Soluciones Ejercicios Sesión 1

1º -

a) El tamaño del registro será la suma de las longitudes de todos los campos que componen el registro:

$L_{registro}$= 30+9+9+40+9+8+1+4+4+1=115 bytes
N registros/ Bloque = $\lfloor 512 / 115 \rfloor = \lfloor 4.452 \rfloor$ = 4 registros / bloque
N bloques = $\lceil 30000 / 4 \rceil$ = 7500 bloques

- Si el campo de búsqueda fuese clave, sólo un registro se tiene que recuperar. El mejor caso sería 1 bloque (está el registro en el primer bloque) y el peor caso sería 7500 bloques (el registro está en el último bloque). De media serían 7500/2 bloques. Si el registro no se encuentra en la tabla, serían 7500 bloques de lectura secuencial.
- Si el campo de búsqueda no es clave, hay valores duplicados, por tanto hay que leer secuencialmente el fichero con lo que son 7500 accesos a bloques

Si el fichero estuviese ordenado según un atributo, la búsqueda binaria de un registro sería de
$\lceil \log_2 (7500) \rceil$ = 13 accesos . Con la búsqueda binaria leemos ya un bloque con datos válidos.

b) Se trata de un índice primario (fichero de datos ordenado) sobre un campo clave con lo que el índice puede ser denso o disperso.

El factor de bloques corresponde con el número de registros / bloque. Para el caso del índice primario con clave de búsqueda según el campo NSS, la longitud del registro índice es:

$L_{registro} = L_{NSS} + P_R = 9 + 7 = 16$ bytes
N registros/ Bloque = $\lfloor 512 / 16 \rfloor$ = 32 registros / bloque

- Si el índice primario fuese denso, habría 30000 entradas, luego el número de bloques del registro índice sería:

N bloques denso = $\lceil 30000 / 32 \rceil$ = 938 bloques
Haciendo una búsqueda binaria en el índice supondría (está ordenado el índice): = $\lceil \log_2 (938) \rceil$ = 10 accesos

- Si es un índice multinivel sobre el denso, los niveles superiores al 1 apuntan a bloques de índices, luego el factor de bloque para los índices de nivel 2 en adelante serían:

$L_{registro} = L_{NSS} + P_B = 9 + 6 = 15$ bytes
N registros/ Bloque = $\lfloor 512 / 15 \rfloor$ = 34 registros / bloque

Entonces los bloques de promedio de cada nivel serían:

Nivel interno: $\lceil 30000 / 32 \rceil$ = 938 bloques para apuntar a los 30000 registros de datos.
Nivel intermedio: $\lceil 938 / 34 \rceil$ = 28 bloques
Nivel externo: $\lceil 28 / 34 \rceil$ = 1 bloque

Número total de bloques del índice multinivel será: 938 + 28 + 1 = 967 bloques
Para el índice multinivel:
N accesos = Número de niveles +1 = 3 +1 = 4 accesos

- Si el índice primario fuese disperso, habría 7500 entradas (apuntamos al primer registro de cada bloque), luego el número de bloques del registro índice sería:

N bloques = $\lceil 7500 / 32 \rceil = 235$ bloques
Haciendo una búsqueda binaria en el índice supondría: = $\lceil \log_2 (235) \rceil = 8$ accesos

- Si es un índice multinivel sobre el disperso, los niveles superiores al 1 apuntan a bloques de índices, luego el factor de bloque para los índices de nivel 2 en adelante serían:

$L_{registro} = L_{NSS} + P_B = 9 + 6 = 15$ bytes
N registros/ Bloque = $\lfloor 512 / 15 \rfloor = 34$ registros / bloque

Entonces los bloques de promedio de cada nivel serían:

Nivel interno: $\lceil 7500 / 32 \rceil = 235$ bloques para apuntar a los 7500 bloques de datos.
Nivel intermedio: $\lceil 235 / 34 \rceil = 7$ bloques
Nivel externo: $\lceil 7 / 34 \rceil = 1$ bloque

Luego con 3 niveles sería el máximo nivel.

- Número total de bloques del índice multinivel será: 235 + 7 + 1 = 243 bloques

- El número de accesos para recuperar un alumno con un NSS dado sería: se debe de recuperar un registro ya que el campo es clave.

Si se efectúa una búsqueda binaria sobre el índice primario disperso de un nivel:
N accesos = $\lceil \log_2 (235) \rceil + 1 = 8 + 1 = 9$ para acceder al dato darían 9 accesos a bloques para índice disperso.
Para el caso de índice denso:
N accesos = $\lceil \log_2 (938) \rceil + 1 = 10 + 1 = 11$ bloques
Para el índice multinivel:
N accesos = Número de niveles +1 = 3 +1 = 4 accesos

c) En este caso, el fichero no se encuentra ordenado según el campo clave NSS, por lo que el número de entradas serán de 30000. Índice secundario sobre campo clave y denso.

La longitud del registro será: 9 + 7 = 16 bytes
Factor de bloques = $\lfloor 512 / 16 \rfloor = 32$ registros / bloque
N bloques registro índice secundario = $\lceil 30000 / 32 \rceil = 938$ bloques

Se puede ver que coincide con el índice primario denso al ser un campo clave.

Para crear el índice multinivel, construimos índices dispersos sobre el índice anterior de la forma más eficiente, realizaríamos.

Nivel interno: $\lceil 30000 / 32 \rceil = 938$ bloques para apuntar a los 30000 registros de datos.
Nivel intermedio: $\lceil 938 / 34 \rceil = 28$ bloques
Nivel externo: $\lceil 28 / 34 \rceil = 1$ bloque

- Número total de bloques del índice multinivel será: 938 + 28 + 1 = 967 bloques

- El número de accesos para recuperar un NSS dado sería: hay que recuperar un registro ya que el campo es clave.

Si se efectúa una búsqueda binaria sobre el índice secundario de un nivel:

N accesos = $\lceil \log_2(938) \rceil + 1 = 10 + 1$ para acceder al dato darían 11 accesos a bloques.
Para el índice multinivel:
N accesos = t +1 =3+1 = 4 accesos

d) El índice es secundario sobre un campo clave, luego tiene que ser denso. En este caso, hay que determinar los valores n de los nodos intermedios y padre y n_h de los nodos hoja antes de proceder, teniendo como restricción que cada nodo debe de caber en un bloque de disco.

Para el nodo intermedio, n indica el número de punteros a bloque que caben en un nodo, luego se debe de cumplir que:

$n*P_{bloque} + (n-1)*L_{NSS} \leq B$
$n*6 + (n-1)*9 \leq 512$
$15*n \leq 521 \Rightarrow n = 34$ punteros, ya que es el mayor valor entero que cumple la condición.

Para el nodo hoja, n_h indica el número de punteros a registro ó claves de búsqueda que caben en un nodo, luego se debe de cumplir que:

$n_h*(P_{registro} + L_{NSS}) + P_{bloque} \leq B$
$n_h*(9 + 7) + 6 \leq 512$
$n_h*16 \leq 506 \Rightarrow n_h = 31$ punteros a registros ó claves de búsqueda.

- Si los nodos se encuentran al 69% de capacidad y el fichero no se encuentra ordenado:

Cada nodo intermedio tendrá 34*0.69 = 23 punteros a nodos y por tanto 22 valores de clave búsqueda
Cada nodo hoja en promedio tendrá: 31*0.69 = 21 punteros a registros de datos.

Como hay 30000 registros de datos, habrá 30000 punteros a registros en los nodos hoja (índice denso):
Como es un índice multinivel especial, para determinar el número de bloques promedio por nivel, se parte desde los nodos hoja hacia el nodo raíz. Luego el número de bloques en cada nivel será:

Nivel hola: $\lceil 30000 / 21 \rceil = 1429$ bloques para apuntar a los 30000 registros de datos.
Nivel intermedio 1: $\lceil 1429 / 23 \rceil = 63$ bloques
Nivel intermedio 2: $\lceil 63 / 23 \rceil = 3$ bloques
Nivel raíz : $\lceil 3 / 23 \rceil = 1$ bloque

El número de bloques totales será: 1429 + 63 + 3 + 1 = 1496 bloques
Tiene altura 4 el árbol, por lo que recuperar un alumno con un NSS dado será 4 + 1 = 5 bloques (hay que recuperar un registro porque el campo es clave).

Para calcular la capacidad máxima del árbol al 69%, el número de registro máximo que podría almacenar en promedio cada nivel sería:

Raiz	1 nodo	22 entradas	23 punteros a nodos
Intermedio 1	23 nodos	23*22=506 entradas	23*23=529 punteros
Intermedio 2	529 nodos	11638 entradas	12167 punteros
Hoja	12167 nodos	255.507 punteros registros	

e) Para el caso de un árbol B, la diferencia se encuentra en los nodos intermedio, siendo los nodos hoja iguales que en el caso anterior. Es un índice secundario sobre campo clave, luego es denso.

Siendo n el número de punteros a nodos, se tiene que cumplir:

$n*P_{bloque} + (n-1)*(P_{registro} + L_{NSS}) \leq B$
$n*6 + (n-1)*(7 + 9) \leq 512$
$22*n \leq 528 \Rightarrow n = 24$ punteros a nodos ó lo que es lo mismo 23 valores de la clave de búsqueda

Para el nodo hoja, se tiene lo mismo que en el caso anterior que serían 31 punteros a registro o valores de búsqueda, que al 69% serían 21 punteros a registro de datos.

Si la capacidad se encuentra al 69%, el número de punteros medio es de 0.69*24 = 17 punteros, lo que equivale a 16 valores de la clave de búsqueda.

Para determinar el número de bloques promedio en cada nivel, al no tener todos los valores en los nodos hojas, no se puede determinar como en el árbol B^+.

Para este caso como los nodos intermedios tienen punteros a registros directamente, se calcula el número de registros que se pueden almacenar con esta capacidad:

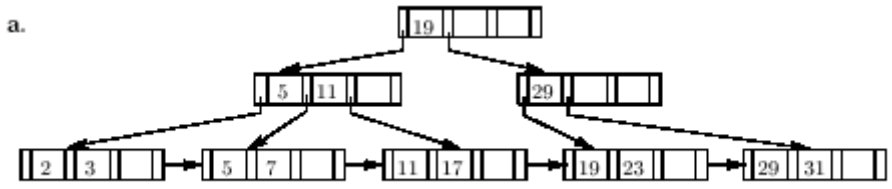
Raiz	1 nodo	16 entradas	17 punteros a nodos
Intermedio 1	17 nodos	17*16=277 entradas	17*17=289 punteros
Intermedio 2	289 nodos	4624 entradas	4913 punteros
Hoja	4913 nodos	103173 punteros registros	

El número de registros totales sería: 16 + 277 + 4624 + 103173 = 107730 registros. Todos los nodos almacenan punteros a registros.

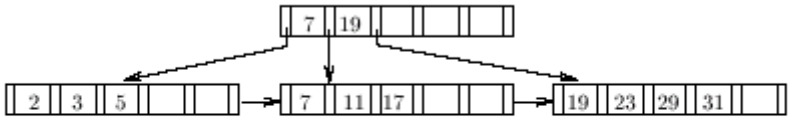
Y se puede ver que el acceso promedio será de 4 + 1 = 5 accesos, ya que hay un mayor número de registros en los nodos hoja

Si el árbol B^+ tendría todas sus entradas ocupadas, se podrían almacenar: 1218424 registros
Si el árbol B tendría todas sus entradas ocupadas, se podrían almacenar: 331775 registros

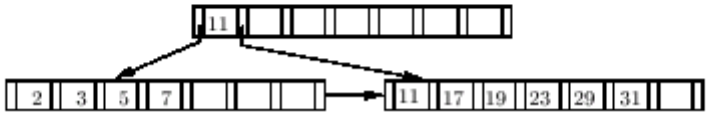
2º - Si n es el número de punteros, un nodo intermedio no puede tener menos de $\lceil n / 2 \rceil$ punteros y un nodo hoja no puede tener menos de $\lceil (n-1) / 2 \rceil$ valores de la clave. Hay que tener en cuenta esa restricción y además que para comenzar el problema se parte de sólo un único nodo y si se desborda, se añade un nodo hermano y se redistribuyen los valores. Entonces se obtiene:



b.

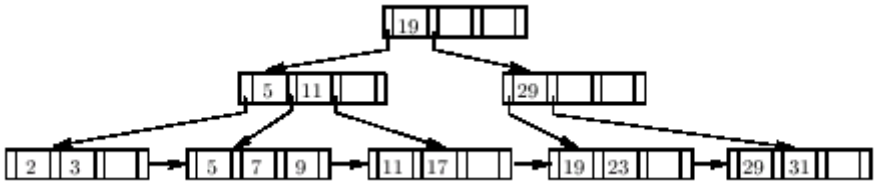


c.

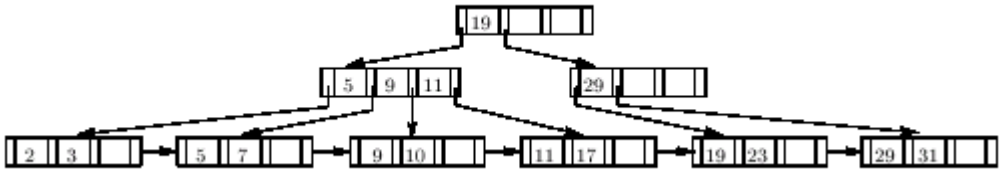


3º - Con estructura 2.a

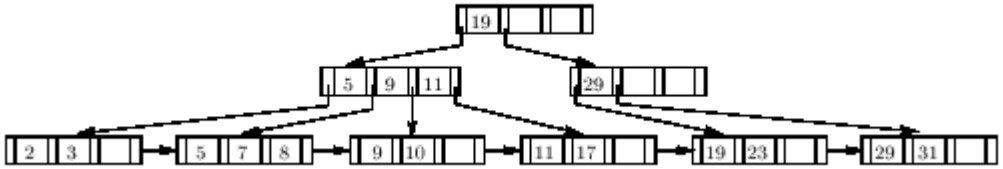
Insertar 9.



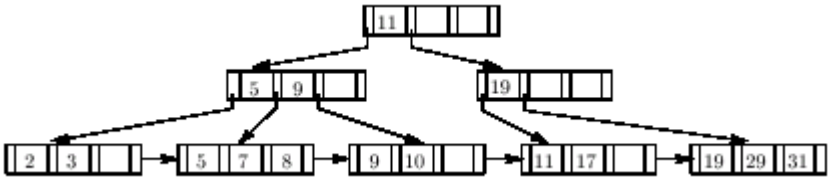
Insertar 10.



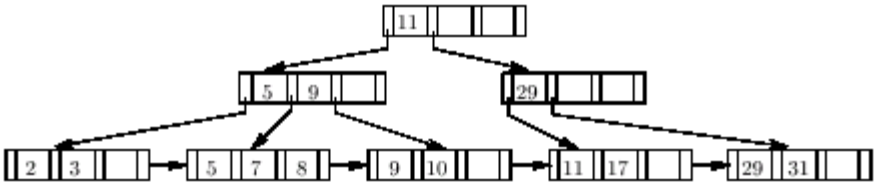
Insertar 8.



Borrar 23.

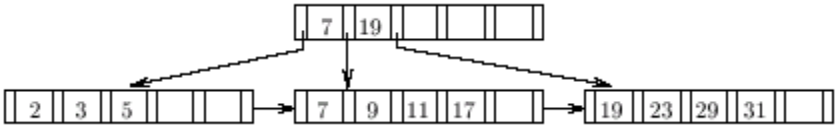


Borrar 19.

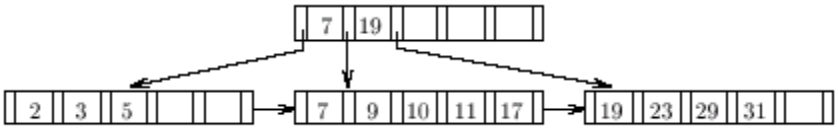


Con estructura 2.b

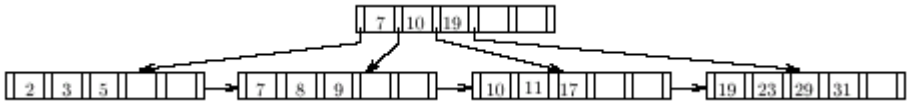
Insertar 9.



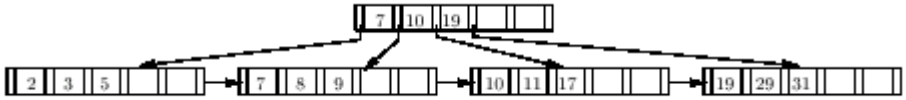
Insertar 10.



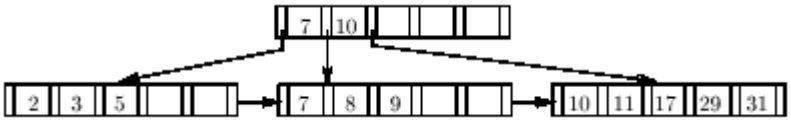
Insertar 8.



Borrar 23.

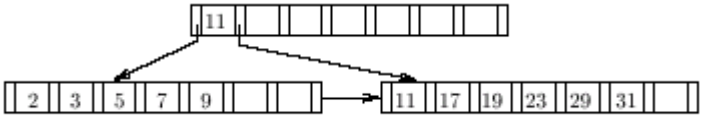


Borrar 19.

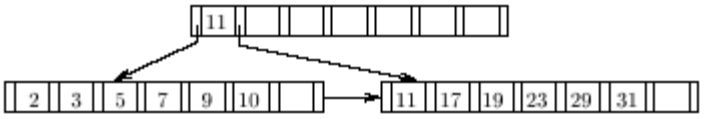


Con estructura 2.c

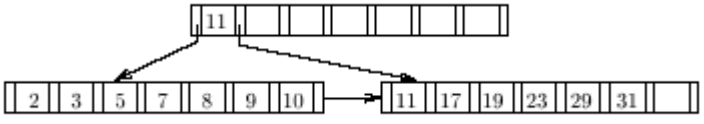
Insertar 9.



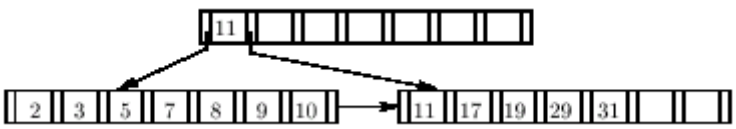
Insertar 10.



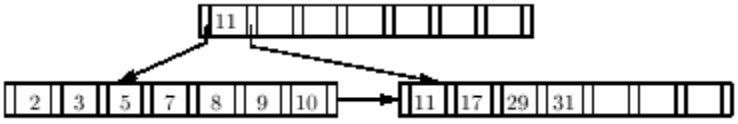
Insertar 8.



Borrar 23.

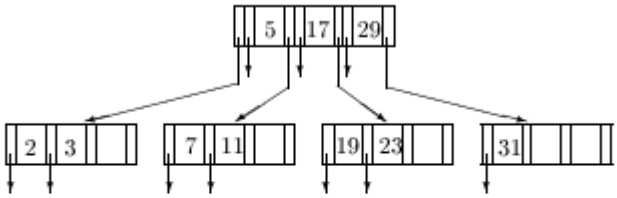


Borrar 19.

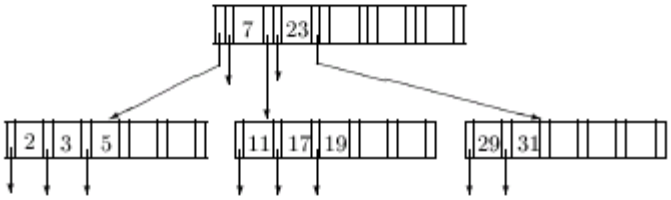


4º- Para un árbol B,

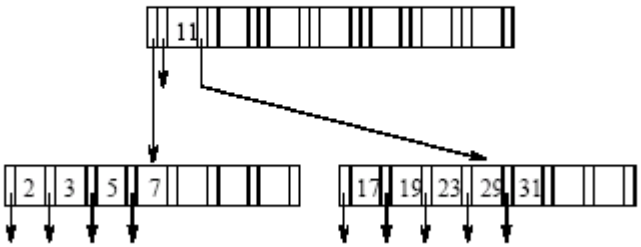
a.



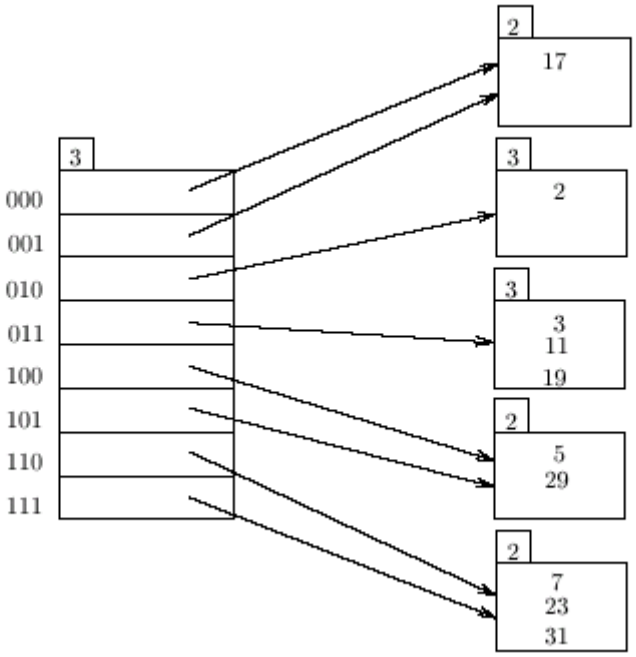
b.



c.

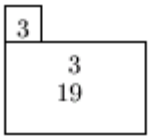


5º- Para una estructura asociativa: Hay que tener en cuenta que cuando se desborda un cajón y hay valores diferentes que tienen diferente función de asociación se tiene que incrementar el número de bits de la tabla de direcciones. Para resolver el problema se tiene que partir de un único cajón (el inicial) y 0 bits de la función de asociación.



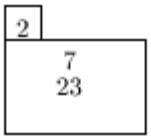
6º - Los cambios de la estructura asociativa anterior son los siguientes para estas operaciones:

a. Borrar 11. Se cambia el tercer cajón a

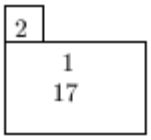


Se podría combinar el tercer y segundo cajón, con lo cuál se podría cambiar el cajón de direcciones de de 8 entradas a 4.

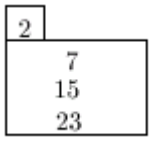
b. Borrar 31. Se cambia el último cajón a



c. Insertar 1. Cambia el cajón 1 a



d. Insertar 15. Cambia el último cajón



6º -

a. El índice de mapa de bits es el siguiente: 6 mapas de bits para cada valor diferente de cada campo.

Nombre-sucursal

Barcelona	1 0 0 0 0 0 0 0 0
Daimiel	0 1 1 0 0 0 0 0 0
Madrid	0 0 0 1 0 0 0 0 0
Pamplona	0 0 0 0 1 1 1 0 0
Reus	0 0 0 0 0 0 0 1 0
Ronda	0 0 0 0 0 0 0 0 1

Niveles de ingresos

L1	0 0 0 0 0 0 0 0 0
L2	0 0 0 0 1 0 0 0 1
L3	0 1 1 1 0 0 1 1 0
L4	1 0 0 0 0 1 0 0 0

b. La consulta trataría los siguientes mapas de bits

Daimiel	0 1 1 0 0 0 0 0 0
L3	0 1 1 1 0 0 1 1 0
L4	1 0 0 0 0 1 0 0 0
L3∪L4	1 1 1 1 0 1 1 1 0
Daimiel	0 1 1 0 0 0 0 0 0
Daimiel∩(L3∪L4)	0 1 1 0 0 0 0 0 0

Por lo que los registros seleccionados son el 1 y 2.

c. 1 mapa de bits ocupa n_R bits, luego en este caso son 9 bits, lo que implica que son $\lceil 9 / 8 \rceil =$ 2 bytes. Si un bloque ocupa 512 bytes, cada mapa de bits ocupa $\lceil 2 / 512 \rceil = 1$ bloque. Como hay 6 + 4 mapas de bits, el índice ocupa 10*1 bloques = 10 bloques.

d. Para la consulta anterior, habría que leer 3 mapas de bits (L3, L4, Daimiel) que son 3 bloques, más recuperar 2 registros que el peor caso sería leer 2 bloques de datos, luego en total serían 3+2=5 bloques.

8º-

a) ¿Cuánto espacio se requiere para almacenar el archivo y sus índices?

Lregistro=20+40+2+16+32=110 bytes
N registros/ Bloque = $\lfloor 512 / 110 \rfloor = \lfloor 4.65 \rfloor = 4$ registros / bloque
Si se llenan los bloques al 65% para el archivo de datos: 4*0.65=2.6 ⇒ 3 registros/bloque
Nº total bloques = $\lceil 100000 / 3 \rceil = 33334$ bloques del archivo de datos

Primer índice B^+ sobre el atributo carnet, que consta de 20 bytes.
Es un índice secundario sobre campo clave, luego es denso. V(carnet,r)=100000

- Nodo raíz, intermedio: hay n valores de la clave y n+1 punteros a bloque

$(n-1)*L_{carnet} + n*P_{bloque} \leq B$
$(n-1)*20 + n*6 \leq 512$
$26*n \leq 532 \Rightarrow n = 20$ punteros a bloque.

- Nodo hoja: hay n_h valores de la clave, n_h punteros a registro y 1 puntero a bloque

$n_h*(P_{registro} + L_{carnet}) + P_{bloque} \leq B$
$n_h*(20 + 7) + 6 \leq 512$
$n_h*27 \leq 506 \Rightarrow n_h = 18$ punteros a registros ó claves de búsqueda.

Como los nodos se encuentran al 69%:

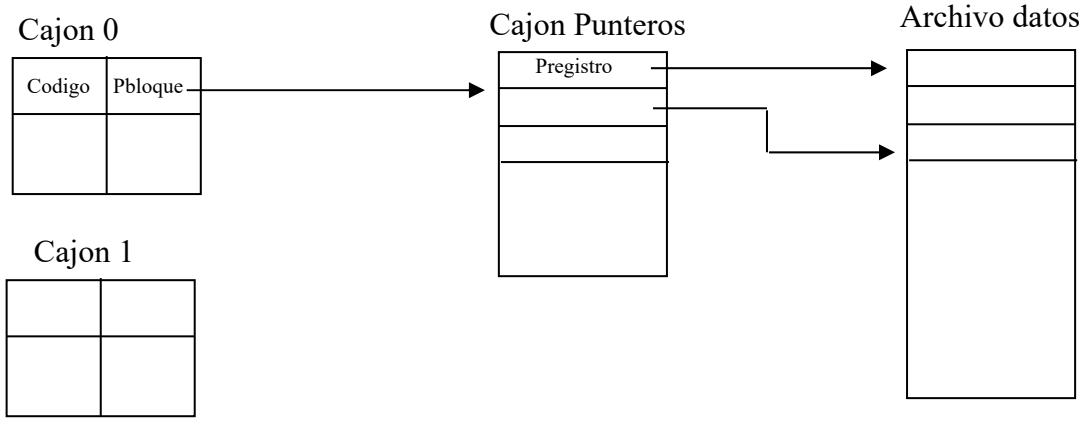
- Nº de valores clave en nodos intermedios: 0.69*20=14 $\Rightarrow$ 14 punteros a bloque y 13 valores.
- Nº de valores clave en nodos hoja: 0.69*18=12.42 $\Rightarrow$ 12 valores y 12 punteros a registro

El Nº de bloques totales para este índice en promedio es:

Nivel hoja: $\lceil 100000 / 12 \rceil = 8334$ bloques para apuntar a los 30000 bloques de datos.
Nivel intermedio 1: $\lceil 8334 / 14 \rceil = 596$ bloques
Nivel intermedio 2: $\lceil 596 / 14 \rceil = 43$ bloques
Nivel intermedio 3: $\lceil 43 / 14 \rceil = 4$ bloques
Nivel raíz : $\lceil 4 / 14 \rceil = 1$ bloque

Número de bloques: 8334+596+43+4+1=8978 bloques

Segundo índice asociativo hash sobre el atributo código_carrera (2 bytes)
Es un índice secundario sobre campo no clave, lo que implica denso y tiene cajones de punteros.
La función es codigo_carrera mod 8, por lo que la función de asociación sólo puede direccionar 8 cajones. Como hay 16 carreras diferentes V(código_carrera,r)=16, y hay 8 cajones, habrá 2 códigos de carrera en cada cajón de media. El esquema será el siguiente:

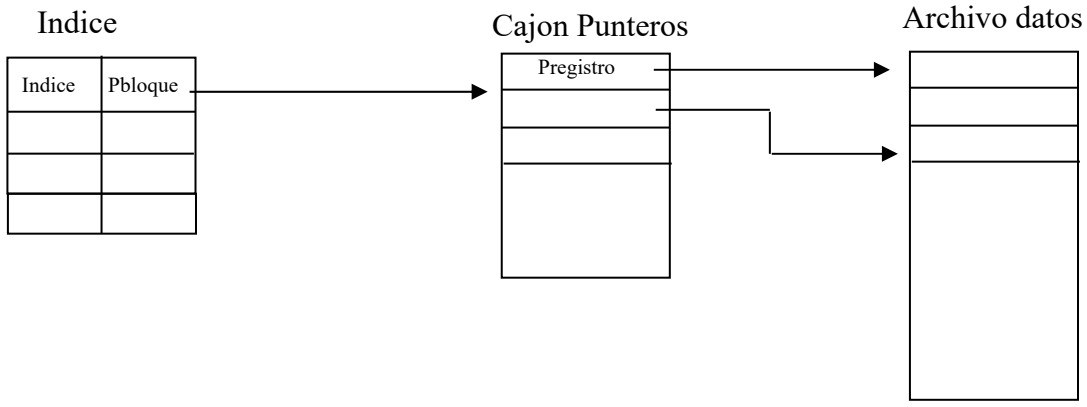


La longitud del registro índice será: Lregistro=6+2=8 bytes
Factor Bloque = $\lfloor 512 / 8 \rfloor$ = 64 registros / bloque
Al 69% son 44 registros cada cajón. Como hay 2 registros por cajón, entonces cada cajón ocupará 1 bloque.

Como las carreras se encuentran distribuidas uniformemente entre los alumnos, habrá 100000/16=6250 alumnos por cada carrera. Cada cajón de punteros deberá de contener 6250 punteros a registro
El factor de bloque del cajón de punteros será: $\lfloor 512 / 7 \rfloor = 73$ punteros, y como los cajones están al 69% equivale a almacenar 50 punteros.
El número de bloques necesarios para almacenar cada cajón de punteros será: $\lceil 6250 / 50 \rceil = 125$ bloques.

El tamaño de este índice será: 8 cajones * 1 bloque + 16 cajones de punteros *125 bloques = 8*1+125*16= 2008 bloques

Tercer índice secundario sobre atributo indice_académico (32 bytes)
Este índice es secundario sobre un campo no clave, lo que implica que es denso y necesita cajones de punteros.
Este índice se debe de construir sobre el índice académico que tiene 8 valores diferentes (V(índice_acdemico,r)=8), luego el esquema del índice será:



La longitud de registro del índice será: Lregistro=Lindice+Pbloque=32+6=38 bytes
Factor Bloque de índice = $\lfloor 512 / 38 \rfloor$ = 13 registros / bloque ,
que al 69% tendrían 9 registros. Luego con 1 bloque sería suficiente para guardar el índice.

Si el índice académico está distribuido uniformemente, cada cajón de punteros debería de tener 100000/8=12500 registros
El factor de bloque del cajón de punteros sería: $\lfloor 512 / 7 \rfloor$ = 73 registros ,
que estando al 69% serían 50 registros. Con lo que cada cajón de punteros a registro tendría:

$\lceil 12500 / 50 \rceil$ = 250 bloques.

La estructura del índice ocuparía: 1+ 250*8 = 2001 bloques.

b) Si se desea insertar un registro nuevo de un estudiante. ¿Cuál es el coste de realizar dicha operación?

Al insertar un nuevo alumno hay que insertar el registro en el archivo de datos y además hay que modificar cada una de las estructuras índice:

1º - Indice primario: habría que recorrer el árbol B^+ hasta localizar la posición donde debería de estar, ordenar el nodo hoja, localizar el bloque de datos donde tiene que ir el nuevo registro (leerlo, modificarlo y escribirlo) y además insertar un puntero a registro en el nodo hoja. Esta operación lleva: 5 bloques para localizar + 1 para insertar el puntero a registro + 1 leer el bloque de datos donde se va a insertar + 1 para escribir el bloque de datos con el registro = 8 bloques. Tener en cuenta que los bloques de datos y de los índices no están llenos y por lo tanto no hace falta crear nuevos bloques.

2º - Insertar puntero con valor de registro para código de carrera en el cajón de punteros correcto: 1 bloque para acceder al cajón + 1 bloque leer el primer bloque de punteros + 1 bloque de escribir el bloque actualizado = 3 bloques.

3º - Recorrer el índice de índice académico para localizar en qué cajón hay que insertar el puntero a registro nuevo= 3 bloques.

Total = 8+3+3=14 bloques.

c) Si se desea buscar un estudiante por su carnet, ¿Cuánto cuesta esta operación?
Hay que recuperar sólo un registro ya que el campo carnet es clave.

Nivel del árbol +1 = 5+1 = 6

d) ¿Cuál es el coste de listar a todos los estudiantes que cursan una carrera dada?. Asumir el peor caso.

Si se quiere recuperar una carrera dada, el número de registros esperados será 100000/V(código_carrera,r)=100000/16=6250 alumos
Hay que procesar la función hash, leer el cajón adecuado, leer todos los punteros a registros y leer del archivo de datos: 1 bloque + 125 bloques + 6250 bloques (peor caso, todos estén en un bloque diferente) = 6376

e) ¿Cuál es el coste de listar a todos los estudiantes que tienen un índice académico mayor que 3.0?. Asumir el peor caso.

Recuperar los alumnos de un índice académico supone obtener 100000/V(índice_academico,r)=100000/8=12500 alumnos
Recorrer el índice encontrando los valores mayores que 3, leer todos los punteros a registros y leer el archivo de datos: 1 bloque + 250*4 (4 valores que cumplen la condición) + 12500*4 (leer cada bloque donde se encuentra el dato) = 51001

f) ¿Cómo se podrían mejorar los procesos de lectura anteriores?

Los procesos de lectura anteriores se podrían mejorar leyendo los punteros a registros, obteniendo el número de bloque donde se encuentran los datos y ordenando secuencialmente por el número de bloque a leer, para así realizar menos accesos a disco, debido a que puede haber varios registros que se deban de leer y caigan en el mismo bloque.

9º -

a) Organización asociativa con 1000 cajones.

Darse cuenta de que en este caso no es un índice + archivo de datos, es sólo un archivo de datos que se organiza de manera asociativa.

N registros/ Bloque = $\lfloor 1000 / 200 \rfloor$ = 5 registros / bloque
Si hay 1.000.000 registros, cada cajón contendrá 1.000.000/ 1000 = 1000 registros
Nº bloques por cajón = $\lceil 1000 / 5 \rceil$ = 200 bloques por cajón.
Luego el tamaño del fichero será: 1000*200= 200.000 bloques.
Luego el número de accesos en promedio para localizar un dato en un cajón sería: 200/2 = 100 bloques. Es un campo clave, mejor caso 1 bloque, peor caso 200 bloques, la media 100 bloques.

b) Para un índice primario de un nivel

Es un índice primario sobre campo clave y además denso.
$L_{registro} = L_{NSS} + P_R = 10 + 5 = 15$ bytes
N registros/ Bloque = $\lfloor 1000 / 15 \rfloor = 66$ registros / bloque
N bloques = $\lceil 1000000 / 66 \rceil = 15152$ bloques
Haciendo un búsqueda binaria en el índice supondría: = $\lceil \log_2 (15152) \rceil + 1 = 15$ accesos

c) Para el árbol B^+
Es un índice primario sobre campo clave y además denso.
Para el nodo intermedio, n indica el número de punteros a bloque que caben en un nodo, luego se debe de cumplir que:

$n*P_{bloque} + (n-1)*L_K \leq B$
$n*5 + (n-1)*10 \leq 1000$
$15*n \leq 1010 \Rightarrow n = 67$ punteros, ó 66 registros de la clave de búsqueda, ya que es el mayor valor que cumple la condición.

Para el nodo hoja, en este caso al ser el puntero de registro igual al puntero de bloque, el nodo intermedio es igual al nodo hoja, por lo que en promedio hay 67 punteros y 66 valores de la clave.

Suponiendo que el árbol tendría todos su nodos al 100% de capacidad, el número de bloques que podría almacenar en promedio cada nivel sería:

Nivel hoja: $\lceil 1000000 / 66 \rceil = 15152$ bloques para apuntar a los 30000 registros de datos.
Nivel intermedio 1: $\lceil 15152 / 67 \rceil = 227$ bloques
Nivel intermedio 2: $\lceil 227 / 67 \rceil = 4$ bloques
Nivel raíz : $\lceil 4 / 67 \rceil = 1$ bloque

El número total de bloques del índice será: 15152 + 227 + 4 +1 = 15384 bloques

Por lo que el árbol tendrá una altura de 4 niveles y por lo tanto el número de accesos para localizar un registro será de 5 (4 +1 de recuperar un registro al ser clave).
En este caso se puede ver que todos los nodos de este árbol son iguales, ya que el puntero a registro es igual al puntero a bloque.

10º -

Campo Índice	**Clave Primaria**	**Clave Secundaria**	**No clave**
No existe índice	① Si los registros están ordenados por ese campo: 1 (∃ c. primaria). Si no están ordenados: 0 (no ∃ c. primaria) ② Búsqueda binaria sobre registros (si ∃ c. primaria).	① Resto de claves – 1 = 4 (∃ c. primaria) Resto de claves = 5 (no ∃ c. primaria) ② Búsqueda secuencial (o lineal) sobre registros.	① Resto de campos = 5 ② Búsqueda secuencial (o lineal) sobre registros.
Existe índice	① Si los registros están ordenados por ese campo: 1 (∃ c. primaria). Si no están ordenados: 0 (no ∃ c. primaria) ② Búsqueda binaria o indirecta sobre fichero de	① Resto de claves – 1 = 4 (∃ c. primaria) Resto de claves = 5 (no ∃ c. primaria) ② Búsqueda binaria o indirecta sobre fichero de	① Resto de campos = 5 ② Búsqueda binaria o indirecta sobre fichero de

Existe índice	índices (según tipo y estructura). Luego acceso directo sobre registros ③ Denso ó Disperso. Cualquier estructura	índices (según tipo y estructura). Luego acceso directo sobre registros ③ Denso. Cualquier estructura	índices (según tipo y estructura). Luego acceso directo sobre registros ③ Denso. Cualquier estructura

11º- Las relaciones que tenemos acerca de las longitudes de las filas son:
Para el fichero de Registros: $L_R = \mathbf{A * L_A}$
Para el fichero de Índices $L_I = L_A + P = \mathbf{L_A}$ según el enunciado: $P << L_A$

Por simplificación, supondremos que las funciones **suelo** ($\lfloor x \rfloor$) y **techo** ($\lceil x \rceil$) devuelven un valor igual al cociente.

a) Primero calcularemos el tamaño de los archivos:
f = factor de bloque –- b = tamaño del archivo en bloques

	Clave Primaria	**Clave Secundaria**	**No clave**
Sin Índices		$f_R = B / L_R$ $b_R = N / f_R$ $= \mathbf{N * L_R / B}$	
Con Índices	$f_I = B / L_I$ $= B / L_A$ $b_I = b_R / f_I$ $= \mathbf{b_R * (L_A / B)}$	$f_I = B / L_I$ $= B / L_A$ $b_I = N / f_I$ $= N * L_A / B$ y como $L_A = L_R / A$ $= \mathbf{b_R / A}$	
Comentarios	Punteros a bloques (normalmente).	Punteros a registros.	
Comparación	El gasto de disco adicional al utilizar índices representa una fracción de $\mathbf{L_A / B}$ del tamaño total del archivo de registros. Si la longitud de un atributo $\mathbf{L_A}$ es grande (comparado con la longitud del bloque **B)**, el tamaño del fichero de índices se hace importante con relación al del archivo de registros.	El gasto de disco adicional al utilizar índices **sólo** representa una fracción de **1/A** del tamaño total del archivo de registros. Si el número de atributos es grande (A>>1), el tamaño del fichero de índices se hace despreciable con relación al del archivo de registros.	

b) Ahora calcularemos el coste de buscar un registro:
C = coste de búsqueda en operaciones de I/O –- b = tamaño del archivo en bloques

	Clave Primaria	**Clave Secundaria**	**No clave**
Sin Índices	Búsqueda binaria sobre registros ($b_R = N * L_R / B$): $C_R = \mathbf{\log_2 b_R}$	Búsqueda lineal sobre registros ($b_R = N * L_R / B$): $C_R = \mathbf{½\ b_R}$	

Con Índices	Búsqueda binaria sobre índices ($b_I = b_R * L_A / B$): $C_I = \log_2 (b_R * L_A / B)$ $= \log_2 b_R + \log_2 (L_A / B)$ $= \mathbf{C_R + \log_2 (L_A / B)}$ $C_{R+I} = C_I + 1$ $= \mathbf{C_R + \log_2 (L_A / B) + 1}$	Búsqueda binaria sobre índices ($b_I = b_R / A$): $C_I = \log_2 (b_R / A)$ $= \mathbf{\log_2 b_R - \log_2 A}$ $C_{R+I} = C_I + 1$ $= \mathbf{\log_2 b_R - \log_2 A + 1}$
Comentarios	Ahorro: $C_R - C_{R+I}$ Normalmente $(L_A / B) < 1$, luego: $\log_2 (L_A / B) < 0$	Ahorro: $C_R - C_{R+I}$
Comparación	El ahorro de coste al utilizar índices es: $\mathbf{-\log_2 (L_A / B) - 1}$. Esto representa un ahorro relativamente pequeño si la longitud de un atributo $\mathbf{L_A}$ es una fracción "razonable" del bloque **B**.	El ahorro de coste al utilizar índices es: $\mathbf{½\ b_R - \log_2 b_R + \log_2 A - 1}$. Para valores "habituales" de A y de b_R ($b_R >> 1$), los términos dominantes son $\mathbf{½\ b_R - \log_2 b_R}$, que tiende realmente a $\mathbf{½\ b_R}$, lo que representa un gran ahorro.

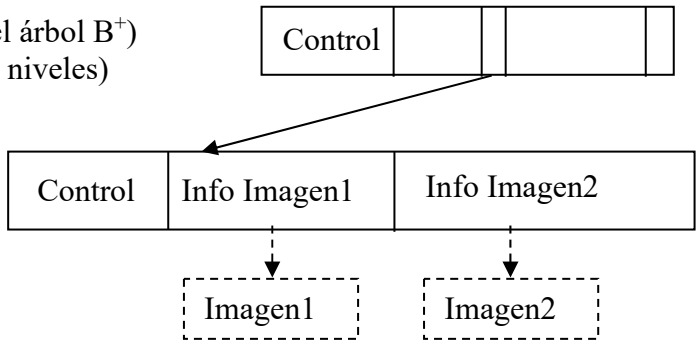
c) Recomendación final:
   a) Espacio en disco: Es cierto que añadir un índice no representa un despilfarro grande de espacio, por lo que se deberían crear también en aquellos atributos sobre los que, aún no siendo clave, se realicen búsquedas sobre ellos (y no sólo en las claves secundarias). Curiosamente el "despilfarro" mayor podría aparecer en la clave primaria, que no siempre es sobre la que se realizan las búsquedas.
   b) Eficiencia en búsquedas: La mayor eficiencia la conseguimos cuando los campos son claves secundarias y campos no clave, por lo que, nuevamente, los índices se deberían crear también en aquellos atributos sobre los que se realicen búsquedas sobre ellos. Otra vez, el ahorro menor (aunque existe) aparece en la clave primaria.

12º -

Se trata de organización de archivo, no hay índice. El problema se puede esquematizar en lo siguiente:

Páginas o bloques de **índices** (nodos del árbol B^+) con su información de control (1 ó más niveles)

Página o bloques de **datos** (información a organizar) con su información de control



(Ficheros de imágenes)

a) Las páginas de datos que contienen la información a organizar tienen una información de control de 196 bytes. La longitud del registro es de $L_R = 128$ bytes que es la información a organizar. El factor de bloques es: $f_R = \lfloor (4096 - 196) / 128 \rfloor = \lfloor 30.46 \rfloor = 30$ registros/bloque. Esta es la cantidad de registros que puede almacear el nodo hoj.

El nodo del árbol B^+ (nodos intermedio y raíz) deberá de tener una información de control de 12 bytes, los valores de la clave de búsqueda y los punteros a bloque:

$12 + n * L_{puntero\_bloque} + (n - 1) * L_{clave} \leq B$
$12 + n * 4 + (n - 1) * 12 \leq 4096$
$16 * n \leq 4.096 \Rightarrow n = 256$ punteros ó 255 valores de la clave de búsqueda

Si los nodos del árbol están al 80% de ocupación:
256 * 0,8 = 205 punteros, y por tanto 204 valores de la clave.
Si la página ó nodos de datos están al 80% de ocupación:
30 * 0,8 = 24 registros.

Hay 100.000 imágenes que hay que organizar, luego primero se calcula el número de nodos hoja para esas 100.000 imágenes, y luego los nodos intermedio más raíz:

Nivel hoja: $\lceil 100000 / 24 \rceil = 4167$ bloques
Nivel intermedio 1: $\lceil 4167 / 205 \rceil = 21$ bloques
Nivel intermedio 2: $\lceil 21 / 205 \rceil = 1$ bloques

Número medio de nodos para 100.000 imágenes será: 1 +21 + 4167 = 4188 bloques

La capacidad máxima del árbol sería para tres niveles al 80%:

Nivel	Nodos	Punteros	Imágenes organizadas
Nivel 1 (Raíz)	1	205	
Nivel 2	205	205 * 205 = 42.025	
Nivel hoja	42.025		42.025 * 24 = 1.008.600

Luego entonces hay 2 niveles de bloques índices del árbol B^+ más 1 nivel de las páginas de datos. En total 3 niveles.
El número máximo de nodos será de: 1 + 205 + 42.025 = 42.231 nodos

Cada nivel indexa como máximo al 80%:
24 para nivel 1
4.920 para nivel 2
1.008.600 para las páginas de datos
Número máximo de páginas de datos: 42.025
Número máximo de registros que se pueden almacenar: 1.008.600
El espacio que ocupa cada nivel como máximo será:
1 nodo para nivel 1 * 4Kbytes = 4 Kbytes.
205 nodos para nivel 2 * 4 Kbytes = 820 Kbytes
42.025 nodos de datos * 4 Kbytes = 168.100 Kbytes
Luego en total son 168.924 Kbytes $\Rightarrow$ 164.1 MBytes. Como máximo

b) En 824 KB de memoria caben la raíz y el nivel 1 por completo del árbol B^+ (4 KB + 820 KB), luego para localizar una imagen se necesitan:
2 accesos a **memoria** (raíz y nivel 1), que no suponen coste efectivo alguno (no es disco)
+ 1 acceso a **disco** para el nodo de datos
+ 1 acceso a **disco** para la página de la imagen

= 2 accesos a **disco**.

Recuperar 100.000 imágenes por la clave, estando las páginas de datos al 80% supondría que es necesario "navegar" o recorrer en secuencia únicamente todos nodos de datos. El número promedio de éstos será de $\lceil 100.000 / 24 \rceil = 4.167$ páginas de datos (o accesos), más recuperar 100.000 imágenes en fichero. En total supondría 104.167 accesos al disco.

13º -

(a) **Mínimo: 3** (Un árbol de altura 2 sólo indexa un máximo de 90 registros)
Máximo: 4 (Un árbol de altura 5 necesita un mínimo de 1.250 registros)

(b) **Mínimo: 17** (Un orden de 17 permite indexar hasta 18 * 17 = 306 registros, mientras que un orden de 16 sólo permitiría indexar hasta 17 * 16 = 272 registros)
Máximo: 300 (Un orden de 300 admite un mínimo de 300 registros con 2 nodos hoja en el segundo nivel, cada uno de los cuales debe indexar al menos 150 registros. Un orden de 301 con un mínimo de dos nodos hoja en el Segundo nivel requeriría que cada uno indexara al menos 151 registros)

14º -

(a) **Número de I/Os: 1** (Cada rango posee 3.000 registros, que caben en 1 bloque cada uno)

(b) Construyamos una tabla simplemente con los datos del enunciado:

Rango	Prob	# Registros	**# Bloques**
[1]	1/15	1.000	**1**
[2]	2/15	2.000	**1**
[3]	3/15	3.000	**1**
[4]	4/15	4.000	**2**
[5]	5/15	5.000	**2**

Usando la tabla anterior, se puede ver que:

Número de I/Os: $\dfrac{\mathbf{1}\times(1000+2000+3000)+\mathbf{2}\times(4000+5000)}{1000+2000+3000+4000+5000} = \mathbf{1,6}$

(c) **Número de I/Os: 5** (El rango [1] contiene 15.000 registros, que caben en 5 bloques, mientras que los demás contienen 0 registros y usan 0 bloques)

(d) **Número de I/Os: 2** (Un árbol B+ con 15.000 registros y 3.300 punteros tiene 2 niveles)

(e) La estructura en retícula es mejor, siempre que los datos no estén *extremadamente* sesgados, como ocurren en el caso (c).
Un índice en retícula proporciona acceso directo a los bloques que contienen los datos en un rango dado. Un árbol B+ requiere atravesar primero desde la raíz a la primera hoja.

(f) Retícula.
Las tablas con estructura asociativa (*hash*) no preservan ninguna clase de orden, por lo que no son buenas en absoluto para consultas de rangos.

15º -

Se tiene un disco que posee las siguientes características conocidas:

- Cada pista está formada por 100 sectores

- Cada sector contiene un espacio no útil del 50%, y los datos útiles ocupan 1.024 bytes
- Un bloque está formado por 2 sectores
- El tiempo de transferencia de 1 bloque es de 0.12 mseg, y el tiempo (medio) de búsqueda es de 10 mseg

1. Calcular la velocidad de rotación del disco **[2 p]**

   $\mathbf{t_{transf}}$ (1 bloque) $= t_{lectura}$ (2 espacios útiles + 1 espacio no útil) $= 3\ t_{lectura}$ (1 KB)

   0.12 mseg $= 3\ t_{lectura}$ (1 KB) $\rightarrow \mathbf{t_{lectura}\ (1\ KB) = 0.04\ mseg}$

   1 pista (ó 1 vuelta) = 100 sectores $\rightarrow \mathbf{t_{rotación}} = 200 * 0.04$ mseg **= 8 mseg**

   y la inversa (en un minuto) será el numero de RPMs

   Velocidad = 60 seg / 8 mseg = 7.500 RPM

2. Calcular el tiempo (medio) que se tardaría en leer un bloque cualquiera **[2 p]**

   El tiempo (medio) de un bloque cualquiera: $\mathbf{t_{búsqueda} + t_{latencia} + t_{transferencia}}$

   - del enunciado $\rightarrow t_{búsqueda} = 10$ mseg
   - $t_{latencia} = 8$ mseg / 2 $= 4$ mseg
   - del enunciado $\rightarrow t_{trans} = 0.12$ mseg

   t = 10 + 4 + 0.12 = 14.12 mseg

3. Calcular la tasa de transferencia de pico y la tasa de transferencia sostenida **[4 p]**

   T_{pico} = 1 bloque de datos (de 1 KB) en 0.04 mseg

   $T_{sostenida}$ = 1 pista completa (de 100 KB) en 8 mseg

   T pico = 1 KB / 0.04 mseg = 25.000 KB/seg
   T sostenida = 100 KB / 8 mseg = 12.500 KB/seg

4. Si en estas condiciones se modifica la geometría del disco y ahora un bloque estuviera formado por 5 sectores ¿Cuál sería el nuevo tiempo de transferencia de 1 bloque? **[2 p]**

   Un bloque contendrá (a efectos de transferencia) 5 partes útiles (de 1 KB) y 4 partes no útiles. En total 9 partes (de 1 KB).

   t_{transf} **= 9 * 0.04 = 0.36 mseg**

16º -

1. Calcule el número N de cajones distintos necesarios. *Justifíquelo*. **[2 p]**
   El número de cajones es, por definición, simplemente el número de valores distintos de la función *hash*. Para la función dada y el rango de valores de entrada dado, obtenemos que h es un entero en el rango $0 \leq h \leq 4$, por lo que:

   N = 5 cajones

2. Suponga que los valores de A están *uniformemente distribuidos*, esto es, un registro cualquiera puede estar en cualquiera de los N cajones con igual probabilidad, y una clave de búsqueda puede estar en cualquier rango con igual probabilidad. ¿Cuál es el número de operaciones de I/O esperado para buscar un registro? *Justifíquelo*. **[3 p]**
   Cada cajón posee 16.000 / 5 = 3.200 registros (< 4.000), que caben en 1 bloque cada uno.

Número de I/Os = 1

3. Ahora suponga que existe un *cierto sesgo* en los datos. En particular, la probabilidad de que un valor de A (tanto en el registro como en la búsqueda) esté en el cajón i es **2*i / N*(N+1)** (tenga en cuenta que la suma de estas probabilidades, desde i=1 hasta N, vale 1). ¿Cuál es el número de operaciones de I/O esperado para buscar un registro? *Justifíquelo*. **[3 p]**

Para **N = 5**, la expresión **2*i / N*(N+1)** queda simplemente **i / 15**

Cajón	Prob.	# Registros	**# Bloques**
[1]	1/15	1.067	**1**
[2]	2/15	2.133	**1**
[3]	3/15	3.200	**1**
[4]	4/15	4.267	**2**
[5]	5/15	5.333	**2**

Usando la tabla anterior, se puede ver que:

$$\textbf{Número de I/Os} = 1\times\frac{1}{15}+1\times\frac{2}{15}+1\times\frac{3}{15}+2\times\frac{4}{15}+2\times\frac{5}{15}=\frac{24}{15} = 1{,}6$$

4. Ahora considere un caso de *sesgo extremo*. La probabilidad de que un valor de A (tanto en el registro como en la búsqueda) esté en el cajón 1 ó 2 es de 1/2 y, por lo tanto, la probabilidad es 0 para los otros rangos. ¿Cuál es el número de operaciones de I/O esperado para buscar un registro? *Justifíquelo*. **[3 p]**

Cajón	Prob.	# Registros	**# Bloques**
[1]	½	8.000	**2**
[2]	½	8.000	**2**
[3]	0	0	**0**
[4]	0	0	**0**
[5]	0	0	**0**

Usando la tabla anterior, se puede ver que:

$$\textbf{Número de I/Os} = 2\times\frac{1}{2}+2\times\frac{1}{2} = 2$$

5. Si se usa un árbol B+ para la misma aplicación, ¿Cuál es el número de operaciones de I/O esperado? Suponga que los nodos de un árbol B+ contienen también 4.000 punteros y claves. *Justifique* porqué. **[2 p]**

Un índice con estructura de árbol B+ con 4.000 claves en cada nodo, tiene **2 niveles** para indexar 16.000 registros. Por lo tanto:

Número de I/Os = 2

6. *Justifique* qué índice, el asociativo o un árbol B+, es mejor para consultas de rangos en dos situaciones: la actual y una general. **[2 p]**

En el presente problema, en **todas** las situaciones, incluso en la de sesgo extremo, el índice asociativo es mejor (o como mucho) igual de costoso que un índice B+.

Sin embargo, los índices asociativos no se construyen generalmente usando funciones similares a la presente (p.e. la función ***h*** usada no posee algunas de las propiedades deseadas en una función *hash*, como la propiedad de dispersión, etc.), por lo que los índices asociativos son

extremadamente ineficientes para búsquedas de rangos (se debería buscar valor a valor), por lo que un índice de árbol B+ resulta más conveniente.

17º -

1. $\sigma_{A < 5.,000}(R)$

   $\sigma_{A < 50.000}(R)$

   Tanto el acceso sobre los datos directamente, como el uso de un índice B+, operan sobre estructuras (datos o índices según el caso) ordenadas. Para esta selección, la elección de acceder directamente al fichero secuencial ordenado es ligeramente mejor en coste a usar un índice con estructura de árbol B+, simplemente por la sobrecarga requerida para navegar por el árbol y buscar cada bloque con el puntero a los datos.

2. $\sigma_{A = 50.000}(R)$

   $\sigma_{A = 50.000}(R)$

   Un índice asociativo será más eficiente aquí, dado que sólo habrá que leer el bloque del cajón de índices correspondiente al valor buscado.

3. $\sigma_{A > 50,000 \wedge A < 50,010}(R)$

   $\sigma_{A > 50.000 \wedge A < 50.010}(R)$

   Un índice con estructura de árbol B+ será más eficiente aquí, dado que para encontrar el primer valor sólo habrá que recorrer el árbol en profundidad y a partir de ese punto son accesos consecutivos a los bloques (ordenados) del árbol. Leer el fichero de datos ordenado hasta encontrar el punto de comienzo del intervalo implica una búsqueda binaria que es mayor que el coste de atravesar un árbol equivalente.

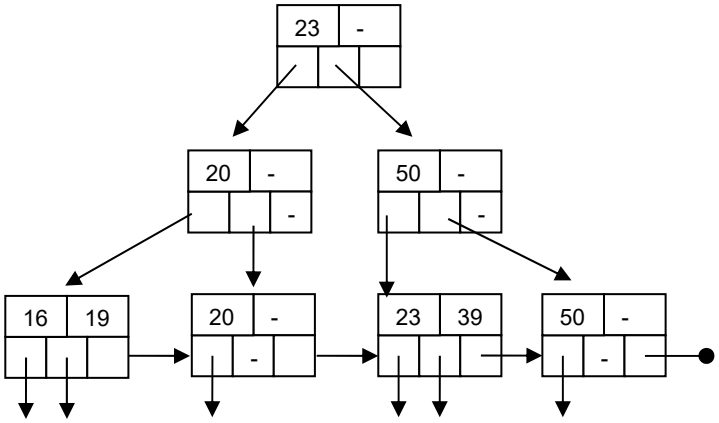
4. $\sigma_{A \neq 50,000}(R)$

   $\sigma_{A \neq 50,000}(R)$

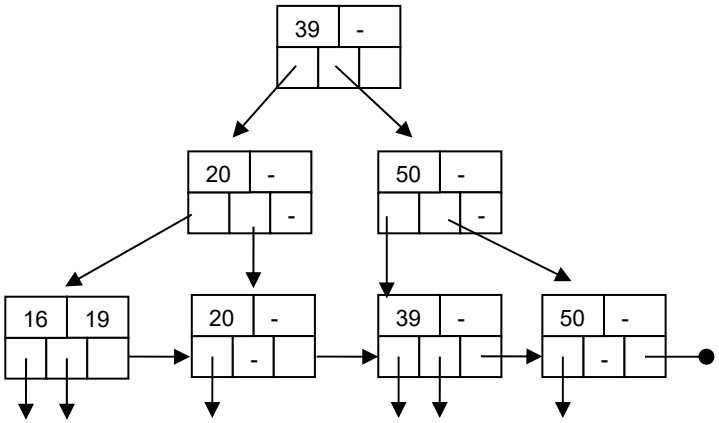
   Puesto que esta selección requerirá analizar (leer) todas las entradas disponibles, y en el caso del árbol B+ vamos a empezar desde el principio del índice ordenado, acceder directamente al fichero de datos ordenado será más eficiente, nuevamente debido al sobrecoste de la búsqueda en el índice.

18º -

1. Muestre el árbol B+ completo (esto es, **con todas** las claves y **con todos** los punteros) que resultaría tras insertar la clave 19.

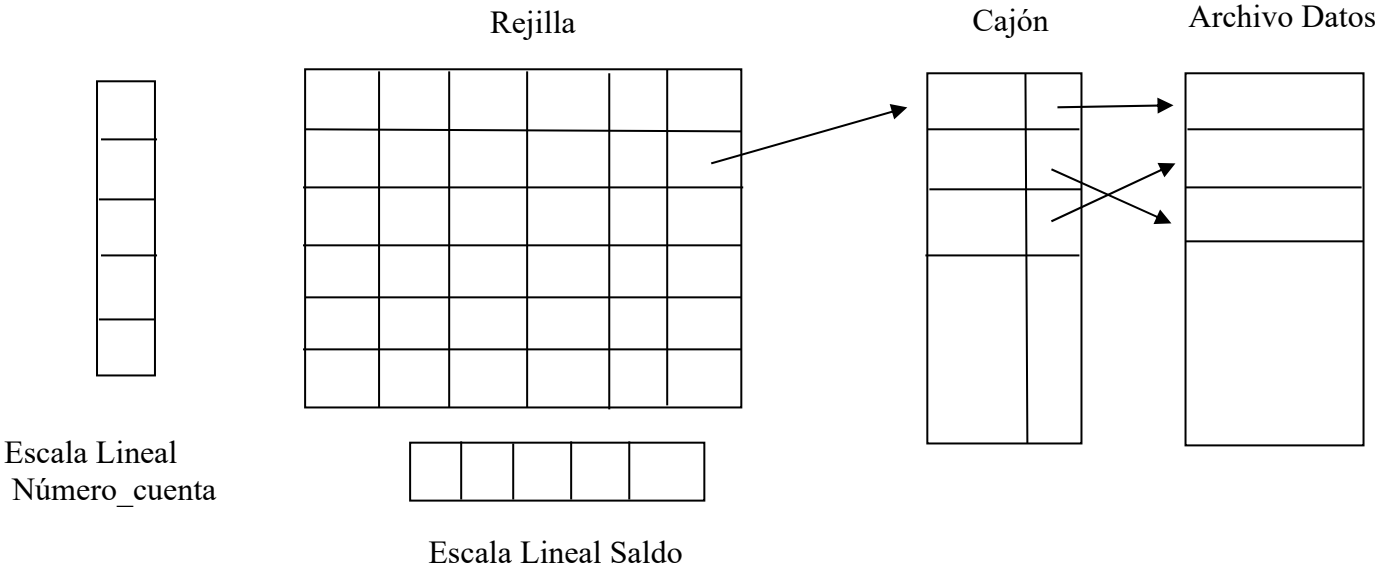


2. A partir del resultado anterior, muestre ahora el árbol B+ completo (esto es, **con todas** las claves y **con todos** los punteros) que resultaría tras borrar la clave 23.



19º-

a) Dibujar el esquema completo índice más archivo de datos.



b) Determinar el tamaño en bloques del archivo de datos.

Para este caso, el tamaño del archivo de datos es:

Longitud registro de cuenta: Lr=3*20=60 bytes. Factor de bloques: fr= $\lfloor 1024 / 60 \rfloor$= 17 reg/bloque. Numero de bloques: $b_R = \lceil 20.000 / 17 \rceil$ = 1177 bloques

Bloques del archivo = 1177 bloques

c) Determinar el número de bloques del índice.

Se trata de un índice secundario sobre dos campos que forman una clave compuesta (todos los valores diferentes).

El número de bloques del índice estará compuesto por las dos escalas lineales + rejilla + los cajones

La rejilla junto con la escala lineal: 6x6+5x2 = 36+10=46 celdas, 46*20=920 bytes que caben en 1 bloque.

Calculamos el factor de bloques de cada cajón:

Longitud registro del cajón sera 2 campos (numero_sucursal,saldo) y el puntero a registro: Lr=3*20=60 bytes. Factor de bloques: fr= $\lfloor 1024 / 60 \rfloor$= 17 reg/bloque.

Cada cajón tendrá de media: $\lceil 20.000 / 36 \rceil$ = 556 registro redondeando y poniendo peor caso.

Numero de bloques: $b_R = \lceil 556 / 17 \rceil$ = 33 bloques

Luego el tamaño del índice será: 1+ 33*36 = 1189 bloques.

Bloques del índice = 1189 bloques

d) Cuál es el número de accesos necesarios para cargar un registro de datos en memoria

Para cargar un registro de datos, habrá que:

acceder a la rejilla: 1 blque

Acceder al puntero correspondiente y leer todo el cajón: 33 bloques.

Recuperar un dato: 1 bloque

Total: 1 + 33 +1 = 35 bloques

Número de accesos = 35 bloques