Archivos

dámian Bongi

Tipos de Archivos

- registros de longitud fija (FILE OF <tipo_de_dato>
- Texto (text)
- bloques de bytes (files)

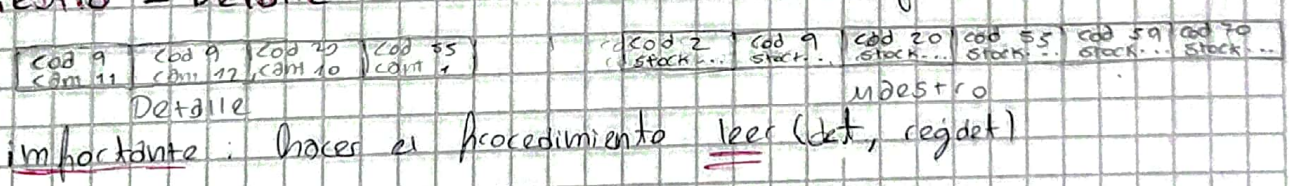
Operaciones

- assign (nom_log, nom_fis) // enlace hardware
- rewrite (nom_log) // crea
- reset (nom_log) // Abre
- close (nom_log) // cierra
- filePos (nom_log) // devuelve pos. del puntero
- read (nom_log, var_dato)
- write (nom_log, var_dato)
- EOF (nom_log) // fin Arch
- fileSize (nom_log) // tam Arch
- seek (nom_log, pos) // posiciona el punt

Algoritmico sobre Archivos

- **Maestro - Detalle**

Maestro: resume info Detalle: mov. sobre el maestro



* importante: hacer el procedimiento leer (det, regdet)

- **Corte de Control**

proceso mediante el cual la información de un archivo es presentada en forma organizada de acuerdo a la estructura que posee el archivo.

- **Merge**

proceso mediante el cual se genera un nuevo archivo a partir de otros archivos existentes

* uso del procedimiento "minimo"

dómian Bonzi

Algoritmica sobre Archivos

Bajas: proceso que permite quitar información de un Archivo

- **baja física:** borra efectivamente la información recuperando el espacio físico

cod 1 lola	cod 2 jorge	cod 3 pepe	cod 4 juan

el reg pasa a tener 3 datos en vez de 4

* siempre tengo el tamaño real del Archivo, ocupa el menor espacio posible
* el Algoritmos es mas dificil de implementar

formas de realizar la baja física

- pasar todo a un Archivo nuevo a excepcion del registro que quiero borrar

- **baja logica:** borra la información del archivo pero sin recuperar el espacio físico

cod 1 lola	cod 2 jorge	cod 3 pepe	cod 4 juan

pongo el cod por ej en -4, para decir que ese reg no interesa

* ocupa mas espacio
* el Algoritmos es mas sencillo

forma de realizar baja logica

- recorrer el archivo y poner la marca de borrado, en el archivo que quiero "borrar"

Reutilizacion del espacio, usando lista invertida

- se tiene un archivo de enteros, en donde el registro en la posicion cero (0) se le llama registro cabecera (es un registro anudado), este contiene la posicion del registro eliminado (a reutilizar), este numero se guarda en negativo para no confundirse, este reg. cabezera contiene la ultima posicion que se elimino y esa posicion tiene la ante ultima posision que se elimino, si tiene un cero es porque es la ultima eliminacion que se hizo, o el ultimo dato a reutilizar

Registro cabecera

0	1	2	3	4	5	6	7	8	9
0	156	304	228	98	116	504	824	59	715

0	1	2	3	4	5	6	7	8	9
0	156	-5	228	98	0	504	-2	59	715

eliminacion de claves:

- 116
- 304
- 824

Arboles

Damian Bonzi

Arboles B: son arboles multicamino con una construccion especial de Arboles que permite mantenerlos balanceados a bajo costo.

- cada nodo del arbol puede contener como maximo M descendientes directos (hijos)
- la raiz no posee descendientes directos o tiene al menos dos (minimo 2 hijos)
- un nodo interno con X hijos contiene X-1 elementos
- todos los nodos terminales (hojas) se encuentran al mismo nivel
- todos los nodos (salvo la raiz) tienen como minimo $\lceil M/2 \rceil - 1$ elementos y como maximo $M-1$ elementos
- cada nodo tiene sus elementos ordenados por clave. Ademas, todos los elementos en el subarbol izquierdo de un elemento son menores o iguales a que dicho elemento y todos los elementos en el subarbol derecho son mayores que ese elemento

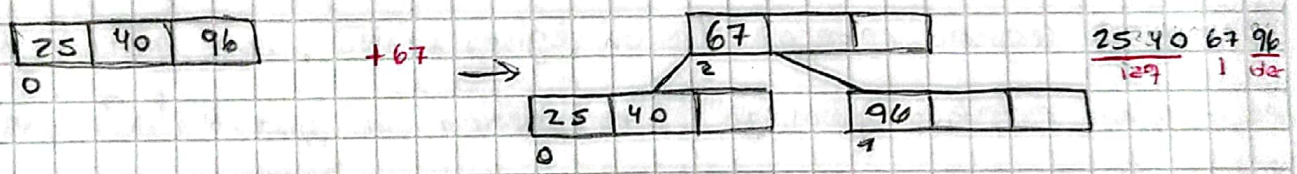
Over flow

1ero Se crea un nuevo nodo

2do la primera mitad de las claves se mantiene en el nodo con overflow

3ero la segunda mitad de las claves se traslada a un nuevo nodo

4to la menor de las claves de la segunda mitad se promociona al nodo padre



* el nodo 0 es el inicial, cuando se produce el overflow, se genera el nodo 1, en el que queda la segunda mitad, que serian en este caso 67 y 97 y por ultimo la menor de esas claves (67) queda como raiz en el nodo 2

Bajas

1. si la clave a eliminar no esta en una hoja, se debe reemplazar con la menor clave del subArbol derecho.
2. si el nodo hoja contiene por lo menos el minimo numero de claves, luego de la eliminacion, no se requiere ninguna accion
3. En caso contrario se debe tratar el underflow

* Importante: siempre las bajas en los arboles se realizan en las hojas

Damian Bongi

Underflow (Bajas)

4- primero se intenta redistribuir con un hermano adyacente. La redistribucion es un proceso mediante el cual se trata de dejar cada nodo lo mas equivalente cargado posible

5- Si la redistribucion no es posible, entonces se debe fusionar con el hermano adyacente

formas de redistribuir

- Politicas para la resolucion de underflow

1. politica izquierda: se intenta redistribuir con el hermano adyacente izquierdo, si no es posible, se fusiona con hermano adyacente izquierdo

2. politica derecha: se intenta redistribuir con el hermano adyacente derecho, si no es posible, se fuciona con hermano adyacente derecho.

3. politica izq. o der: se intenta redistribuir con el hermano adyacente izq si no es posible, se intenta con el hermano adyacente derecho, si tampoco es posible, se fusiona con hermano adyacente izquierdo.

4. politica der. o izq: se intenta redistribuir con el hermano adyacente derecho, si no es posible, se intenta con el hermano adyacente izquierdo, si tampoco es posible, se fusiona con hermano adyacente derecho.

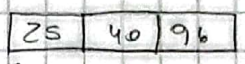
Casos especiales: en cualquier politica si se tratase de un nodo hoja de un extremo del arbol debe intentarse redistribuir con el hermano adyacente que el mismo posea

- en caso de underflow lo primero que se intenta SIEMPRE es redistribuir si el hermano adyacente se encuentra en condicion de hacer la redistribucion y no se produce underflow en el

Altas en Arboles B

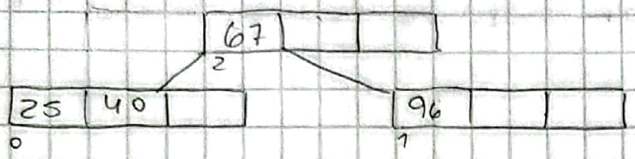
Deimian Bonzi

+25
+40
+96

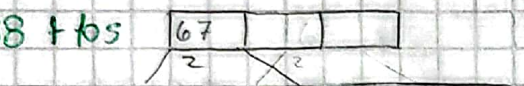


+67
(division de la raiz se incrementa la Altura del Arbol)

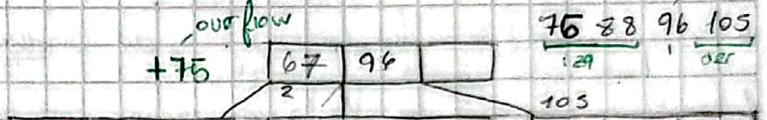
- Se produce OVERFLOW, sigo los pasos del mismo
- (creo nuevo nodo y divido las claves, la 1era mitad se queda la otra se va)
- por ultimo se promociona la menor de las claves de la segunda mitad



+88 +105



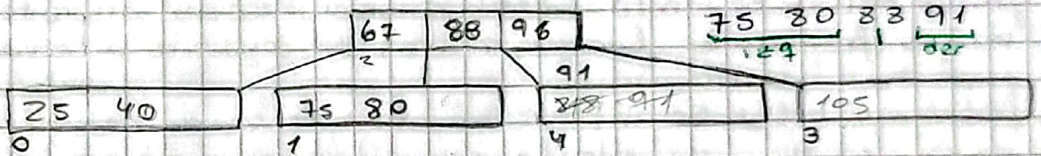
+75



Promociono
76 88 96 105
izq der

+91

+80 → overflow



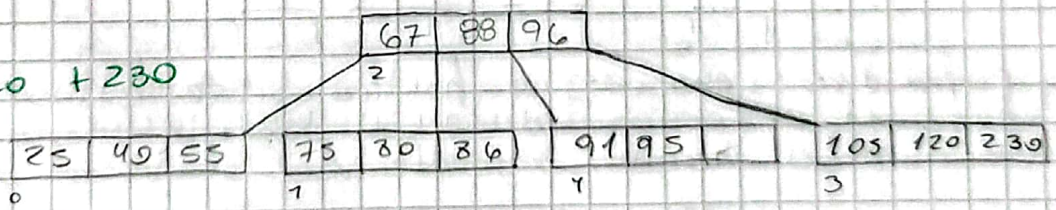
75 80 88 91
izq der

- lecturas y escrituras para el Alta de la clave 80 L2, L1, E1, E4, E2

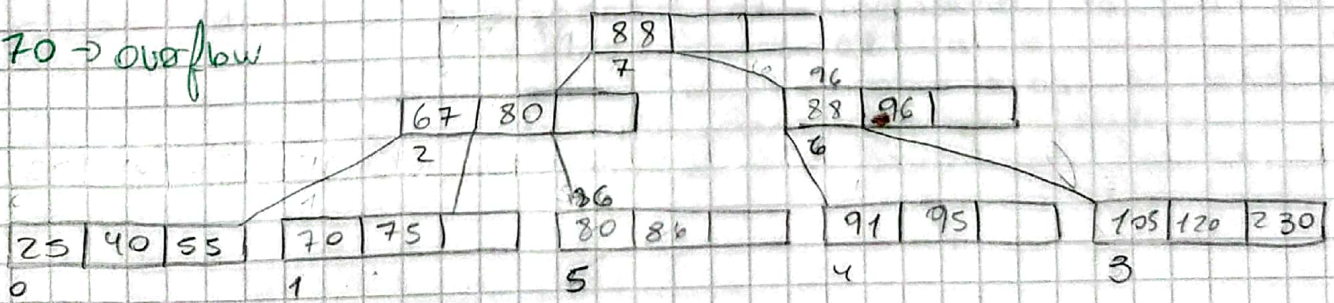
la escritura de los nodos, se puede hacer en cualquiera de los recorridos (in order, post-order, pre-order), PERO, SIEMPRE, Respetar el mismo recorrido

esta esto en in order

+86 +120 +230
+95 +55



+70 → overflow



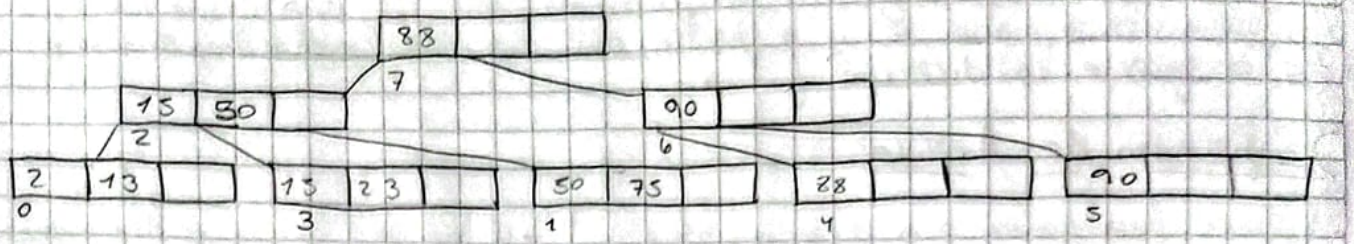
70 75, 80 86
izq raiz der

67 80 88 96
izq raiz der

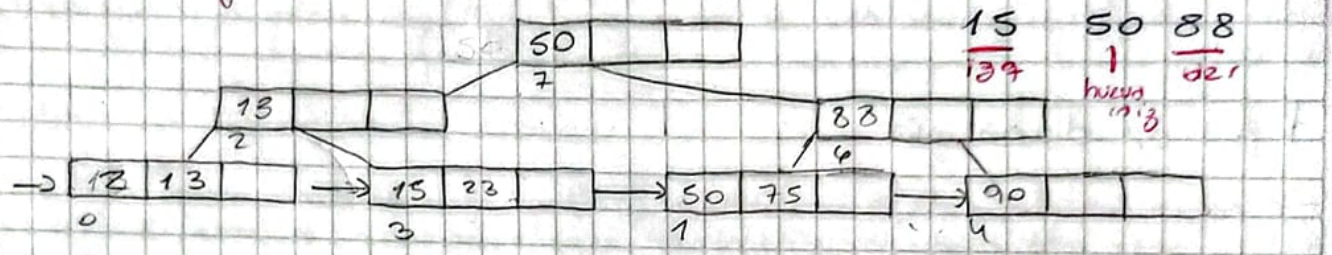
- Se propago el overflow al nodo 2, (anteriormente el overflow fue en el nodo 1)

la propagacion del overflow a la raiz hizo que se divida la misma y aumento la altura del Arbol

Démian Bonzi

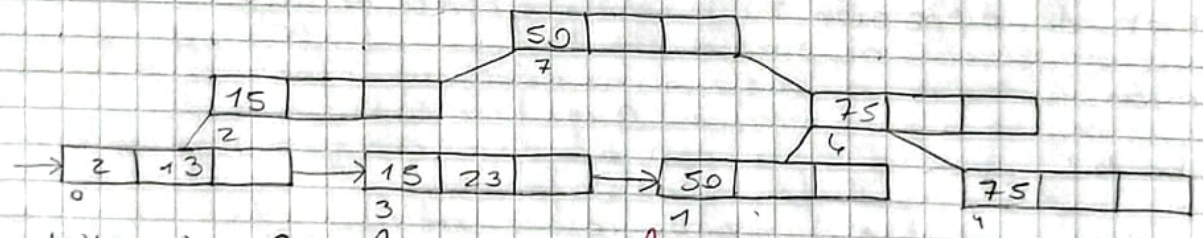


-88 → underflow



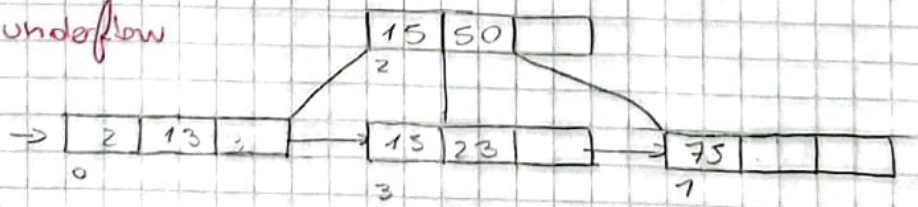
Underflow en nodo 4. No es posible redistribuir y se fusionan los 4 y 5. Se libera el nodo 5. Se propaga el underflow. Redistribucion entre los nodos 2, 7 y 6.

-90 → underflow



• La baja del 90 produce underflow en el nodo 4, se puede balancear las cargas. importante No bajar el 88

-50 → underflow

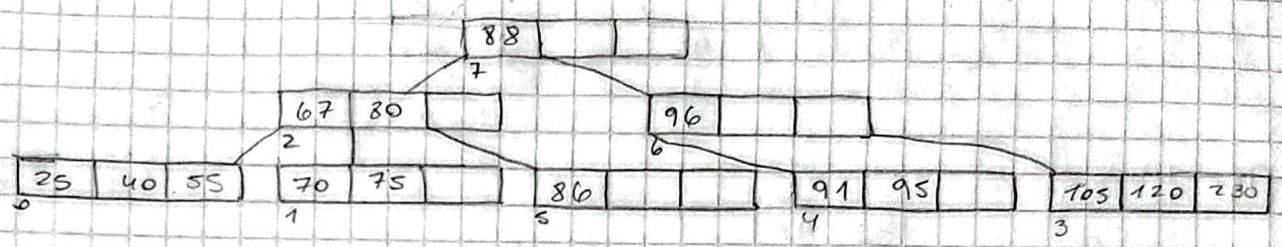


• La baja del 50 produce underflow, pero su unico hermano que es el 75 (nodo 4) esta al minimo entonces no se puede balancear la carga (redistribuir), hay que fusionar el nodo 4 y el nodo 4, pero se propaga el underflow al nodo 6, y el nodo 2 que es el hermano del 6 no puede redistribuir porque tiene la cantidad minima, fusiono el nodo 6 y 4, esto lleva a disminuir en uno el nivel del Arbol

Damian Bonzi

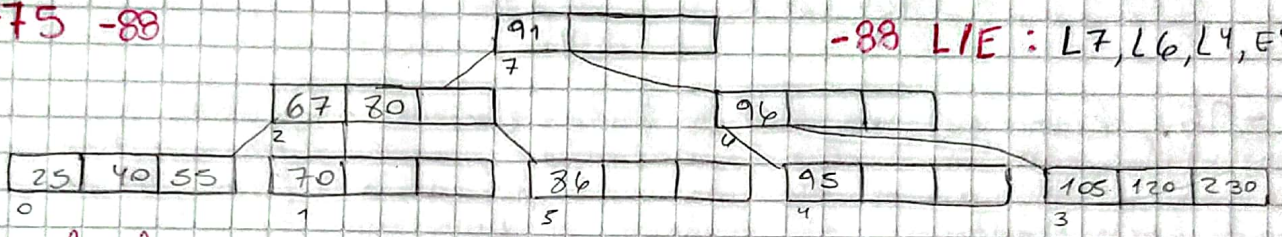
Bajas en Arbodes B

orden del Arbol = 4 Minimo elementos = 1



-75 -88

-88 L/E : L7, L6, L4, E4, E7

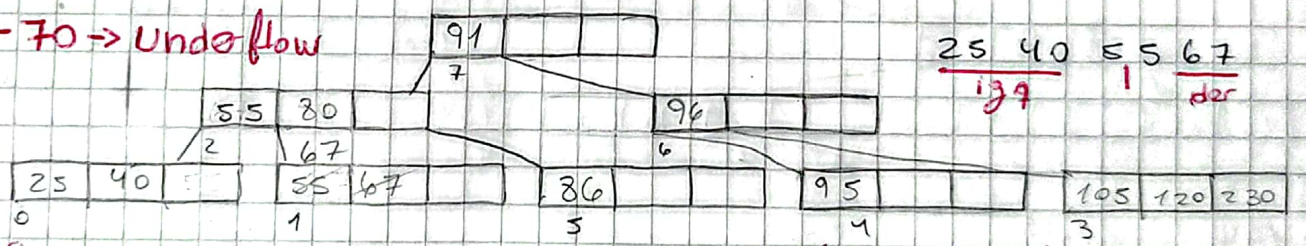


Sigo los pasos de bajas 1ero si no esta en hoja se reemplaza con la menor derecha, 2do, si la hoja tiene por lo menos el minimo de elementos, no se requiere nada mas, 3ero, si no se genera underflow.

En este caso 75 estaba en hoja y quedo con el minimo asi que eliminamos, el 88, lo cambie por el minimo derecho (91) y como tiene el minimo de elemento, no se requiere ninguna accion mas

- baja usando politica derecha o izquierda

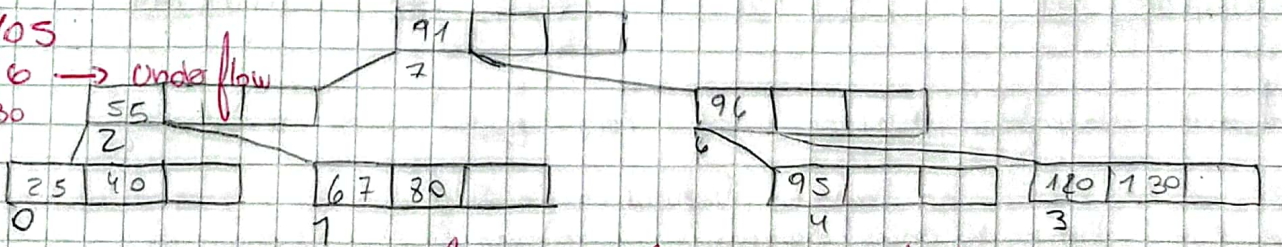
-70 -> Underflow



25 40 55 67
izq der

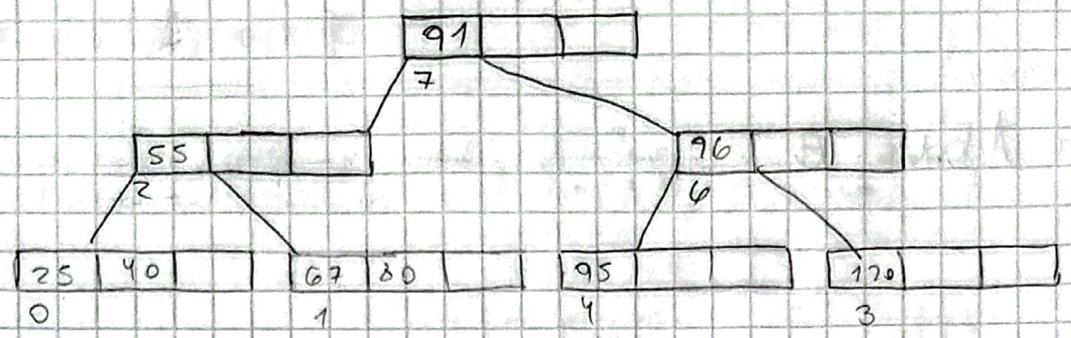
- La eliminacion de la clave 70 en el nodo 1 produce underflow
- Se intenta redistribuir con el hermano derecho. No es posible porque contiene la cant minima de claves
- Se intenta redistribuir con el hermano izquierdo. Si se puede y se rebalancea la carga entre los nodos 130

-105
-86 → underflow
-230

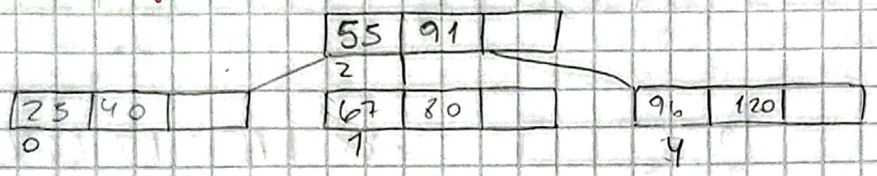


- La eliminacion del 86 produce overflow, se intenta redistribuir con derecho, no tiene, se intento con izquierdo no se puede porque no tiene las claves suficientes, entonces se procede a la fusion con su hermano.

Damian Bonzi

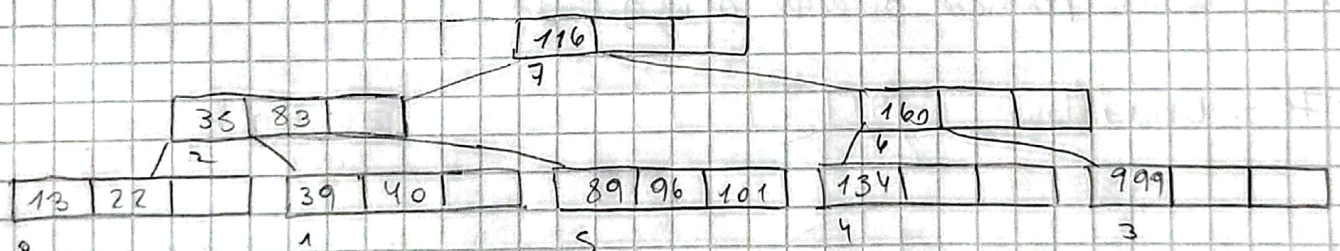


-95 → underflow

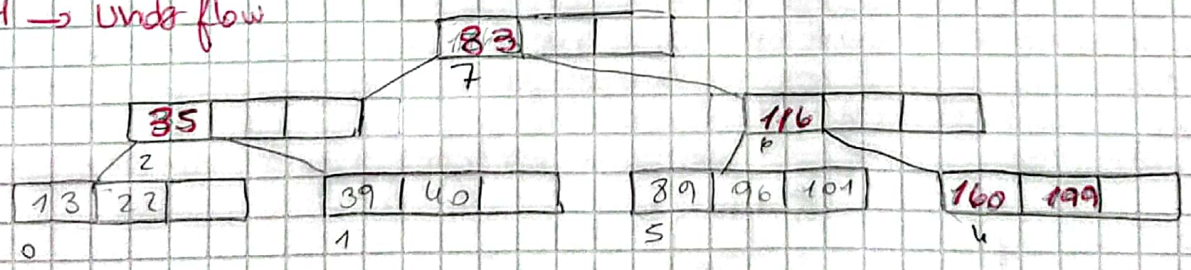


• -95, como estamos usando política derecho o izquierda, se intento redistribuir con el hermano derecho, no se puede porque tiene la cantidad minima de elementos, se procede a la fusion, PERO no se puede, entonces se propaga el UNDERFLOW al nodo 6, y devuelta el mismo procedimiento, se intento redistribuir con derecho, no tiene, se intento con izquierdo, no se puede por que tiene la cantidad minima, entonces fusiono los hermanos (y la raiz) disminuyendo la altura del Arbol.

Redistribucion en nodo interno



-134 → underflow



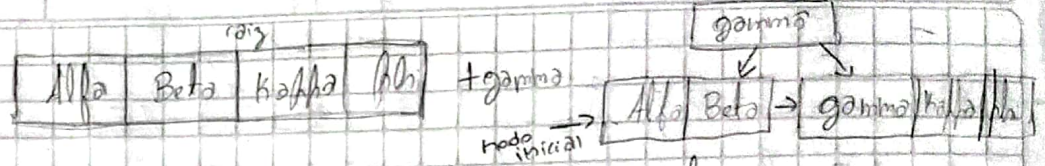
Tengo underflow a nivel medio pero puede balancear las cargas

en este caso no importa la politica porque cuando voy a eliminar el 134, tiene un unico hermano intento hacer la fusion del nodo 4 y 3 y propago el underflow al nodo 6, pero en este caso si puedo hacer balanceo de carga (distribucion), porque el hermano tiene mas que el minimo de elementos, junto la raiz 35 83 116

35 | 83 | 116
izq | der

Damian Bongi

Arboles B+



Los Arboles B+ son una mejora de los Arboles B, porque conservan el Acceso AlEatorio Rapido y ademas permiten un Recorrido Secuencial Rapido

* todos los datos reales ("verdaderos") estan en los nodos terminales (hojas)

- permite recorrer ordenadamente todos los datos, porque tiene un enlace entre cada nodo hoja y porque los datos en los Arboles B+, a diferencia de los arboles B, estan solo en las hojas.

Busqueda B+: las busquedas comienzan en la raiz como en los arboles B pero van a llegar si o si a las hojas, donde se determina si el elemento esta o no.

Insercion B+: la insercion es igual a la del Arbol B, lo unico que cambia es cuando hay **overflow**, si tengo overflow, creo un nuevo nodo, divido equitativamente las claves y se promociona una copia de la menor de las claves mayores o una copia de la clave del medio, si es que hay
- la copia de la clave solo se realiza en un overflow ocurrido a nivel de hoja

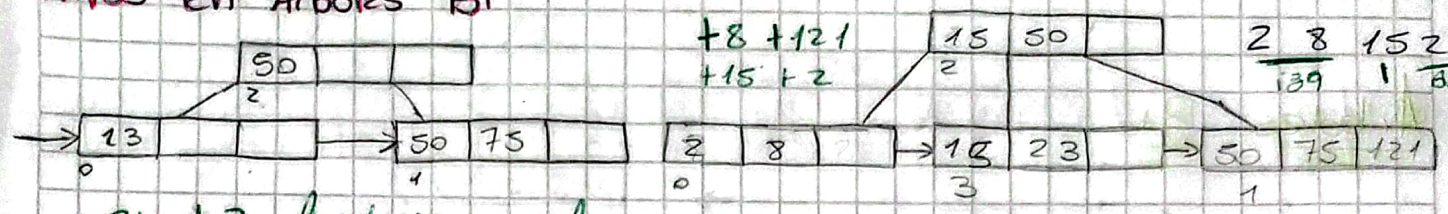
Bajas en B+: las claves a eliminar siempre se encuentran en los nodos hojas.
Si al eliminar la clave, la cantidad de claves que queda es mayor o igual que $[n/2]-1$, entonces termina la operacion

Underflow

- Si al eliminar una clave, la cantidad de llaves es menor a $[n/2]-1$, entonces debe realizarse una redistribucion de claves, tanto en el indice como en las hojas.
- Si la redistribucion no es posible, entonces debe realizarse una fusion entre los nodos

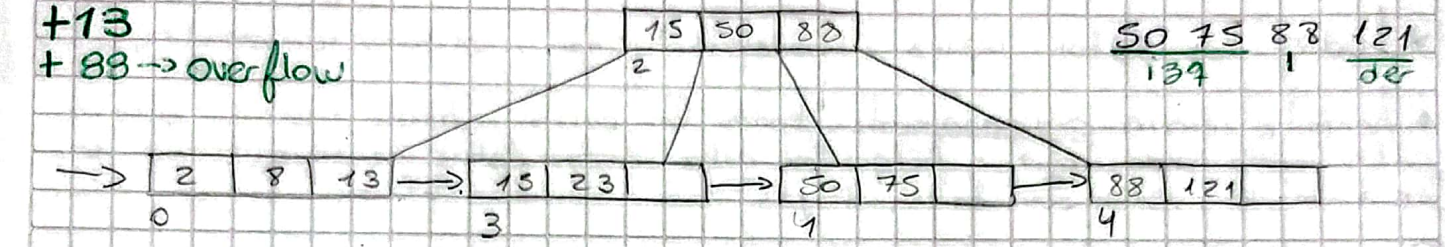
las politicas son las mismas que en B

Altas en Arboles B+

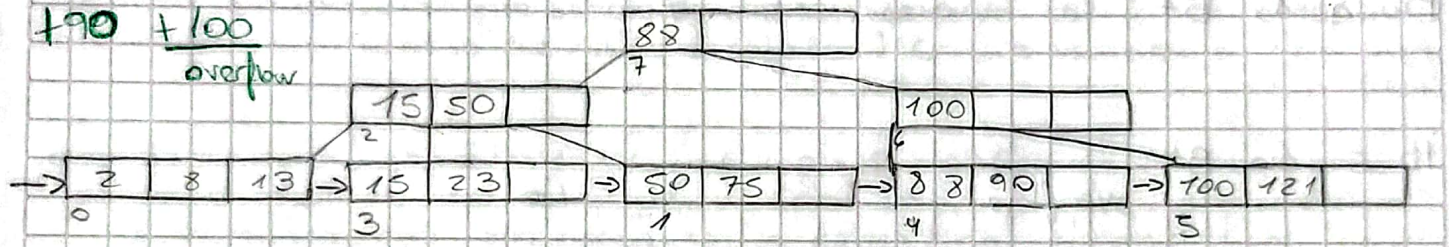


• el +2 producio overflow, se creo el nuevo nodo y se dividieron las claves 2 y 8 quedaron a la izquierda y 12 y 23 quedaron a la derecha, el 15 se promociona PERO se copia

+13
+88 → overflow



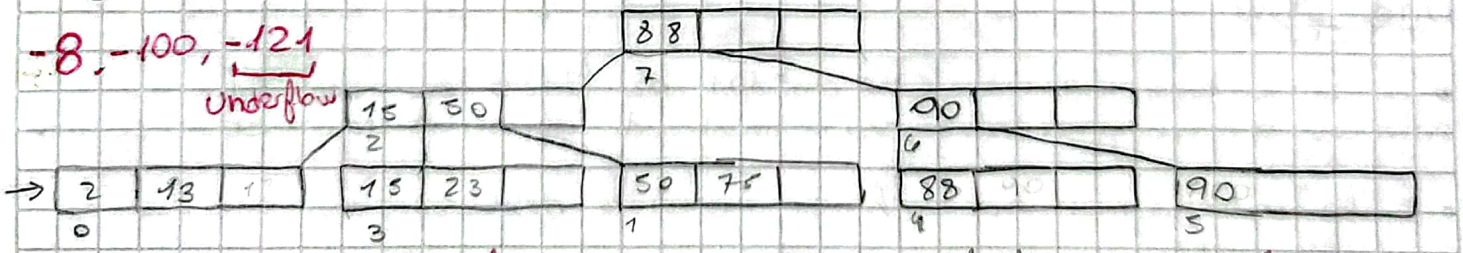
+90 +100
overflow



• el nodo 4 se producio overflow cuando di de alta a el 100, entonces, hago el nuevo nodo 5, el 88 y 90 quedan en el 4 y el 100 y 121 van al 5, se promociona la copia del 100 y se produce overflow en el nodo 2, creo el nuevo nodo 6, el 15 y 20 quedan, el 100 y 88 en el nodo 6, el 88 se promociona y es la nueva raiz.

Bajas en Arboles B+

-8, -100, -121
underflow



• el -121 genera underflow en el nodo 5 y ~~se propagan en underflow~~, balanceo cargas con su unico hermano que es el 4, el 100 no es un dato valido es un señalador, queda el 90 como señalador

Damian Bonzi

Hashing → sirve para acceder lo más rápido posible a la clave

es una técnica para generar la dirección base de una clave dada, yo tengo una clave x y a partir del hashing lo que voy a hacer es asignarle la dirección física

Dispersion de Archivos

- técnica para generar una dirección base única para una clave dada.
- convierte la clave en un número aleatorio, que luego sirve para determinar donde se almacena la clave.

- Utiliza una función de dispersión para mapear cada clave con una dirección física de almacenamiento
- Utilizada cuando se requiere acceso rápido por clave.

tipos de dispersión

- direccionamiento o dispersión estática: espacio fijo de direcciones
- direccionamiento o dispersión dinámica: espacio necesario, en función lo que necesite

Parametros a considerar

- cap de almacenamiento de cada dirección
- Densidad de empaquetamiento

- función de hash
- Metodo de tratamiento de desbordes

***Funcion de dispersión**: caja negra que a partir de una clave genera la dirección física de almacenamiento.

*** Colisión**: situación en la que mi clave o registro a partir de la función de dispersión va a una dirección que ya tiene almacenado algo

*** Desborde**: situación en la cual una clave carece de lugar en la dirección asignada por la función de dispersión.

*** Densidad de empaquetamiento**: relación entre el espacio disponible para el archivo de datos y la cantidad de registros que integran el mismo.

$$\boxed{\text{Densidad de empaquetamiento (DE)} = \text{número de registros} / \text{espacio total}}$$

Damian Bonzi

Hashing Extensible

tecnica de resoluciones de hashing extensible

ejemplo:

- funcion de dispersión: Retorna 10 bits
- capacidad para 2 registros por dirección
- se van a dispersar 8 claves en total

CLAVES A DISPERSAR

1- Colapinto (1...1100) 2- Verstappen (1...1000) 3- Russell (1... 1001)
4- Stroll (1..1010) 5- Alonso (1... 1000) 6- Hamilton (1...1011)
7- Sainz (1... 1111) 8- Leclerc (1... 0111).

Estado inicial del Archivo:

Tabla de Dispersion bits de dispersion: 0	
Sufijo	#bloque
(0)	0

Archivo de Datos			
#Bloque	Bits	clave R1	clave R2
0	0		

que el numero de bits de dispersion este en 0, indica que no es necesario ningun bit de la secuencia obtenida por la funcion de dispersión.

- Se agregan colapinto (1...1100) y verstappen (1...1000)

tabla de dispersion bits de dispersion: 0	
sufijo	#Bloque
(0)	0

Archivo de datos			
#Bloque	Bits	clave R1	clave R2
0	0	colapinto (1...1100)	verstappen (1...1000)

Ambos se Agregan sin inconvenientes en el Bloque 0

- Se agrega Russell (1... 1001) → produce desborde

Tabla de dispersión bits de dispersion: 1	
sufijo	#Bloque
(0)	1
(1)	0

Archivo de datos			
#Bloque	Bits	clave R1	clave R2
0	1	Russell (1...1001)	
1	1	colapinto (1...1100)	verstappen (1...1000)

Se produce desborde en el bloque 0. Aumentamos en 1 los bits de dispersion locales del bloque 0 y creamos un nuevo bloque (bloque 1) con la misma cantidad de bits locales. Luego comparamos los bits locales del bloque con los bits de dispersion globales de la tabla: como ahora el bloque tiene mas bits locales que globales aumenta en uno los bits globales y duplicamos la cant. de direcciones. Por ultimo se distribuyen las claves entre el bloque original y el nuevo segun el bit menos significativo correspondiente.

Démian Bonzi

- Se agrega Stroll (1...1010) → produce desborde.

Tabla de dispersión	
bits de dispersión 2	
Sufijo	#Bloque
(00)	2
(01)	0
(10)	1
(11)	0

Archivo de datos			
#Bloque	Bits	clave R1	clave R2
0	1	Russell (1...1001)	
1	2	Stroll (1...1010)	
2	2	Colahinto (1...1100)	Verstappen (1...1000)

Se produce desborde en el Bloque 1. Aumentamos en uno los bits de dispersion locales del bloque 1 y creamos el bloque 2 con la misma cantidad de bits locales. Como los bits locales superan los bits globales de la tabla, se aumenta en uno los bits de dispersion de la tabla y se duplican la cantidad de direcciones. Por ultimo las claves se redistribuyen entre el bloque original y el nuevo segun los dos bits menos significativos. La direccion donde ocurrio el desborde ahora apunta a el nuevo bloque.

- Se agrega Alonso (1...1000) → produce desborde

tabla de dispersión	
bits de dispersion: 3	
sufijo	#Bloque
(000)	3
(001)	0
(010)	1
(011)	0
(100)	2
(101)	0
(110)	1
(111)	0

Archivo de datos			
#Bloque	Bits	clave R1	clave R2
0	1	Russell (1...1001)	
1	2	Stroll (1...1010)	
2	3	Colahinto (1...1100)	
3	3	Alonso (1...1000)	Verstappen (1...1000)

- Se agrega Hamilton (1...1011) → entra normal

tabla de dispersión	
bits de dispersion: 3	
sufijo	#Bloque
(000)	3
(001)	0
(010)	1
(011)	0
(100)	2
(101)	0
(110)	1
(111)	0

Archivo de datos			
#Bloque	Bits	clave R1	clave R2
0	1	Russell (1...1001)	Hamilton (1...1011)
1	2	Stroll (1...1010)	
2	3	Colahinto (1...1100)	
3	3	Alonso (1...1000)	Verstappen (1...1000)

Damian Banzi

- Se agrega Sainz (1... 1111) → produce desborde

Tabla de dispersión bits de dispersión: 3	
Sufijo	#Bloque
(000)	3
(001)	0
(010)	1
(011)	4
(100)	2
(101)	0
(110)	1
(111)	4

Archivo de Datos			
#Bloque	Bits	Clave R1	Clave R2
0	2	Rossell (1...1001)	
1	2	Stroll (1...1010)	
2	3	Colahinto (1...1100)	
3	3	Alonso (1...1000)	Verstappen (1...1000)
4	2	Hamilton (1...1011)	Sainz (1...1111)

- Se agrega Leclerc (1... 0111) → produce desborde

Tabla de dispersión bits de dispersión: 3	
Sufijo	#Bloque
(000)	3
(001)	0
(010)	1
(011)	4
(100)	2
(101)	0
(110)	1
(111)	5

Archivo de datos			
#Bloque	Bits	Clave R1	Clave R2
0	2	Rossell (1...1001)	
1	2	Stroll (1...1010)	
2	3	Colahinto (1...1100)	
3	3	Alonso (1...1000)	Verstappen (1...1000)
4	3	Hamilton (1...1011)	
5	3	Leclerc (1...0111)	Sainz (1...1111)

Demian Bongi

- doy de baja a Verstappen (1...1000) → Se borra normal

Tabla de dispersion bits de dispersion: 3	
sufijo	#Bloque
(000)	~~3~~ 2
(001)	0
(010)	1
(011)	4
(100)	2
(101)	0
(110)	1
(111)	5

Archivo de Datos			
#Bloque	Bits	clave R1	clave R2
0	2	Russell (1...1001)	
1	2	Stroll (1...1010)	
2	~~3~~ 2	Colapinto (1...1100)	
3	~~3~~	~~Alonso (1...1000)~~	~~Verstappen (1...1000)~~
4	3	Hamilton (1...1011)	
5	3	Leclerc (1...0111)	Sainz (1...1111)

- doy de baja a Alonso (1...1000)

Al dar de baja a Alonso, el bloque 3 queda vacío. este bloque tenía nivel de dispersion local igual a global (3). se identifica su bloque hermano: aquel que comparte el mismo sufijo, salvo en el bit mas significativo utilizado para el direccionamiento, en este caso el bloque 2, sufijo (100). Como el bloque hermano tambien tiene un nivel de dispersión local de 3 y contiene claves validas, se libera el bloque 3, y su rango es absorbido por el bloque hermano. Luego se reduce en uno el nivel de dispersión local del bloque resultante y se actualizan las entradas de la tabla que apuntaban al bloque eliminado para que redirigan al bloque que permanece

- doy de baja a Stroll (1...1010)

Tabla de dispersión bits de dispersión	
sufijo	#Bloque
(000)	2
(001)	0
(010)	~~1~~ 2
(011)	4
(100)	2
(101)	0
(110)	~~1~~ 2
(111)	5

Archivo de Datos			
#Bloque	Bits	clave R1	clave R2
0	2	Russell (1...1001)	
1	~~2~~	~~Stroll (1...1010)~~	
2	~~2~~ 1	Colapinto (1...1100)	
4	3	Hamilton (1...1011)	
5	3	Leclerc (1...0111)	Sainz (1...1111)

al eliminar Stroll, el bloque 1 queda vacío el mismo tenía 2 bits de dispersion local y estaba referenciado por las direcciones de la tabla que terminan en 10, para saber si se puede liberar el bloque 1, se evalua si sus dirhermanas apuntan al mismo bloque son hermanas aquellas que terminan en 00. Las entradas terminadas en 00 apuntan al mismo bloque (2), entonces se puede liberar el bloque 1: se sustituyen las referencias al bloque 1 por referencias al bloque 2, se reduce en uno la dispersión local y el bloque 1 puede eliminarse

Damian Bonzi

- doy de baja a Russell (1...1001)

Tabla de dispersión bits de dispersión: 3	
SUFIJO	# Bloque
(000)	2
(001)	0
(010)	2
(011)	4
(100)	2
(101)	0
(110)	2
(111)	5

Archivo de datos			
# Bloque	Bits	clave R1	clave R2
0	~~2~~	~~Russell (1...1001)~~	
1			
2	1	colophinto (1...1100)	
3			
4	3	hamilton (1...1011)	
5	3	Leclerc (1...0111)	Saing (1...1111)

Al eliminar Russell, el bloque 0 queda vacío. El mismo tenía 2 bits de dispersión local y estaba referenciado por las direcciones de la tabla que terminan en 01. Para saber si se puede liberar el bloque 0, se evalua si sus direcciones hermanas apuntan todas al mismo bloque. Se consideran hermanas aquellas que terminan en 11, ya que si se reduce la dispersion local comparirían el mismo sufijo (1). Las entradas terminadas en 11 no apuntan al mismo bloque. (011 apunta al 4 y 111 apunta al 5), por ende no se puede liberar el bloque 0. Se escribe el bloque 0 vacío

- Estado final

Tabla de dispersión bits de dispersión: 3	
SUFIJO	# Bloque
(000)	2
(001)	0
(010)	2
(011)	4
(100)	2
(101)	0
(110)	2
(111)	5

Archivo de Datos			
# Bloque	Bits	clave R1	clave R2
0	2		
2	1	colophinto (1...1100)	
4	3	Hamilton (1...1011)	
5	3	Leclerc (1...0111)	Saing (1...1111)

* Los bloques 1 y 3 quedaron liberados cuando se eliminaron sus respectivas claves, el bloque 0 quedo vacío pero no se pudo liberar.