Fundamentos de Organización de Datos

Clase 6

Agenda

Arboles	• Binarios • AVL • Multicamino • Balanceados
Arboles Balanceados	• Características • B, B*, B+ • Operaciones • Prefijos simples

Arboles → introducción

Problemas con los índices?

- La búsqueda binaria aun es costosa
- Mantener los índices ordenados es costoso
- Solución → RAM
- Objetivo → persistencia de datos

Árboles

- Estructuras de datos que permiten localizar en forma más rápida información de un archivo, tienen intrínsecamente búsqueda binaria

Arboles binarios

Que es un árbol binario?

- Estructuras de datos donde cada nodo tiene dos sucesores, a izquierda y a derecha

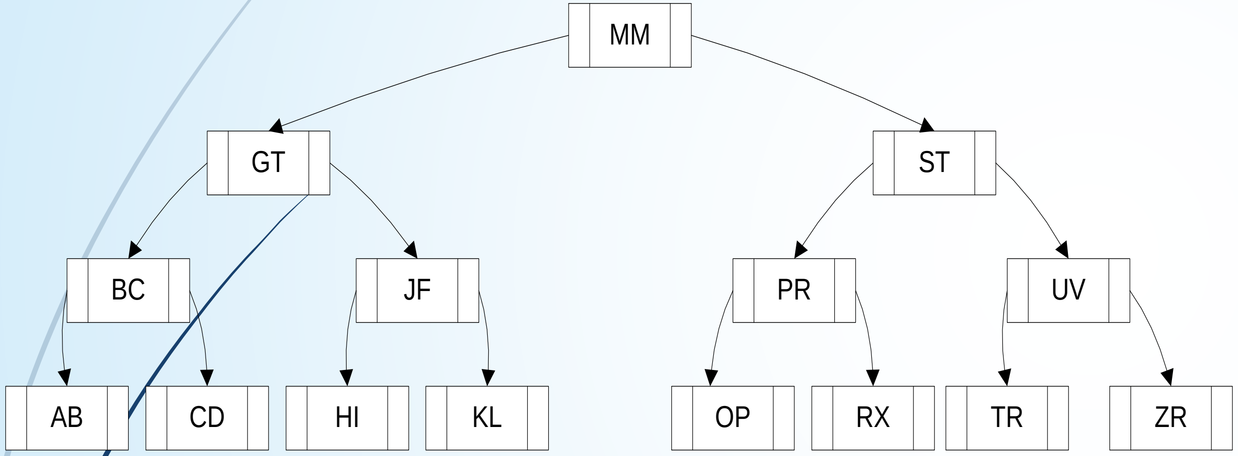
Un árbol binario, puede implantarse en disco?

- Como lograr la persistencia?

Ejemplo → supongamos estas claves

- MM ST GT PR JF BC UV CD HI AB KL TR OP RX ZR

Arboles binarios



Raíz → 0

	Clave	Hijo izq	Hijo Der
0	MM	1	2
1	GT	3	4
2	ST	8	11
3	BC	5	6
4	JF	7	14
5	AB	-1	-1
6	CD	-1	-1
7	HI	-1	-1

	Clave	Hijo izq	Hijo Der
8	PR	9	10
9	OP	-1	-1
10	RX	-1	-1
11	UV	12	13
12	TR	-1	-1
13	ZR	-1	-1
14	KL	-1	-1

Construccion de un arbol binario

```
Type arbol = record;
        elemento: tipodedato;
        hijo_izq, hijo_derecha: ^arbol;
End

Type  arbol = record
        elemento: tipodedato;
        hijo_izq, hijo_derecha: integer;
End
 indice: file of arbol;
```

Construccion de un arbol binario

Al principio

elemento	H Izq	H Derecha

Llega el primer elemento MM

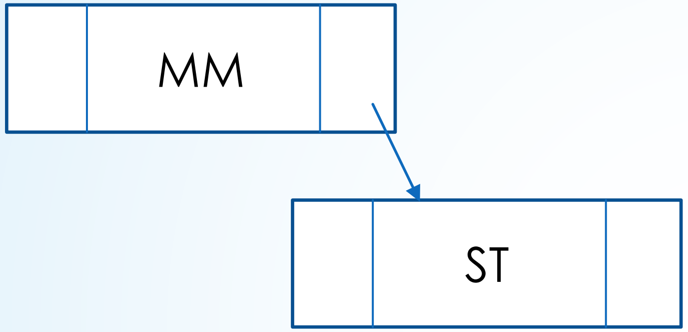
	MM	

Raiz --> 0

	elemento	H Izq	H Derecha
0	MM	-1	-1

Construccion de un arbol binario

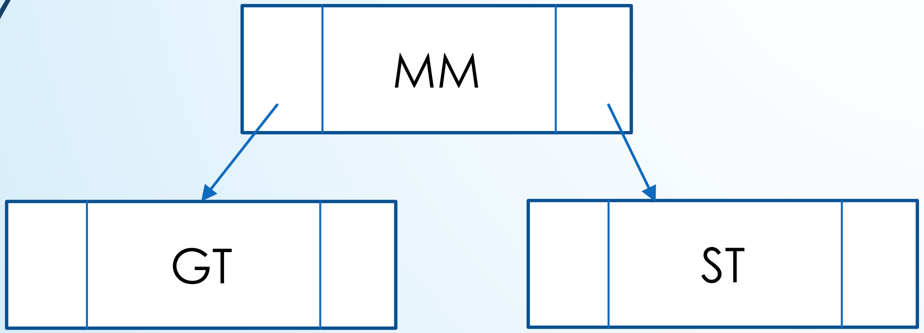
Llega el elemento ST



Raiz --> 0

	elemento	H Izq	H Derecha
0	MM	-1	-1 1
1	ST	-1	-1

Llega el elemento GT

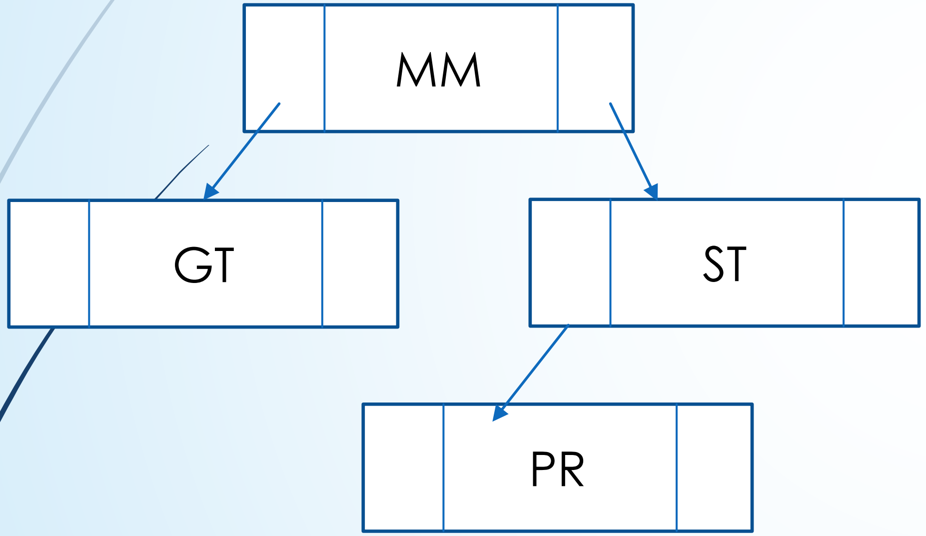


Raiz --> 0

	elemento	H Izq	H Derecha
0	MM	-1 2	1
1	ST	-1	-1
2	GT	-1	-1

Construccion de un arbol binario

Llega el elemento PR

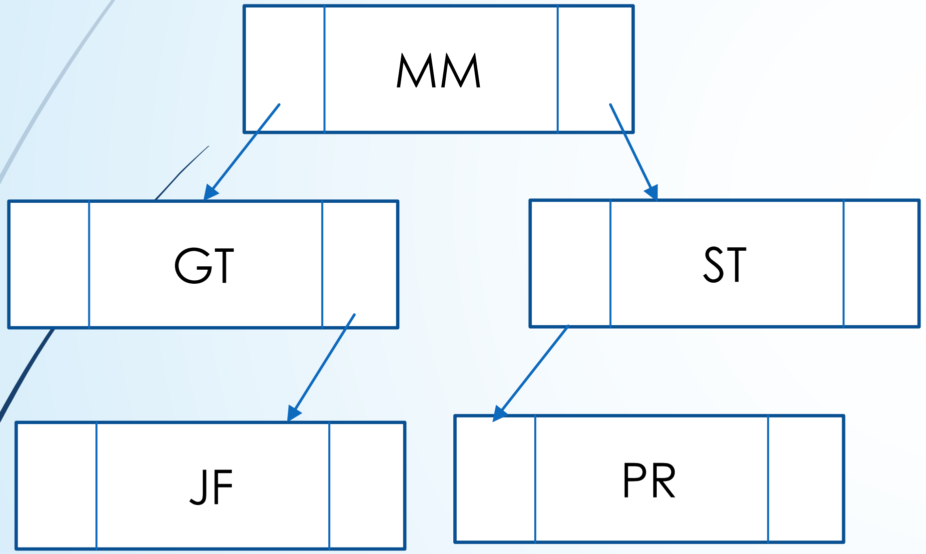


Raiz --> 0

	elemento	H Izq	H Derecha
0	MM	2	1
1	ST	-1 3	-1
2	GT	-1	-1
3	PR	-1	-1

Construccion de un arbol binario

Llega el elemento JF

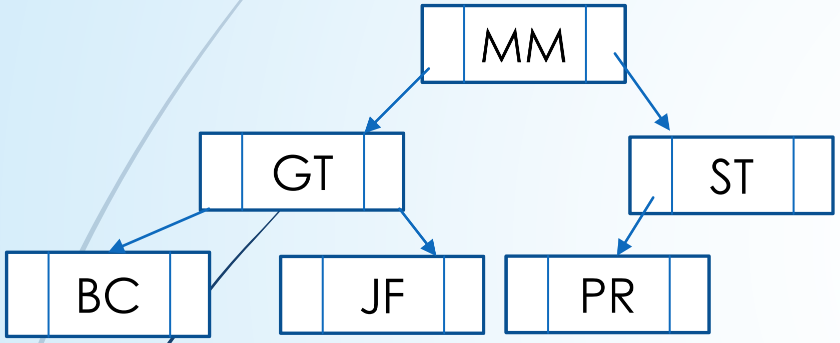


Raiz --> 0

	elemento	H Izq	H Derecha
0	MM	2	1
1	ST	3	-1
2	GT	-1	-1 4
3	PR	-1	-1
4	JF	-1	-1

Construccion de un arbol binario

Llega el elemento BC

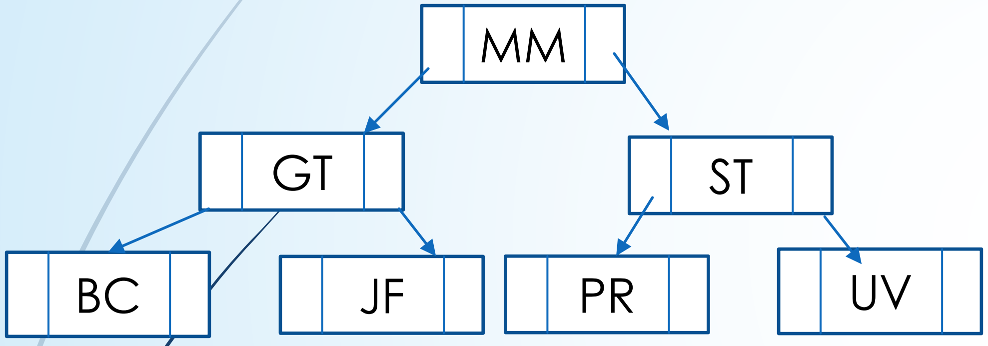


Raiz --> 0

	elemento	H Izq	H Derecha
0	MM	2	1
1	ST	3	-1
2	GT	~~-1~~ 5	4
3	PR	-1	-1
4	JF	-1	-1
5	BC	-1	-1

Construccion de un arbol binario

Llega el elemento UV

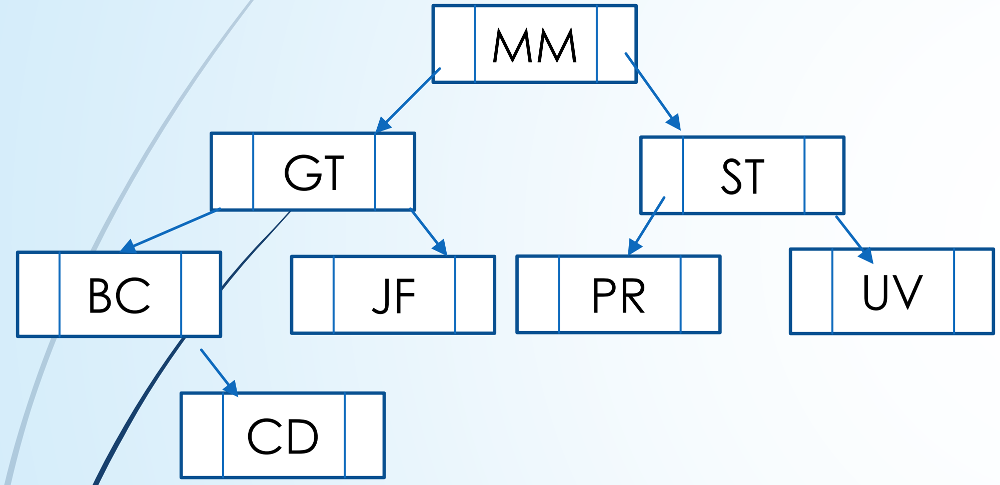


Raiz --> 0

	elemento	H Izq	H Derecha
0	MM	2	1
1	ST	3	-1 6
2	GT	5	4
3	PR	-1	-1
4	JF	-1	-1
5	BC	-1	-1
6	UV	-1	-1

Construccion de un arbol binario

Llega el elemento CD



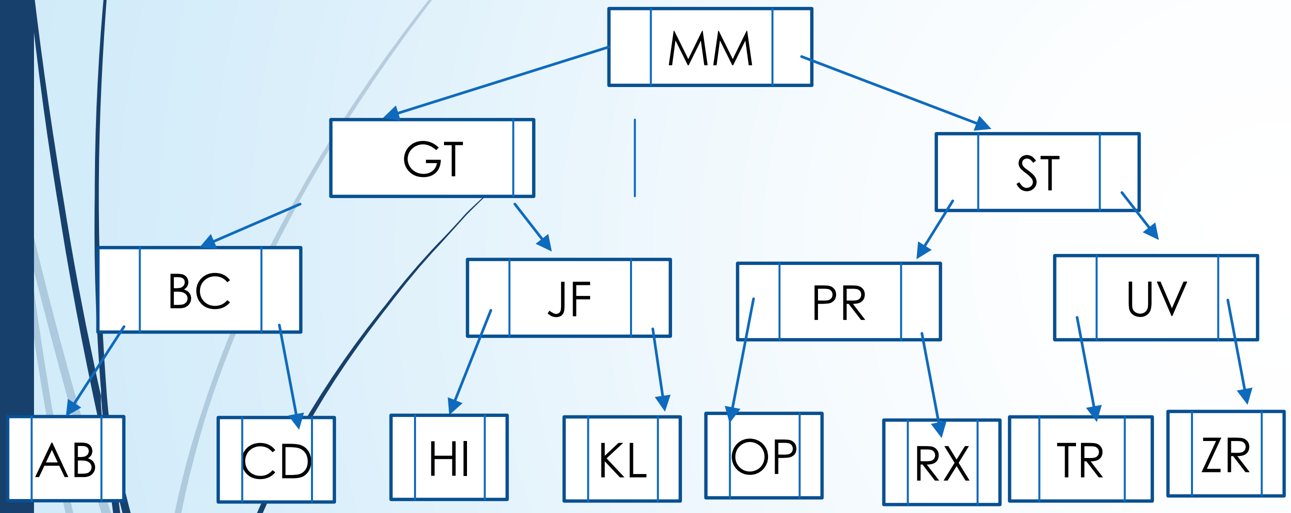
Raiz --> 0

	elemento	H Izq	H Derecha
0	MM	2	1
1	ST	3	6
2	GT	5	4
3	PR	-1	-1
4	JF	-1	-1
5	BC	-1	~~-1~~ 7
6	UV	-1	-1
7	CD	-1	-1

Construccion de un arbol binario

Llega el elemento HI AB KL TR OP RX ZR

Raiz --> 0



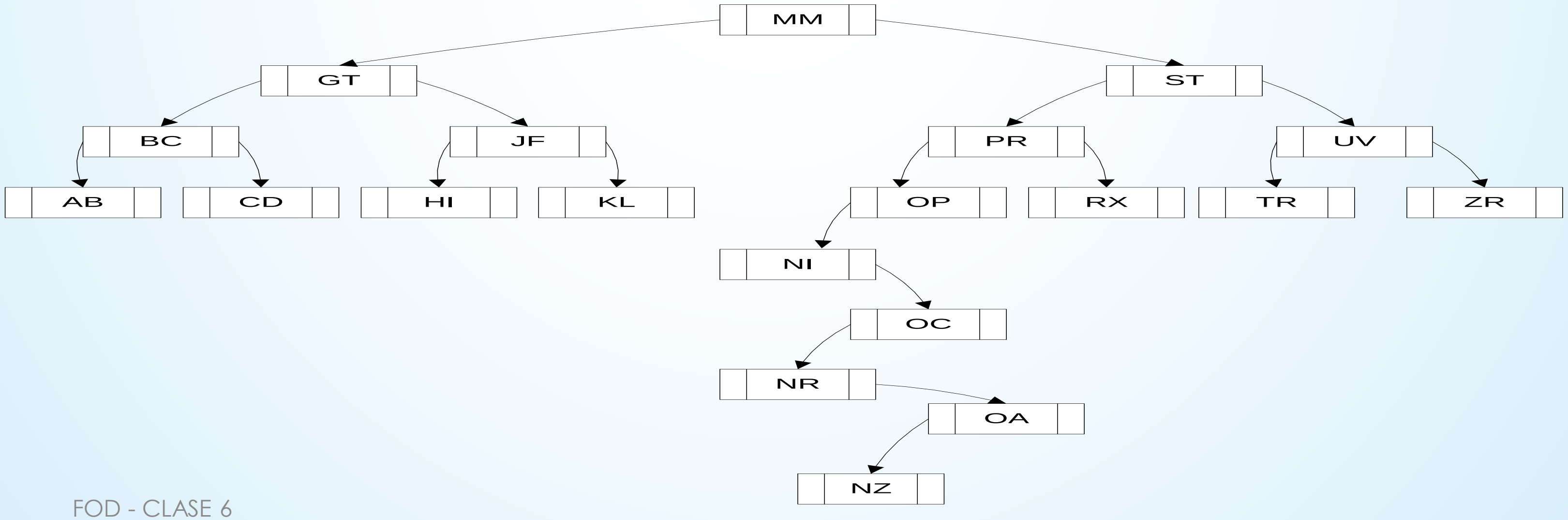
	elemento	H Izq	H Derecha
0	MM	2	1
1	ST	3	6
2	GT	5	4
3	PR	-1 12	-1 13
4	JF	-1 8	-1 10
5	BC	-1 9	7
6	UV	-1 11	-1 14
7	CD	-1	-1
8	HI	-1	-1
9	AB	-1	-1
10	KL	-1	-1
11	TR	-1	-1
12	OP	-1	-1

Arboles binarios

Árbol balanceado: un árbol está balanceado cuando la altura de la trayectoria más corta hacia una hoja no difiere de la altura de la trayectoria más grande.

Inconveniente de los binarios: se desbalancean fácilmente.

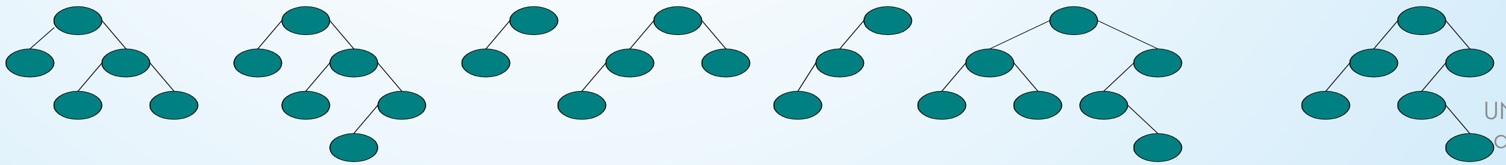
Supongamos que llegan las claves : NI OC NR OA NZ



Árboles AVL

Árboles AVL

- Árbol binario balanceado en altura (BA(1)) en el que las inserciones y eliminaciones se efectúan con un mínimo de accesos.
- Árbol balanceado en altura:
  - Para cada nodo existe un límite en la diferencia que se permite entre las alturas de cualquiera de los subárboles del nodo (BA(k)), donde k es el nivel de balance)
  - Ejemplos:



Arboles AVL y Binarios

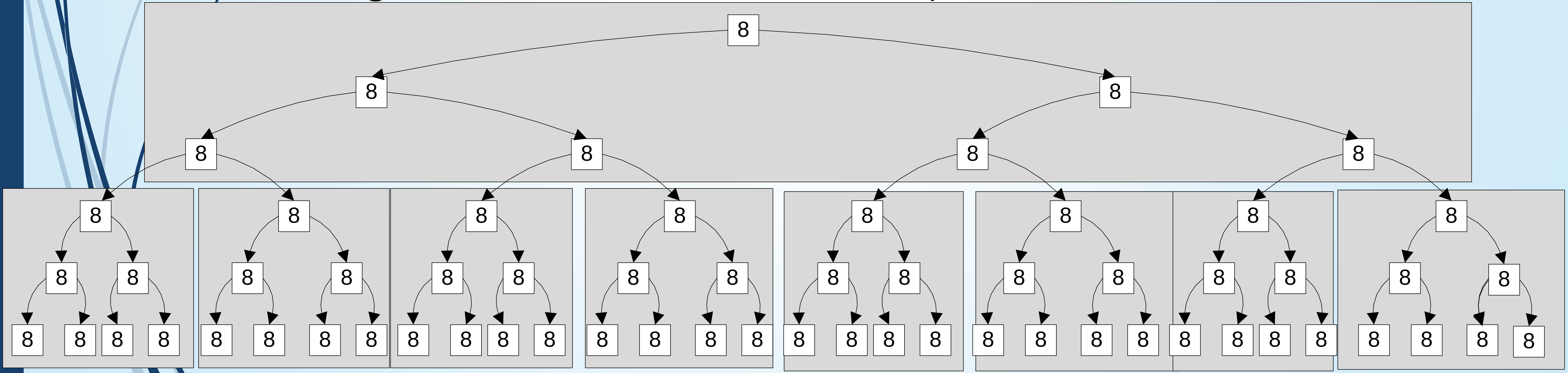
Características/Conclusiones

- Estructura que debe ser respetada
- Mantener árbol, rotaciones restringidas a un área local del árbol
  - Binario: → Búsqueda: $Log_2(N+1)$
  - AVL: → Búsqueda: $1.44 \log_2(N+2)$
  - Ambas performance por el peor caso posible

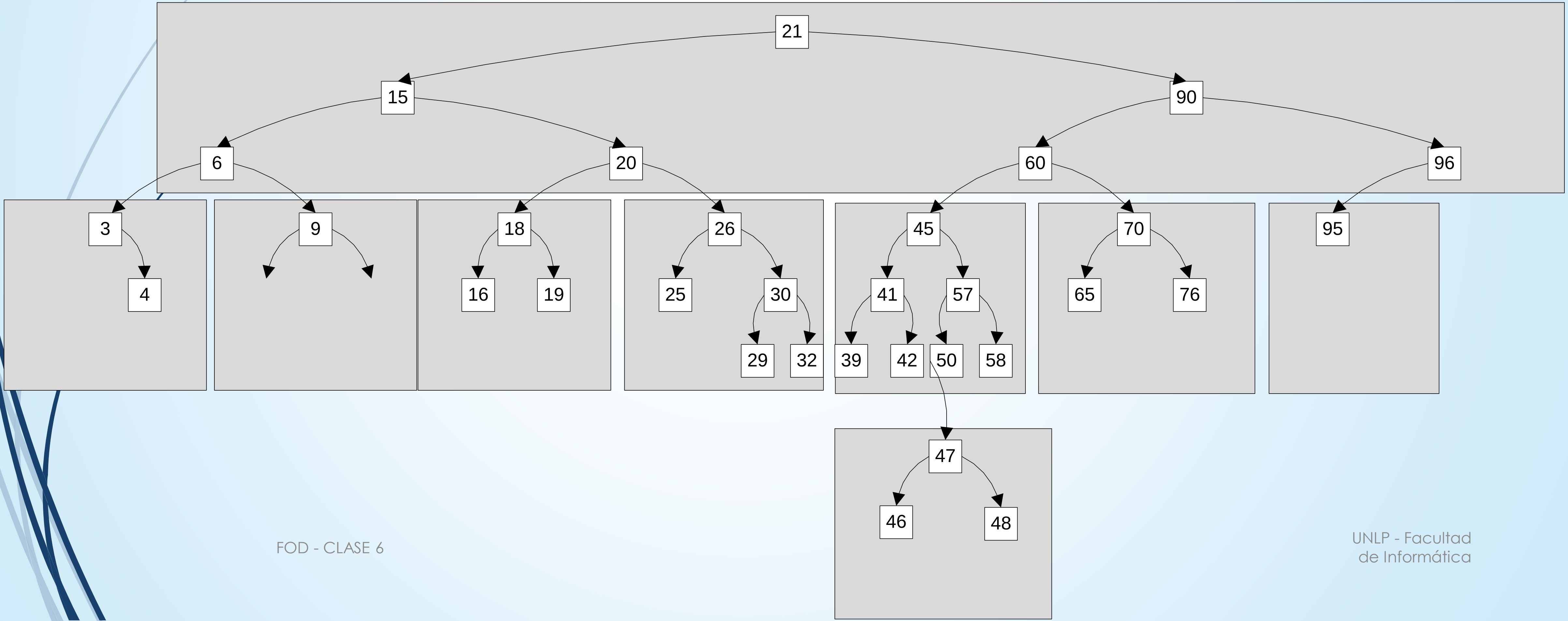
Árboles Binarios Paginados

Árboles binarios paginados

- Problemas de almacenamiento secundario, buffering, páginas de memoria, varios registros individuales, minimiza el número de accesos
- Problema: construcción descendente, como se elige la raíz?, cómo va construyendo balanceado?



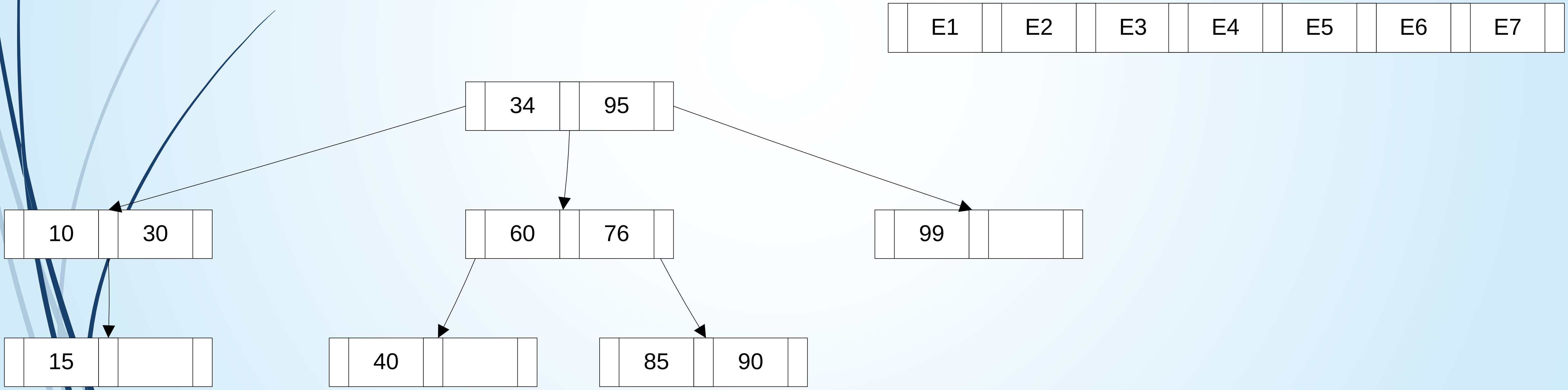
Árboles Binarios Paginados



Árboles multicamino

Generalización de árboles binarios, c/nodo tiene k punteros y k-1 claves (o registros), disminuye la profundidad del árbol,

- Orden del árbol.



Arboles balanceados

Son árboles multicamino con una construcción especial en forma ascendente que permite mantenerlo balanceado a bajo costo.

Propiedades de un árbol B de orden M:

- Ningún nodo tiene más de M hijos
- C/nodo (menos raíz y los terminales) tienen como mínimo [M/2] hijos
- La raíz tiene como mínimo 2 hijos (o sino ninguno)
- Todos los nodos terminales a igual nivel
- Nodos no terminales con K hijos contienen K-1 registros. Los nodos terminales tienen:
  - Mínimo [M/2] –1 registros
  - Máximo M – 1 registros

PO	R1	P1	R2	P2	R3	P3	R4	P4	R5	P5	Nro de registros

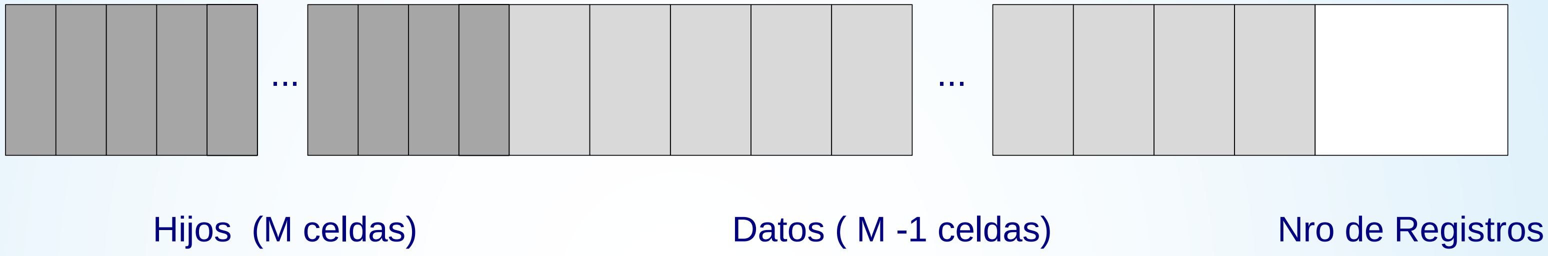
Arbol balanceado

- Estructura

```
Type arbolb = record;
        claves:  array[ 1.. N ] of tipodeclave;
        punteros:  array [0.. N] of integer;
        Cantelementos:  integer:
End;
```

Arboles Balanceados

- Formato del nodo

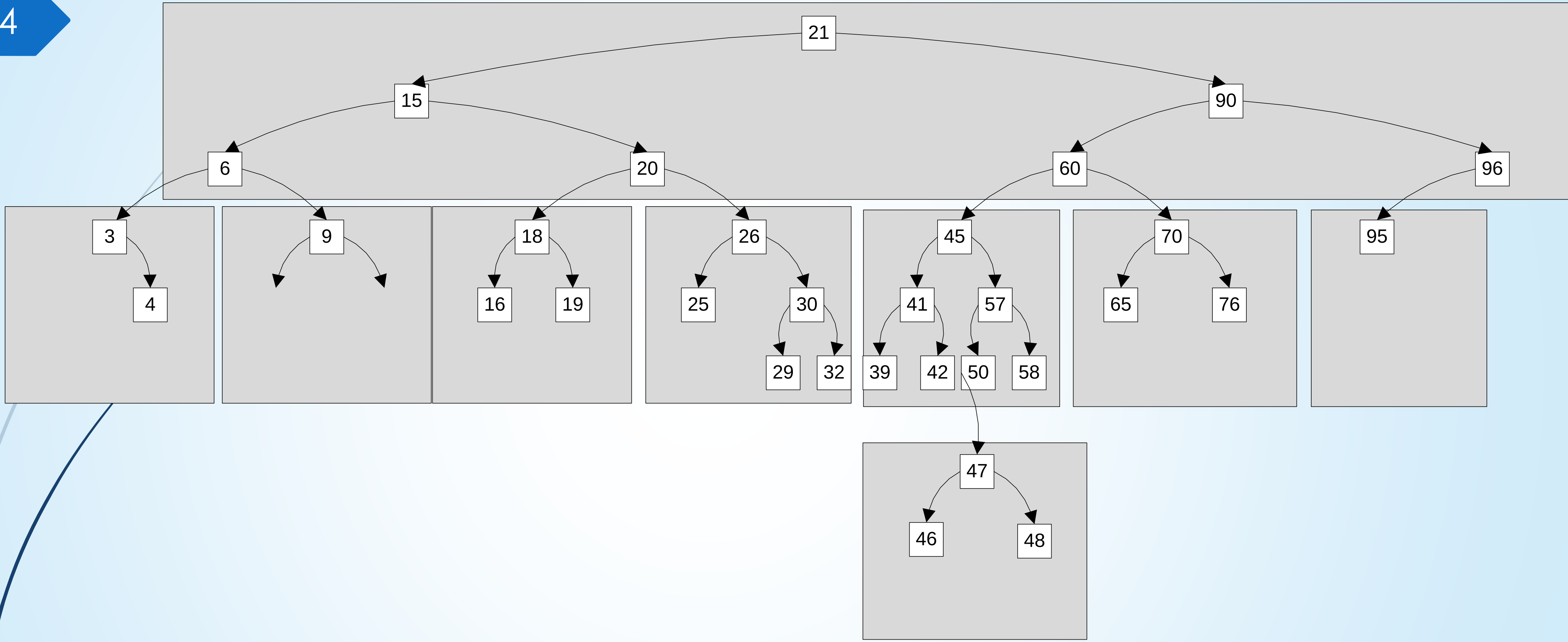


Formato del Nodo para archivo del índice arbol b

P0	Dato 1	P1	Dato 2	P2	…..	PM-2	Dato M-2	PM-1	Dato M-1	PM

Formato Gráfico del Nodo del índice arbol B

Árboles Binarios Paginados



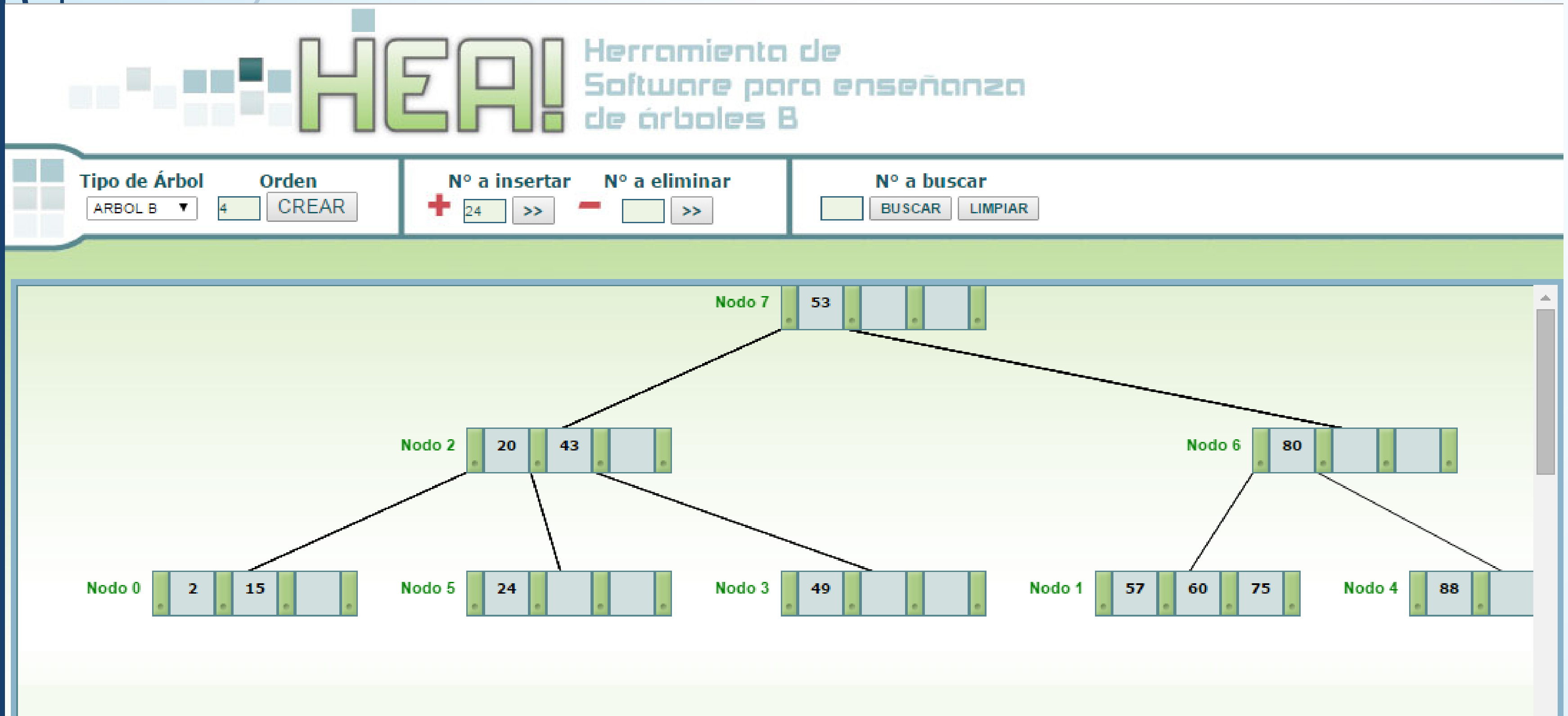
Arboles balanceados

Creacion:

- Dadas las claves: 43 2 53 88 75 80 15 49 60 20 57 24
- Como se construye el árbol?
- Como se general el archivo de datos que persiste el árbol?

Arboles Balanceados

	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



Arbol Balanceado

Raiz = ??								
	punteros				clave (datos)			#reg
Nodo 0								

Llega el elemento 43

Nodo 0

Raiz = 0								
	punteros				clave (datos)			#reg
Nodo 0	-1	-1			43			1

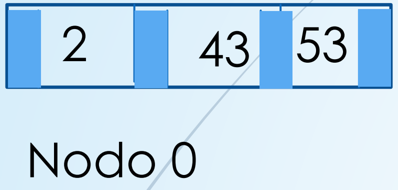
Llega el elemento 2

Nodo 0

Raiz = 0								
	punteros				clave (datos)			#reg
Nodo 0	-1	-1	-1		2	43		2

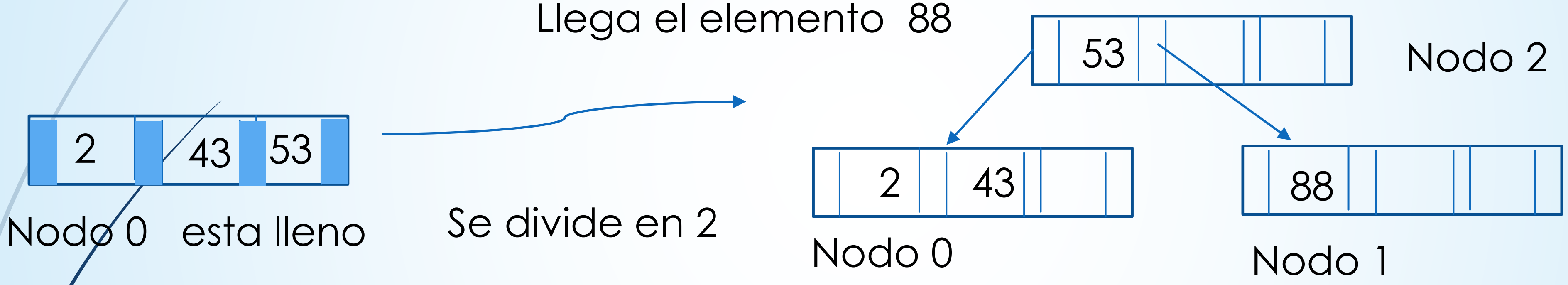
Arbol Balanceado

Llega el elemento 53



Raiz = 0								
	punteros				clave (datos)			#reg
Nodo 0	-1	-1	-1	-1	2	43	53	3

Llega el elemento 88

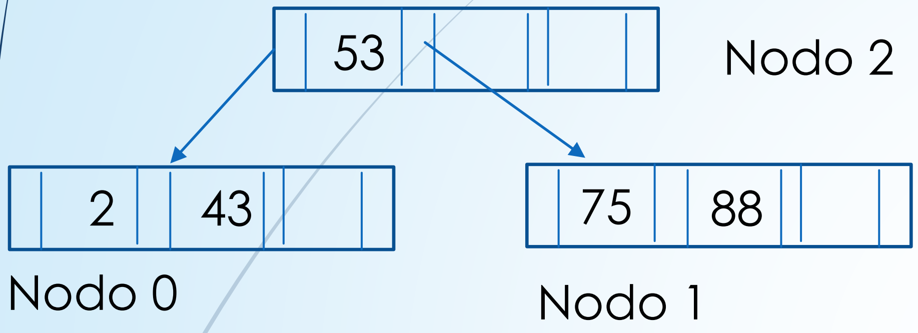


2 43 53 88

Raiz = 2								
	punteros				clave (datos)			#reg
Nodo 0	-1	-1	-1		2	43		2
Nodo 1	-1	-1			88			1
Nodo 2	0	1			53			1

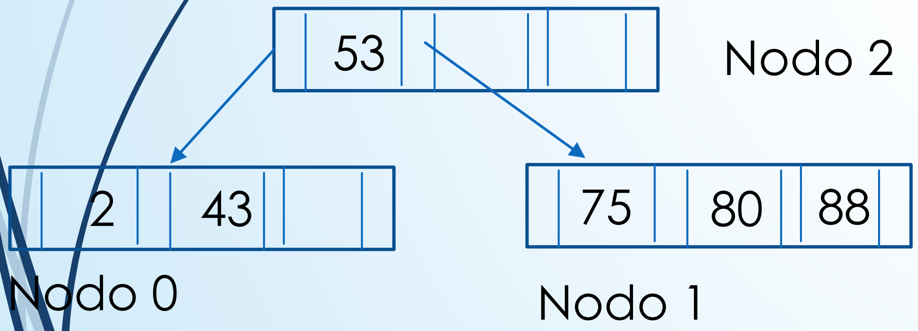
Arbol Balanceado

Llega el elemento 75



Raiz = 2								
	punteros				clave (datos)			#reg
Nodo 0	-1	-1	-1		2	43		2
Nodo 1	-1	-1	-1		75	88		2
Nodo 2	0	1			53			1

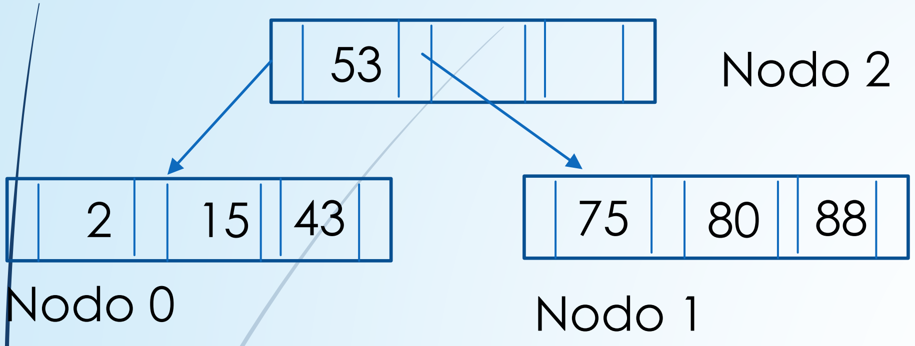
Llega el elemento 80



Raiz = 2								
	punteros				clave (datos)			#reg
Nodo 0	-1	-1	-1		2	43		2
Nodo 1	-1	-1	-1	-1	75	80	88	3
Nodo 2	0	1			53			1

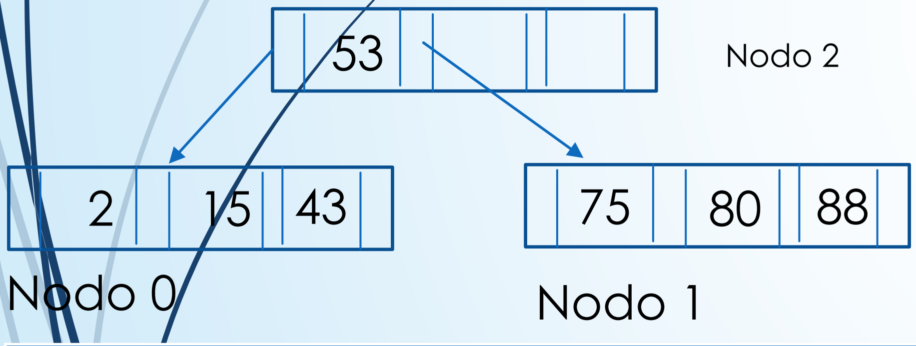
Arbol Balanceado

Llega el elemento 15

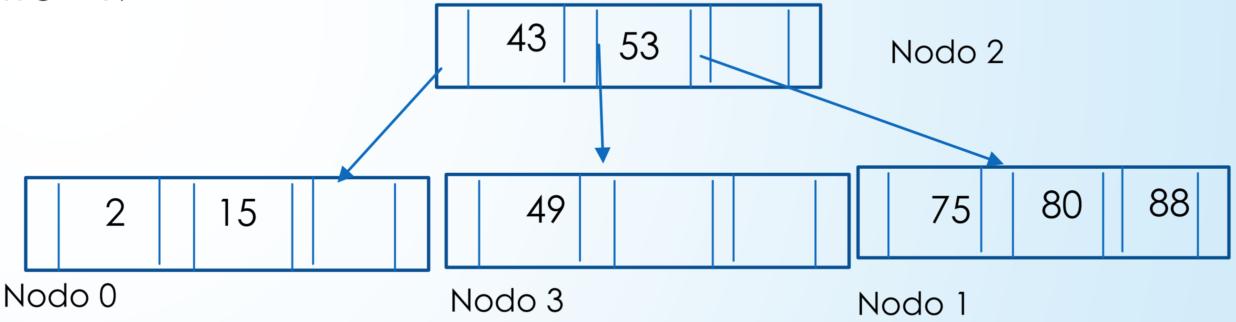


Raiz = 2								
	punteros				clave (datos)			#reg
Nodo 0	-1	-1	-1	-1	2	15	43	3
Nodo 1	-1	-1	-1	-1	75	80	88	3
Nodo 2	0	1			53			1

Llega el elemento 49



Se divide el nodo 0

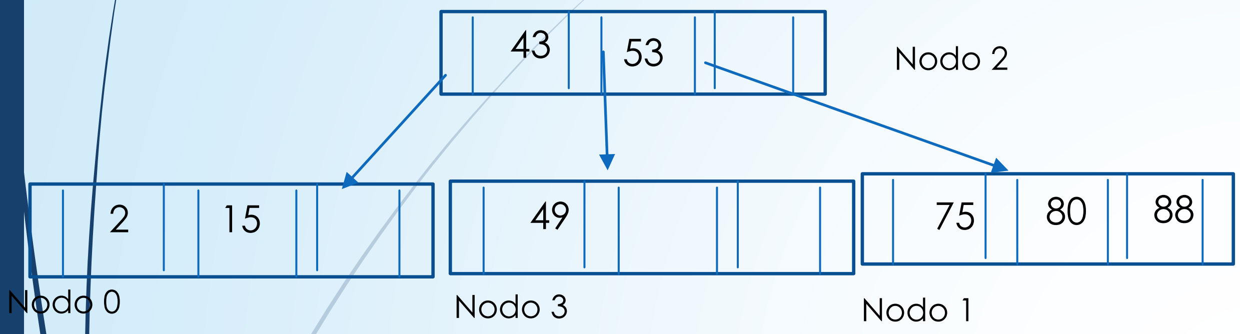


Raiz = 2								
	punteros				clave (datos)			#reg
Nodo 0	-1	-1	-1		2	15		2
Nodo 1	-1	-1	-1	-1	75	80	88	3
Nodo 2	0	3	1		43	53		2
Nodo 3	-1	-1			49			1

2 15 43 49

Arbol Balanceado

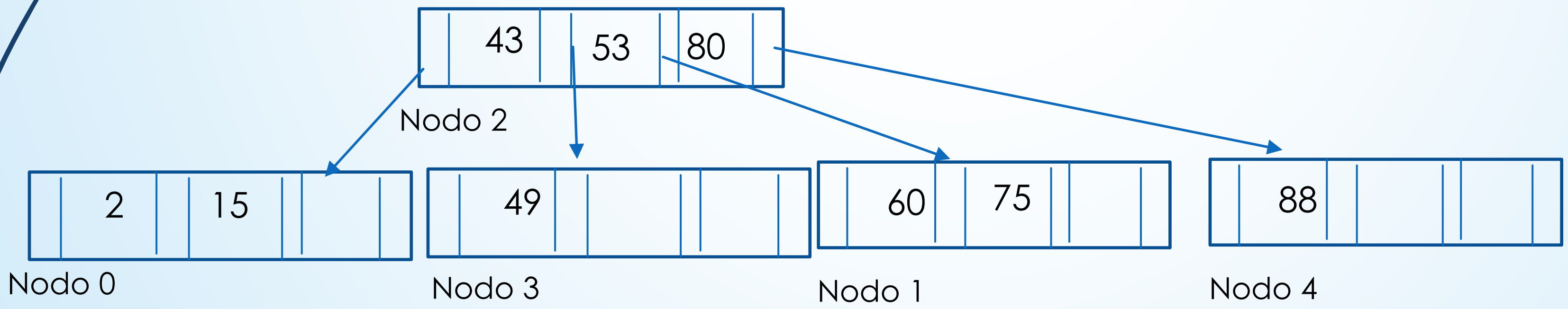
Llega el elemento 60



Raiz = 2

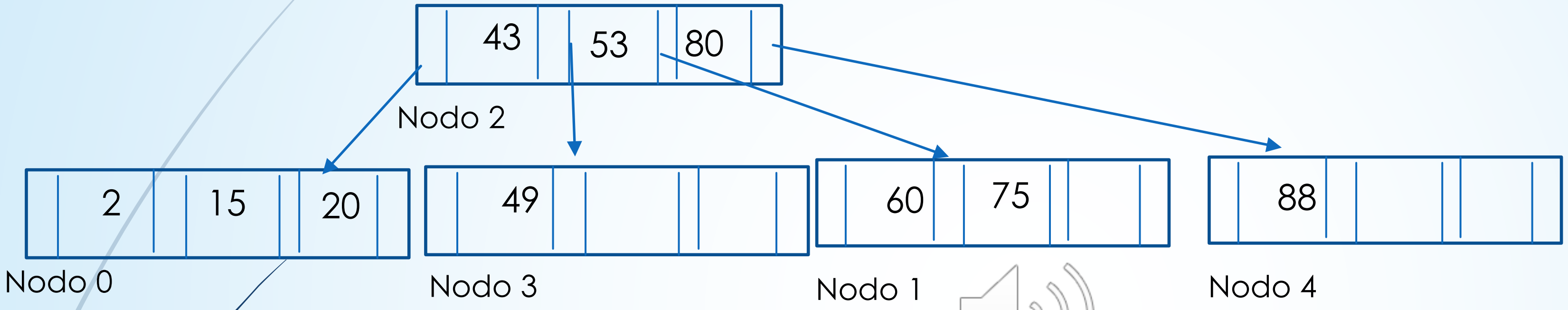
	punteros				clave (datos)			#reg
Nodo 0	-1	-1	-1		2	15		2
Nodo 1	-1	-1	-1		60	75		2
Nodo 2	0	3	1	4	43	53	80	3
Nodo 3	-1	-1			49			1
Nodo 4	-1	-1			88			1

60 75 80 88



Arbol Balanceado

Llega el elemento 20

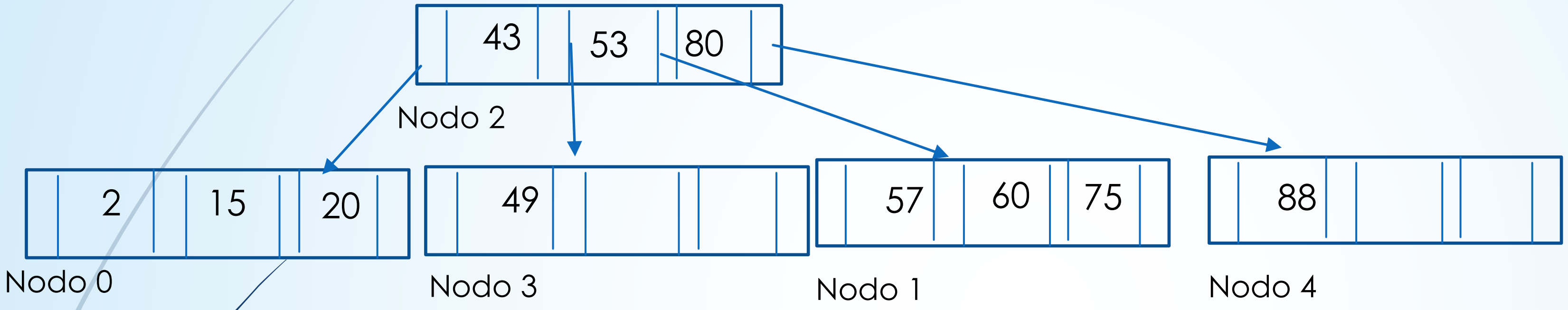


Raiz = 2

	punteros				clave (datos)			#reg
Nodo 0	-1	-1	-1	-1	2	15	20	3
Nodo 1	-1	-1	-1		60	75		2
Nodo 2	0	3	1	4	43	53	80	3
Nodo 3	-1	-1			49			1
Nodo 4	-1	-1			88			1

Arbol Balanceado

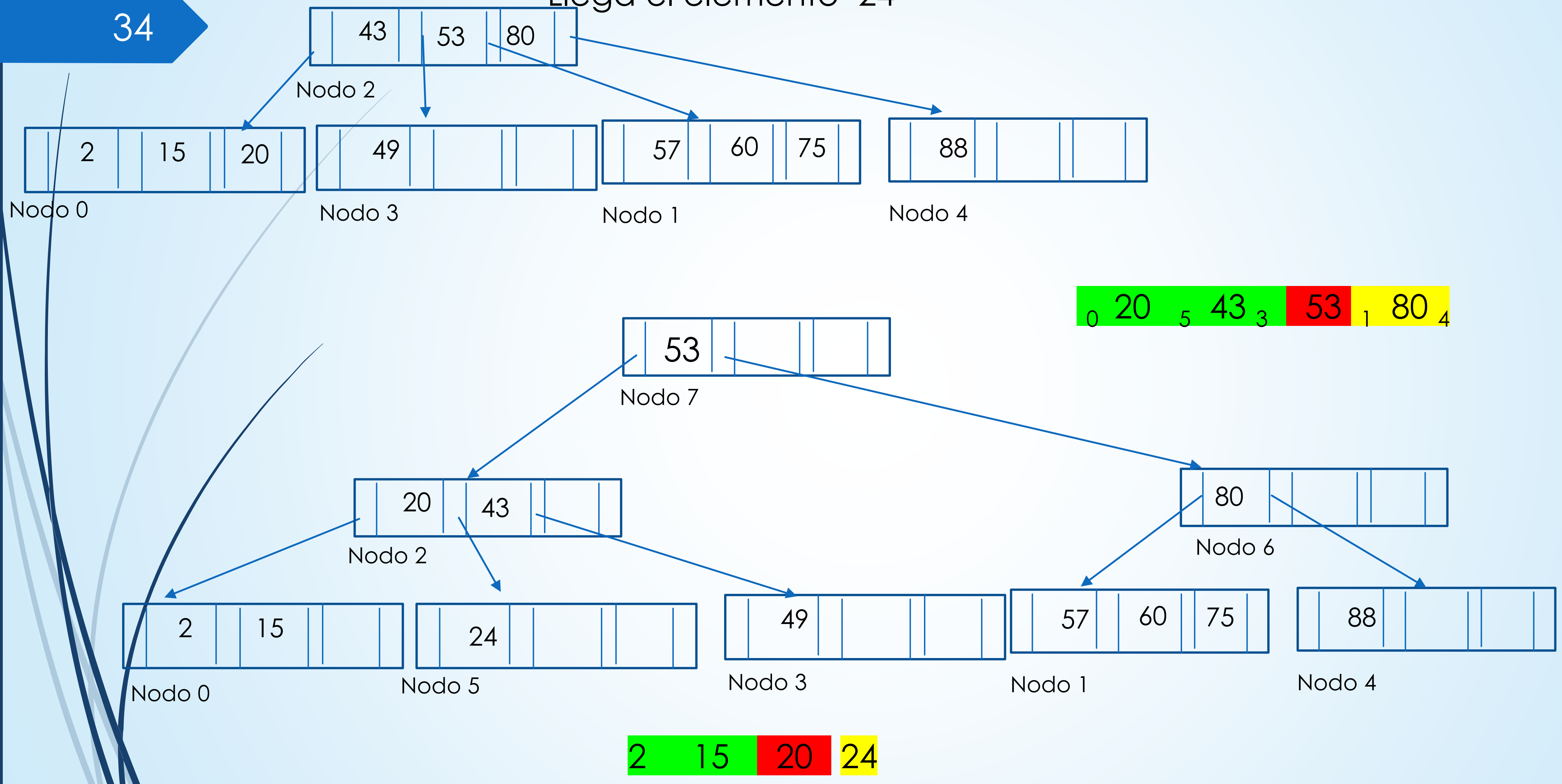
Llega el elemento 57



Raiz = 2								
	punteros				clave (datos)			#reg
Nodo 0	-1	-1	-1	-1	2	15	20	3
Nodo 1	-1	-1	-1	-1	57	60	75	3
Nodo 2	0	3	1	4	43	53	80	3
Nodo 3	-1	-1			49			1
Nodo 4	-1	-1			88			1

Arbol Balanceado

Llega el elemento 24

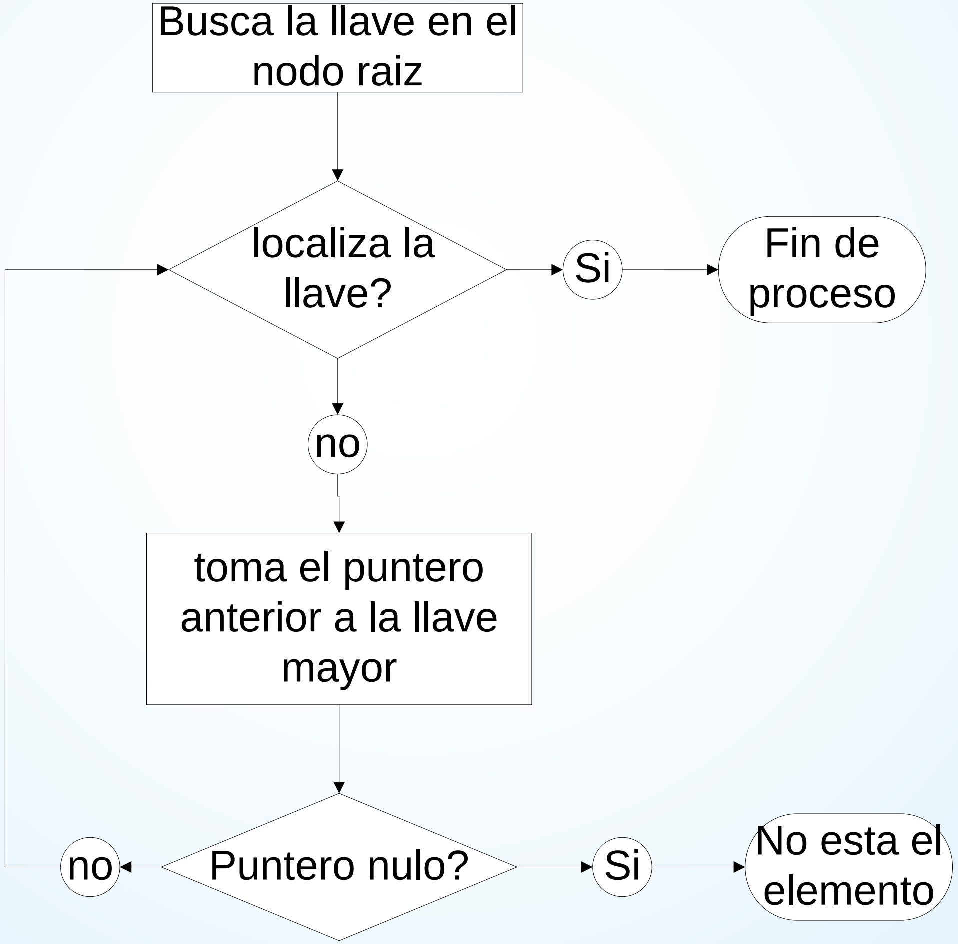


Arbol Balanceado

Raiz = 7								
	punteros				clave (datos)			#reg
Nodo 0	-1	-1	-1		2	15		2
Nodo 1	-1	-1	-1	-1	57	60	75	3
Nodo 2	0	5	3		20	43		2
Nodo 3	-1	-1			49			1
Nodo 4	-1	-1			88			1
Nodo 5	-1	-1			24			1
Nodo 6	1	4			80			1
Nodo 7	2	6			53			1

Árboles Balanceados

- Busqueda de información:



Arboles Balanceados

Performance de 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.

Árboles Balanceados

Cota para h

Nivel	# mínimo de descendientes
1	2
2	2 * [M/2]
3	2 * [M/2] * [M/2]
....	
h	$2 * [M/2]^{h-1}$

Relación entre h y # de nodos

$N+1 \ >= \ 2 * [M/2]^{h-1}$

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

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

Arbol balanceado

- La raiz tiene como minimo 2 descendientes

- El nodo verde tiene como minimo [M/2] lo mismo el amarillo
  - Por ende luego del nivel 2 hay → 2 * [M/2] hijos posibles
- Luego cada nodo del nivel tres tendra como minimo [M/2] hijos
  - En el nivel 3 hay 2*[M/2] nodos
  - Por ende luego del nivel 3 hay 2 * [M/2] * [M/2] hijos
- Esto se repite hasta el nivel h
  - En el nivel h hay 2 * [M/2] *… [M/2] punteros nulos (hijos) y eso es $2 * [M/2]^{h-1}$

Árboles Balanceados

Performance de la inserción

- Mejor caso (sin overflow)
  - H lecturas
  - 1 escritura
- Peor caso (overflow hasta la raíz, aumenta en uno el nivel del árbol)
  - H lecturas
  - 2h+1 escrituras (dos por nivel más la raíz)
- Estudios realizados
  - M = 10 25% divisiones
  - M = 100 2% divisiones

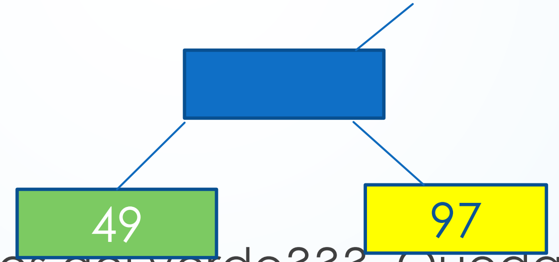
Árboles Balanceados

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

Arboles balanceados

- Definición: nodo adyacente hermano
  - Dos nodos son adyacentes hermanos si tienen el mismo padre y son apuntados por punteros adyacentes en el padre.
- Supongamos el siguiente caso orden del arbol 100, minima cantidad de elementos en un terminal 49

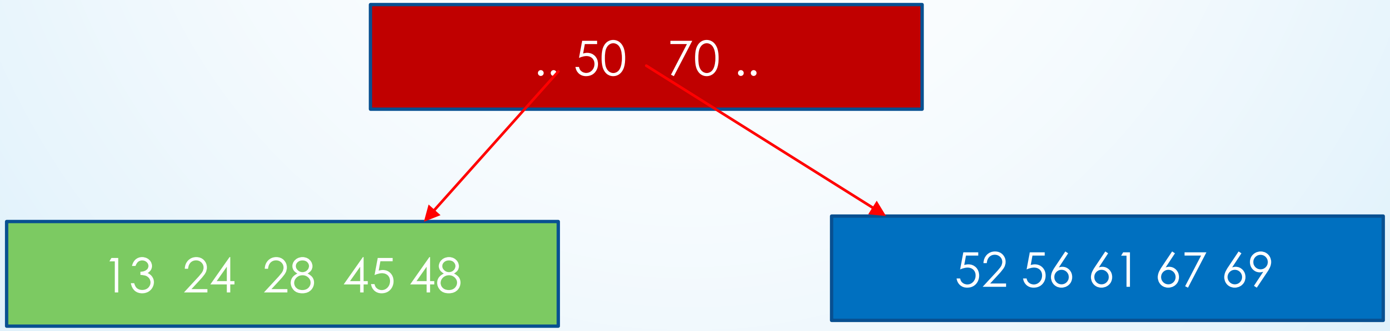


- Que pasa si borramos del verde??? Quedan 48 → underflow
  - Lo opuesto del overflow
  - Juntar con el amarillo → imposible 48 + 97 + elemento del padre = 146

Arboles balanceados

- No siempre se puede concatenar en caso de un underflow. Ej: arbol de orden 10:

- Se borra el elemento 39 → nodo verde entra en underflow
  - No se puede concatenar
  - Se debe redistribuir 13 24 28 45 48 50 52 56 61 67 69



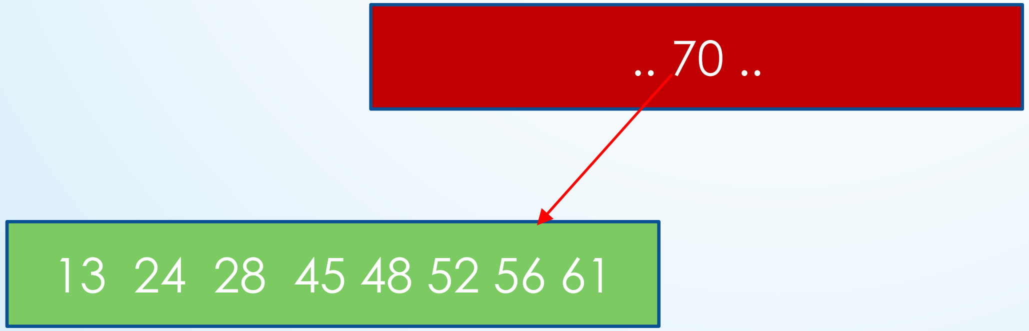
Arboles balanceados

- Suponga ahora que el caso es el siguiente (ordel del arbol 10)

- Quiero borrar 39 → en este caso es imposible redistribuir

  13 24 28 45 48 52 56 61   entra en underflow el nodo azul

- La unica opcion es concatenar



Árboles Balanceados

Redistribuir

- Cuando un nodo tiene underflow puede trasladarse llaves de un nodo adyacente hermano (en caso que este tenga suficientes elementos)

Concatenación:

- Si un nodo adyacente hermano está al mínimo (no le sobra ningún elemento) no se puede redistribuir, se concatena con un nodo adyacente disminuyendo el # de nodos (y en algunos casos la altura del árbol)

Árboles Balanceados

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

Árboles Balanceados → B*

Eliminación

- Redistribución
- Concatenación

Inserción

- ¿¿¿¿¿¿
- División

Árboles Balanceados → B*

La redistribución podría posponer la creación de páginas nuevas

Se pueden generar árboles B más eficientes en términos de utilización de espacio

Árboles Balanceados → B*

Árbol B especial en que cada nodo está lleno por lo menos en 2/3 partes

Propiedades (orden M)	
	Cada página tiene máximo M descendientes
	Cada página, menos la raíz y las hojas, tienen al menos [(2M – 1) / 3] descendientes
	La raíz tiene al menos dos descendientes (o ninguno)
	Todas las hojas aparecen en igual nivel
	Una página que no sea hoja si tiene K descendientes contiene K-1 llaves
	Una página hoja contiene por lo menos [(2M – 1) / 3] –1 llaves, y no más de M-1.

Árboles Balanceados → B*

Operaciones de Búsqueda

- Igual que el árbol B común

Operaciones de Inserción

- Tres casos posible
  - **Derecha**: redistribuir con nodo adyacente hermano de la derecha (o izq. si es el último)
  - **Izquierda o derecha**: si el nodo de la derecha está lleno y no se puede redistribuir, se busca el de la izquierda.
  - **Izquierda y derecha**: busca llenar los tres nodos, estos tendrán un ¾ parte llena.

Ejemplos

Arboles B*

Creacion:

- Dadas las claves: 43 2 53 88 75 80 15 49 60 20 57 24
- Como se construye el árbol B*?
- Como se general el archivo de datos que persiste el árbol?

Arbol B*

Raiz = ??								
	punteros				clave (datos)			#reg
Nodo 0								

Llega el elemento 43

Nodo 0

Raiz = 0								
	punteros				clave (datos)			#reg
Nodo 0	-1	-1			43			1

Llega el elemento 2

Nodo 0

Raiz = 0								
	punteros				clave (datos)			#reg
Nodo 0	-1	-1	-1		2	43		2

Arbol B*

Nodo 0

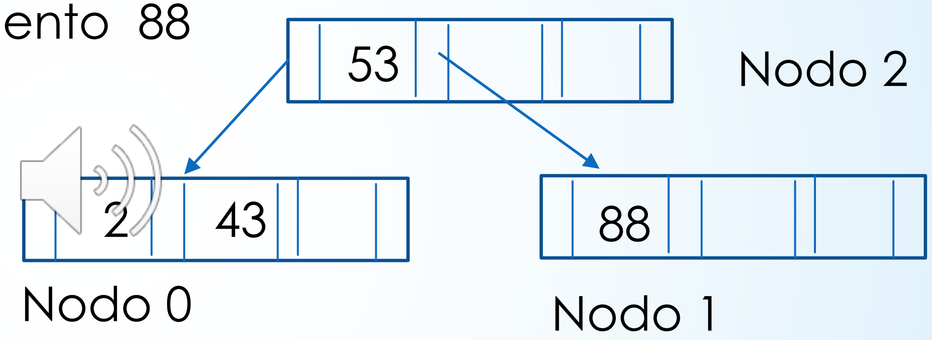
Llega el elemento 53

Raiz = 0								
	punteros				clave (datos)			#reg
Nodo 0	-1	-1	-1	-1	2	43	53	3

Llega el elemento 88

Nodo 0 esta lleno

Se divide en 2

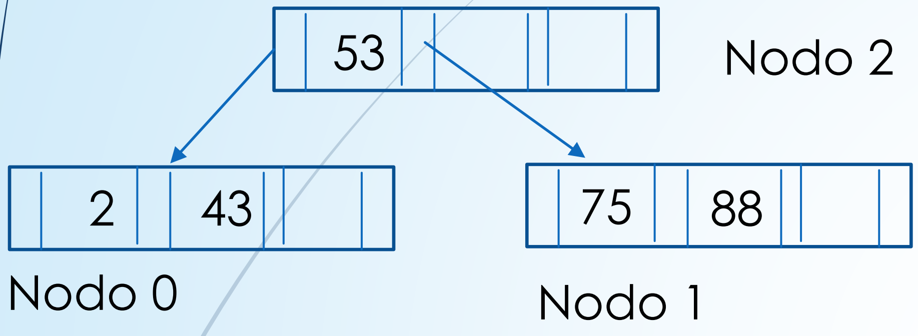


2 43 53 88

Raiz = 2								
	punteros				clave (datos)			#reg
Nodo 0	-1	-1	-1		2	43		2
Nodo 1	-1	-1			88			1
Nodo 2	0	1			53			1

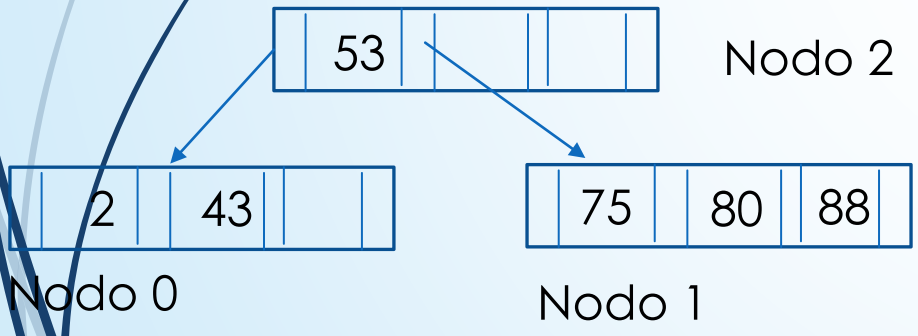
Arbol B*

Llega el elemento 75



Raiz = 2								
	punteros				clave (datos)			#reg
Nodo 0	-1	-1	-1		2	43		2
Nodo 1	-1	-1	-1		75	88		2
Nodo 2	0	1			53			1

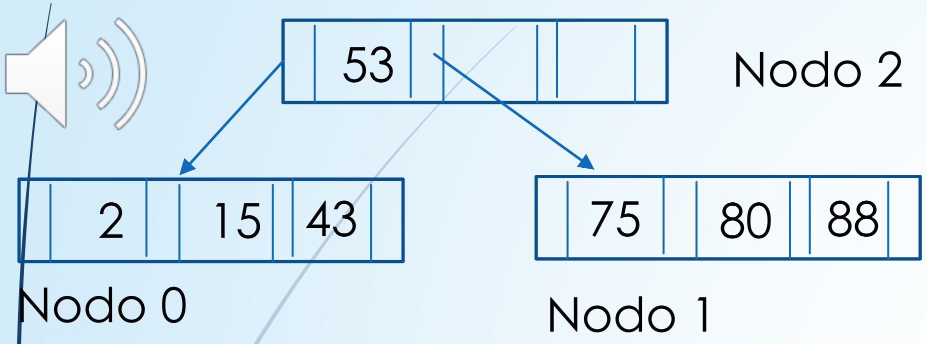
Llega el elemento 80



Raiz = 2								
	punteros				clave (datos)			#reg
Nodo 0	-1	-1	-1		2	43		2
Nodo 1	-1	-1	-1	-1	75	80	88	3
Nodo 2	0	1			53			1

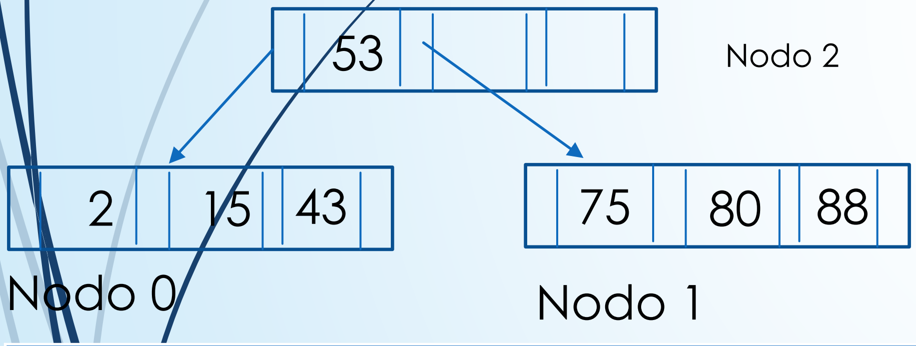
Arbol B*

Llega el elemento 15

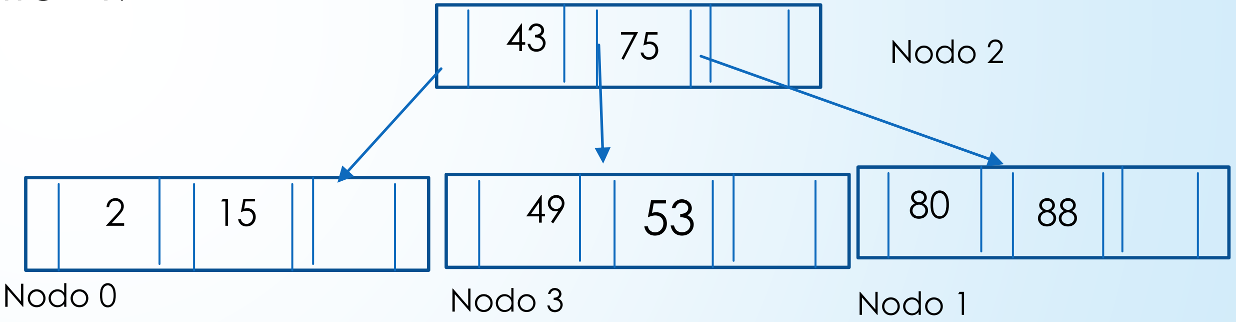


Raiz = 2								
	punteros				clave (datos)			#reg
Nodo 0	-1	-1	-1	-1	2	15	43	3
Nodo 1	-1	-1	-1	-1	75	80	88	3
Nodo 2	0	1			53			1

Llega el elemento 49



Se divide el nodo 0

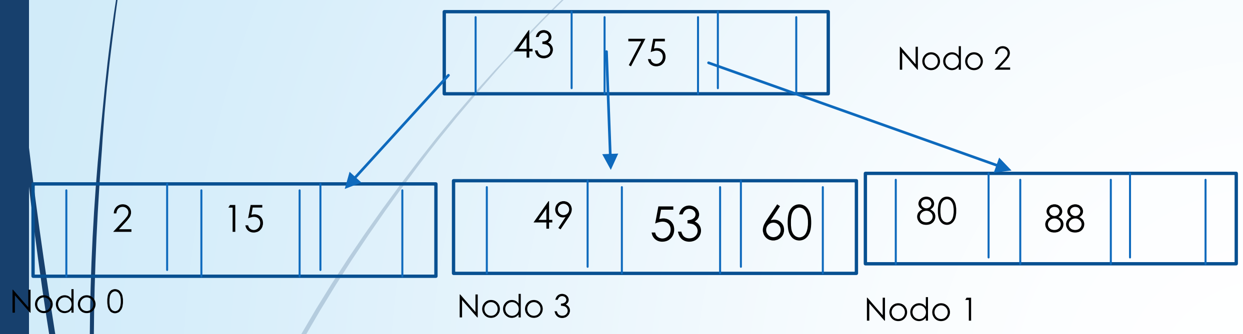


2 15 43 49 53 75 80 88

Raiz = 2							
	punteros			clave (datos)			#reg
Nodo 0	-1	-1	-1		2	15		2
Nodo 1	-1	-1	-1		80	88		2
Nodo 2	0	3	1		43	75		2
Nodo 3	-1	-1	-1		49	53		2

Arbol B*

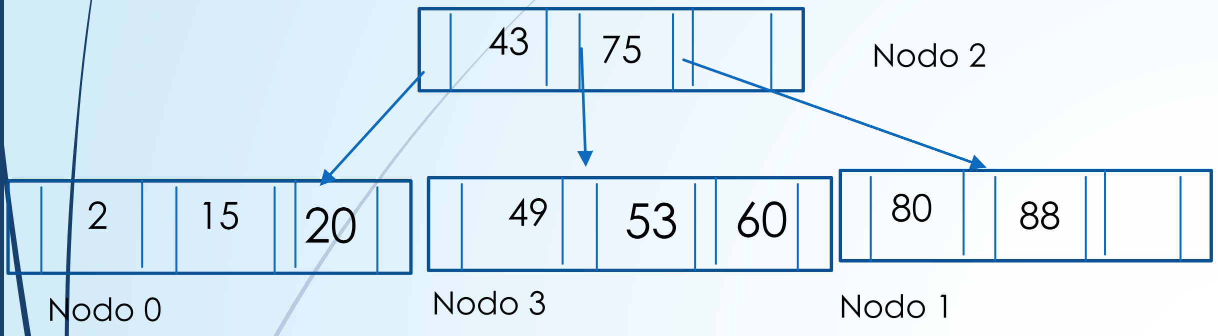
Llega el elemento 60



Raiz = 2								
	punteros				clave (datos)			#reg
Nodo 0	-1	-1	-1		2	15		2
Nodo 1	-1	-1	-1		80	88		2
Nodo 2	0	3	1		43	75		2
Nodo 3	-1	-1	-1	-1	49	53	60	3

Arbol b*

Llega el elemento 20

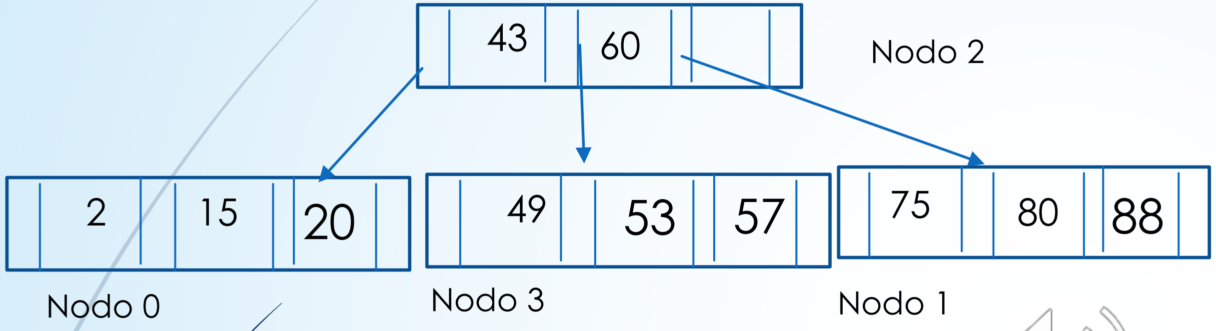


Raiz = 2

	punteros				clave (datos)			#reg
Nodo 0	-1	-1	-1	-1	2	15	20	3
Nodo 1	-1	-1	-1		80	88		2
Nodo 2	0	3	1		43	75		2
Nodo 3	-1	-1	-1	-1	49	53	60	3

Arbol B*

Llega el elemento 57



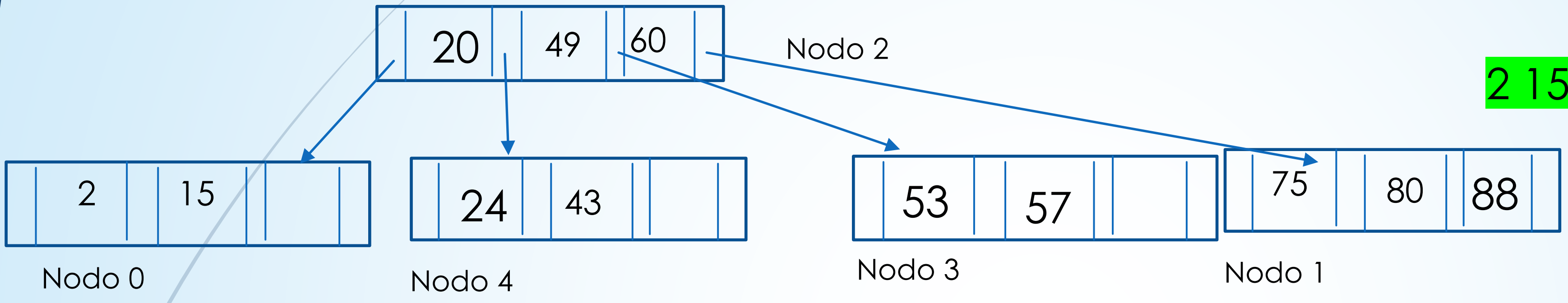
Raiz = 2

	punteros				clave (datos)			#reg
Nodo 0	-1	-1	-1	-1	2	15	20	3
Nodo 1	-1	-1	-1	-1	75	80	88	3
Nodo 2	0	3	1		43	60		2
Nodo 3	-1	-1	-1	-1	49	53	57	3

49 53 57 60 75 80 88

Arbol B*

Llega el elemento 24

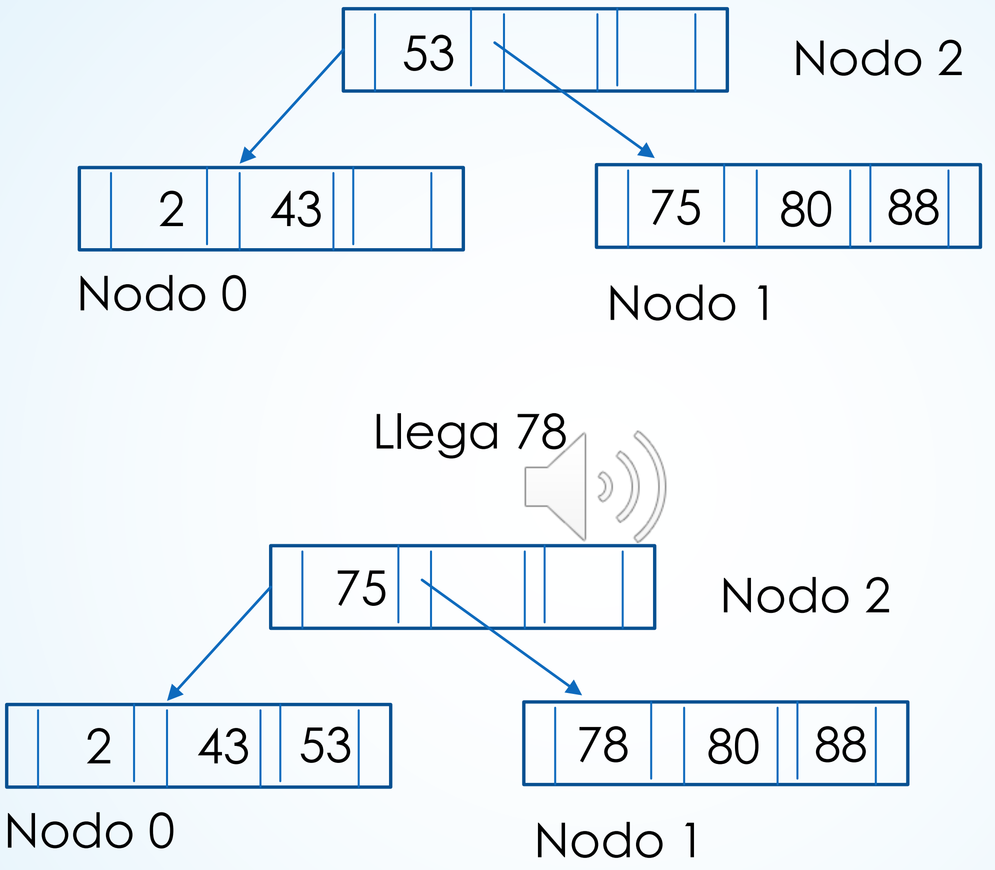


2 15 20 24 43 49 53 57

Raiz = 2

	punteros				clave (datos)			#reg
Nodo 0	-1	-1	-1		2	15		2
Nodo 1	-1	-1	-1	-1	75	80	88	3
Nodo 2	0	4	3	1	20	49	60	3
Nodo 3	-1	-1	-1		53	57	57	2
Nodo 4	-1	-1	-1		24	43		2

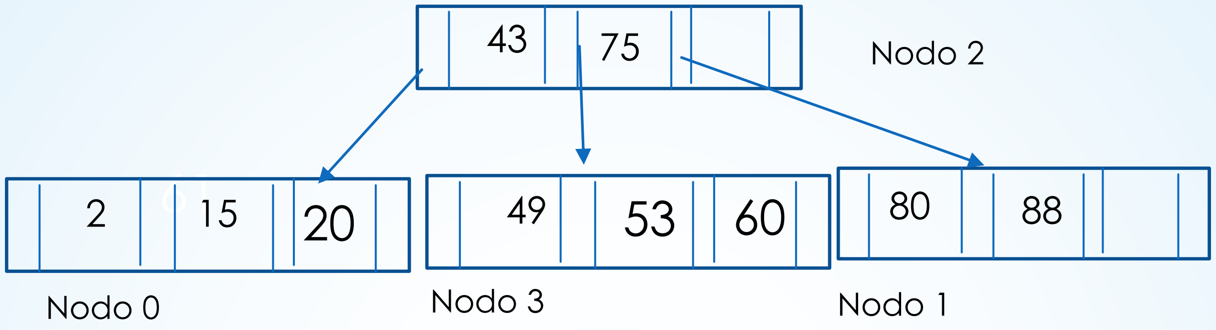
Arbol B*

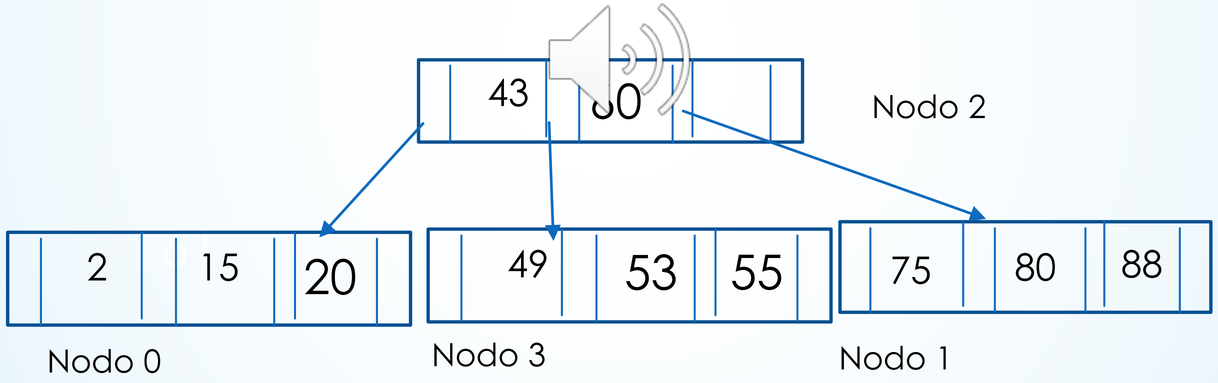


53
Nodo 2
2
43
75
80
88
Nodo 0
Nodo 1
Llega 78
75
Nodo 2
2
43
53
78
80
88
Nodo 0
Nodo 1

Arbol B*

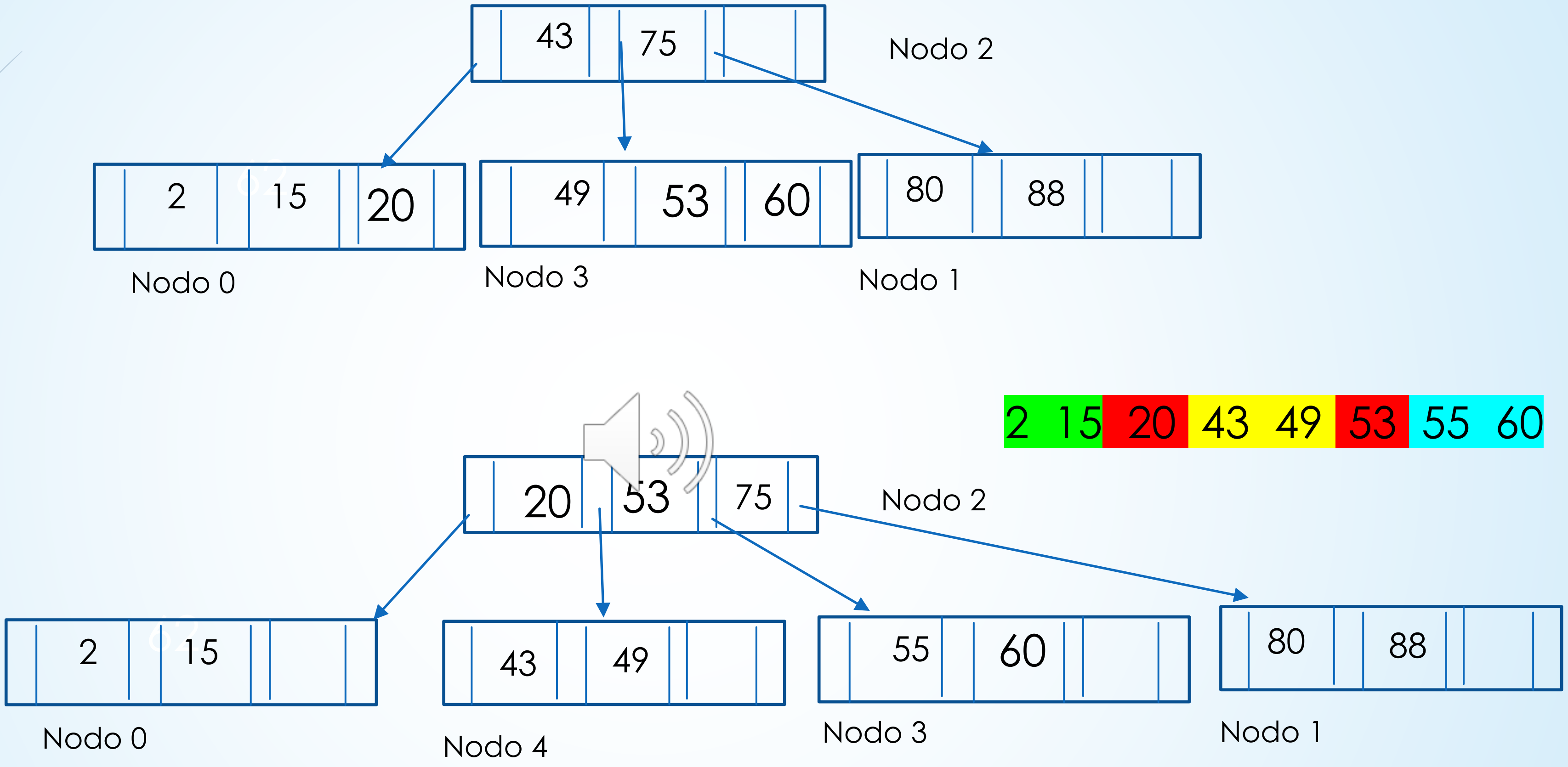
Llega el elemento 55 politica de derecha





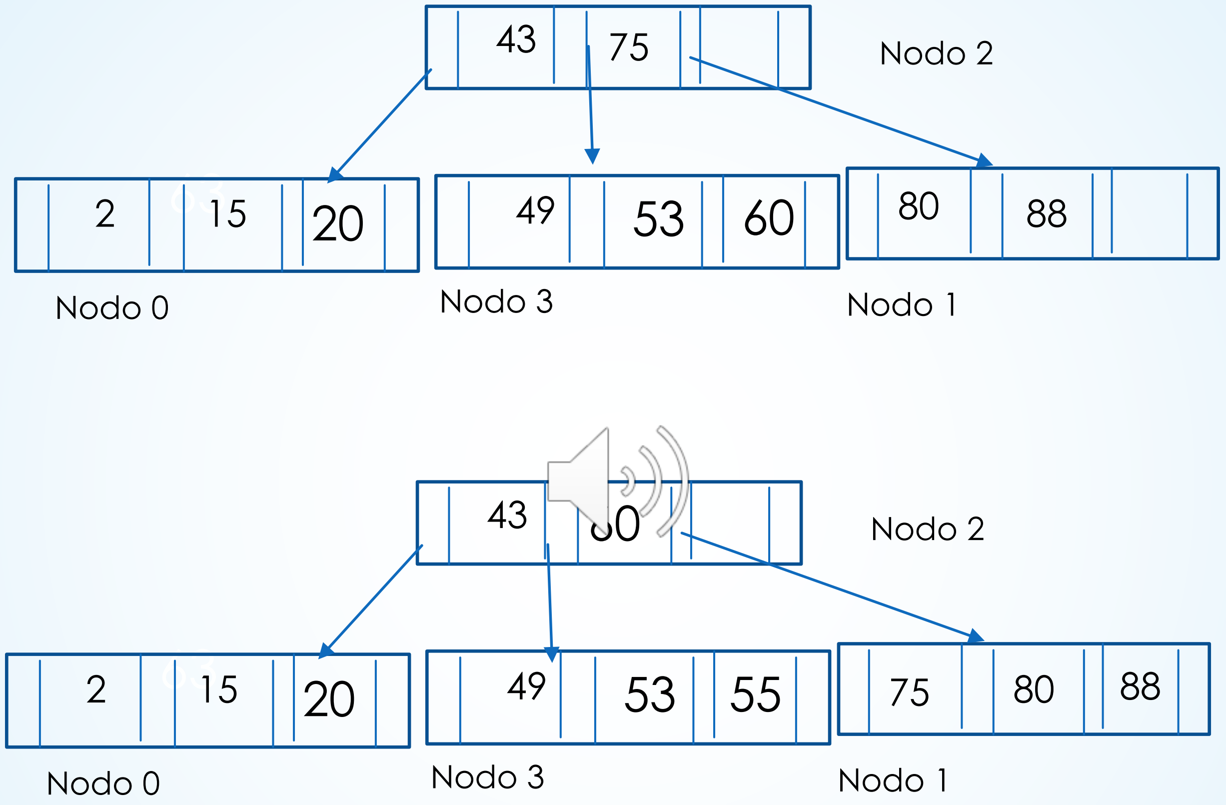
Arbol B*

Llega el elemento 55 politica de izquierda



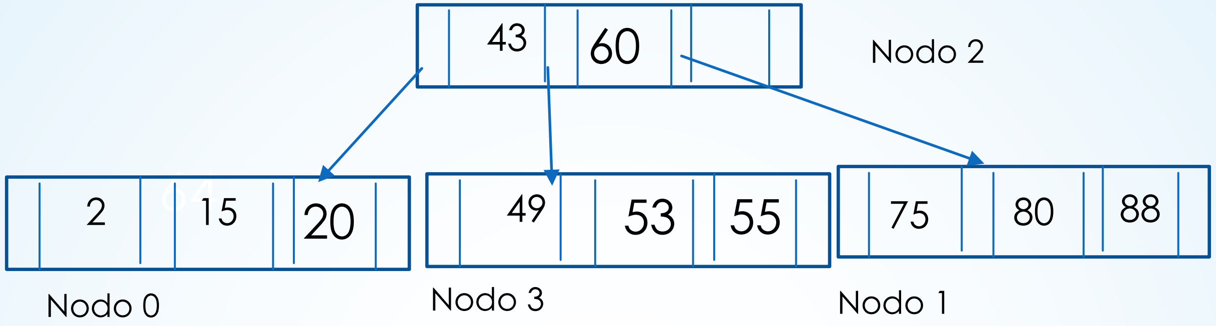
Arbol B*

Llega el elemento 55 izquierda o derecha

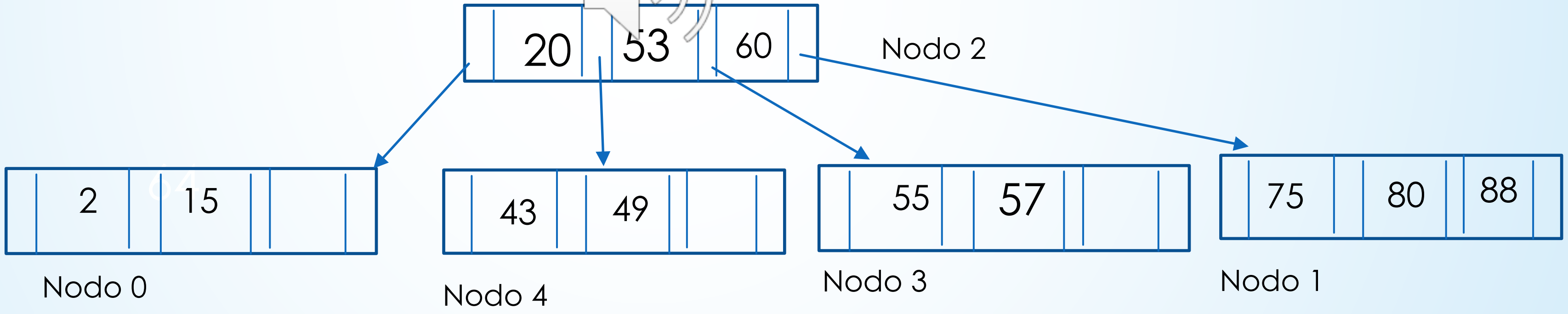


Arbol B*

Llega el elemento 57 izquierda o derecha

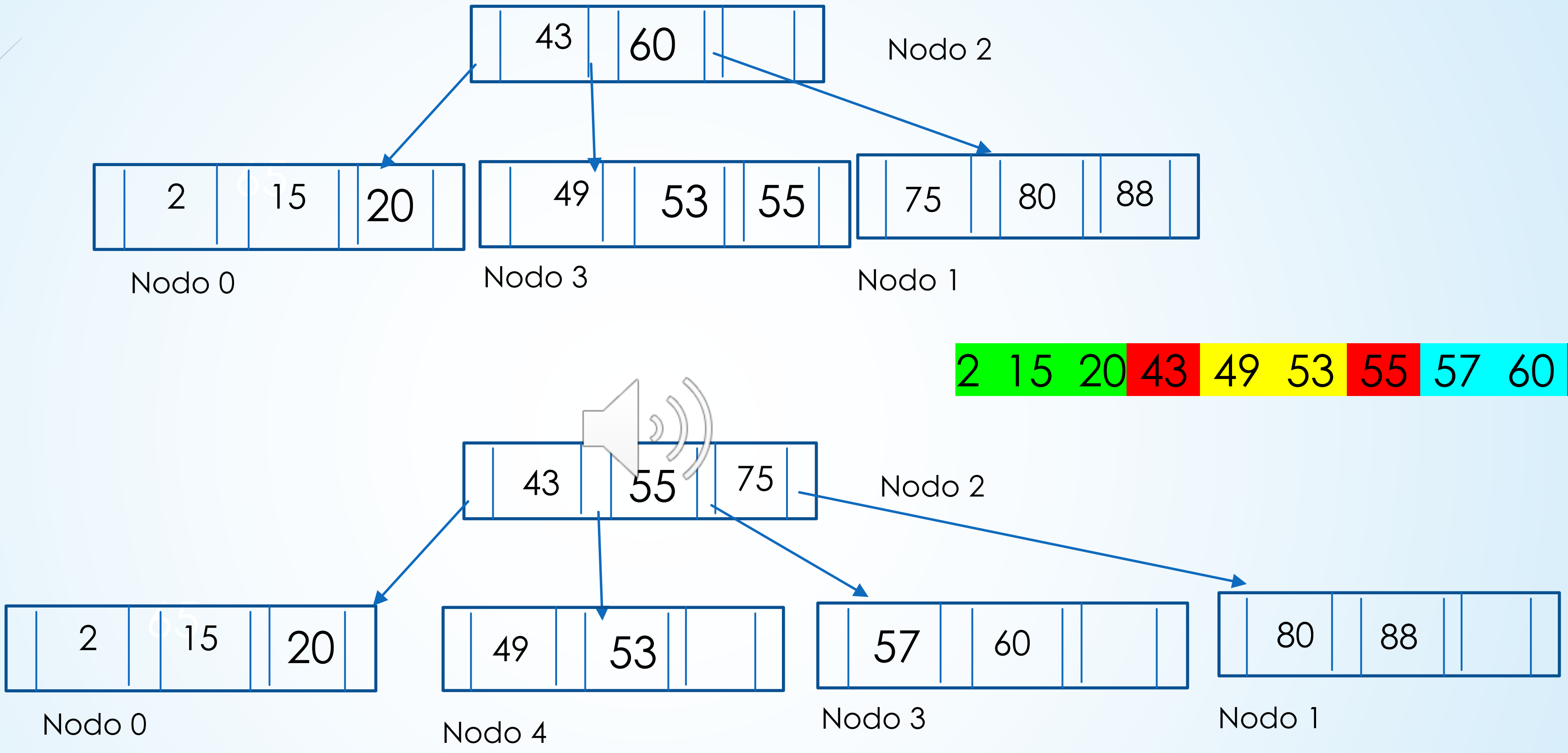


2 15 20 43 49 53 55 57



Arbol B*

Llega el elemento 57 izquierda y derecha



2	15	20	43	49	53	55	57	60	75	80	88

Árboles Balanceados → B*

Costo de la redistribución

	Mejor	Peor
Derecha	RRWW	RRWWW
Izq o der	RRWW	RRRWWW (divido solo dos)
Izq y der	RRWW	RRRWWWW

Árboles Balanceados

Técnicas de paginado

estrategias de reemplazo: LRU (last recently used)

Análisis numérico

llaves = 2400 # páginas = 140 Altura = 3 niveles

1	5	10	20
3.00	1.71	1.42	0.97

Árboles Balanceados

Archivos secuenciales indizados	Permiten una mejor recorrida por algún tipo de orden	Indizado (ordenado por una llave) Secuencial (acceder por orden físico, devolviendo el registro en orden de llave)
	Hasta ahora métodos disjuntos, se opta:	rápida recuperación (Árbol) Recuperación ordenada (secuencial)
	Debemos encontrar una solución que agrupe ambos casos	

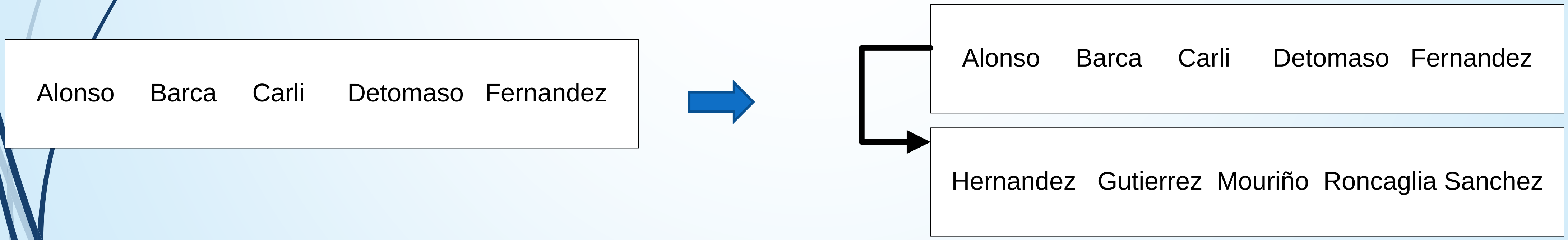
Árboles Balanceados

Conjunto de secuencias

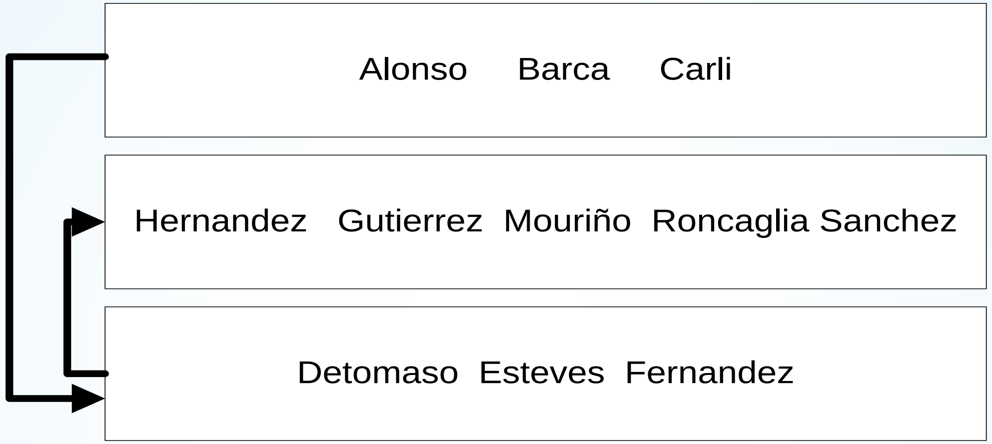
- Conjunto de registros que mantienen un orden físico por llave mientras que se agregan o quitan datos, si podemos mantenerlo podemos indizarlos

Posible solución

- Mantener bloques de datos
- Cada bloque con registros y puntero al siguiente



Árboles Balanceados



Costo

- Aumenta el tamaño del archivo (fragmentación interna)
- No hay orden físico salvo dentro del un bloque.
- Tamaño del bloque
  - Debe permitir almacenar varios bloques en RAM (redistribucióon)
  - Las E/S deben ser rápidas y sin necesidad de desplazamientos
- Como logramos ahora una rápida búsqueda?

Árboles Balanceados → B+

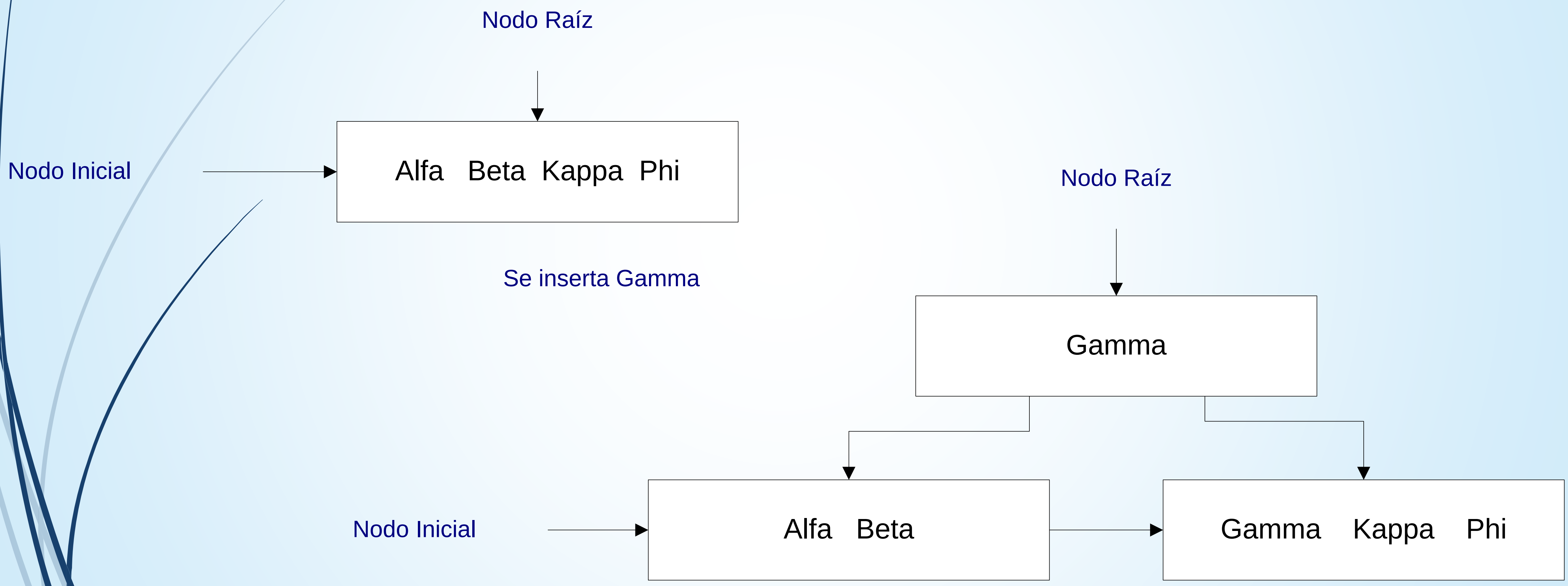
Consiste en un conjunto de grupos de registros ordenados por clave en forma secuencial, junto con un conjunto de índices, que proporciona acceso rápido a los registros.

Propiedades

- Cada página tiene máximo M descendientes
- Cada página, menos la raíz y las hojas, tienen entre [M/2] y M hijos
- La raíz tiene al menos dos descendientes (o ninguno)
- Todas las hojas aparecen en igual nivel
- Una página que no sea hoja si tiene K descendientes contiene K-1 llaves
- Los nodos terminales representan un conjunto de datos y son linkeados juntos.

Los nodos no terminales no tienen datos sino punteros a los datos.

Árboles Balanceados → B+



Árboles Balanceados → B+

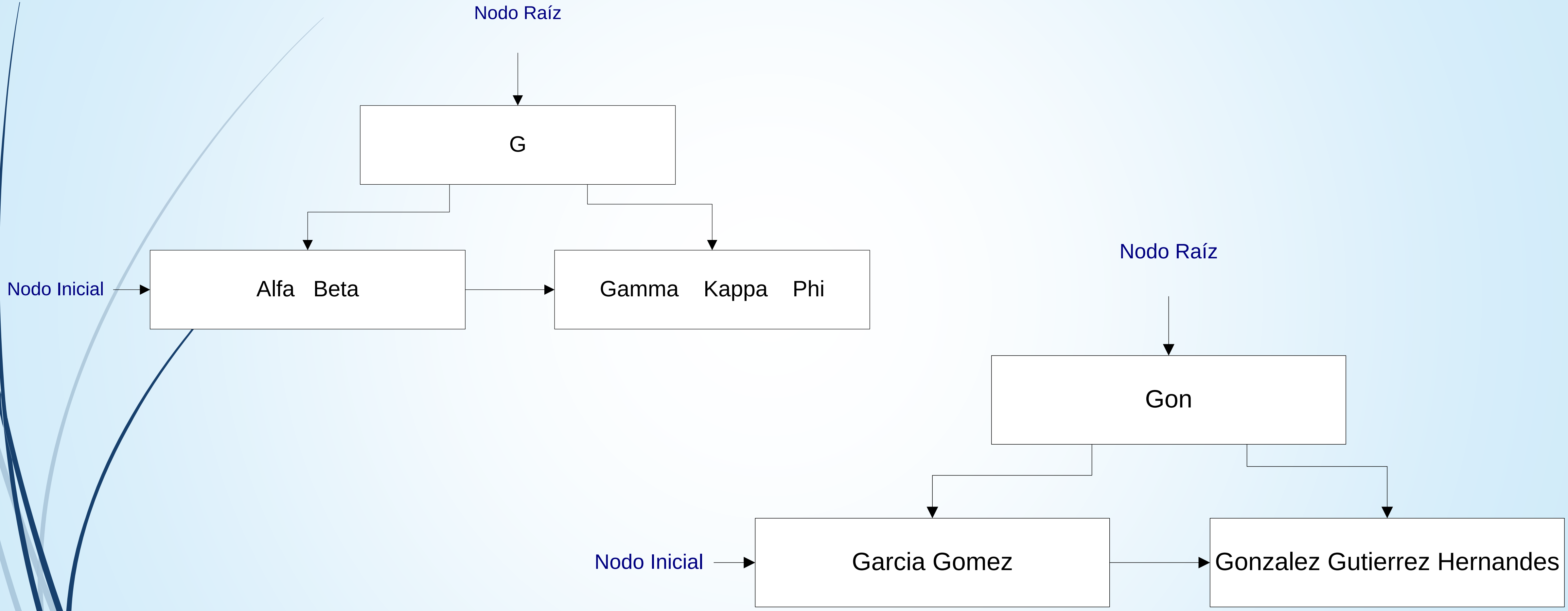
Separadores

- Derivados de las llaves de los registros que limitan un bloque en el conjunto de secuencia
- Separadores más cortos, ocupan espacio mínimo

Árbol B+ de prefijos simples

- Árbol B+ en el cual el conjunto índice está constituido por separadores más cortos

Árboles Balanceados → B+



Árboles Balanceados → conclusiones

	Árbol B	Árbol B+
Ubicación de datos	Nodos (cualquiera)	Nodo Terminal
Tiempo de búsqueda	=	=
Procesamiento secuencial	Lento (complejo)	Rápido (con punteros)
Inserción eliminación	Ya discutida	Puede requerir + tiempo

Árboles

- Operaciones clásicas
- Comparaciones

	Árbol B	Árbol B+
Ubicación de datos	Nodos (cualquiera)	Nodo Terminal
Tiempo de búsqueda	=	=
Procesamiento secuencial	Lento (complejo)	Rápido (con punteros)
Inserción eliminación	Ya discutida	Puede requerir + tiempo