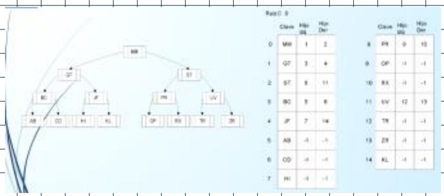
Clase 6 -Árboles B

jueves, 20 de abril de 2023 15:17

Problemas con Índices

- Búsqueda b. b. costosa.
- Mantener orden es costoso } solucionado usando RAM

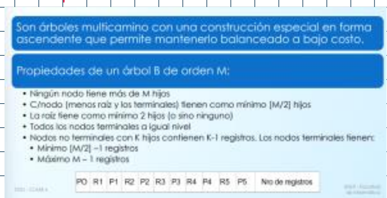
Árboles binarios en disco (archivo)



Este árbol está balanceado

- Hojas al = nivel
- = distancia de raíz a cualquier hoja
- como máx 4 accesos.
- Puedo tener parte del a. en ram y primero
- Los a. binarios se desbalancean muy fácil.
- balanceado = lleno

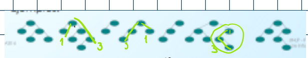
¡Importante!



Árbol AVL

- + fácil q se mantenga balanceado.

Para cada nodo hay un límite de la diferencia de altura de cualquier subárbol



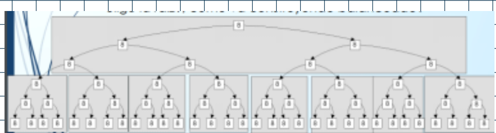
NO NO NO NO NO

Vamos a usar (BA(1)) diferencia max 1

Búsqueda en AVL es mejor

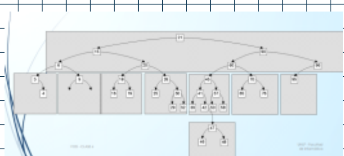
para todos los nodos, la altura de la rama izquierda no difiere en más de una unidad de la altura de la rama derecha o viceversa.

Árbol b. paginado



me voy llevando de a un bloque a ram.

Recorro como mucho 2 bloques

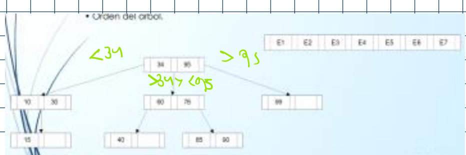


se desbalancea fácil

Difícil de implementar en la práctica

Árboles generales

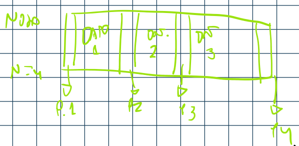
- Cada nodo puede tener + de 2 hijos



Árbol B: árboles generales q' se forman de abajo hacia arriba

propiedades → orden → M es cant. de hijos q tiene cada nodo

Con M=4, Max cant de claves = 3



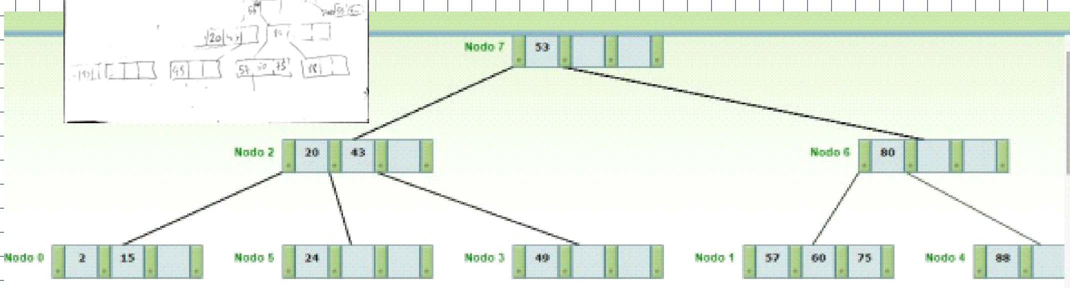
- Ningún nodo tiene + de N hijos
- Cada nodo (menos hoja y raíz) tiene mínimo [M/2] hijos → Si M=5, Mínimo 2
- Raíz tiene min 2 hijos (o ninguno)
- Todas las hojas a = nivel
- Nodos no terminales con K hijos tienen K-1 elem.
- Algoritmo, tope y min
  - → [M/2]-1 registros → Hojas completas + o - hasta la mitad del orden.
  - → M-1

Creación → ¿Cómo se crea un árbol?

M=4 Nodo

- Dadas las claves: 43 2 53 88 75 80 15 49 60 20 57 24

- Cuando hay overflow, creo otro nodo igual
- divido el nodo en 2 (en este caso es impar, me debo llevar 1 o 2)
- promociono (agrego otro nodo y pongo el + chico o el más grande (siempre = criterio))
- Pongo este nuevo nodo arriba.



ARCH de DATOS

Nodo Raiz: 7								
	Punteros				Datos			Nro Datos
0	-1	-1	-1		2	15		2
1	-1	-1	-1	-1	57	60	75	3
2	0	5	3		20	43		2
3	-1	-1			49			1
4	-1	-1			88			1
5	-1	-1			24			1
6	1	4			80			1
7	2	6			53			1

ARCH DE ÍNDICES.

Árboles B

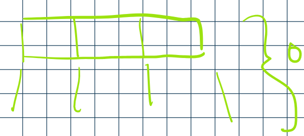
- A. multicaminos
- Se mantienen balanceados a bajo costo
- Como máx. tienen M descendientes y M-1 elem.
- La raíz o no tiene hijos o tiene min 2
- Nodo con x hijos tiene x-1 elem.
- Cada nodo tiene min [M/2]-1 elem y max M-1 elem
- Nodos terminales a = nivel

```
const M = ... ; {orden del árbol}
type
   nodo = record
   cant_claves:  integer;
   claves:       array[1..M-1] of <tipo_dato>;
   hijos:        array[1..M] of integer;
      end;
```

→ clave primaria o de uno los datos?

Mejor (aunque necesito 1 índice)

→ + enlace: array [1.. M−1] of integer → ≤ nodo



orden 4. (3 elem, 4 hijos)

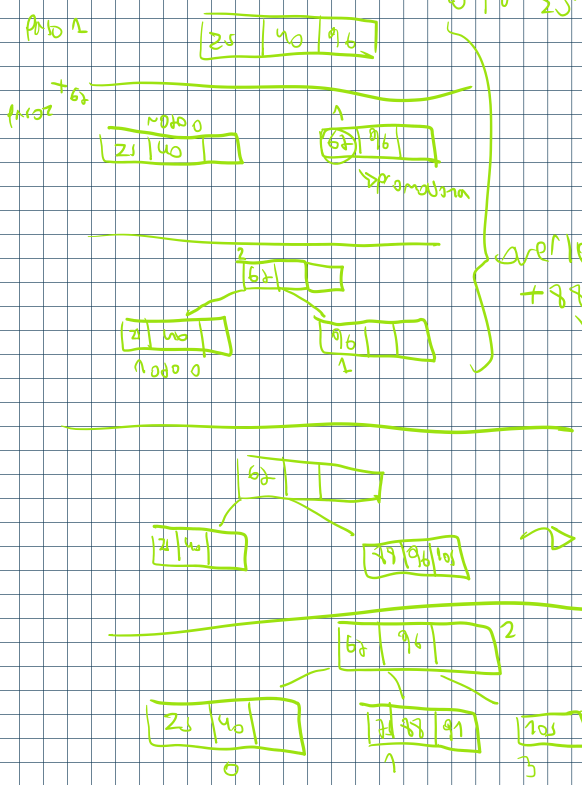
40 96 25 67 → overflow → desplaza

→ creo un nodo nuevo

→ mitad + grande va nuevo nodo

promociono el más chico al padre

Se crea un nuevo nodo.
La primera mitad de las claves se mantiene en el nodo con overflow.
La segunda mitad de las claves se traslada al nuevo nodo.
La menor de las claves de la segunda mitad se promociona al nodo padre.



paso 2

overflow

+88, +105

→ > q' 67

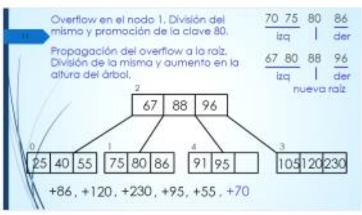
→ busca L. der.

llegué a una hoja, si hay espacio inserto

+85

→ overflow

+91



Overflow en el nodo 1. División del mismo y promoción de la clave 80.
Propagación del overflow a la raíz. División de la misma y aumento en la altura del árbol.

nuevo nodo raíz

Principal perk de árboles B es la búsqueda

- Mejor caso: 1 lectura
- Pero caso: h lecturas (con h altura del árbol)
- Cual es el valor de h?
  - Axioma: árbol balanceado de Orden M, si el número de elementos del árbol es N □ hay N+1 punteros nulos en nodos terminales.

$h <= [1 + \log_{[M/2]} ((N+1)/2)]$

Si M = 512 y N = 1000000 □ h <= 3.37 (4 lecturas encuentra un registro)

→ p/ saber con cuántas lecturas máx. encuentro un reg.

$\boxed{\text{son árboles gordos y bajitos}}$

Al borrar tengo q' revisar que la cant. de nodos que quedan sea mayor a $[M/2]-1$ → underflow

Eliminación

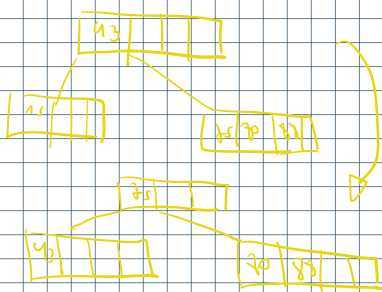
- Siempre eliminar de nodos terminales (trabajamos con árboles)
- Si se va a eliminar un elemento que no esta en nodo terminal □ llevarlo primero a nodo terminal
- Posibilidades ante eliminación
  - Mejor caso: borra un elemento del nodo y no produce underflow, solo reacomodos (# elementos >= [M/2]-1
  - Peor caso: se produce underflow, #elementos < [M/2] – 1
- Dos soluciones
  - Redistribuir
  - concatenar

FOD - CLASE 6 · UNLP - Facultad de Informática

Orden → cant de hijos
↳ NO elementos, elem +1

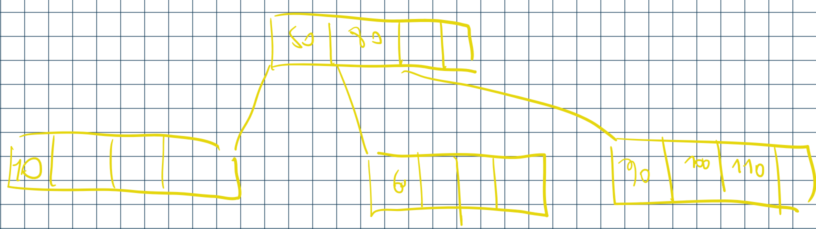
Más grande de izq, más chico q' der ↗ sobre 60 ↗ hacer esto
por convención de la cát. regla.

Si quiero borrar uno q' no está en hoja, cambio el que quiero borrar por el de una hoja y borro el de hoja.
Nada más hay q' tener cuidado con los underflow (menos del min).
Si queda en underflow, miro a los del hermano adyacente = padre, no importa si está pegado o no

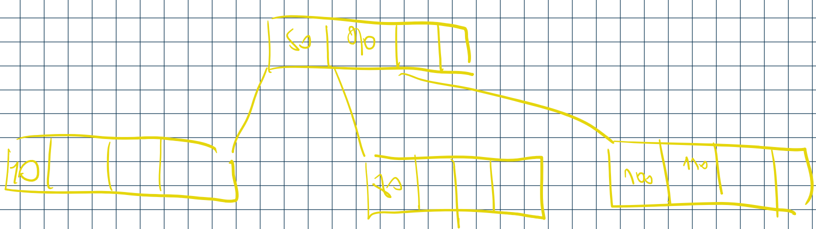


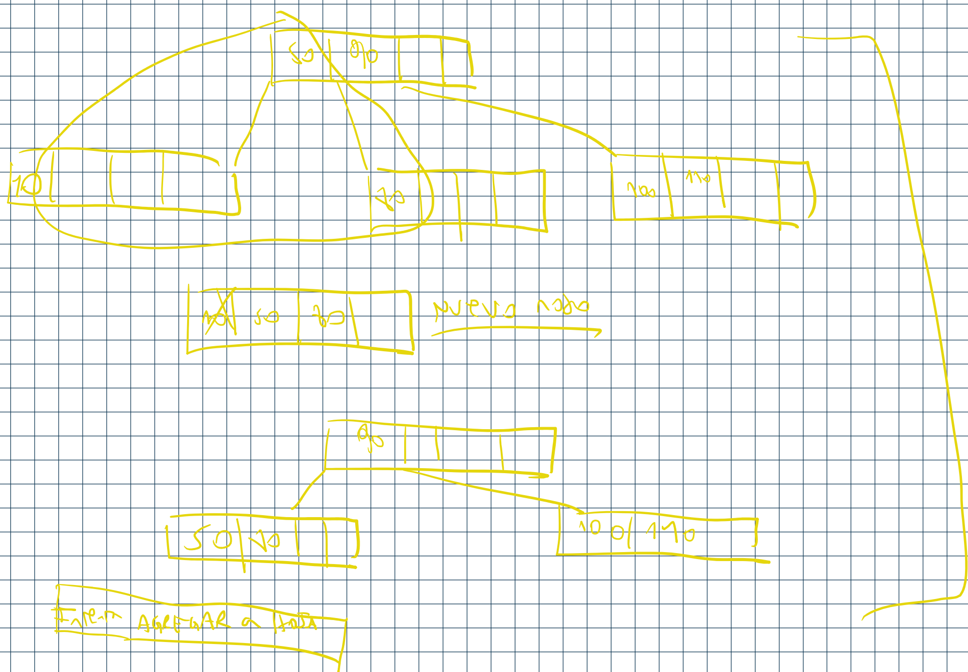
Redistribuir.

Si no tengo hermano p. distribuir (porque me queda en underflow)
Concateno - fusiono.



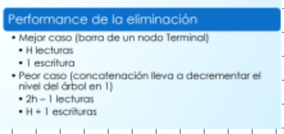
Borrar el 60 → fusiono.

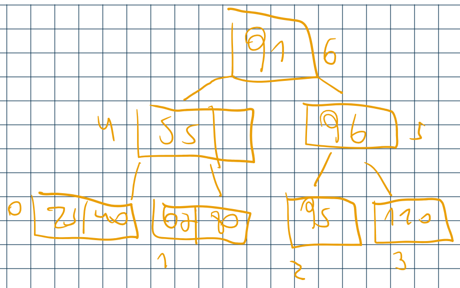




Borrar el 10.

Concatenar → involucra siempre 3 nodos
el q' quiero borrar + padre + hermano.


Performance de la eliminación
• Mejor caso (borra de un nodo Terminal)
• H lecturas
• 1 escritura
• Peor caso (concatenación lleva a decrementar el nivel del árbol en 1)
• 2h – 1 lecturas
• H + 1 escrituras



—95