

Diseño de Conjuntos y Diccionarios

Esteban.

Ejemplos de diccionarios.

Cuando no nos interesa el significado
 \Rightarrow tenemos un conjunto

Diccionario :

Generalización de un conjunto

Representación secuencial de conjuntos y diccionarios.

Array $\langle \text{clave}, \text{sig} \rangle^*$

C:S
C:S
C:S

\emptyset definir def? obtener
 1 1 n n

ordenado

C:S
C:S
C:S

1 n $\log n$ $\log n$

Lista $\langle c, s \rangle$

1 1 n n

Lista Ord $\langle c, s \rangle$

1 n n n

Func. Característica

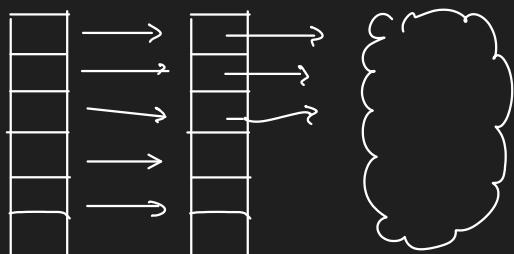
Todos los claves posibles de findar, con clave 0/1
dependiendo si está o no definido

0	5
1	5
1	5
0	5
...	...
...	...
0	5



Otra visión:

Claves Punteros Significan

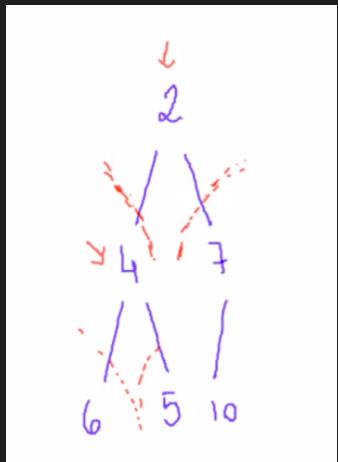


Ejemplo

Queríamos + Tiempo sublineal (por debajo de orden n)

Representación de Conj. y Dicts con

Arboles binarios



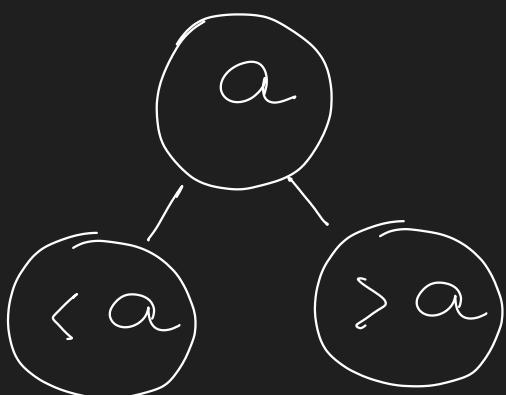
	\emptyset	def	def?	obt
	1	n	n	n

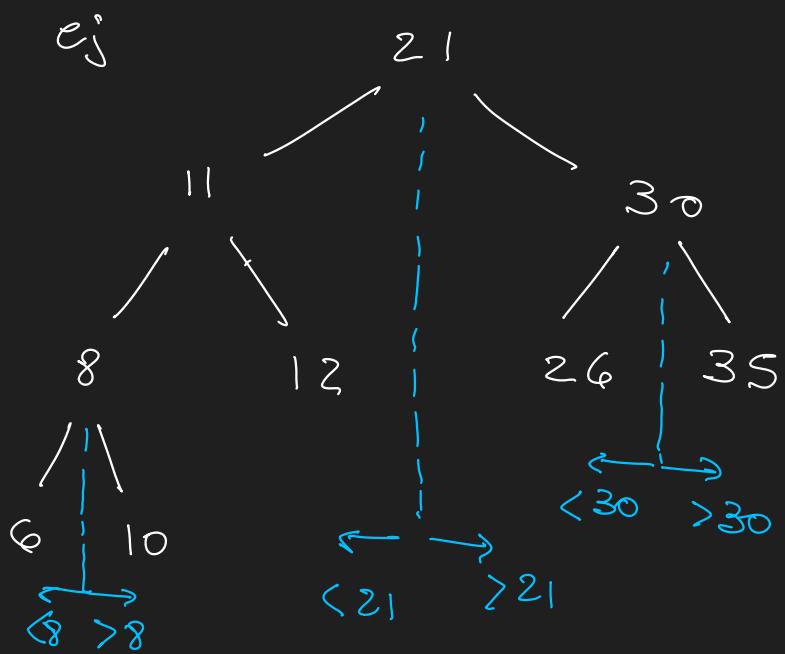
regla de completa nivel
en vez de agregar en cualquier.

ABB (BST: Binary Search Tree)

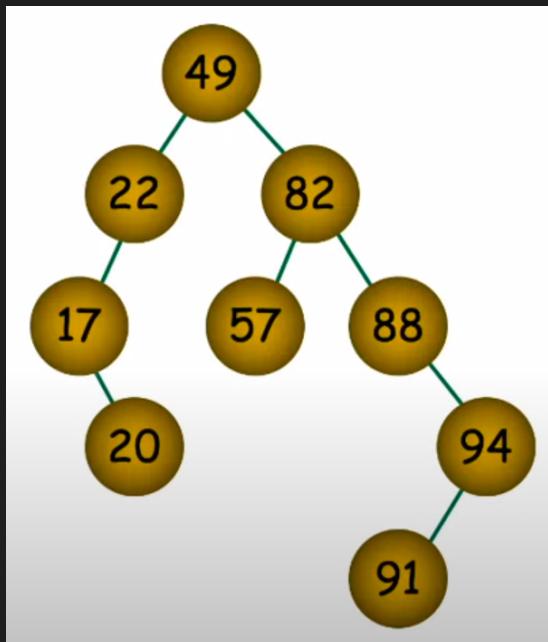
Es un AB que satisface la siguiente propiedad:

- Para todo nodo, los valores de los elementos en su subárbol izquierdo son menores que el valor del nodo, y los valores de los elementos de su subárbol derecho son mayores que el valor del nodo
- Dicho de otra forma, el valor de todos los elementos del subárbol izquierdo es menor que el valor de la raíz, el valor de todos los elementos del subárbol derecho es mayor que el valor de la raíz, y tanto el subárbol izquierdo como el subárbol derecho....son ABB.



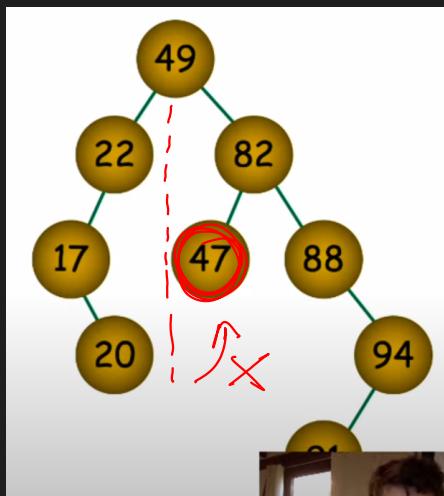


Es ABB ?



- Sí : Vale la prop. en cada nodo

Però alcanz \checkmark solo ver
cada nodo ?



No lo es !

■ Invariante de Representación:



ABB: ab(nodo) → boolean

$\forall e: ab(nodo) \xrightarrow{\text{es ABB si es nil}} , \text{sino:}$

$$ABB(e) \equiv \text{Nil?}(e) \vee_L$$

$$[\{(\forall c: \text{clave}: \text{Está}(c, \text{Izq}(e)) \Rightarrow (c <_{\text{clave}} \text{LaClave}(\text{Raíz}(e)))\} \wedge \text{↑}]$$

$$[(\forall c: \text{clave}: \text{Está}(c, \text{Der}(e)) \Rightarrow (c >_{\text{clave}} \text{LaClave}(\text{Raíz}(e)))\} \wedge \text{↑}]$$

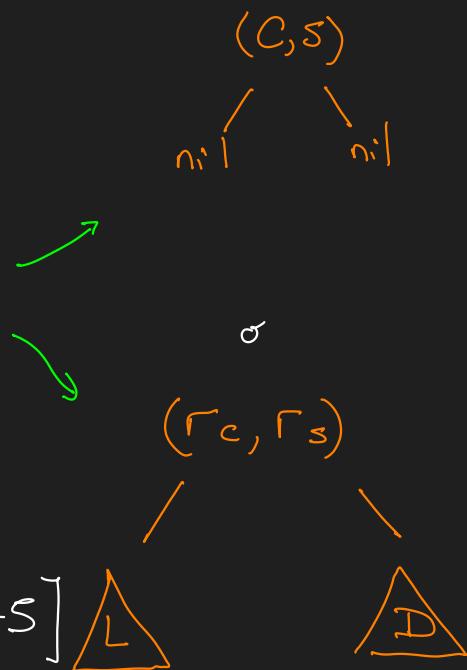
$$ABB(\text{Izq}(e)) \wedge \text{↑}$$

$$ABB(\text{Der}(e))]$$

necesitamos poder comparar claves
(relación de orden)

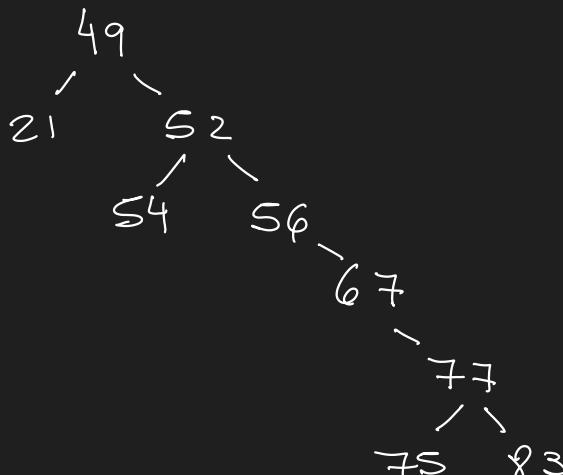
Algoritmos

- vacío()
 - nil
- definir(c,s,A)
 - Case
 - A=nil then bin(nil,(c,s),nil) else
 - A=bin(L,(r_c,r_s),D) then
 - if c<r_c then bin (definir(c,s,L), (r_c,r_s),D)
 - else bin(L, (r_c,r_s),definir(c,s,D))



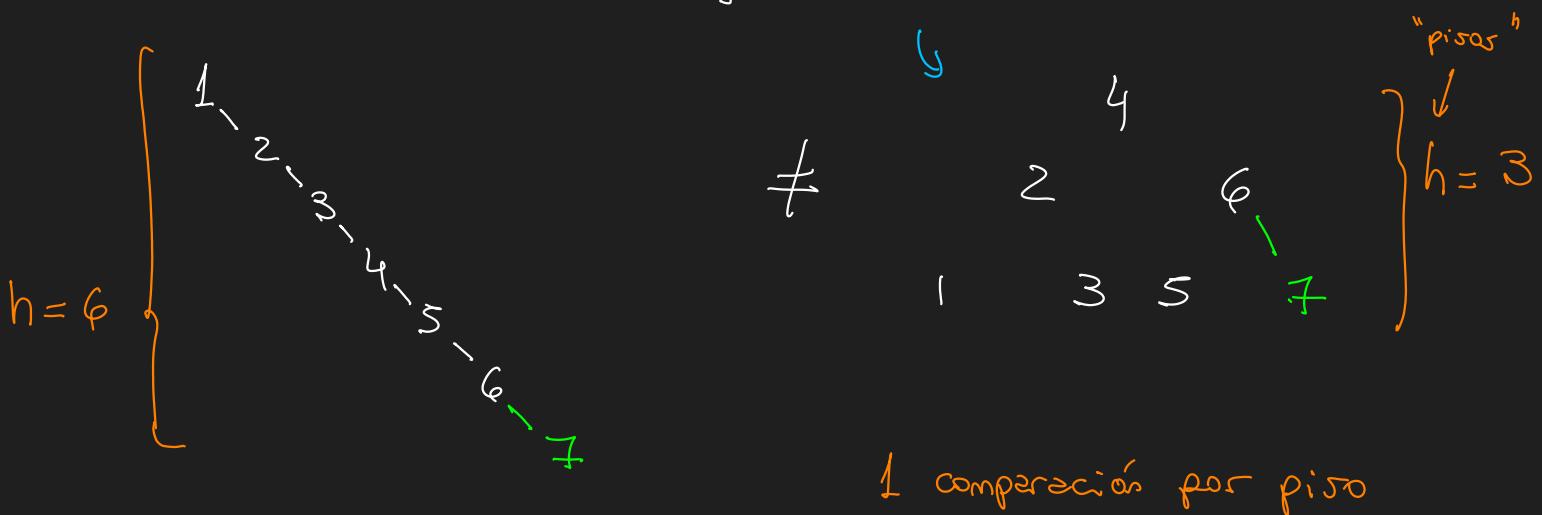
Llegan como

[49, 21, 52, 54, 67, 54, 77, 83, 75]



Distintos órdenes

$$S_1 = [1, 2, 3, 4, 5, 6, 7, 8]$$



Desbalanceados \rightarrow Listas enlazadas !

Inserción :

En el peor caso

$\hookrightarrow \Theta(n)$

En el caso promedio ($\text{Claves} \sim \text{Unif}$)

$\hookrightarrow \mathcal{O}(\log n)$

$$\text{Pero } h = \log_2(n+1) - 1$$

Borrado

- Borrar(u, A)
- Tres casos
 1. u es una hoja
 2. u tiene un solo hijo
 3. u tiene dos hijos

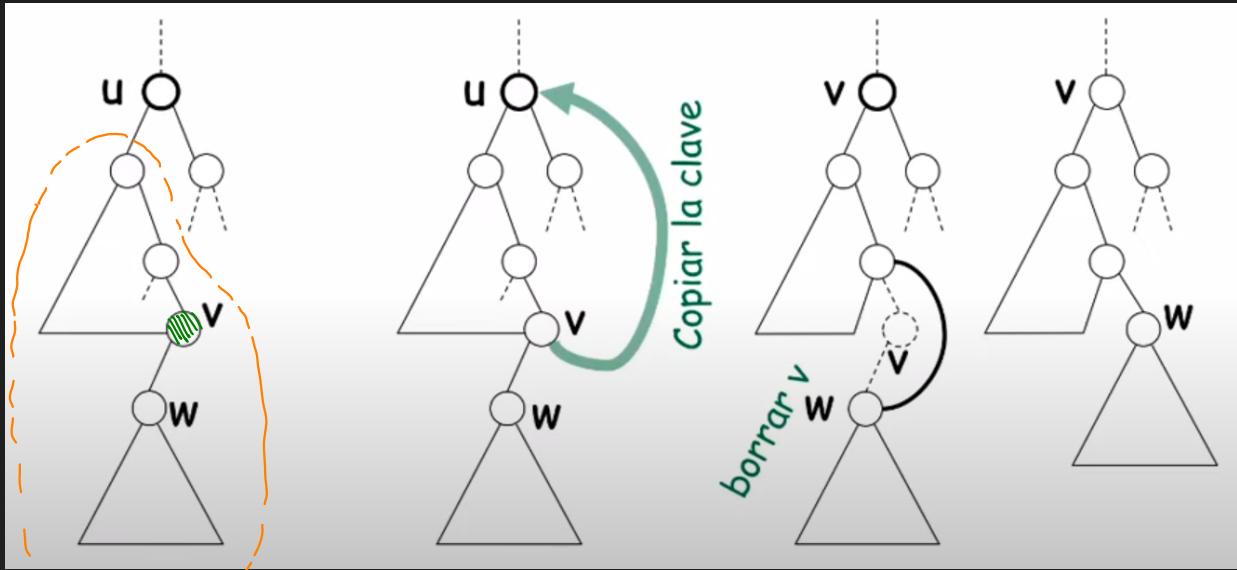
2. Borrar un nodo u con un solo hijo v

- Buscar al padre w de u
- Si existe w , reemplazar la conexión (w,u) con la conexión (w,v)



3. Borrado de un nodo u con dos hijos

- Encontrar el “predecesor inmediato” v (o sucesor inmediato) de u
 - v no puede tener dos hijos, en caso contrario no sería el predecesor inmediato (sucesor)
- copiar la clave de v en lugar de la de u
- Borrar el nodo v
 - v es hoja, o bien tiene un solo hijo, lo que nos lleva los casos anteriores



2.

más de

"v es el que está más a la derecha, de los que están a la izquierda de u."

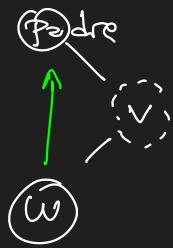
o tiene 1 hoja

o tiene 1 hijo IZQUIERDO

(Nunca hijo Derecho!)

4. Si v es hoja \Rightarrow solo lo muevo

• Si no \Rightarrow conecto w con el padre de v



Teniendo min y max

↳ Tengo predecesor y sucesor inmediato.

~ Vuelve Esteban ~~~

El costo depende de la profundidad del árbol

↳ Balanceo



Agregar a un ABB balanceado, lo desbalancea

Cant de nodos $2^n - 1$

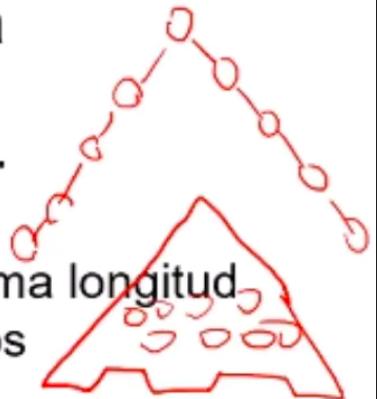
2-1	→	1			
4-1	↖	2	3		
8-1	↖	4	5	6	7

Por eso fijo en buscar:

$\log_2 n \leftarrow$ altura del árbol

Problema "equivalente" → búsquedas binarias,

- Pero...no podemos pretender tener siempre árboles completos (¡mantenerlos completos sería demasiado caro!)
- Quizás con alguna propiedad más débil...
- Noción intuitiva de balanceo
 - Todas las ramas del árbol tienen "casi" la misma longitud
 - Todos los nodos internos tienen "muchos" hijos
- Caso ideal para un arbol k -ario
 - Cada nodo tiene 0 o k hijos
 - La longitud de dos ramas cualesquiera difiere a lo sumo en una unidad



Demo de Teo

■ Teo: Un árbol binario perfectamente balanceado de n nodos tiene altura $\lfloor \lg_2 n \rfloor + 1$

Dem: Si cada nodo tiene 0 o 2 hijos

$$n_h = n_i + 1$$

n_h = # hojas
 n_i = # nodos internos
 $n = n_h + n_i$

luego de agregar

$$n'_h = n_h + 2 - 1$$

$$= n_h + 1$$

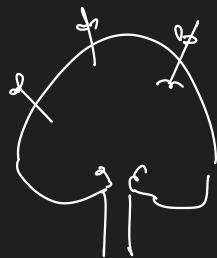
$$= n_i + 1 + 1$$

$$= n'_i + 1$$

Obs

>50% del árbol son hojas!

Podar



1º hojar de ramas largas

2º todo las hojas

Podemos poderlo

$\log_2 n$ veces (su altura)

Costo de

- Búsqueda
 - Inserción
 - Borrado
- $\} \quad O(\log n)$

Pero se desbalancea el nodo hacerlo.

Quiero algo intermedio

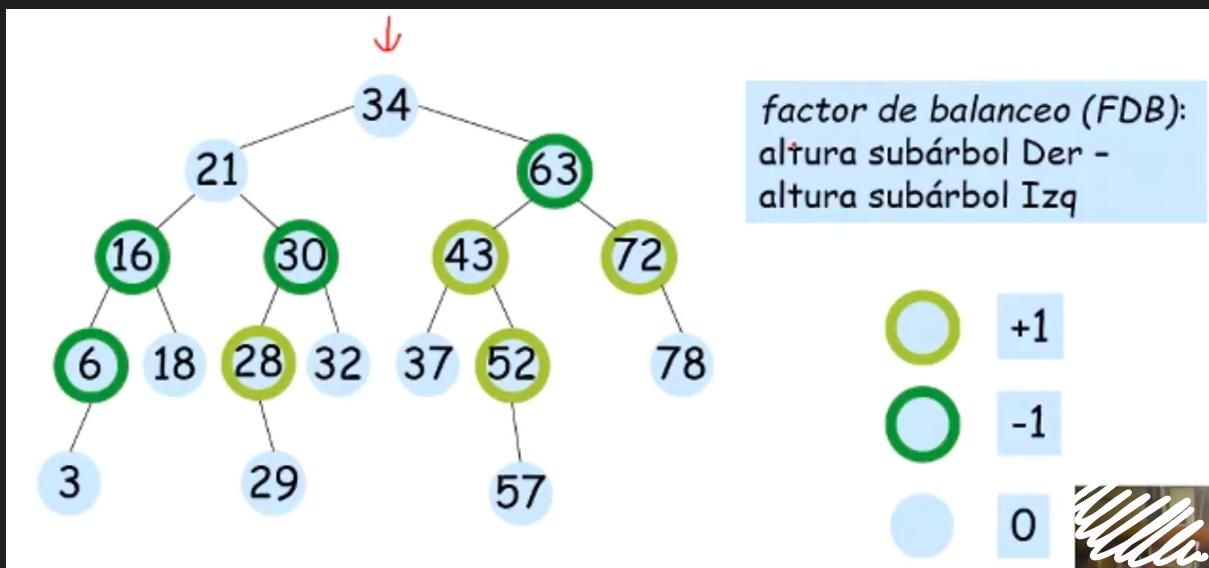
Balances en altura.

- Un árbol se dice balanceado en altura si las alturas de los subárboles izquierdo y derecho de cada nodo difieren en a lo sumo una unidad

Los árboles balanceados en altura se llaman
árboles AVL

Factor de Balanceo

$$\text{alt}(\text{Subárbol Der}) - \text{alt}(\text{Subárbol Izq})$$

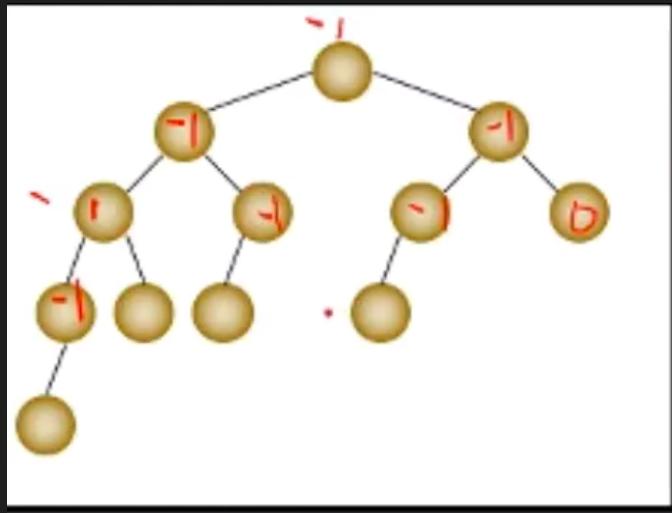


En un árbol balanceado en altura

$$|FDB| \leq 1$$

para cada nodo.

Todos factores son iguales



¿ Cuán altos son los AVL ?

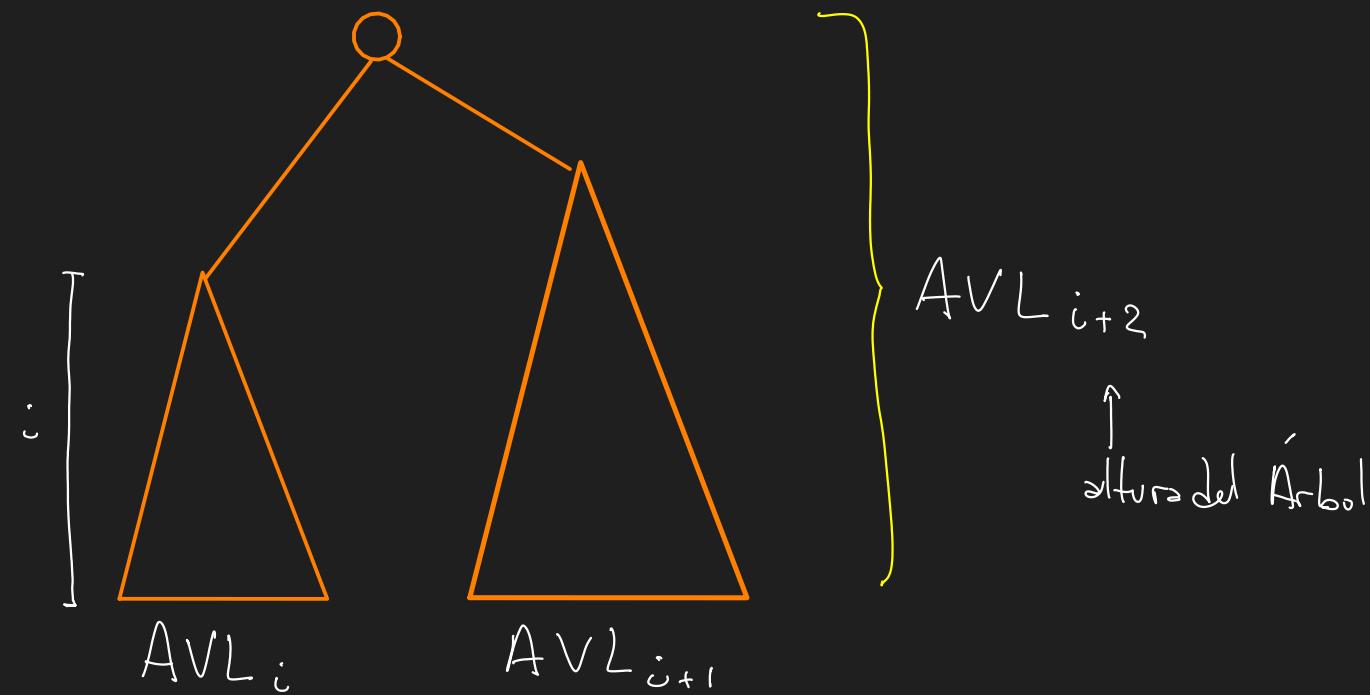
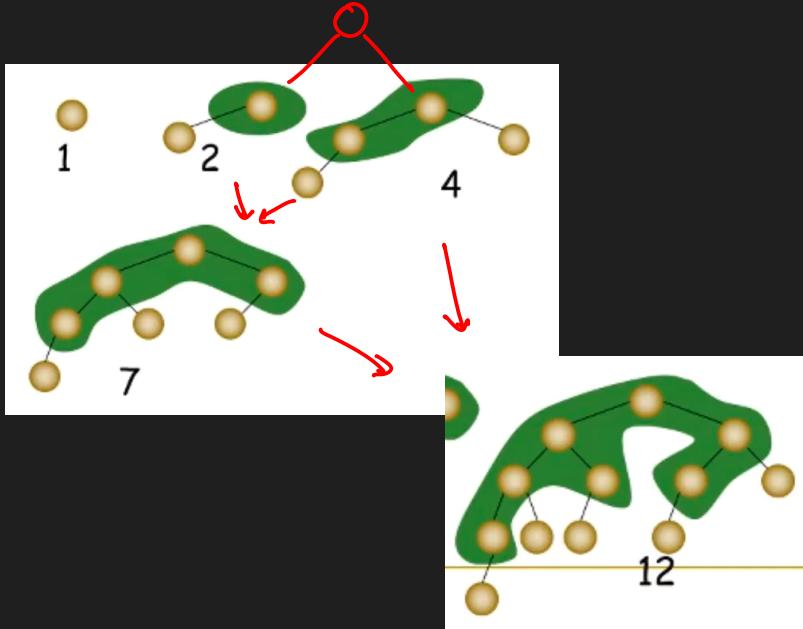
- Si quiero demostrar que la altura de un árbol con n nodos es menor que $f(n)$, eso es lo mismo que demostrar que un árbol con altura x tiene más que $f^{-1}(x)$ nodos
- Por si no se entendió: Si quiero demostrar que para empaquetar n caramelos me alcanza con $f(n)$ cajas, eso es lo mismo que demostrar que en x cajas caben más que $f^{-1}(x)$ caramelos
- Por si todavía no se entendió: Si quiero demostrar que para empaquetar n caramelos me alcanza con $n/20$ cajas, eso es lo mismo que demostrar que en x cajas caben más que $x^{*}20$ caramelos



Por lo tanto, si quiero demostrar que la altura de un árbol con n nodos es menor que $\log(n)$, eso es lo mismo que demostrar que un árbol con altura x tiene más que $\exp(x)$ nodos.

Árboles de Fibonacci

Árboles AVL con el mínimo número de nodos
(dada la altura)



- $\text{AVL}_{i+2} = \text{AVL}_{i+1} + \text{AVL}_i + 1$

h	F_h	AVL_h
0	0	0
1	1	1
2	1	2
3	2	4
4	3	7
5	5	12
6	8	20
7	13	33

Fibonacci

$$F_{i+2} = F_i + F_{i+1}$$

- Gran propiedad

$$\text{AVL}_i = F_{i+2} - 1$$

F_i es exponencial en i

(fórmula cerrada de Fibonacci)

∴ la altura de un árbol con n nodos

es $\mathcal{O}(\log n)$

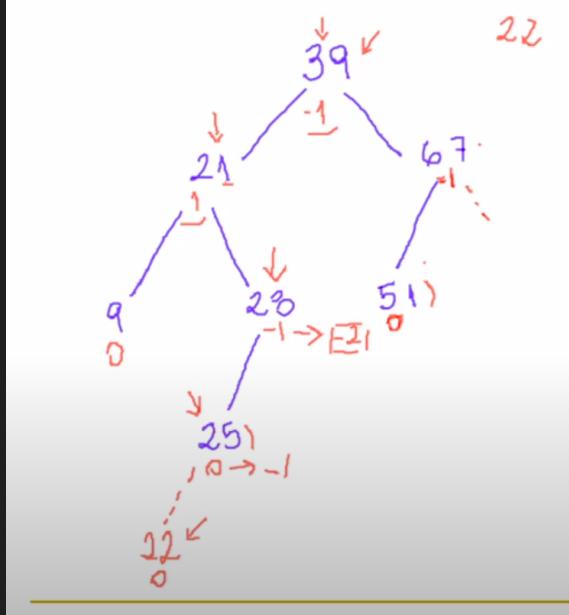
más precisamente.

$$1.44 \cdot \log(n+2) - 0.328$$

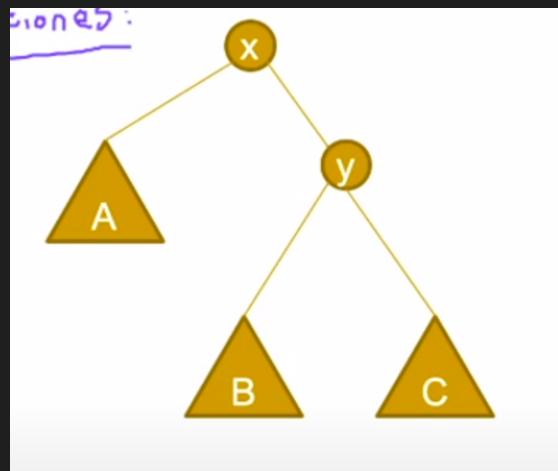
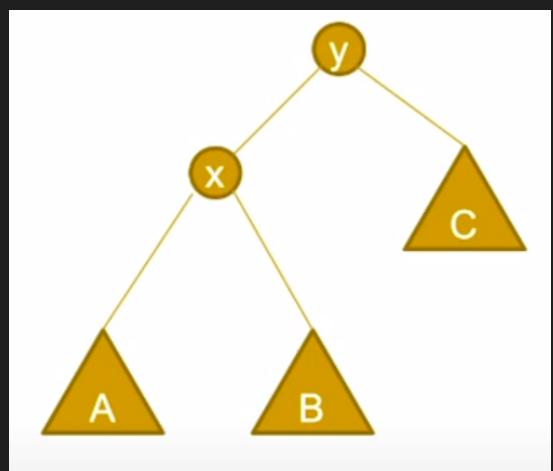
⇒ un AVL de n nodos tiene altura

$\Theta(\log n)$

Inserción en AVL



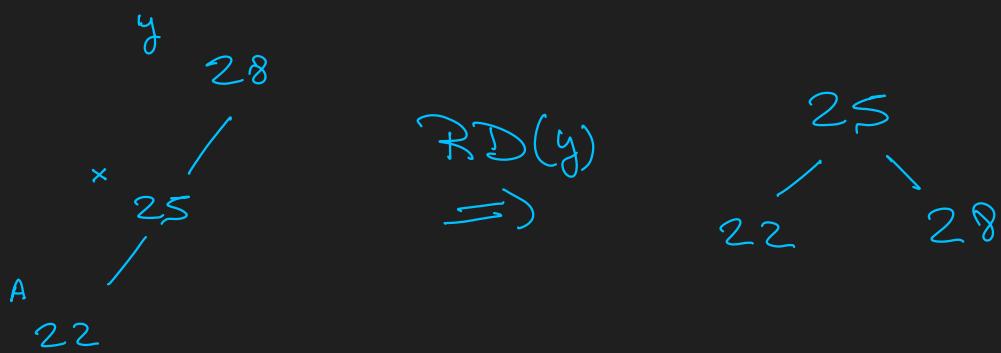
Rotaciones

Rotación Izquierda ($RL(x)$) $RL(x)$ Rotación Derecha ($RD(y)$) $RD(y)$

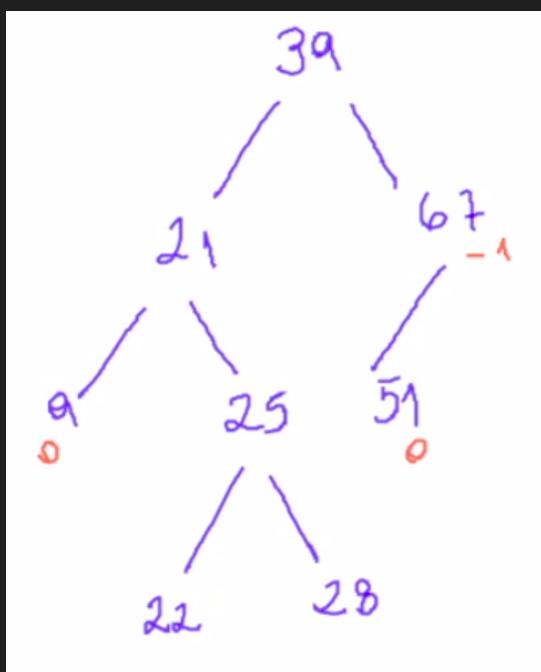
¿A dónde aparece la relación de balanceados?

Donde se ve la nueva "flexibilidad"?

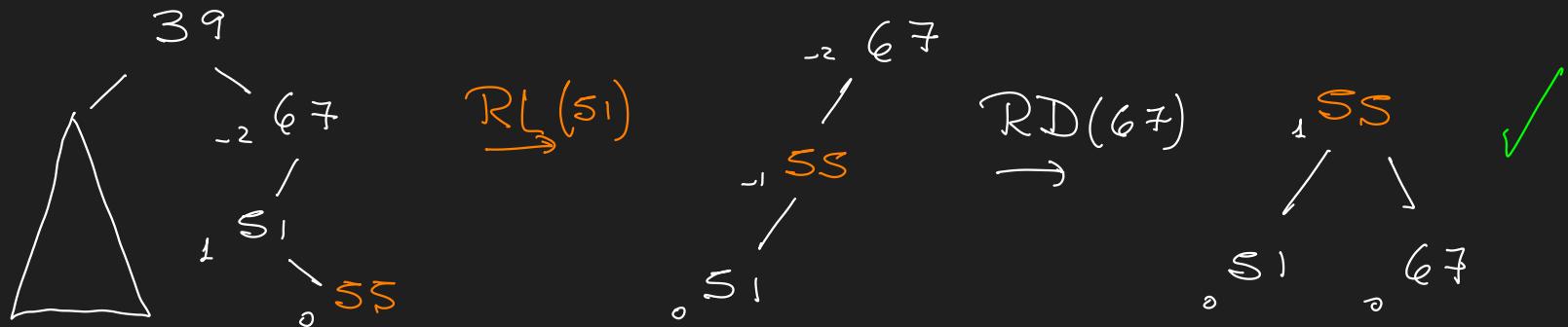
- No depende de la cant. de nodos → Dura más sin desbalancearse
Es más fácil balancearlos.
- Depende de una cant. fija de reasignaciones de punteros.



Balanciado con $|FB| \leq 1$ ✓

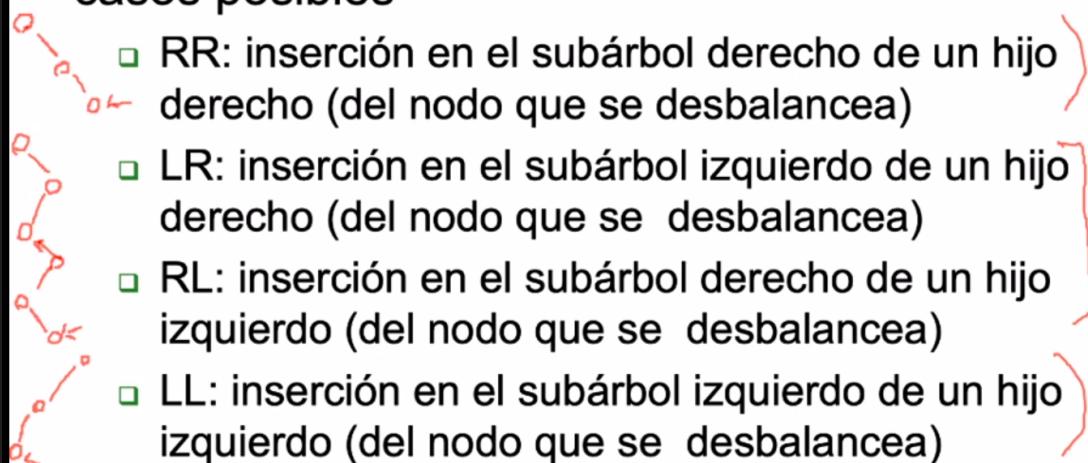


Inserción SS :

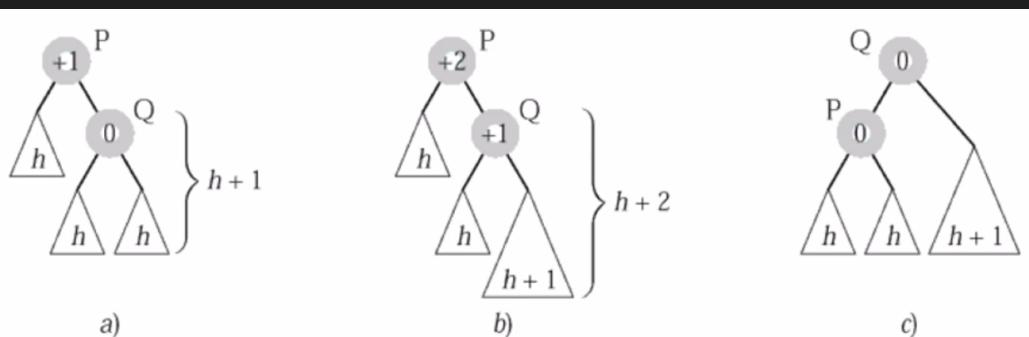


1. Insertar el nuevo nodo como en un ABB “clásico”
 - el nuevo nodo es una hoja
2. Recalcular los factores de balanceo que cambiaron por la inserción
 - sólo en la rama en la que ocurrió la inserción (los otros factores no pueden cambiar!), de abajo hacia arriba
3. Si en la rama aparece un factor de balanceo de ± 2 hay que rebalancear
 - A través de “rotaciones”

casos posibles

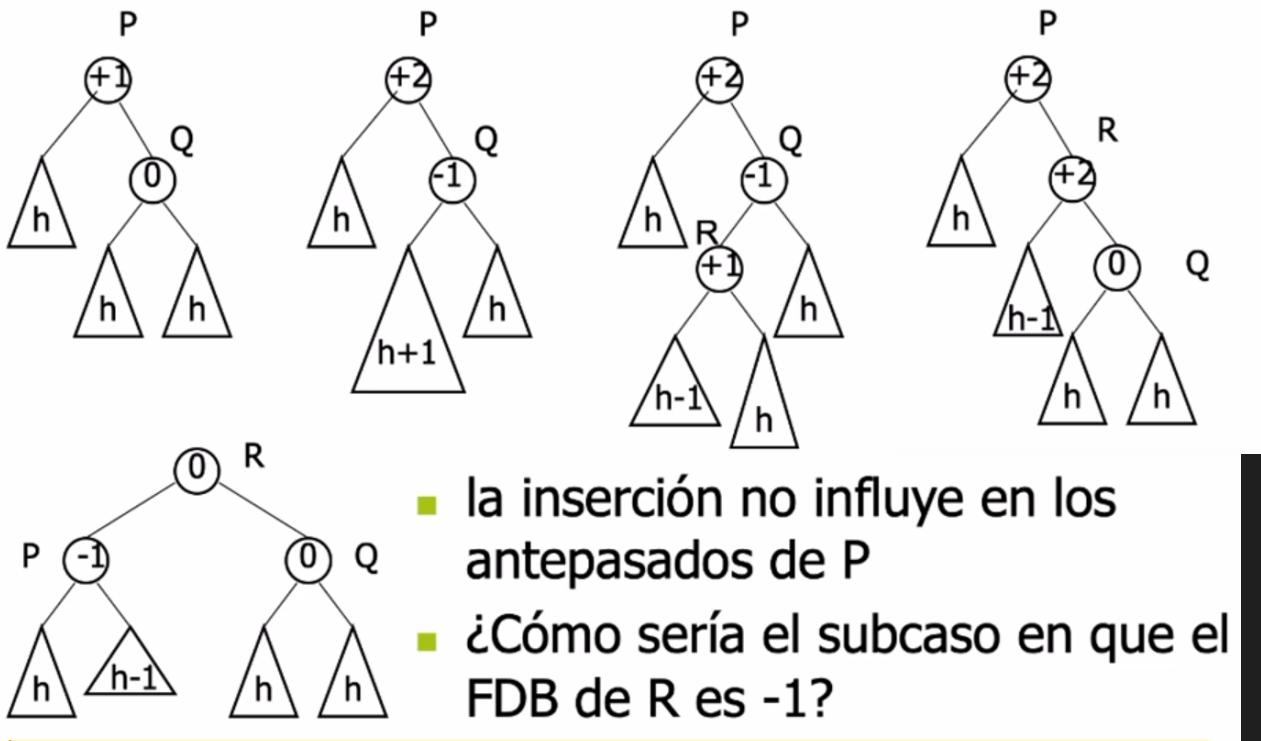
- 
- RR: inserción en el subárbol derecho de un hijo derecho (del nodo que se desbalancea)
 - LR: inserción en el subárbol izquierdo de un hijo derecho (del nodo que se desbalancea)
 - RL: inserción en el subárbol derecho de un hijo izquierdo (del nodo que se desbalancea)
 - LL: inserción en el subárbol izquierdo de un hijo izquierdo (del nodo que se desbalancea)

rotación simple (caso RR)



- La inserción no influye en los antepasados de P porque luego de la rotación recuperan su factor de balanceo anterior

rotación doble (caso LR)



inserción en los AVL/costo

- paso 1: proporcional a la altura del árbol $\Theta(\lg n)$
- paso 2: proporcional a la altura del árbol $\Theta(\lg n)$
- paso 3: $O(1)$ (se hace una o dos rotaciones por inserción)

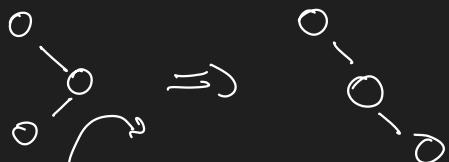
En total: $\Theta(\lg n)$

Borra ↴

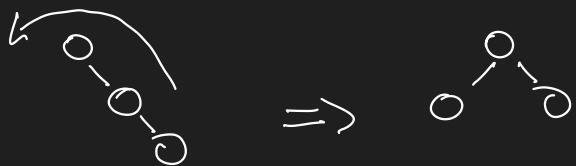
1. Borrar el nodo como en un ABB "clásico"
2. recalcular los factores de balanceo que cambiaron por el borrado
 - sólo en la rama en que ocurrió el borrado, de abajo hacia arriba
3. para cada nodo con factor de balanceo ± 2 hay que hacer una rotación simple o doble
 - $O(\lg n)$ rotaciones en el caso peor

Cuando hago 2 rotaciones

La 1º estira la "rodilla"



Luego 2º vuelve al caso de 1 rotación.



borrado en los AVL/costo

- en el caso peor hay que hacer rotaciones (simples o dobles) a lo largo de toda la rama
- paso 1: proporcional a la altura del árbol $\Theta(\lg n)$
- paso 2: proporcional a la altura del árbol $\Theta(\lg n)$
- paso 3: $\Theta(\lg n) \cdot \Theta(1)$

En total: $\Theta(\lg n)$