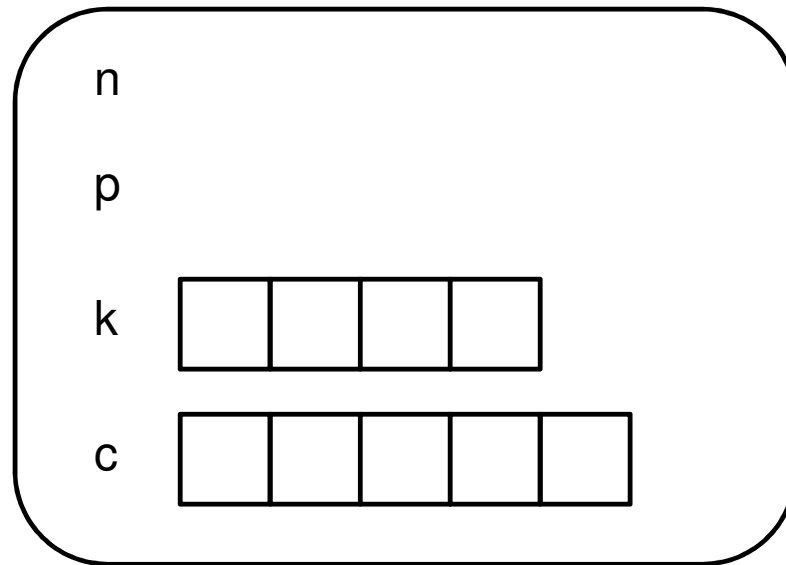
□ Mejor Caso:

□ $h(t) = \log_{2t} n$

□ Peor Caso:

□ $h(t) = \log_t n$

Árboles B. Implementación



nodoB

n: entero (cantidad de claves)

p: apuntador al nodo padre

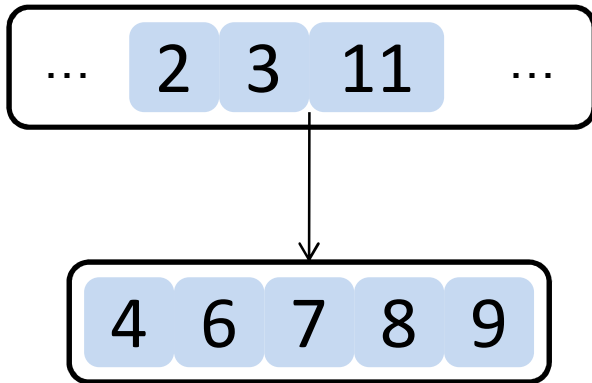
k: arreglo de claves

c: arreglo de apuntadores a los nodos hijo

Árboles B. Búsqueda

buscar(nodoB: x, tipoElemento: k): nodoB	
1	i = 1
2	MIENTRAS (i ≤ claves(x) AND k > clave(x, i)) i = i + 1
3	SI (i ≤ claves(x) AND k = clave(x, i)) retornar x
4	SI (nodo_hoja(x)) retornar NULO
5	SINO x = hijo(x, i) retornar buscar(x, k)

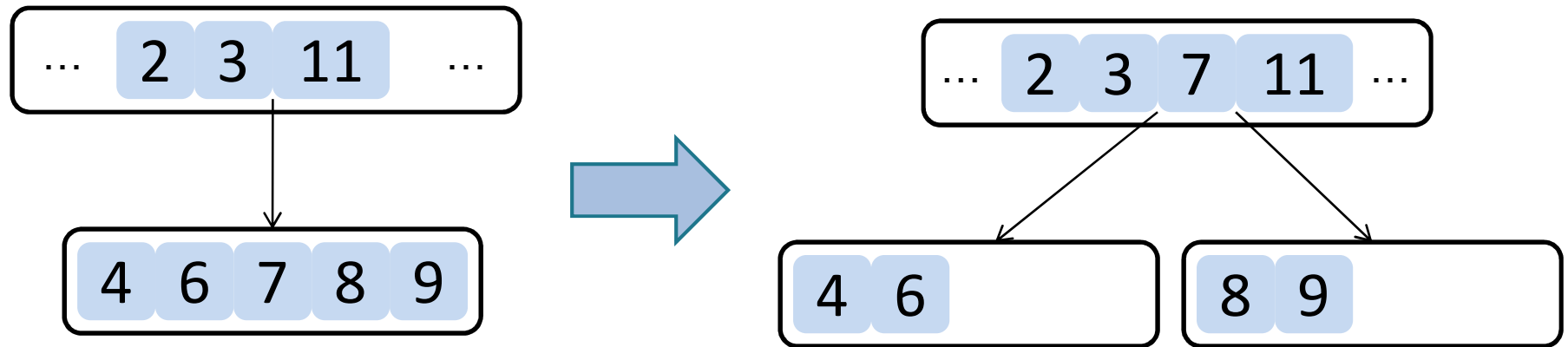
Árboles B. Split



$t=3$

Máximo Número de claves = $2t-1 = 5$

Árboles B. Split



Cada nodo queda con $t - 1 = 2$ claves

Árboles B. Split

split(nodoB: x, entero: i, nodoB: y)		
1	z = nuevoNodo()	x: nodo padre
2	nodo_hoja(z) = nodo_hoja(y)	y: i-ésimo hijo de x
3	claves(z) = t - 1	
4	REPITA PARA(j = 1 HASTA t - 1)	
5	clave(z, j) = clave(y, j+t)	
	SI (NOT nodo_hoja(y))	
	REPITA PARA(j = 1 HASTA t)	
	hijo(z, j) = hijo(y, j+t)	
	claves(y) = t - 1	
	REPITA PARA(j = claves(x) + 1 HASTA i + 1)	
	hijo(x, j+1) = hijo(x, j)	
	hijo(i+1) = z	
	REPITA PARA(j = claves(x) HASTA i)	
	clave(x, j+1) = clave(x, j)	
	clave(x, i) = clave(y, t)	
	claves(x) = claves(x) + 1	

Inserción en Árboles B

$t = 2$

En cada nodo:

$1 \leq \text{claves} \leq 3$

$2 \leq \text{hijos} \leq 4$

Inserción en Árboles B

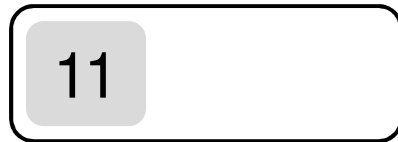


Insertar(11)



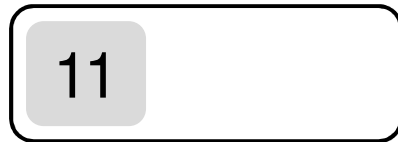
Inserción en Árboles B

Insertar(11)



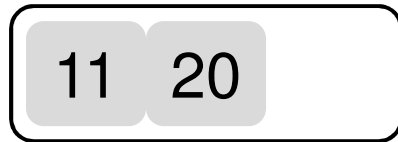
Inserción en Árboles B

Insertar(20)



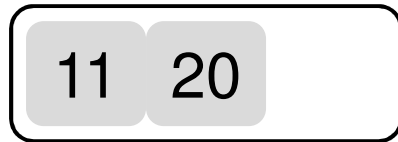
Inserción en Árboles B

Insertar(20)



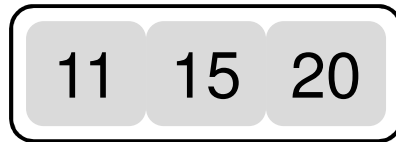
Inserción en Árboles B

Insertar(15)



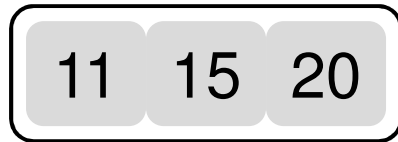
Inserción en Árboles B

Insertar(15)



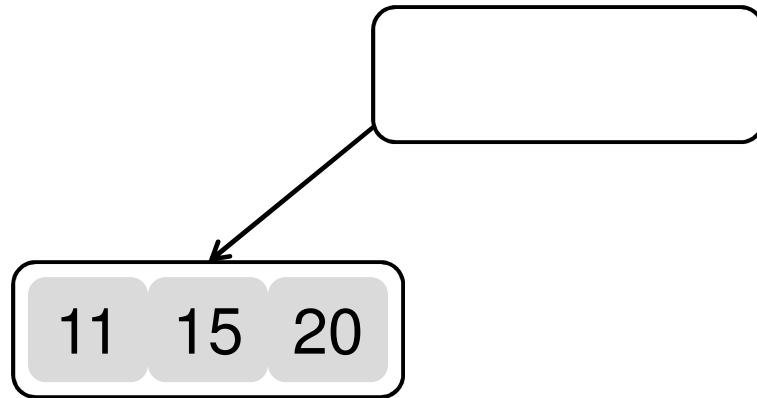
Inserción en Árboles B

Insertar(8)



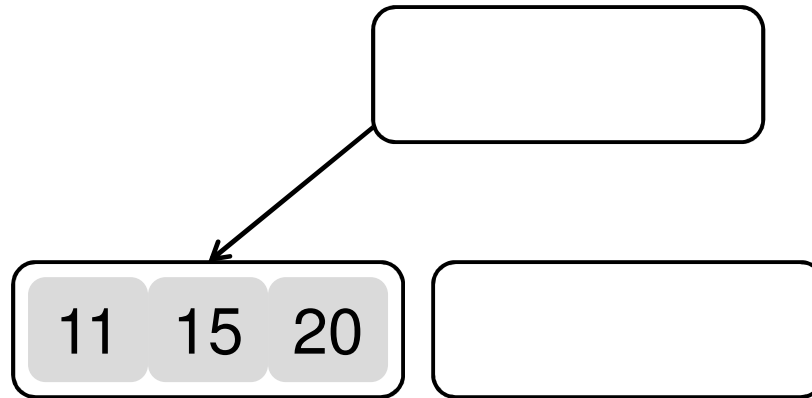
Inserción en Árboles B

Insertar(8)



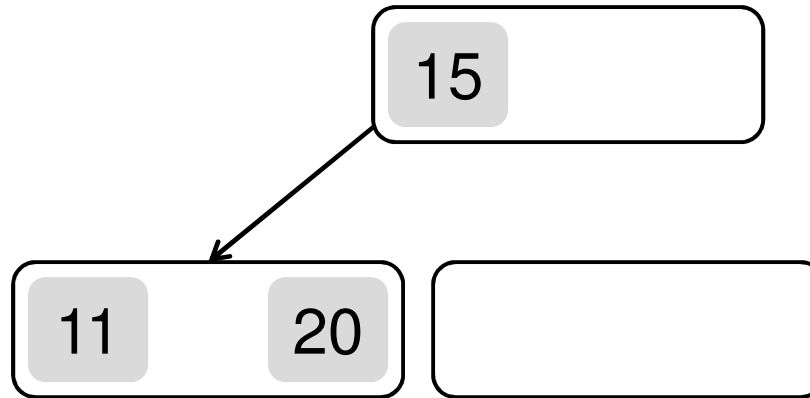
Inserción en Árboles B

Insertar(8)



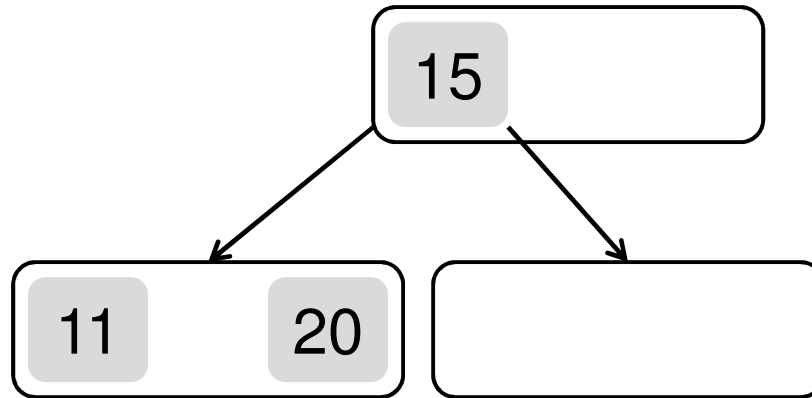
Inserción en Árboles B

Insertar(8)



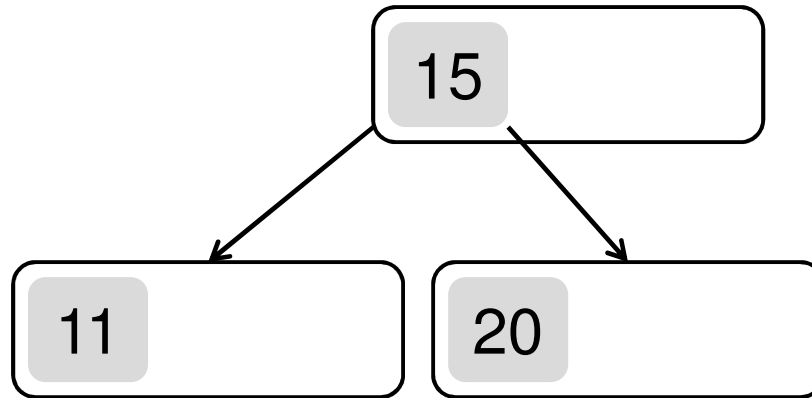
Inserción en Árboles B

Insertar(8)



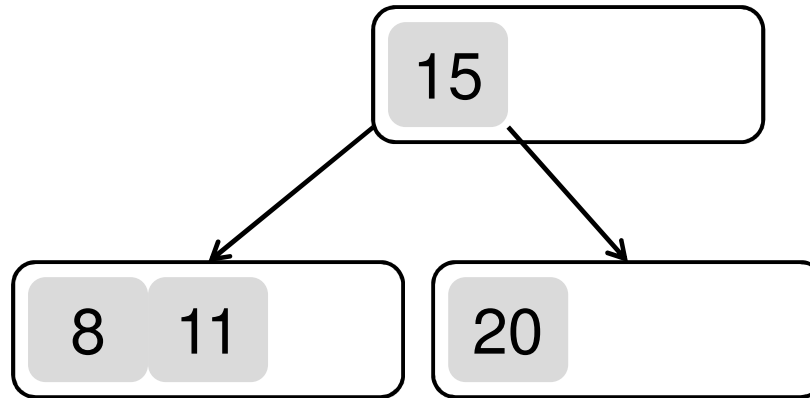
Inserción en Árboles B

Insertar(8)



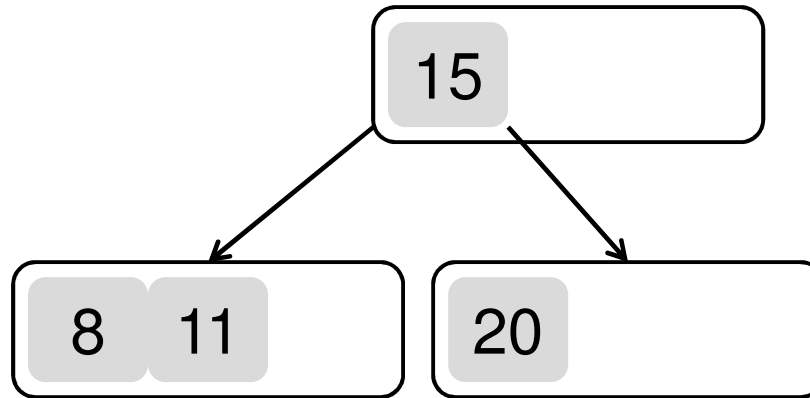
Inserción en Árboles B

Insertar(8)



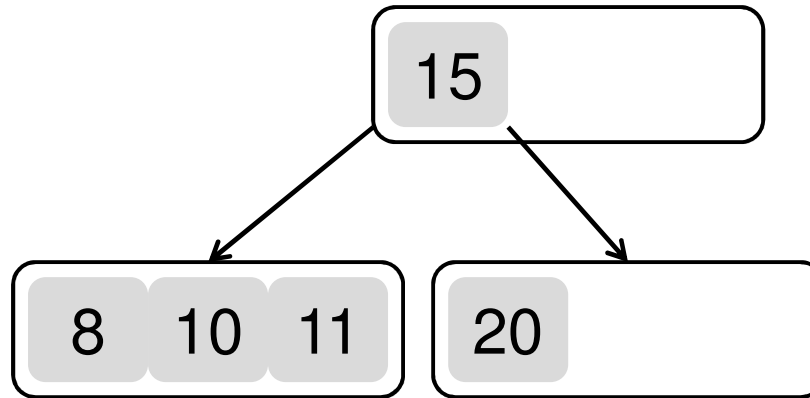
Inserción en Árboles B

Insertar(10)



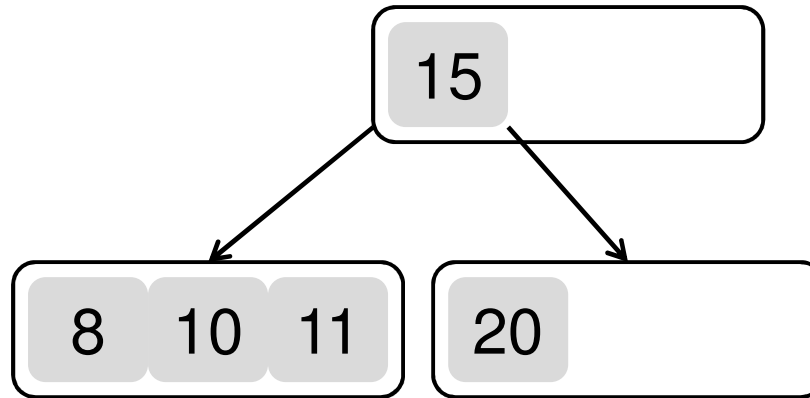
Inserción en Árboles B

Insertar(10)



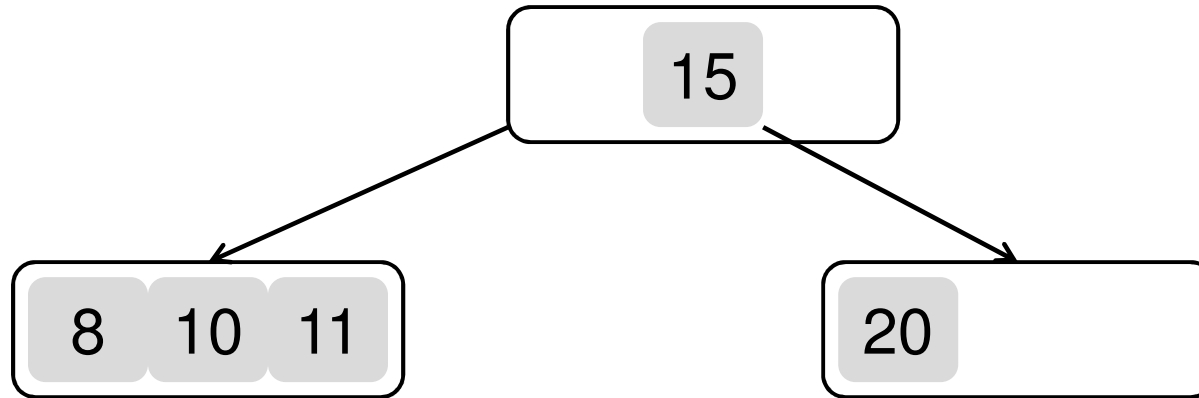
Inserción en Árboles B

Insertar(14)



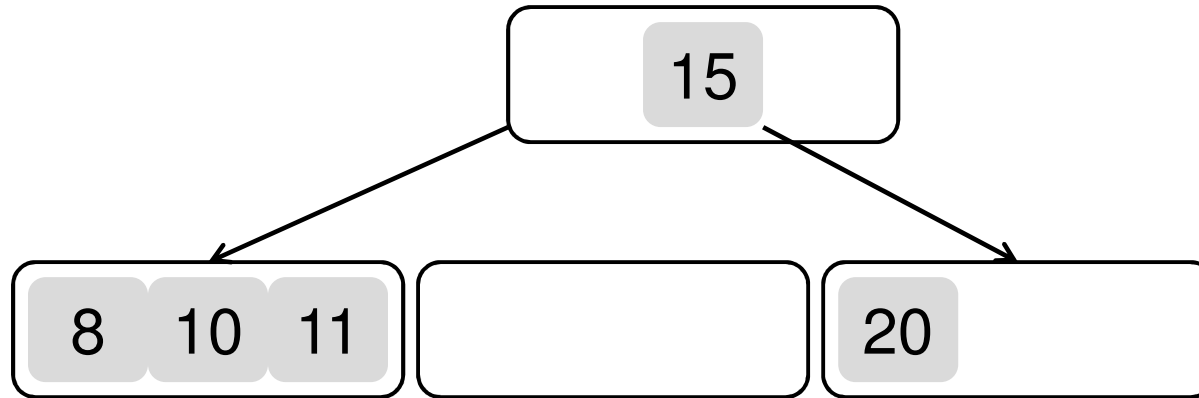
Inserción en Árboles B

Insertar(14)



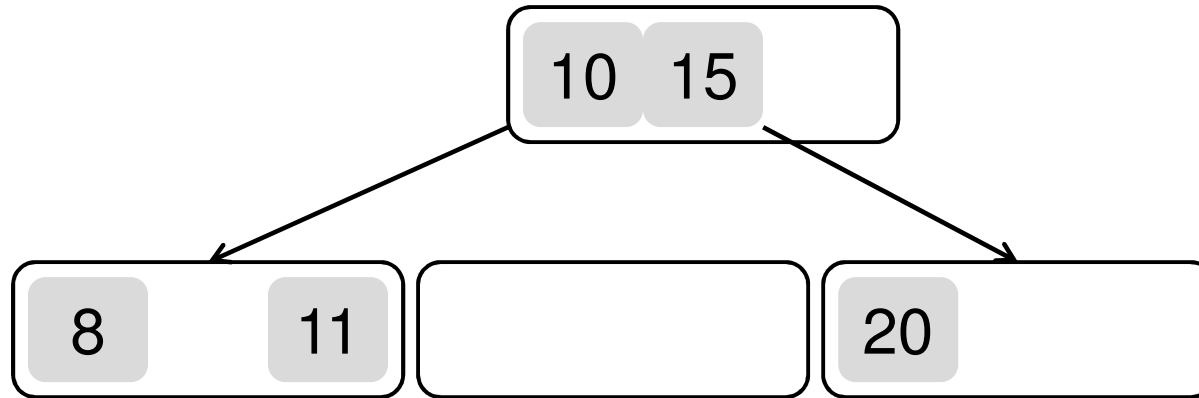
Inserción en Árboles B

Insertar(14)



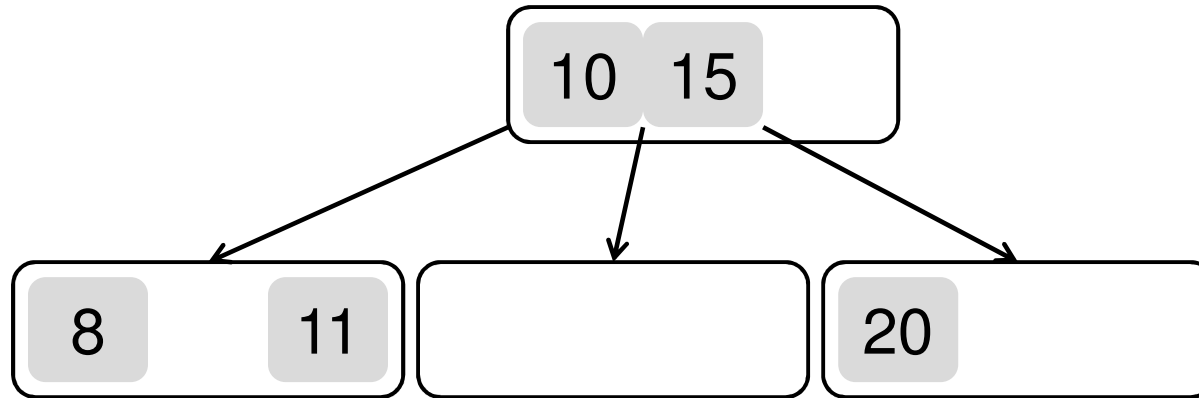
Inserción en Árboles B

Insertar(14)



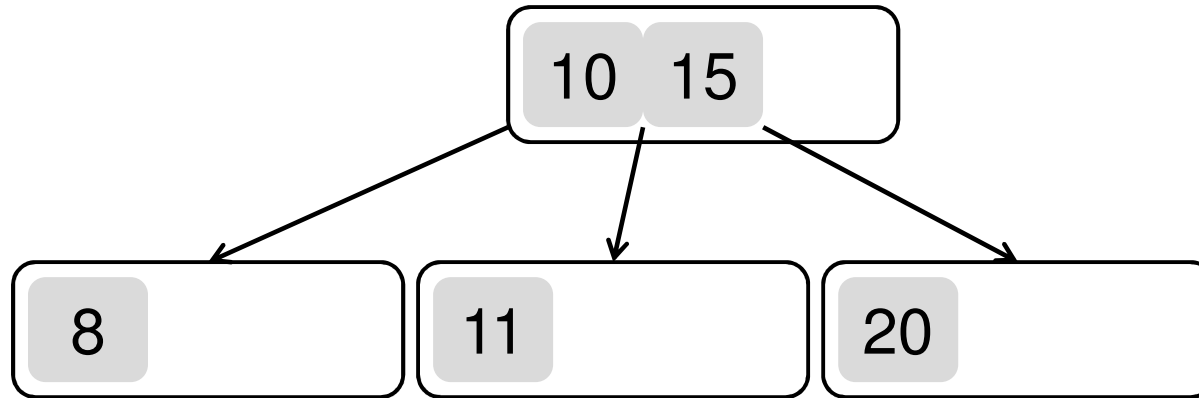
Inserción en Árboles B

Insertar(14)



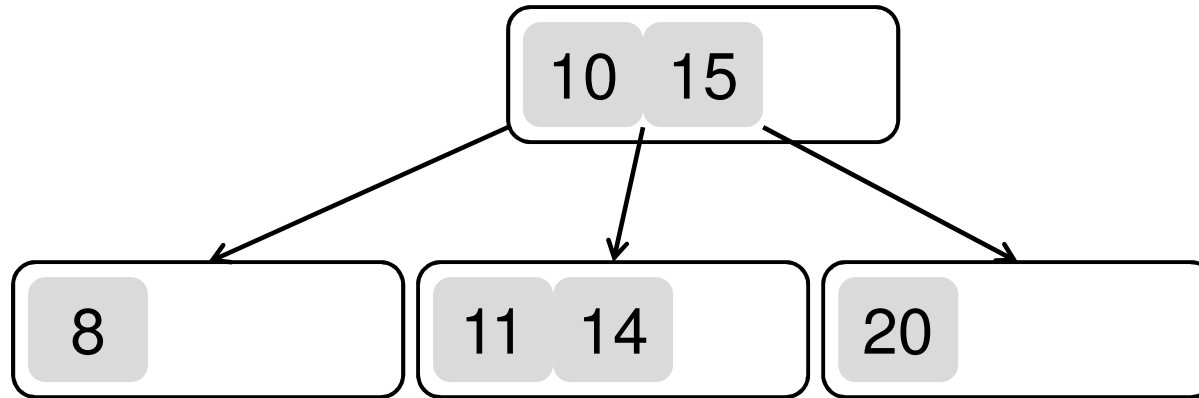
Inserción en Árboles B

Insertar(14)



Inserción en Árboles B

Insertar(14)



Árboles B. Inserción

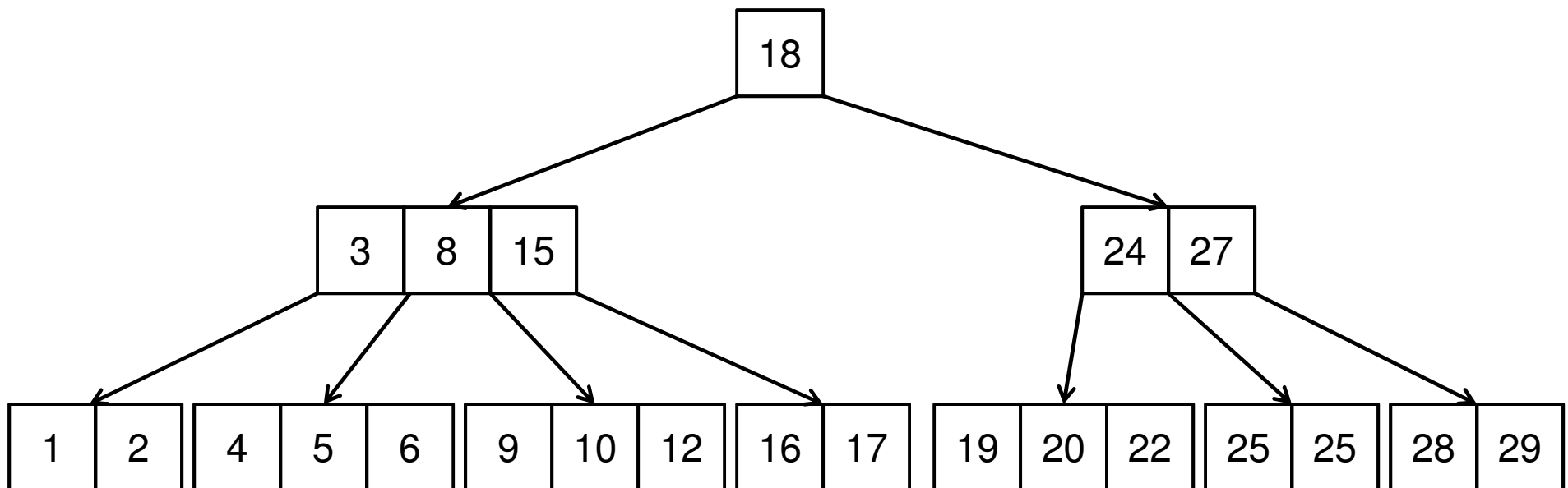
insertar(nodoB: r, tipoElemento: k)	
1	SI ($\text{claves}(r) = 2*t - 1$) s = nuevoNodo() raiz(T) = s nodo_hoja(s) = FALSE claves(s) = 0 hijo(x, 1) = r split(s, 1, r) insertar2(s, k)
2	SINO insertar2(r, k)

Árboles B. Inserción

insertar2(nodoB: x, tipoElemento: k)	
1	i = nodos(x)
2	SI (nodo_hoja(x)) MIENTRAS($i \geq 1$ AND $k < \text{clave}(x, i)$) $\text{clave}(x, i+1) = \text{clave}(x, i)$ $i = i - 1$ $\text{clave}(x, i+1) = k$ $\text{claves}(x) = \text{claves}(x) + 1$
3	EN CASO CONTRARIO MIENTRAS($i \geq 1$ AND $k < \text{clave}(x, i)$) $i = i - 1$ $i = i + 1$ $y = \text{hijo}(x, i)$ SI ($\text{claves}(y) = 2*t - 1$) split(x, i, y) SI ($k > \text{clave}(x, i)$) $i = i + 1$ insertar2(y, k)

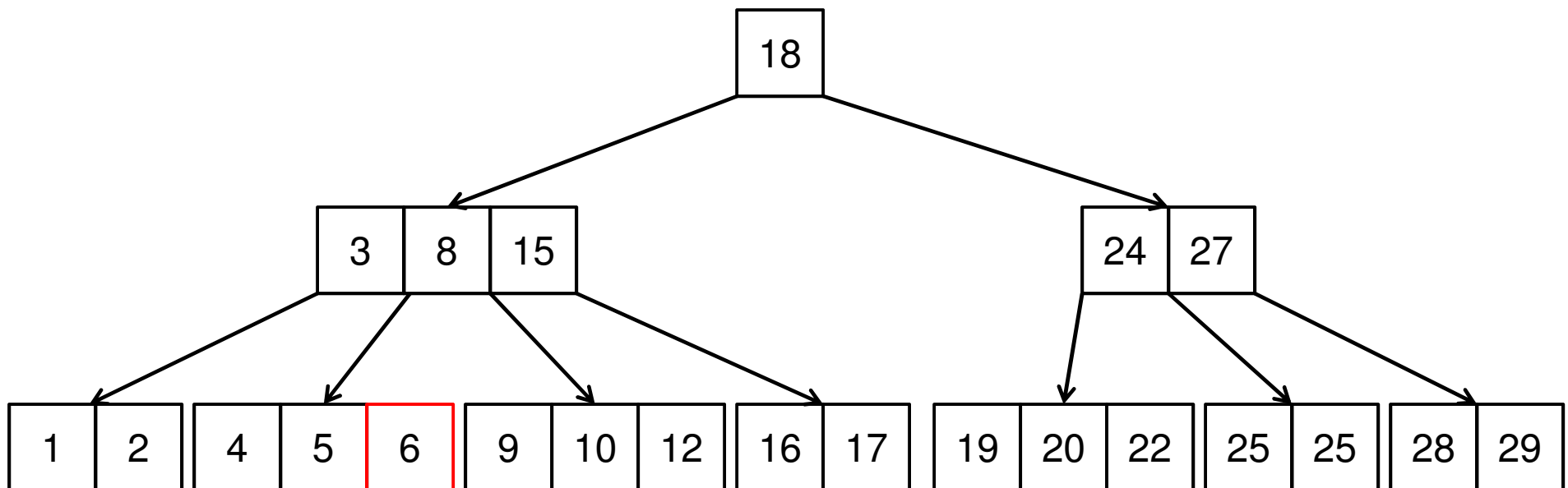
Eliminación en Árboles B

Eliminar(6)



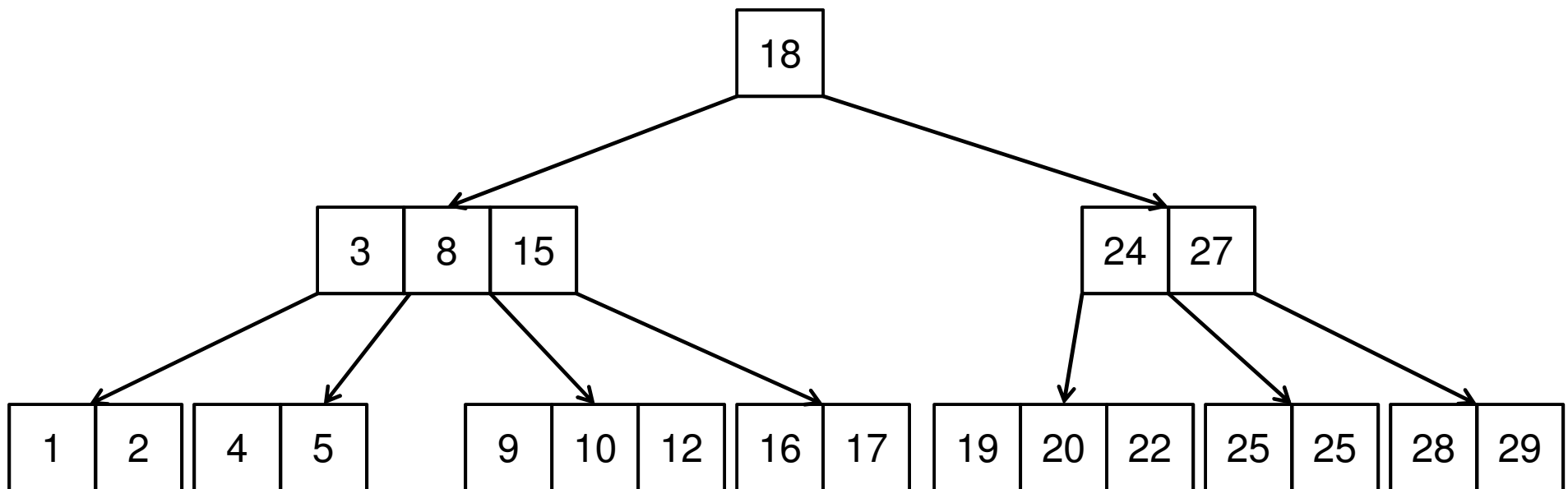
Eliminación en Árboles B

Eliminar(6)



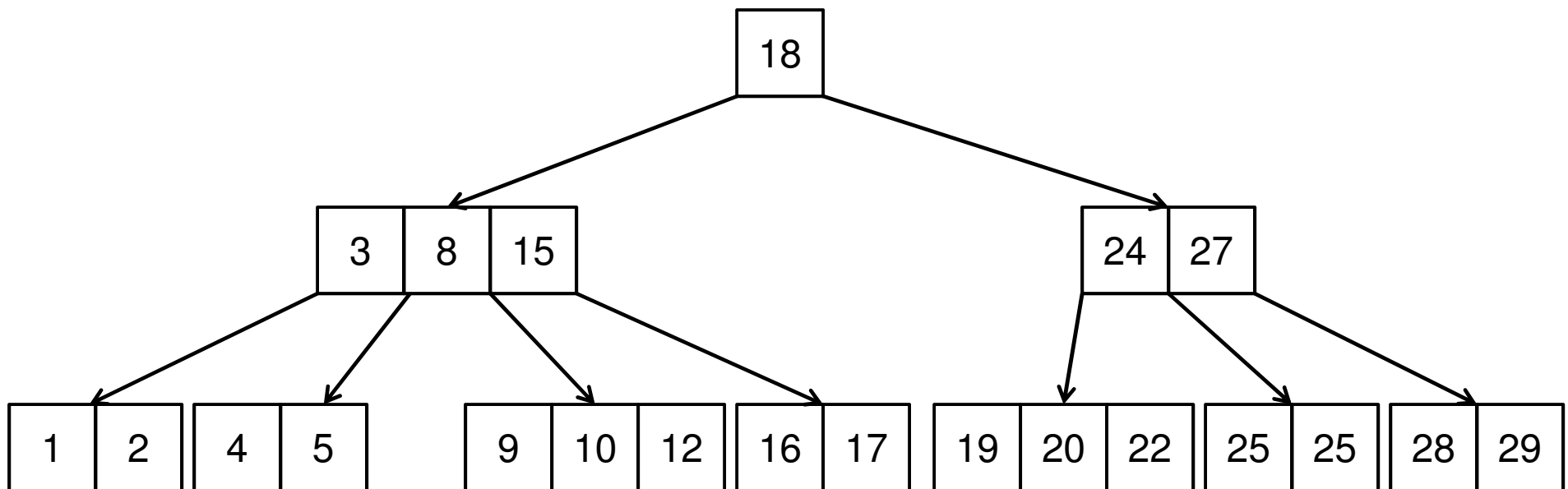
Eliminación en Árboles B

Eliminar(6)



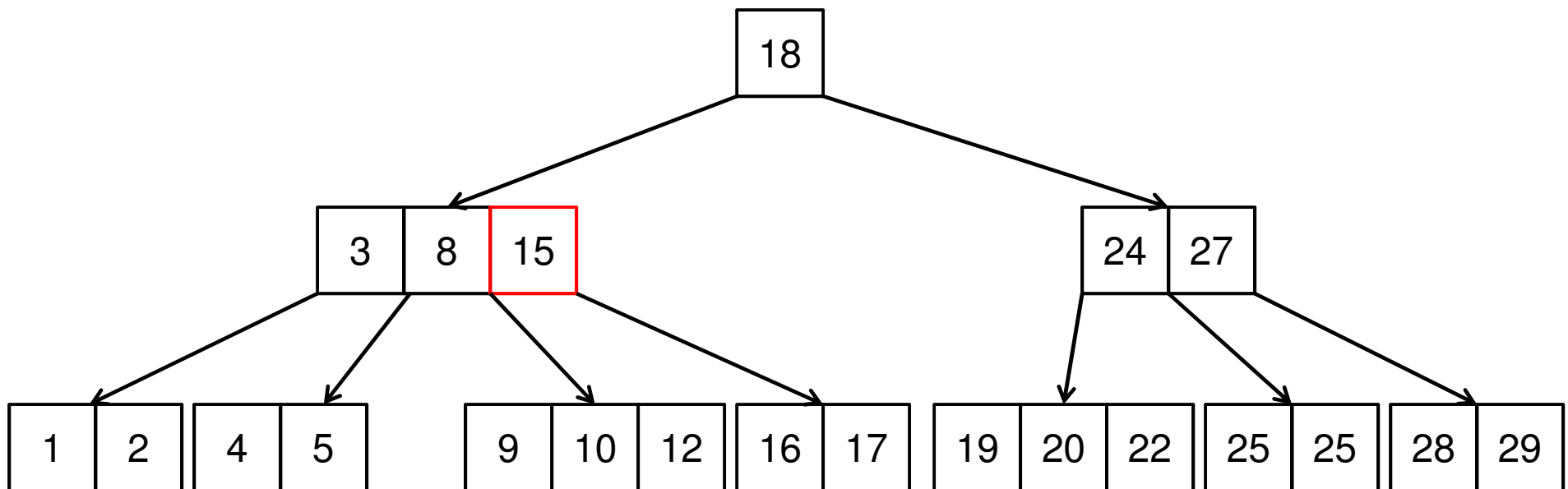
Eliminación en Árboles B

Eliminar(15)



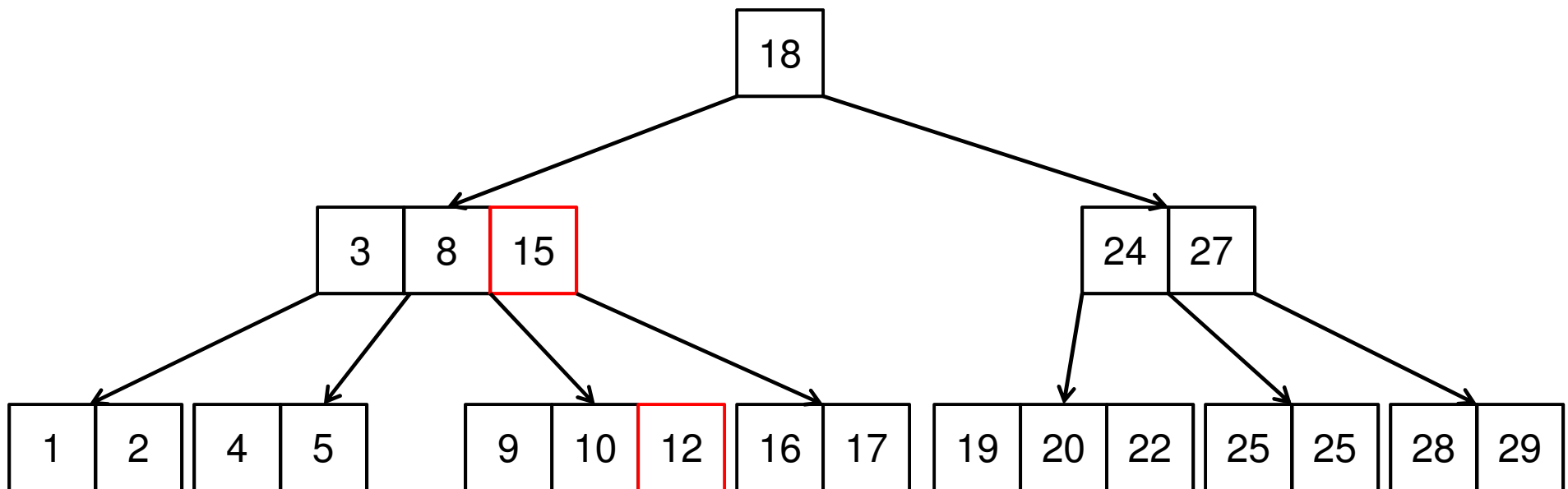
Eliminación en Árboles B

Eliminar(15)



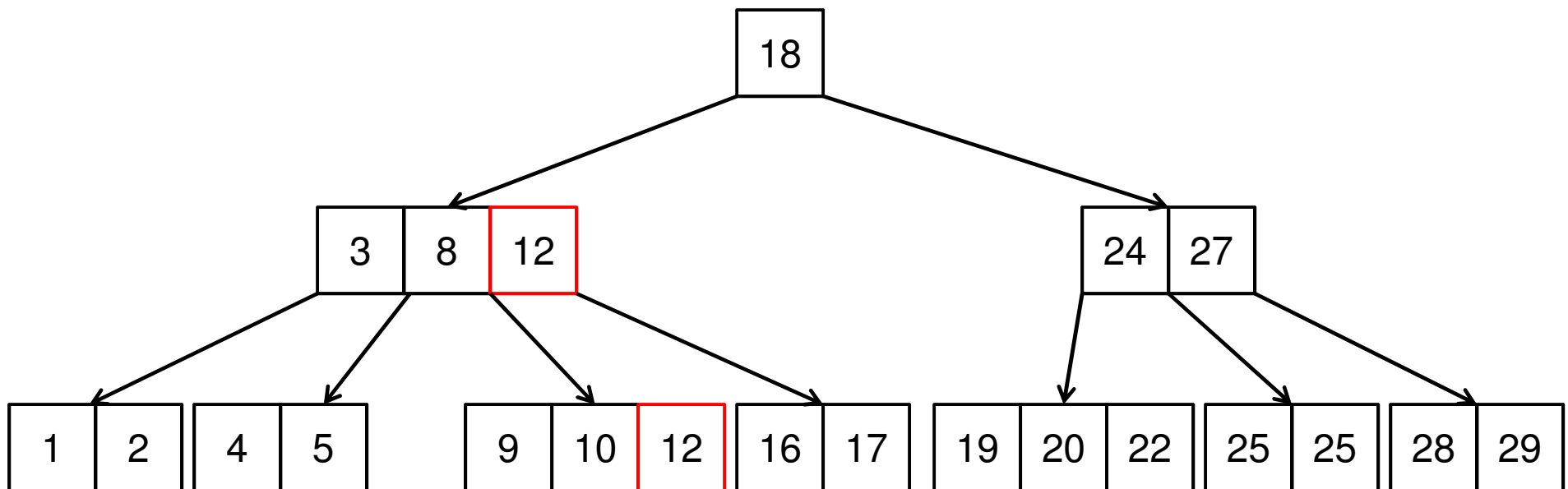
Eliminación en Árboles B

Eliminar(15)



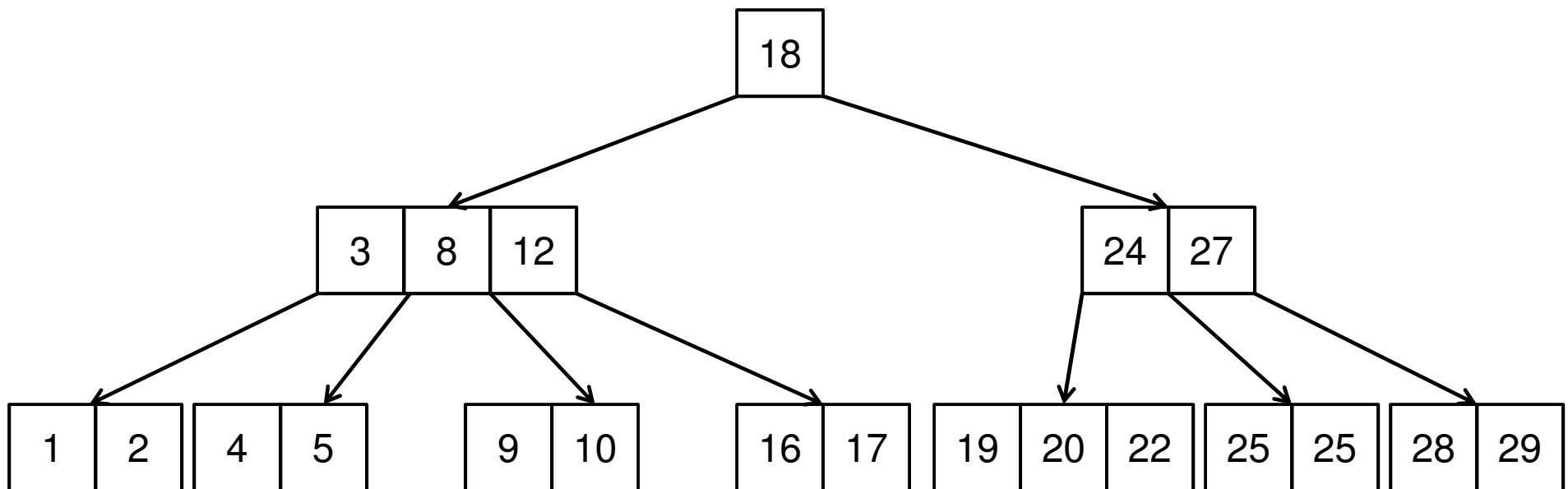
Eliminación en Árboles B

Eliminar(15)



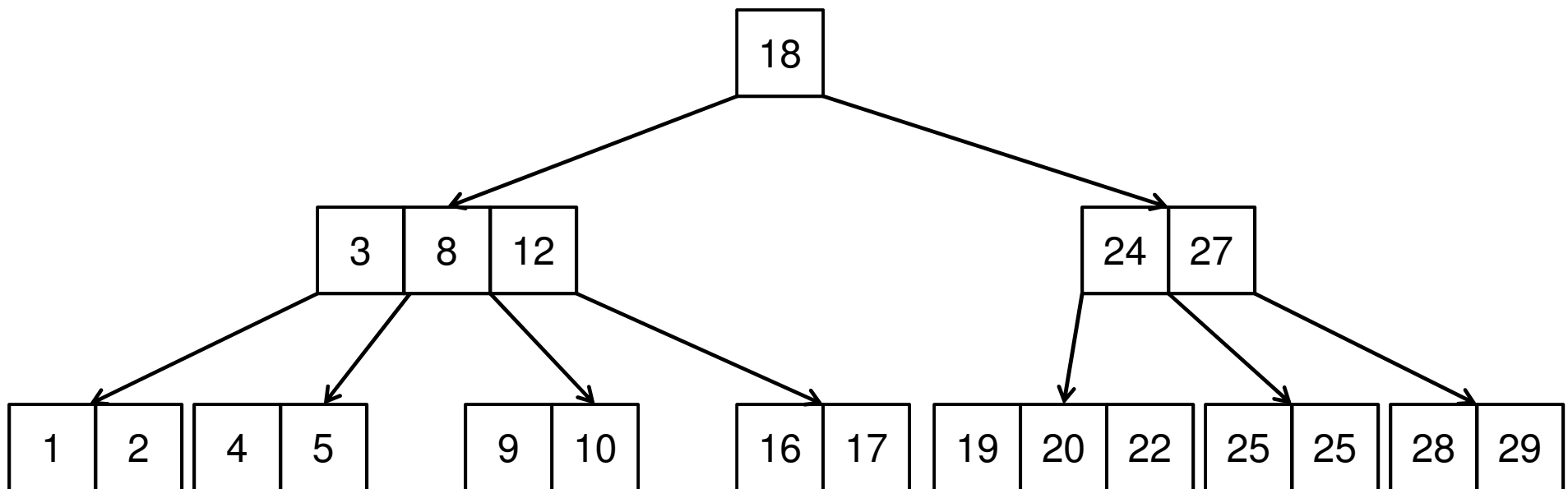
Eliminación en Árboles B

Eliminar(15)



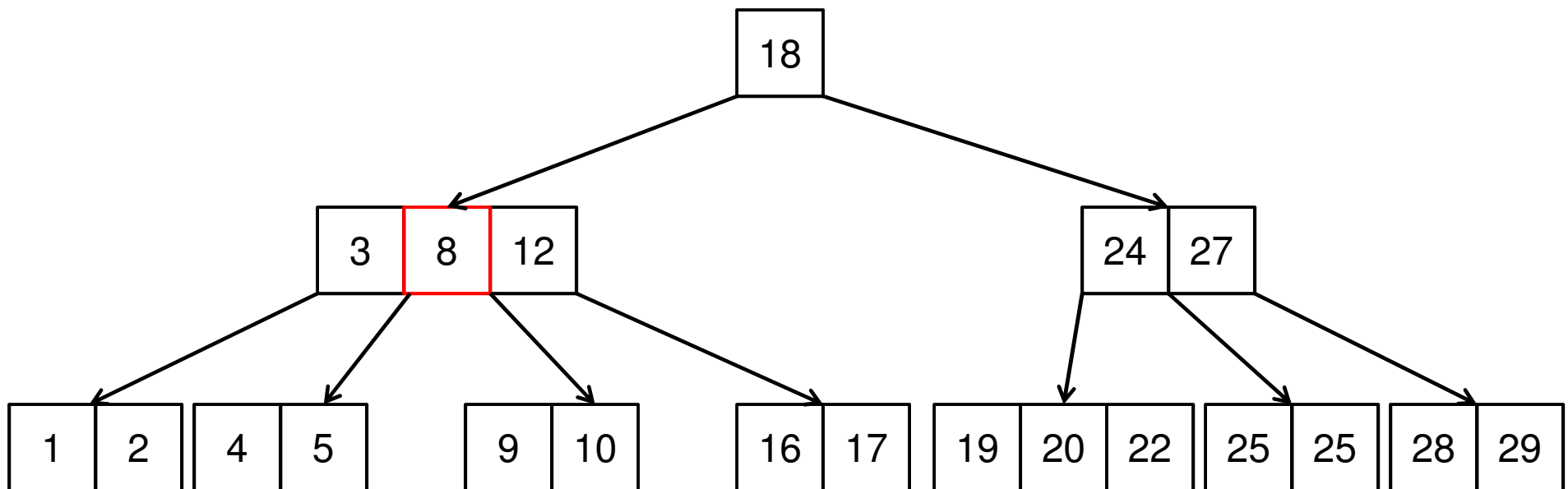
Eliminación en Árboles B

Eliminar(8)



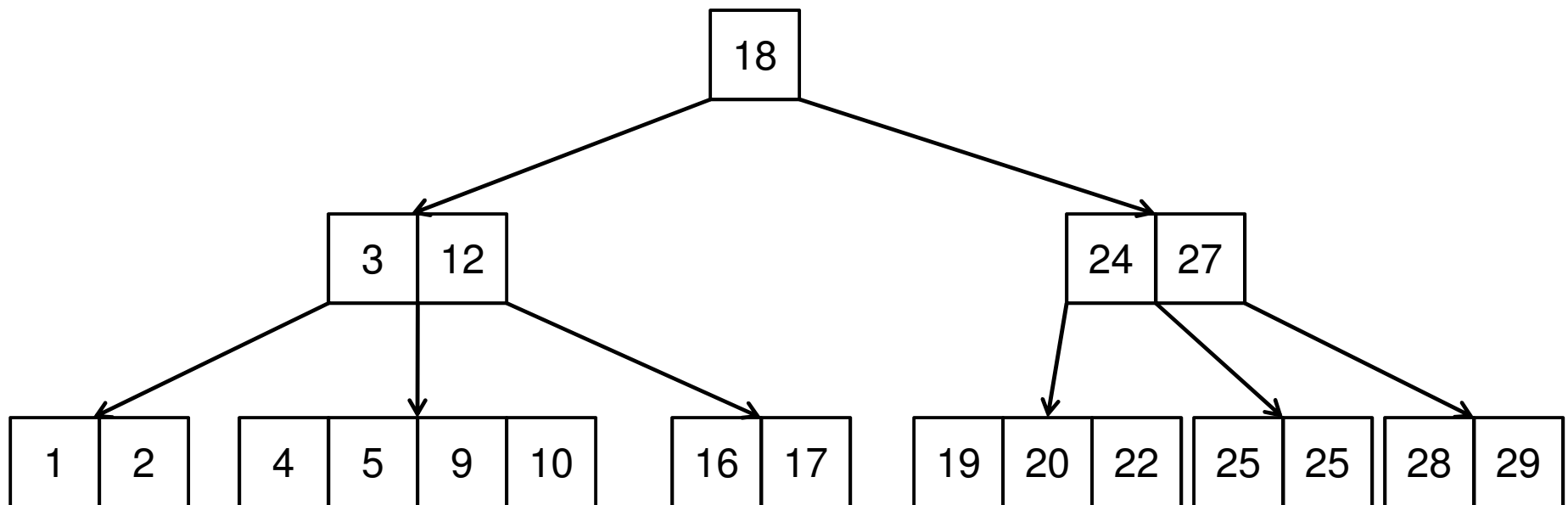
Eliminación en Árboles B

Eliminar(8)



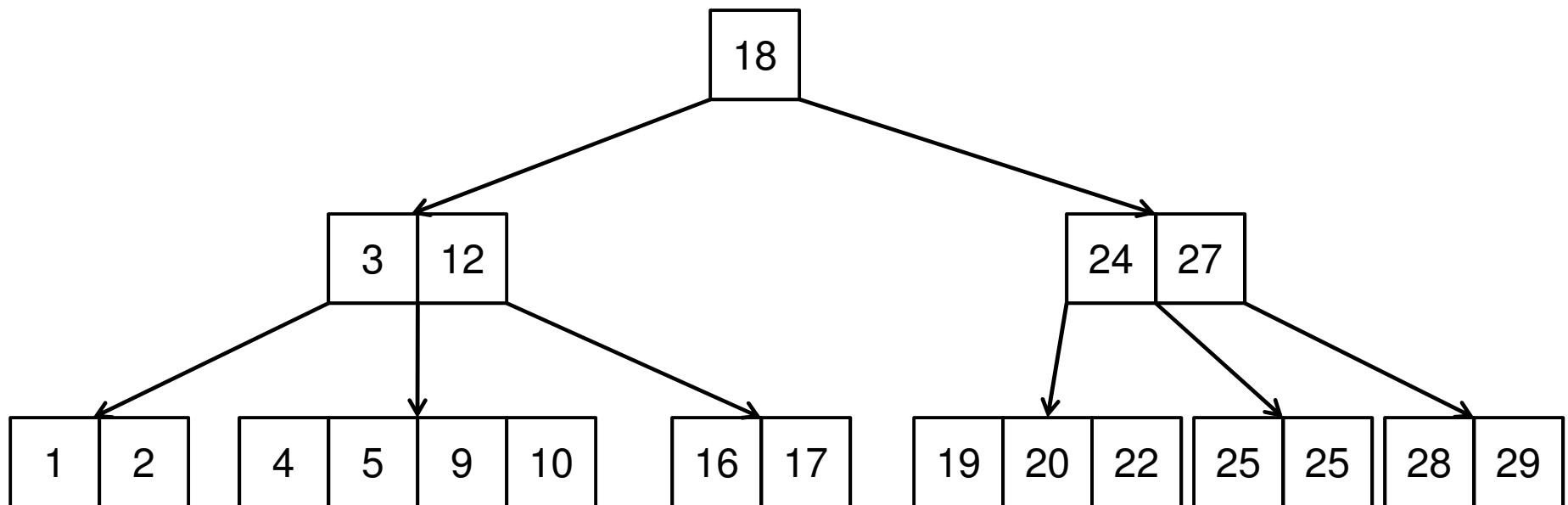
Eliminación en Árboles B

Eliminar(8)



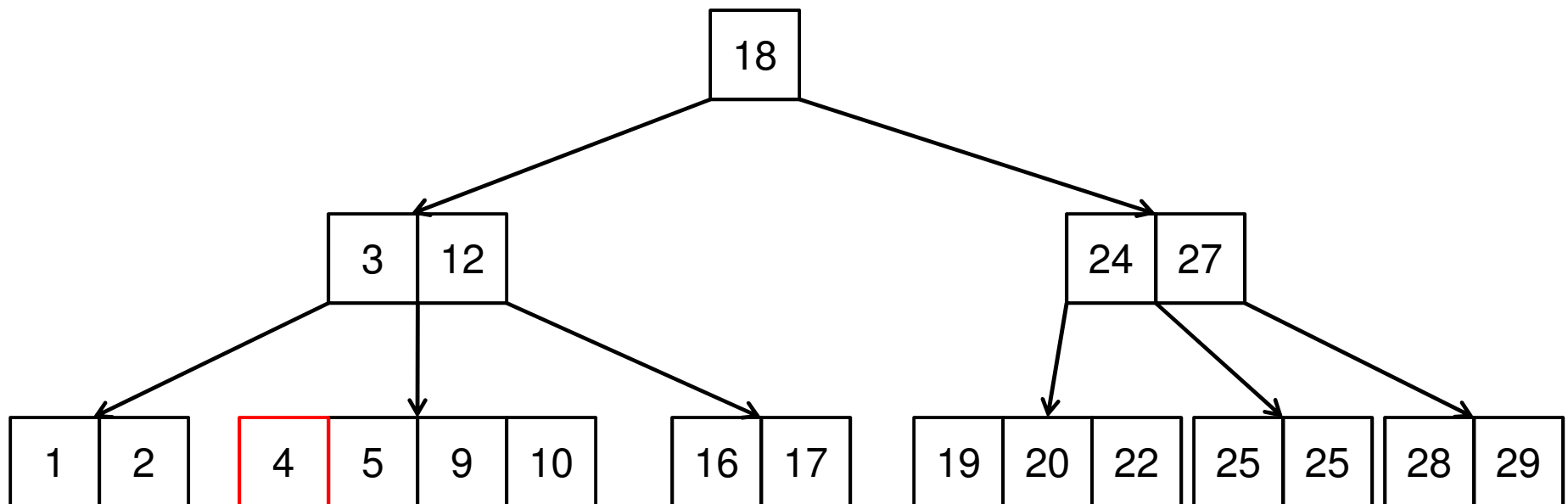
Eliminación en Árboles B

Eliminar(4)



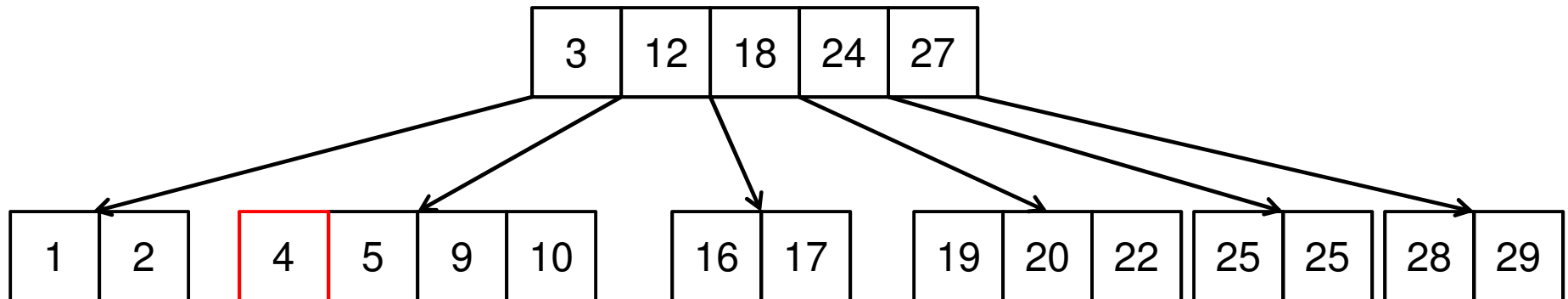
Eliminación en Árboles B

Eliminar(4)



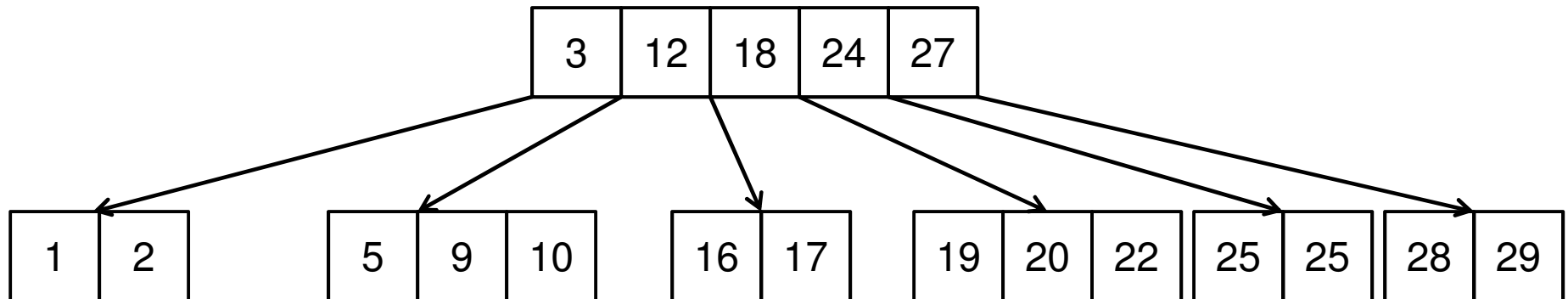
Eliminación en Árboles B

Eliminar(4)



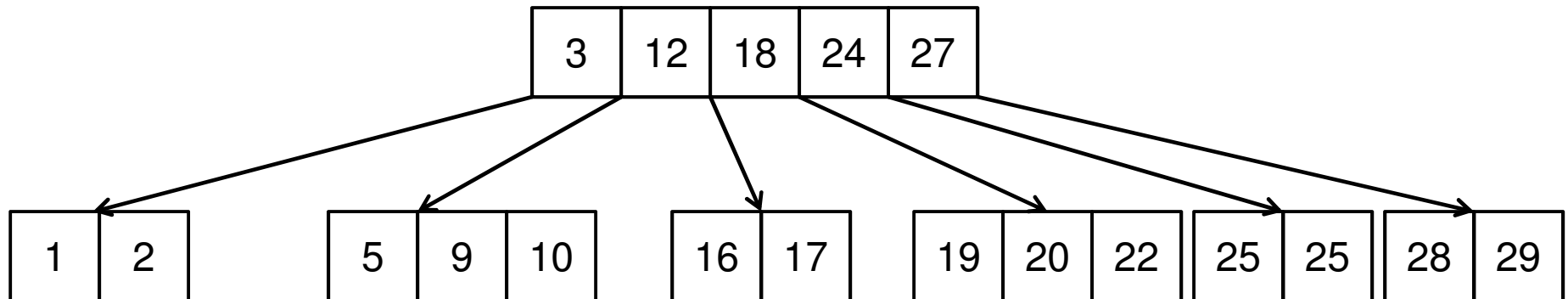
Eliminación en Árboles B

Eliminar(4)



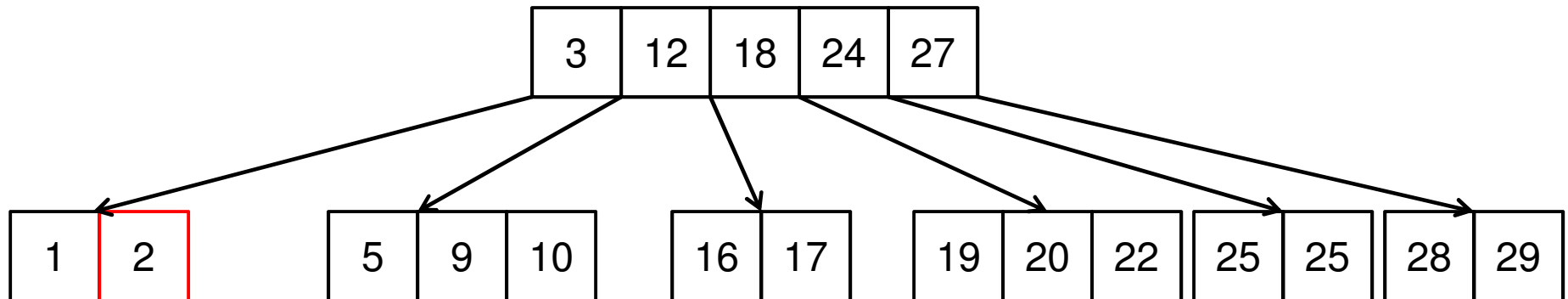
Eliminación en Árboles B

Eliminar(2)



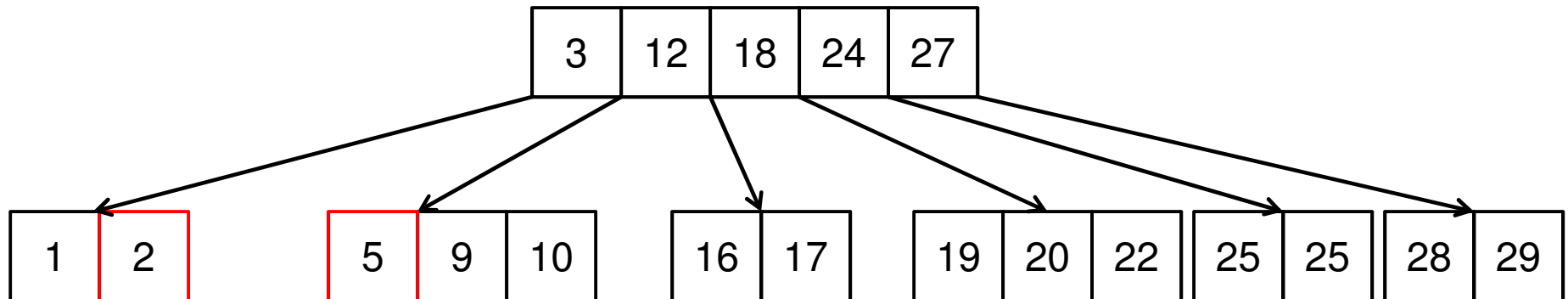
Eliminación en Árboles B

Eliminar(2)



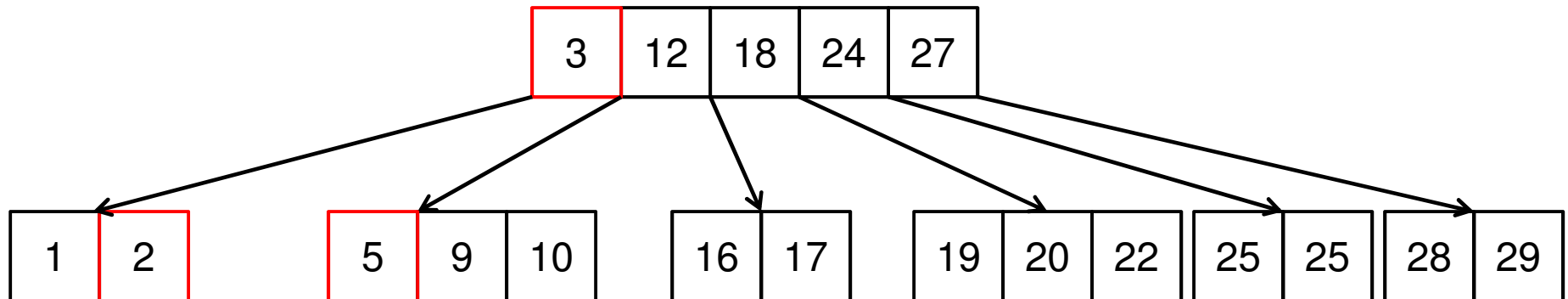
Eliminación en Árboles B

Eliminar(2)



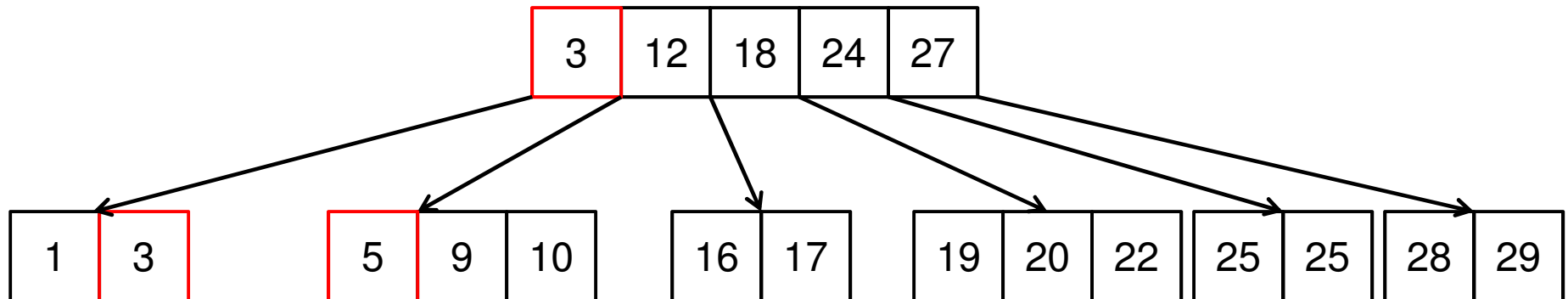
Eliminación en Árboles B

Eliminar(2)



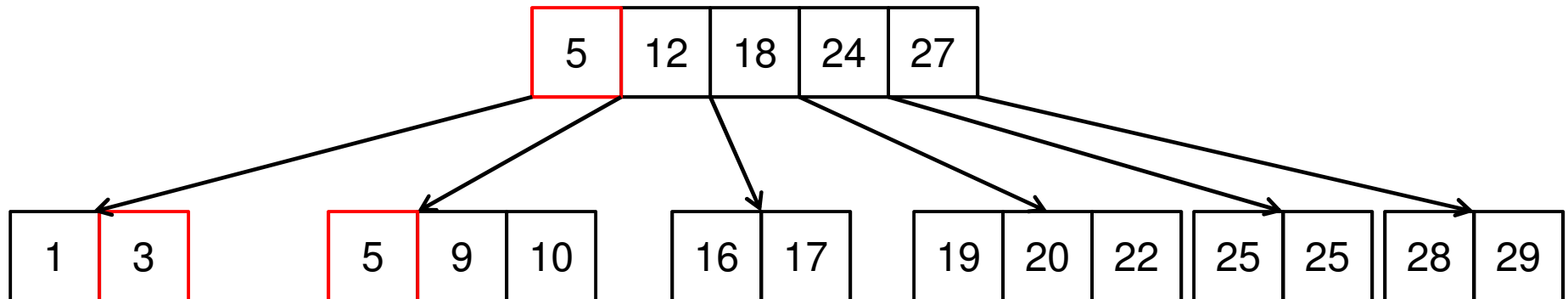
Eliminación en Árboles B

Eliminar(2)



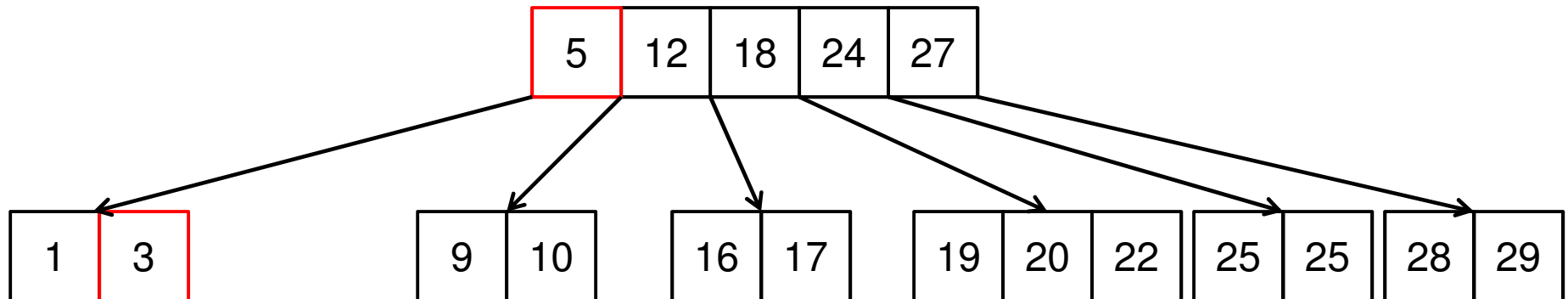
Eliminación en Árboles B

Eliminar(2)



Eliminación en Árboles B

Eliminar(2)

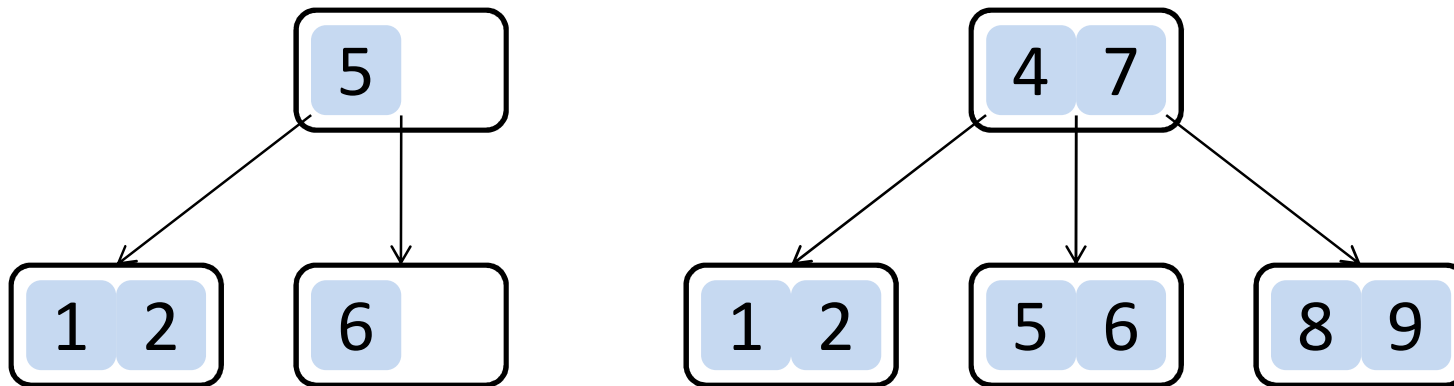


Árboles B. Análisis

Operación	Costo
búsqueda	$O(\log_t n)$
creación	$O(1)$
split	$O(t)$
inserción	$O(t \cdot \log_t n)$
eliminación	$O(t \cdot \log_t n)$

Árboles 2-3

- Son Árboles B en los que cada nodo puede tener 1 o 2 claves, y por consiguiente 2 o 3 hijos

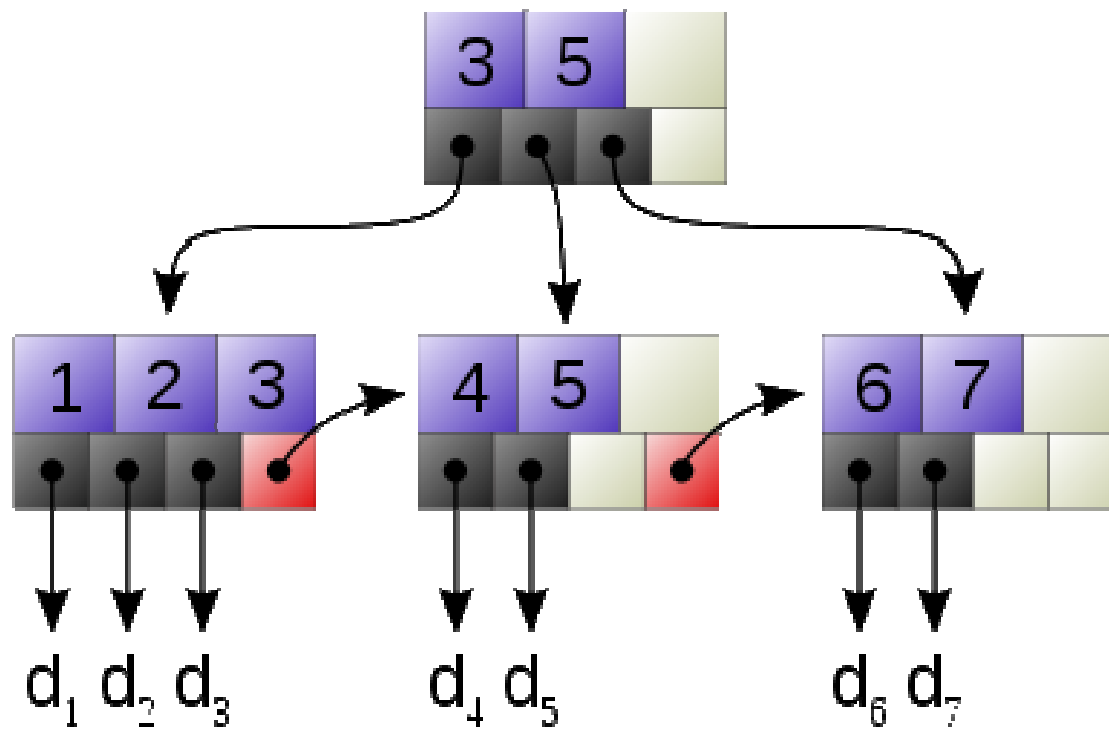


Árboles B+



- Variante de los Árboles B.
- Los datos se almacenan sólo en los nodos hoja.
- En los nodos no hoja se almacenan índices
- Los nodos hoja se encuentran enlazados

Árboles B+



Árboles B*



- Variante de los Árboles B
- Todos los nodos (a excepción de la raíz) deben estar al menos $2/3$ llenos
- Disminuye la sobrecarga de reorganizar el árbol en la inserción y eliminación
- Utilizados en los sistemas de archivos Reiser4 y HFS

Árboles B*



- Split
 - ▣ Pasar elementos al vecino
 - ▣ Cuando el vecino también esté lleno, convertir los 2 nodos en 3

- Mezcla
 - ▣ 3 nodos se convierten en 2