1. Introducción

- **Filosofía**: Utilizar estructuras de datos adicionales que faciliten el proceso de la búsqueda, actuando como **índices** sobre el fichero de datos.

- **Requisitos**:
  - Acceso directo al fichero
  - La utilización de estructuras de datos adicionales (*índices*).

- **Tipos de índices**:
  - Indices organizados en tablas.
    - Indices densos.
    - Indices no densos (organización *secuencial - indexada*).
  - Indices organizados en árboles
    - Arboles binarios de búsqueda.
    - Arboles AVL.
    - Arboles B y variantes (B^+ y B^*)

2. Métodos simples de acceso por índices

2.1. Indices densos

0	4	70	Juarez Valladares	0
1	1	18	Ruperez Galiano	1
2	5	68	Carrasco Martinez	2
3	3	38	Domingo Lucas	3
4	6	12	Abad Ruiz	4
5	7	21	Berbabe Perez	5
6	2	44	Martin Perez	6
7	0	52	Fernandez Sostoa	7

A

0	4	70	Juarez Valladares	0
1	5	18	Ruperez Galiano	1
2	2	68	Carrasco Martinez	2
3	3	38	Domingo Lucas	3
4	7	12	Abad Ruiz	4
5	0	21	Berbabe Perez	5
6	6	44	Martin Perez	6
7	1	52	Fernandez Sostoa	7

B

Figura 1: Fichero *clientes* indexado por: A) campo *codigo*. B) campo *apellido*

Orden	codigo	apellido	*Orden*	codigo	apellido
1	12	Abad Ruiz	5	44	Martin Perez
2	18	Ruperez Galiano	6	52	Fernandez Sostoa
3	21	Bernabe Perez	7	68	Carrasco Martinez
4	38	Domingo Lucas	8	70	Juarez Valladares

Orden	codigo	apellido	*Orden*	codigo	apellido
1	12	Abad Ruiz	5	52	Fernandez Sostoa
2	21	Bernabe Perez	6	70	Juarez Valladares
3	68	Carrasco Martinez	7	44	Martin Perez
4	38	Domingo Lucas	8	18	Ruperez Galiano

2.2. Acceso secuencial-indexado

172 SOLA AGUILERA | **TALAVERA RUIZ 173**

SOLA AGUILERA, A.
Cam. Ronda 134 576239

SOLA ALONSO, M.C.
Recogidas 46 678656

SOTO JUAREZ, V.
Gaviota, 16 467650

SOLA LINARES, P.
Pta. Guzmanes, 4 936745

TALAVERA ARBOLEDA, M.
Rucio, 45 997369

TALAVERA RUIZ, M.J.
S. Francisco, 1 847264

174 TALENS PIO | **TITO ROJO 175**

TALENS PIO, M.
Rio Ebro, 6 662840

TARRAGO NUÑEZ. C.
Maria Lejarraga, 9 ... 552288

TARRAGONA CAMACHO, L.
Gabriel Miro, 3 113095

TITO ROJO, B.
S. Genaro, 66 267384

176 TOCON PEREZ | **TOVAR SOTO 177**

TOCON PEREZ, J.
Luis Braille, 7 150102

TORRES CAMACHO, D.
S. Anton, 44 749968

TORRUS LOPEZ, A.
Arabial, 29 754321

TOVAR SOTO, G.
Duquesa, 44 672975

178 TRAPERO CHICA | **VALDIVIA MILLA 179**

TRAPERO CHICA, C.
Sto. Domingo, 4, ... 982641

URBANO ALONSO, P.
Fontiveros, 3 549812

URBANO BERMUDEZ, S.
Motril, 67 552873

VALDIVIA MILLA, Z.
A. Gracia, 33 667739

Figura 2: Unas páginas de la guía telefónica

`apellido`	*página*
...	...
Talavera Ruiz	173
Tito Rojo	175
Tovar Soto	177
Valdivia Millan	179
...	...

Figura 3: Un índice para la guía telefónica

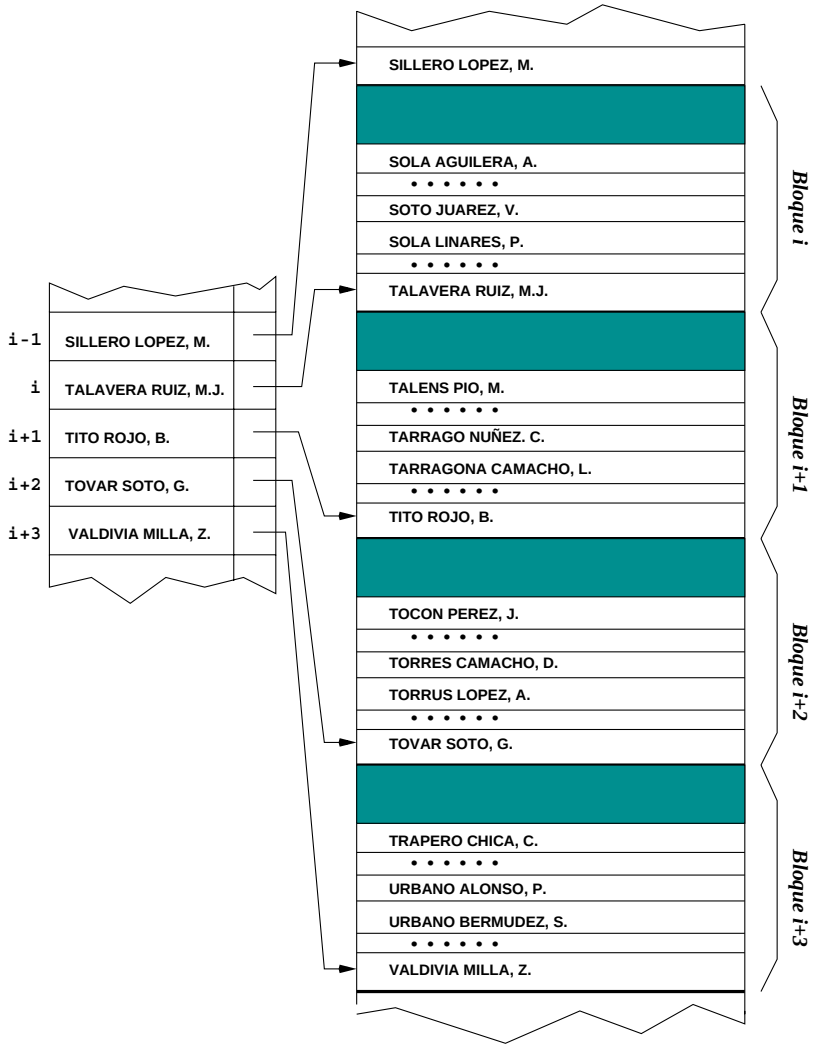


Figura 4: Acceso secuencial indexado. Indice y fichero indexado

3. Indices estructurados en forma de árboles

3.1. Arboles binarios de búsqueda

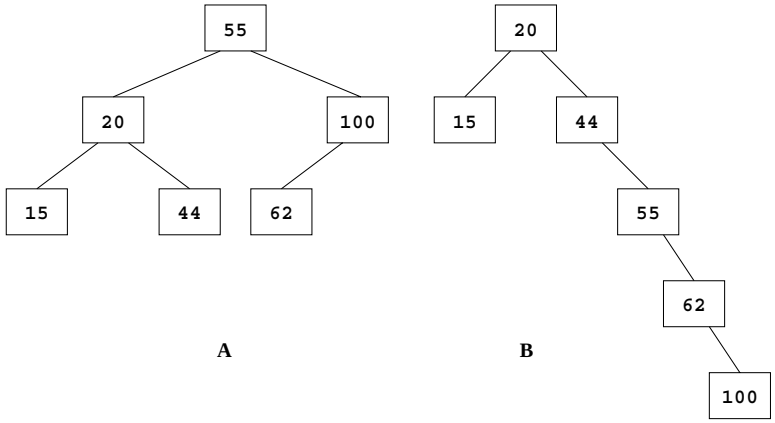


Figura 5: Arboles binarios de búsqueda sobre el mismo conjunto

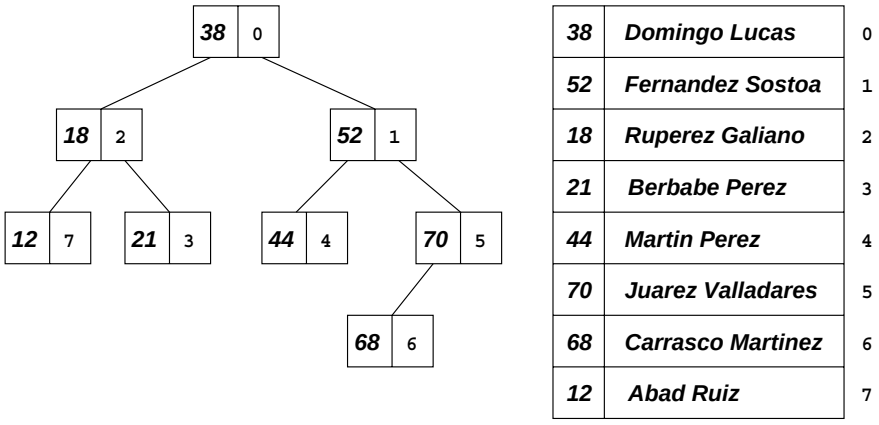


Figura 6: Un ABB que actúa como índice sobre un fichero

3.2. Arboles AVL

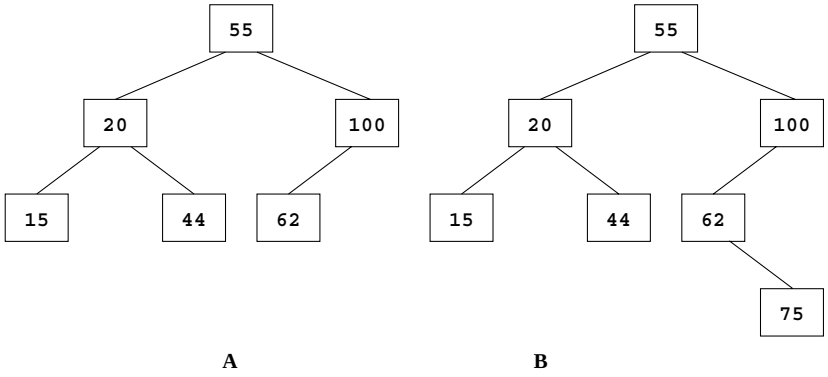


Figura 7: ABBs sobre el mismo conjunto. A) el árbol es AVL. B) el árbol no es AVL

- **Eficiencia**.

  La altura h de un AVL con n nodos está acotada por:

  $$\log_2(n+1) \leq h < 1.44\ \log_2(n+2) - 0.33$$

  La longitud de los caminos de búsqueda (la altura) para un AVL de n nodos nunca excede al 44% de la longitud de los caminos de un árbol completamente equilibrado con esos n nodos.

  Consecuencia: en el peor de los casos, la búsqueda está en un orden de $O(\log_2 n)$.

- **Importante**: Coste adicional para mantener el equilibrio.

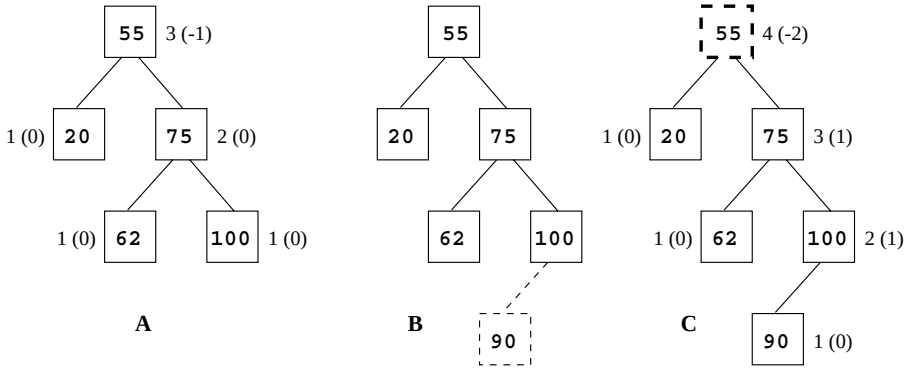


Figura 8: a) Arbol AVL inicial. B) Inserción de la clave 90. C) El árbol deja de ser AVL

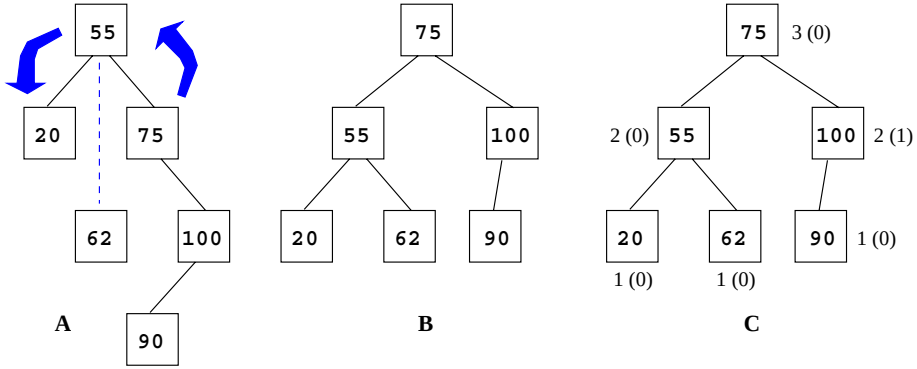


Figura 9: a) Operaciones para una rotación simple a la izquierda. B) Resultado de la rotación. C) El árbol resultante es AVL

3.3. Arboles con múltiples hijos

- **Motivación**: Acceso a grandes ficheros de datos.
- **Restricción**: Los índices residen en disco.
- **Problema**: Accesos a disco muy costosos.
- **Solución**: Organizar árboles con múltiples claves, n, por nodo.
  - Recuperar un nodo de este árbol (un acceso al fichero índice) selecciona una entre n alternativas, frente a una entre dos (AB).
  - El índice puede diseñarse para que el tamaño de cada nodo coincida con el de un bloque de disco.

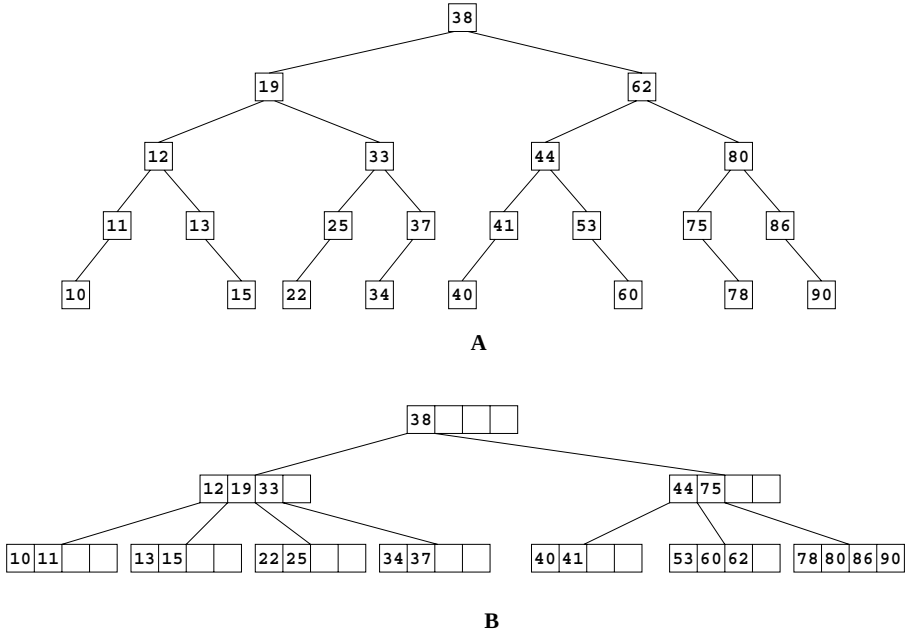


Figura 10: A) Arbol AVL. B) Arbol B de orden 5

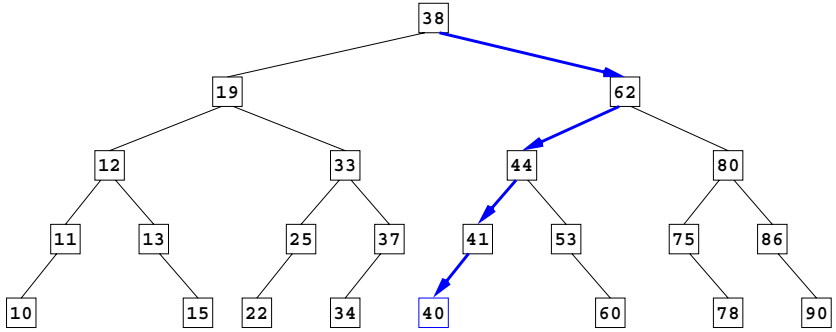


Figura 11: Búsqueda de la clave 40 en un AVL

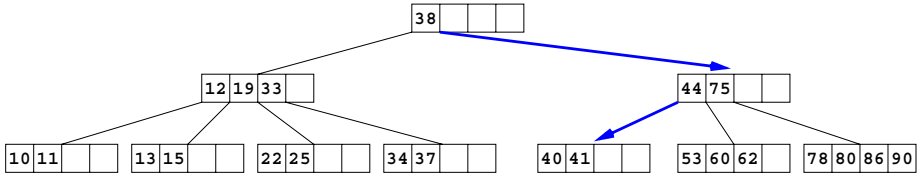


Figura 12: Búsqueda de la clave 40 en un árbol B de orden 5

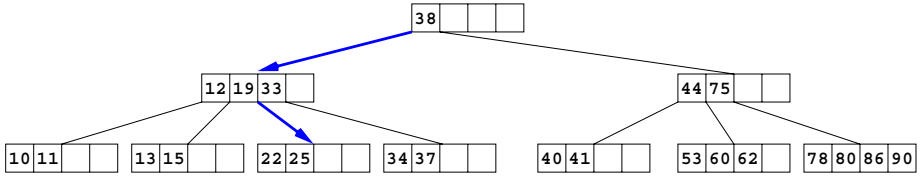


Figura 13: Búsqueda de la clave 25 en un árbol B de orden 5

4. Arboles B

4.1. Definición y estructura de los árboles B

Un **árbol B de orden** n se caracteriza por cinco propiedades básicas. Para que un árbol con múltiples hijos sea un árbol B debe verificar las siguientes propiedades:

1. Cada nodo tiene como máximo n descendientes.
2. La raiz tiene al menos dos descendientes o es una hoja.
3. Cada nodo interno tiene al menos $\lceil \frac{n}{2} \rceil$ descendientes.
4. Un nodo no hoja con k descendientes tiene $k-1$ claves.
5. Todas las hojas están en el mismo nivel.

Orden	3	4	5	6	7	8
máximo	3	4	5	6	7	8
mínimo	2	2	3	3	4	4
núm. claves	1-2	1-3	2-4	2-5	3-6	3-7

Table 1: Máximo y mínimo número de descendientes y rango del número de claves

Todos los nodos tienen la misma estructura:

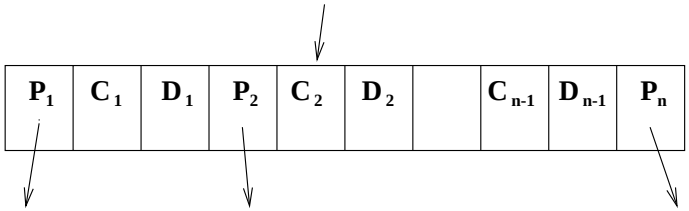


Figura 14: Estructura de un nodo de un árbol B

- $P_i \quad i = 1, 2, \ldots, n$ son **indicadores** a otro nodo en el árbol.
- $C_i \quad i = 1, 2, \ldots, n-1$ son las **claves** asociadas al nodo.
- $D_i \quad i = 1, 2, \ldots, n-1$ son **direcciones** sobre el fichero de datos.

$$C_1 \ < \ C_2 \ < \ \ldots \ < \ C_{n-1}$$

Si $\overline{P_i}$ es el conjunto de claves en el nodo referenciado por P_i:

- $\overline{P_1} \ < \ C_1$.
  Todas las claves del nodo referenciado por P_1 son menores que C_1.

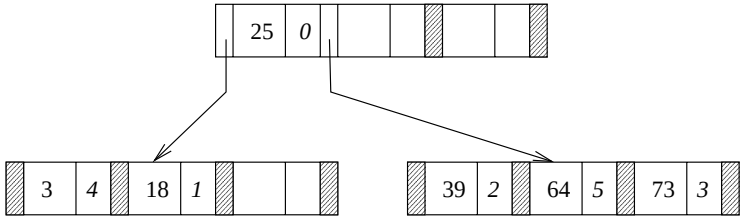
- $C_{n-1} \ < \ \overline{P_n}$.
  Todas las claves del nodo referenciado por P_n son mayores que C_{n-1}.

- $C_{i-1} \ < \ \overline{P_i} \ < \ C_i \quad i = 2, 3, \ldots, n-1$.
  Entre cada pareja de claves, C_{i-1} y C_i, existe un indicador que referencia a un nodo en el que sus claves son mayores que C_{i-1} y menores que C_i.

Ejemplo 1

Arbol B de orden 4 que actúa como índice sobre un fichero que contiene 6 registros.

Indice (arbol B de orden 4)



Fichero de datos

0	25	Lopez Torres, Alfredo
1	18	Garcia Garcia, Fernando
2	39	Baldomero Ruiz, Pilar
3	73	Mantas Ortega, Olga
4	3	Vera Lozano, Rafael
5	64	Garcia Bonaire, Vicente

Figura 15: árbol B de orden 4 que indexa un fichero de 6 registros

Ejemplo 2

Arbol B de orden 5 que actúa como índice sobre un fichero de 23 registros.

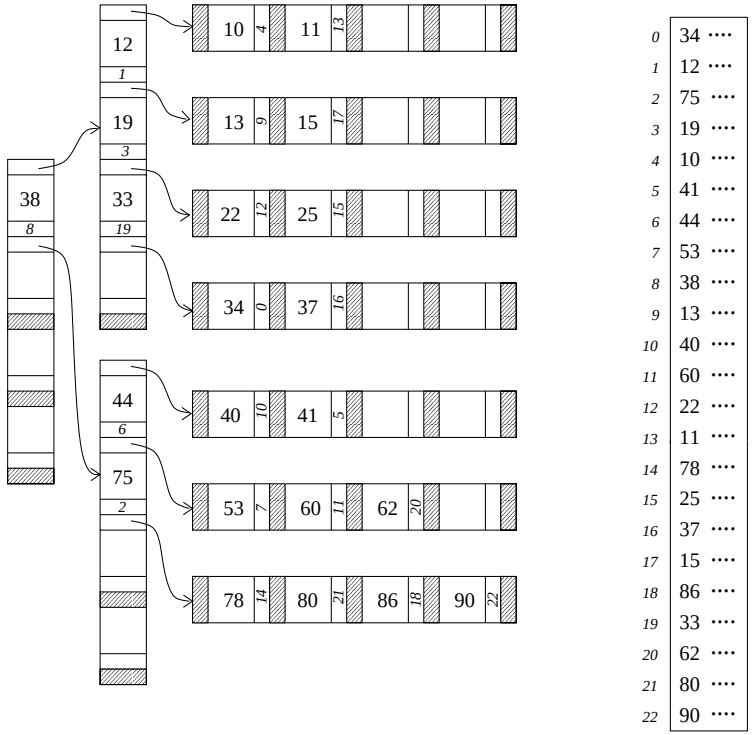


Figura 16: árbol B de orden 5 que indexa un fichero de 23 registros

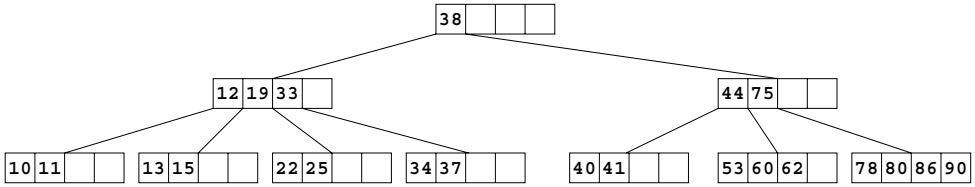


Figura 17: Representación simplificada del árbol B anterior

Algoritmo de búsqueda en un árbol B

1. Sea `NodoActual` la raiz del árbol B.
2. Buscar el valor x entre las claves C_i de `NodoActual`.
   (a) Si está,
   Sea $C_m = x$,
   La clave está en el fichero.
   Leer registro D_m del fichero.
   Ir al paso 3 (FIN).
   (b) Si no está,
   Seleccionar el puntero adecuado P_i según x y C_i.
   Si P_i = `NULL` (estamos en un nodo hoja)
   La clave no está en el fichero.
   Ir al paso 3 (FIN).
   Si no,
   Sea `NodoActual` el nodo apuntado por P_i.
   Volver al paso 2.
3. FIN.

Ejemplo

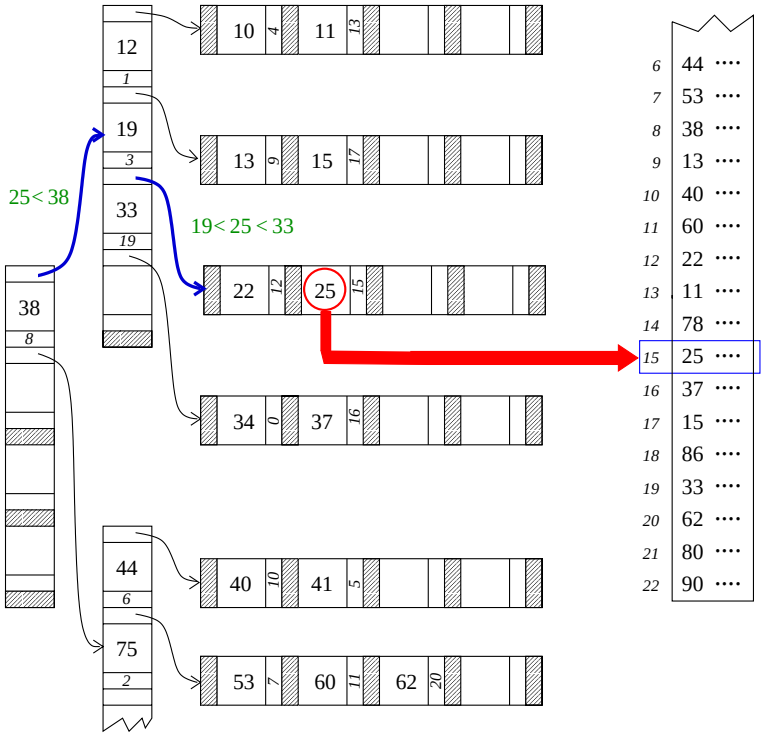


Figura 18: Ejemplo de búsqueda (clave 25) en un árbol B de orden 5

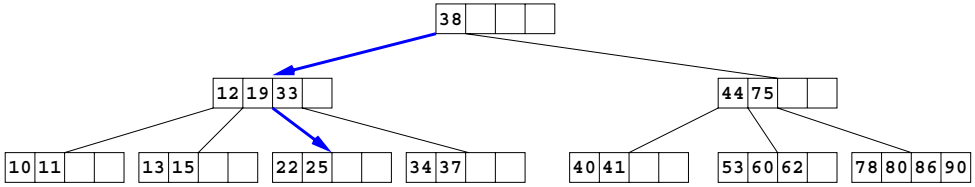


Figura 19: Búsqueda de la clave 25 en un árbol B de orden 5. Esquema simplificado

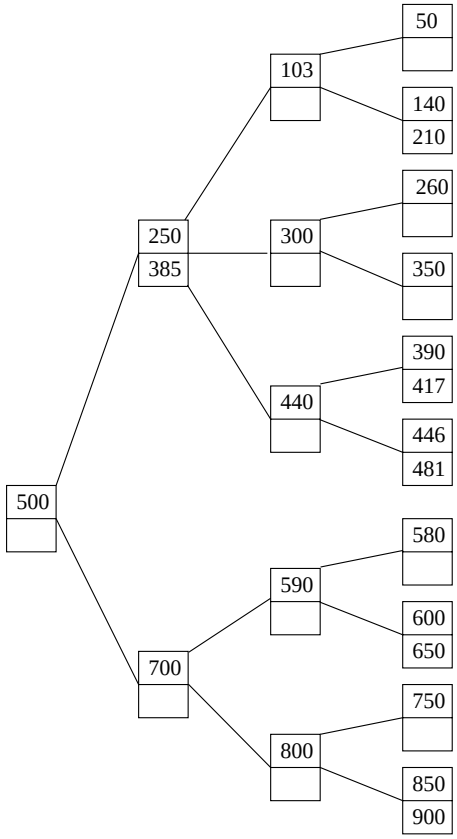
Algoritmo de inserción en un árbol B

1. Buscar **la hoja** donde debería insertarse el valor x.
2. Comprobar si la hoja está llena.
   (a) Si no está llena,
   Insertar la clave de forma ordenada y terminar.
   (b) Si está llena,
   1. Crear un nuevo nodo.
   2. Redistribuir las claves entre los dos nodos y el nodo padre de forma que la mediana se promociona al nodo padre.
   3. Si el nodo padre está lleno,
   Repetir el proceso hasta que la inserción no produzca la promoción de claves.
   Este proceso puede afectar al nodo raiz, de forma que si está lleno se divide y se crea una nueva raiz con una sola clave, aumentando la altura del árbol en una unidad.

Alternativa

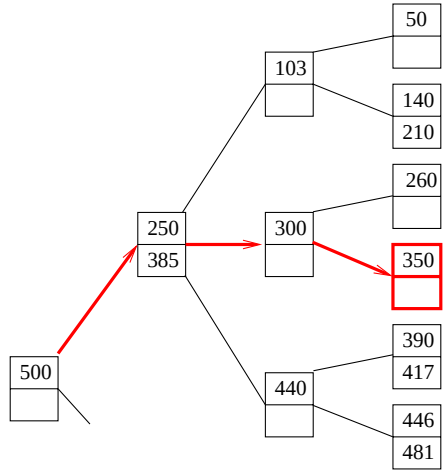
- Comprobar, antes de dividir un nodo, si algún hermano adyacentes tiene huecos.
- Si los tiene, se trata de redistribuir las claves de éste, más la clave del padre que las separa, más la clave a insertar.
- La redistribución consiste en promocionar al nodo padre la mediana de todas las claves involucradas.

Ejemplo

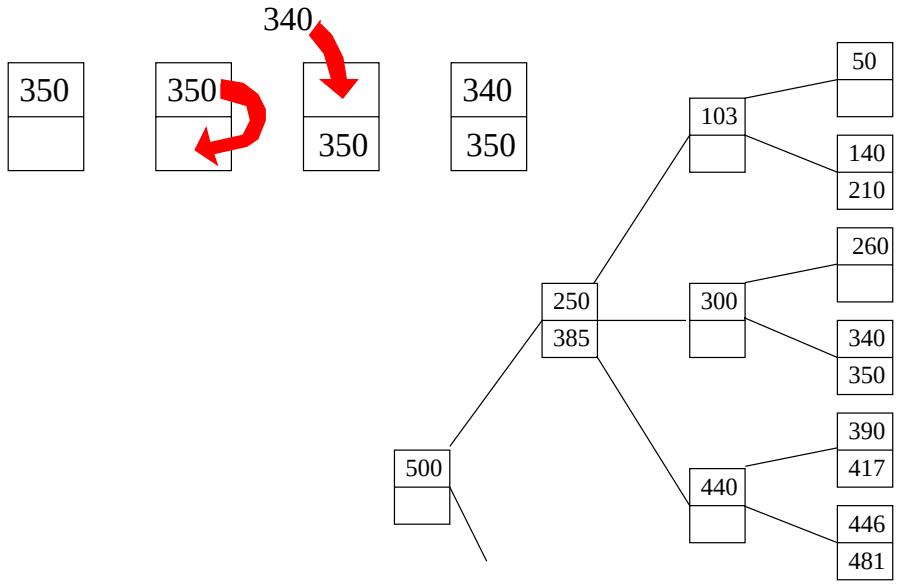


Arbol B de orden 3 inicial

1. Inserción de 340.

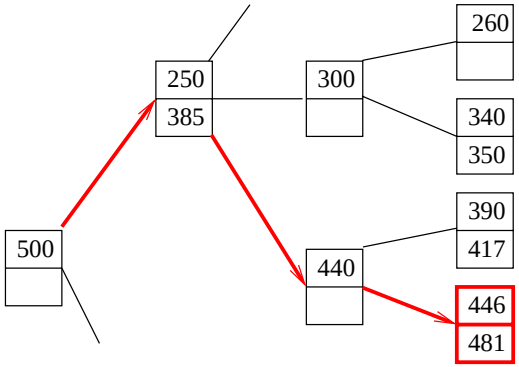


Búsqueda de la hoja en la que insertar 340

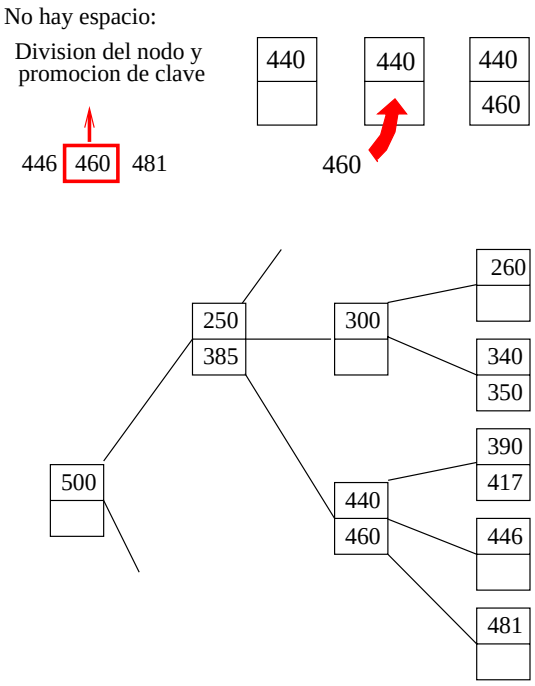


Inserción ordenada en la hoja y estado final del subárbol

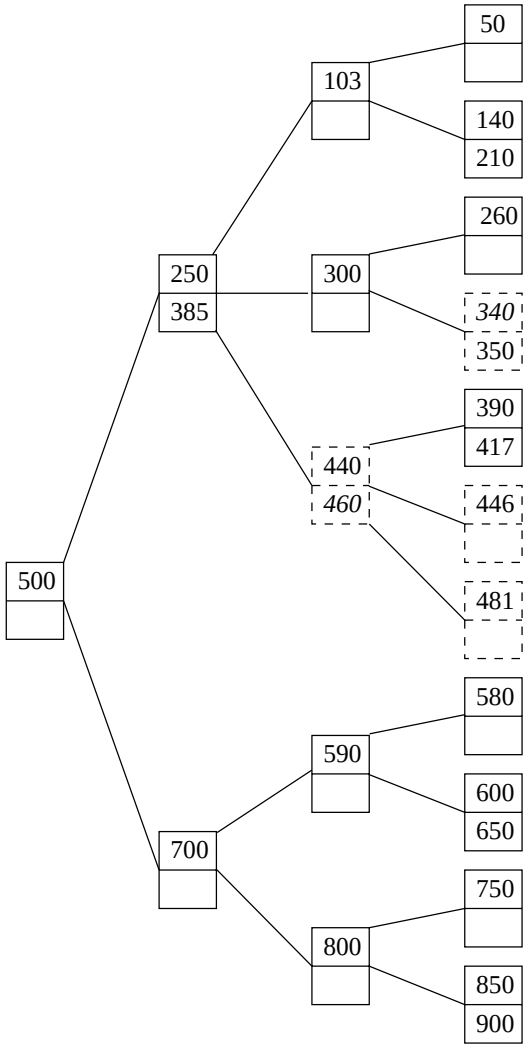
2. Inserción de 460.



Búsqueda de la hoja en la que insertar 460

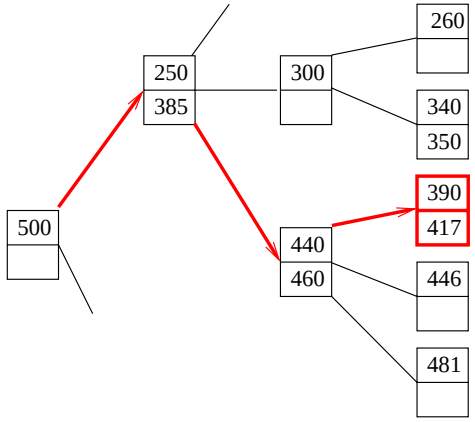


Inserción ordenada en el padre y estado final del subárbol

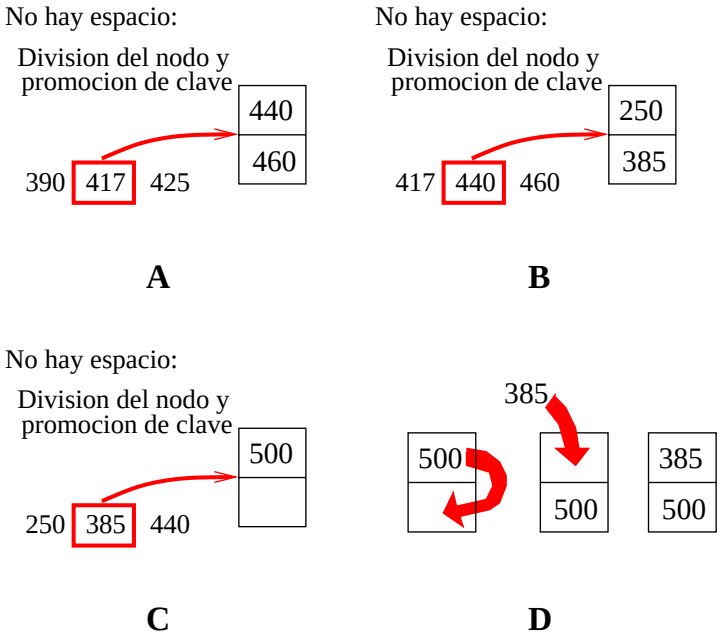


Arbol tras la inserción de 340 y 460

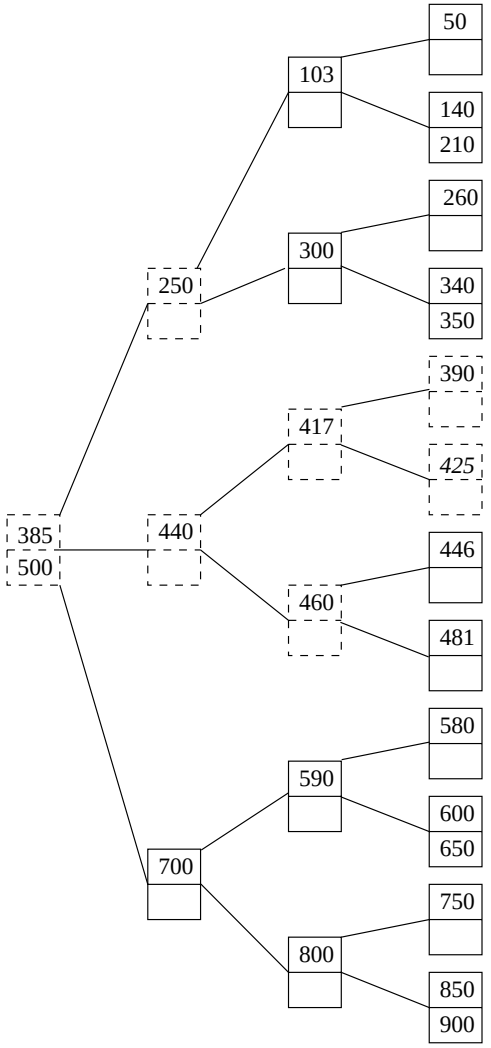
3. Inserción de 425.



Búsqueda de la hoja en la que insertar 425

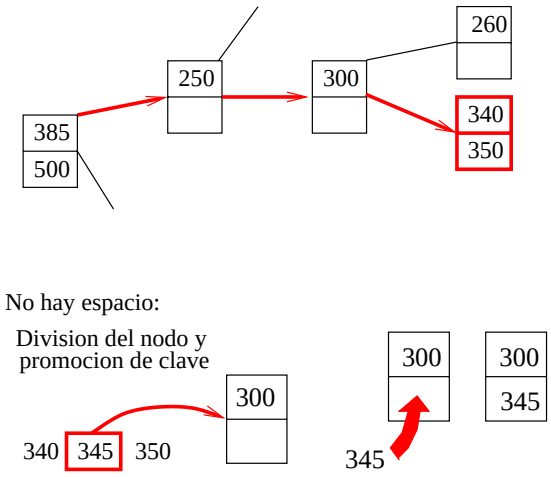


Proceso de actualización del árbol para insertar 425



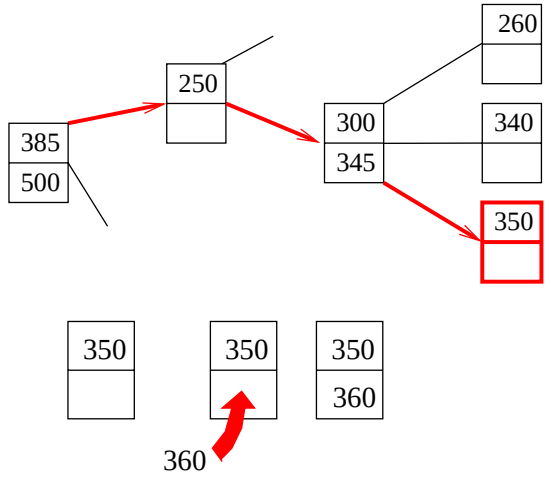
Arbol tras la inserción de 425

4. Inserción de 345.

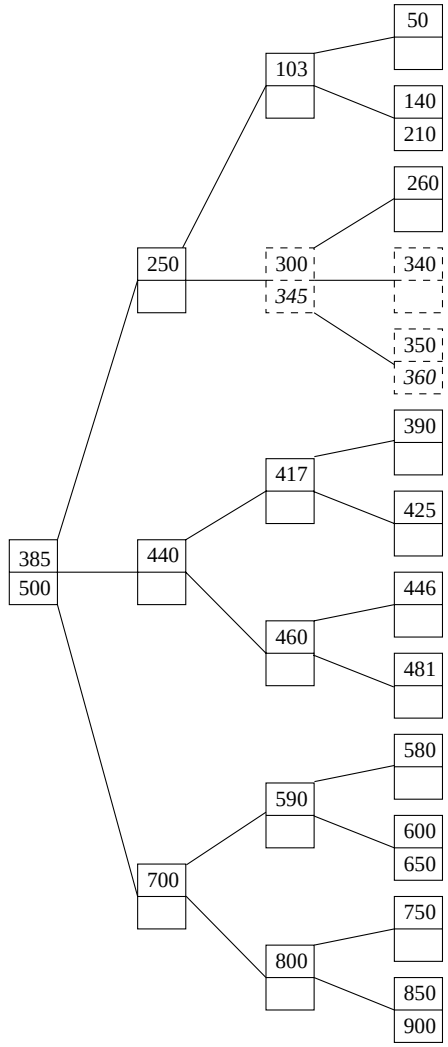


Inserción de la clave 345

5. Inserción de 360.

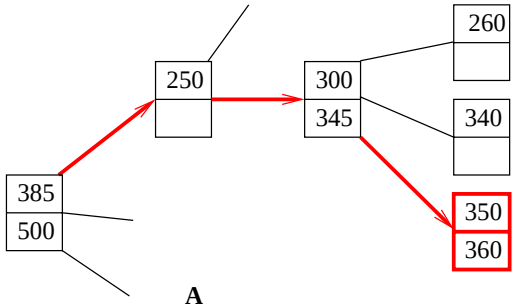


Inserción de la clave 360



Arbol tras la inserción de 345 y 360

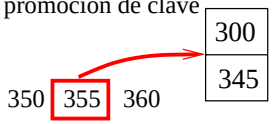
6. Inserción de 355.



A

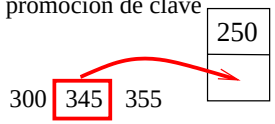
No hay espacio:

Division del nodo y promocion de clave

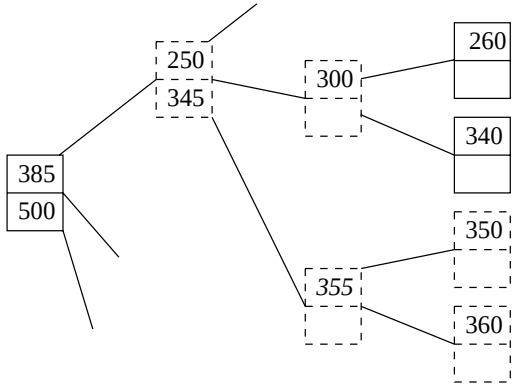


No hay espacio:

Division del nodo y promocion de clave



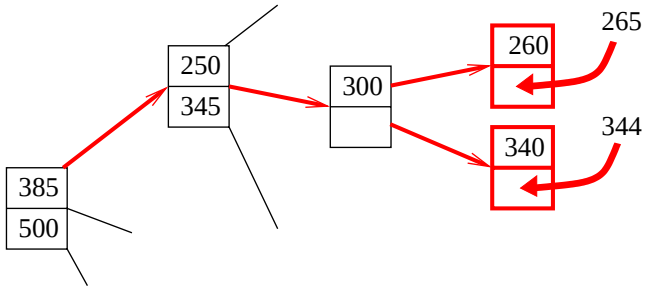
B



C

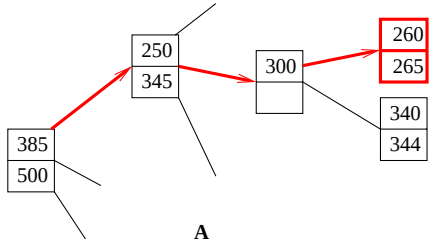
INSERCIÓN DE LA CLAVE 355

7. Inserción de 265 y 344.



INSERCIÓN TRIVIAL DE LAS CLAVES 265 Y 344

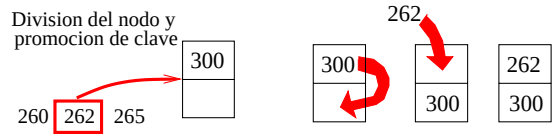
8. Inserción de 262.



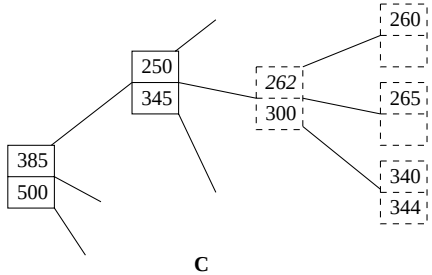
A

No hay espacio:

Division del nodo y promocion de clave

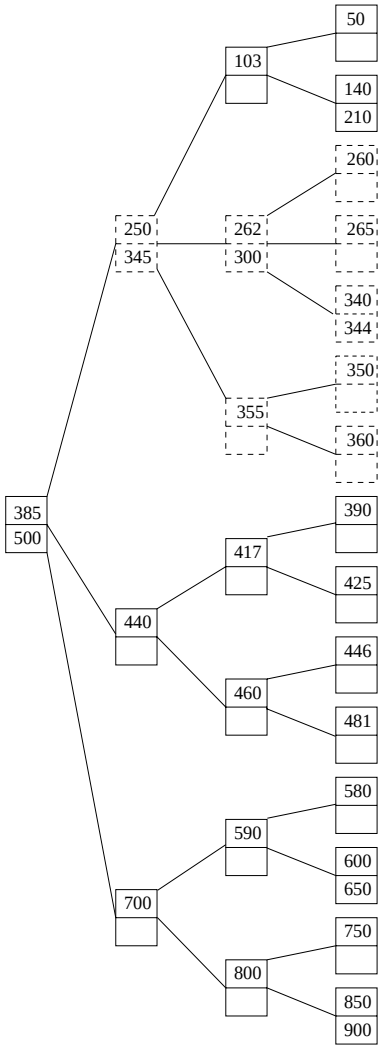


B



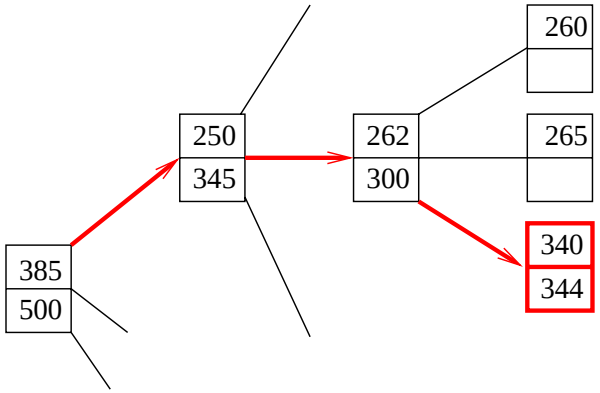
C

INSERCIÓN DE LA CLAVE 262

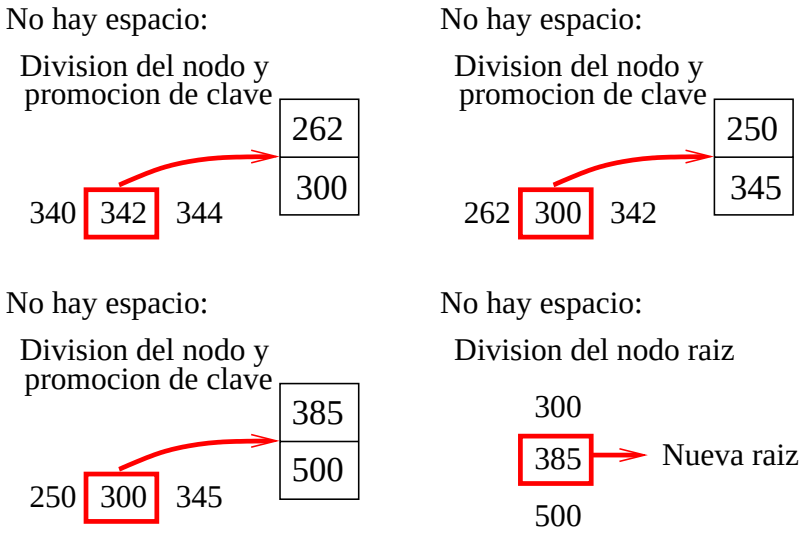


Arbol tras la inserción de 355, 265, 344 y 262

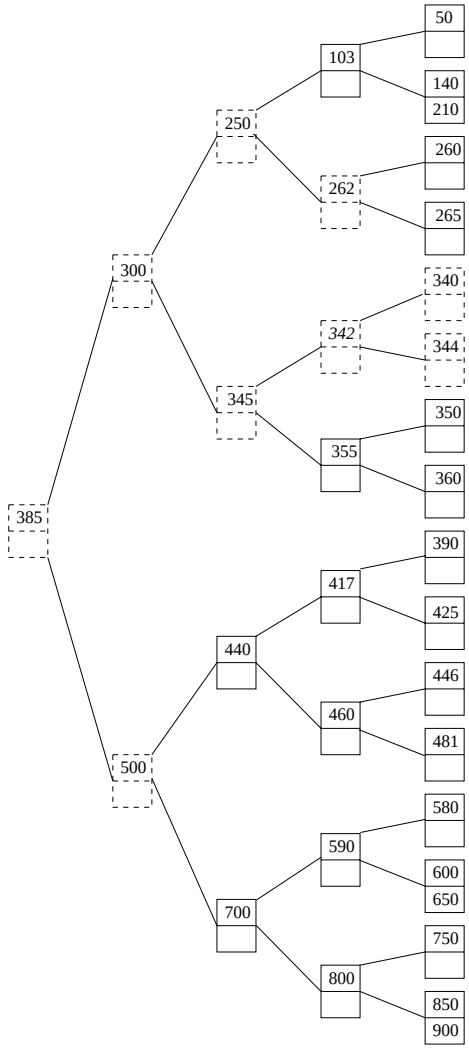
9. Inserción de 342.



Búsqueda de la hoja en la que insertar 342



Proceso de actualización del árbol para insertar 342



Arbol tras la inserción de 342

Algoritmo de borrado en un árbol B

Buscar el nodo donde está la clave x.

(a) Si es un nodo hoja,

Borrar la clave y reordenar las restantes.
Verificar y ajustar el árbol.

(b) Si es un nodo interior,

Sustituir la clave a borrar por la inmediatamente superior, que ha de estar forzosamente en un nodo hoja.
Verificar y ajustar el árbol.

Verificar y ajustar el árbol

(**1**) Si el número de claves del nodo modificado es válido,

Terminar.

Si no,

(**2**) Si algún hermano adyacente tiene más claves que el mínimo,

Redistribución.
Terminar.

(**3**) Si no,

Unión.
Volver a verificar y ajustar el árbol para el nodo padre.

1. **Redistribución**.

   Se aplica cuando el nodo afectado queda con menos claves que el mínimo permitido y algún hermano adyacente tiene más claves que el mínimo y puede, por lo tanto, "prestar" una clave al nodo afectado.

   La clave del padre de ambos nodos pasa al nodo donde se ha borrado y ésta se sustituye por una clave del otro nodo:

   - la menor, si el hermano está a su derecha,
   - la mayor, si el hermano está a su izquierda.

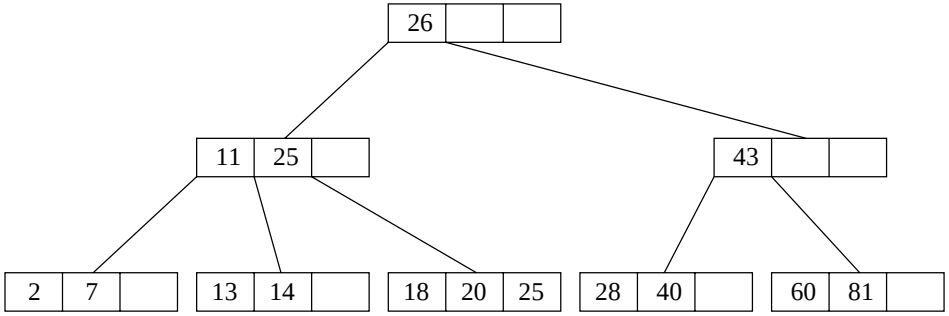
2. **Unión**.

   Se aplica cuando no sea posible la redistribución y la única opción es, por lo tanto, compactar dos hermanos adyacentes en uno.

   Se hace un sólo nodo entre el nodo donde se ha borrado, un hermano y la clave que las separa en el nodo padre.

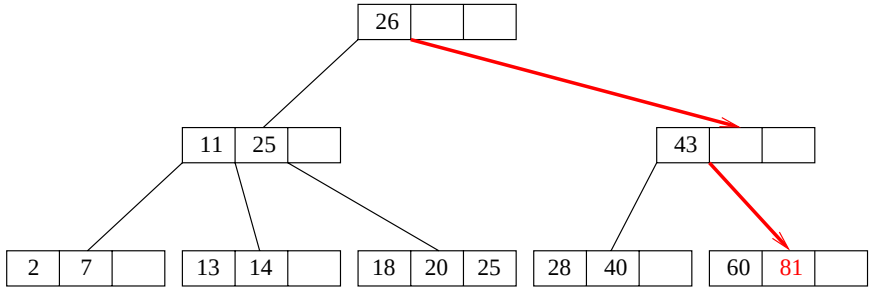
   - La unión puede obligar a ajustar el árbol, tomando ahora como referencia el nodo padre.
   - Este efecto puede propagarse hacia la raiz, produciendo, en algunos casos el decrecimiento en altura del árbol.

Ejemplo



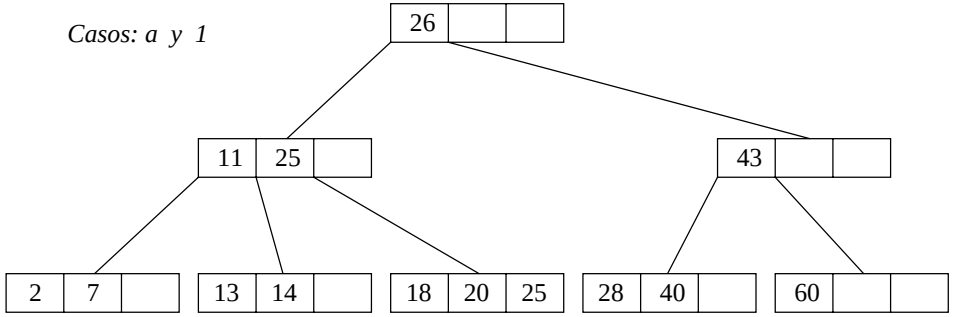
Arbol B de orden 4 inicial

1. Borrado de 81.



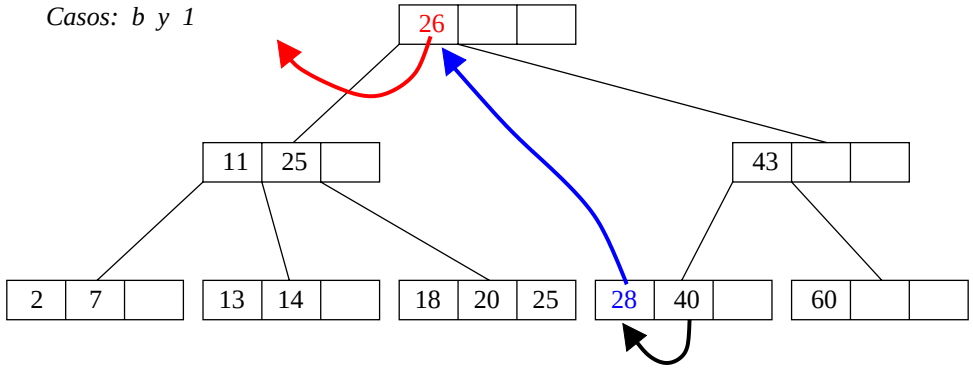
Búsqueda del nodo en el que borrar 81

Casos: a y 1

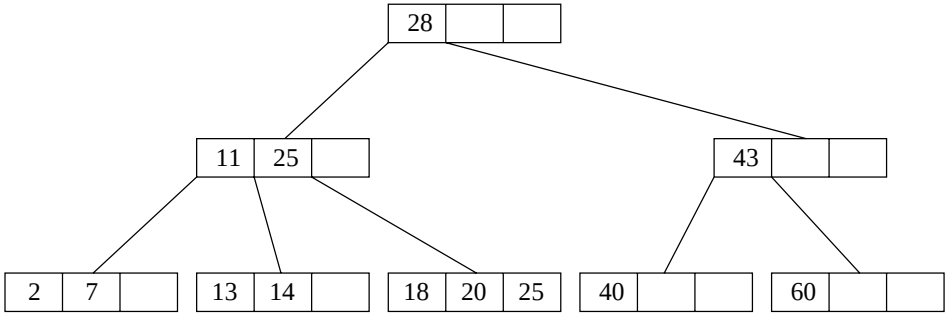


Arbol después de borrar la clave 81

2. Borrado de 26.

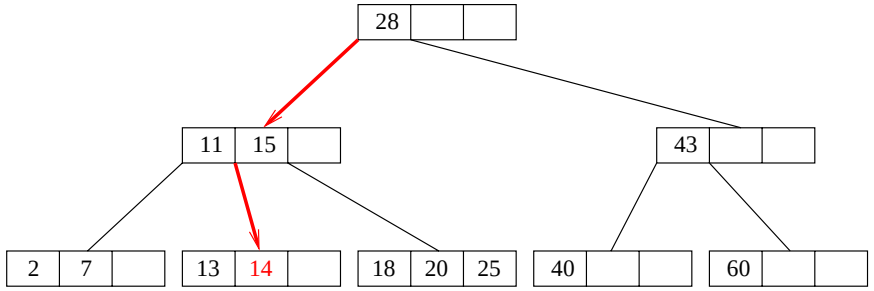


Operaciones requeridas para borrar la clave 26

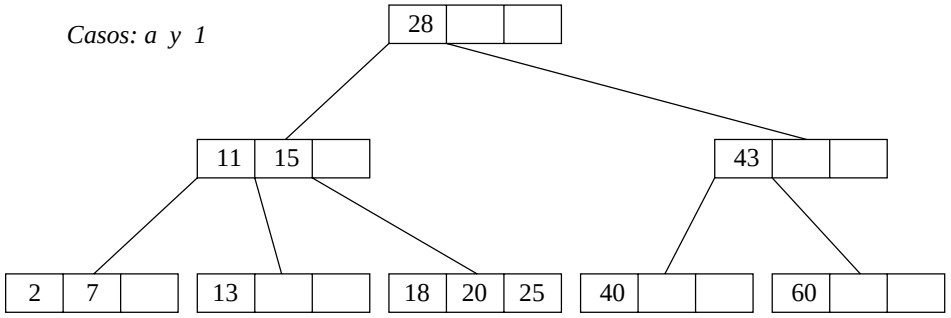


Arbol después de borrar la clave 26

3. Borrado de 14.

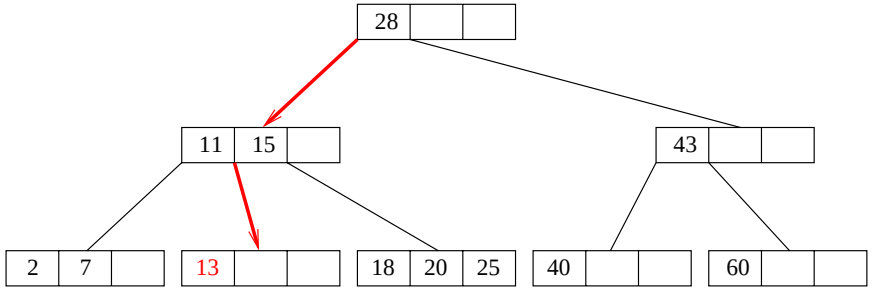


Búsqueda del nodo en el que borrar 14

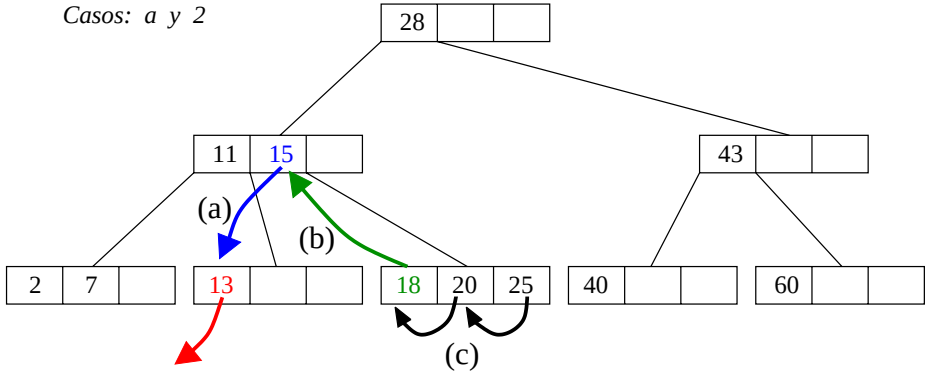


Arbol después de borrar la clave 14

4. Borrado de 13.



Búsqueda del nodo en el que borrar 13

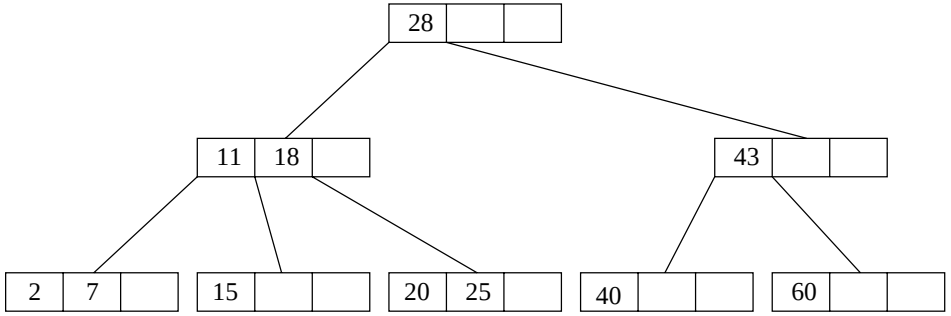


Distribución de claves en el borrado de la clave 13

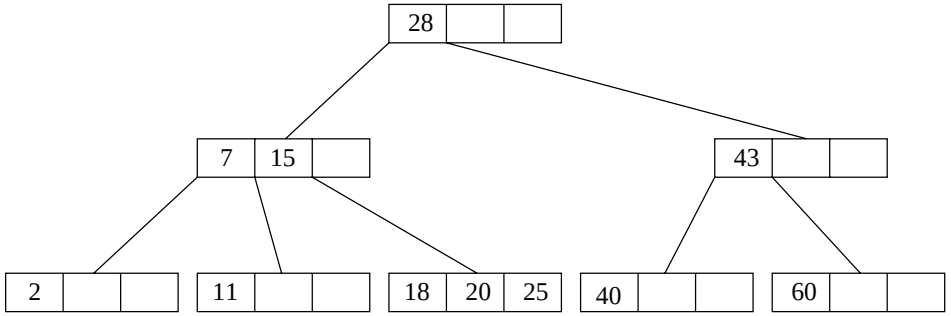
(a) La clave 15, que está en el padre de los dos nodos involucrados, y que los separa, pasa al nodo que ha quedado vacío.

(b) La clave menor del hermano a la derecha del nodo afectado, 18, pasa al nodo padre, ocupando el lugar que tenía la clave 15.

(c) Se reordenan las claves del hermano a la derecha del nodo afectado.

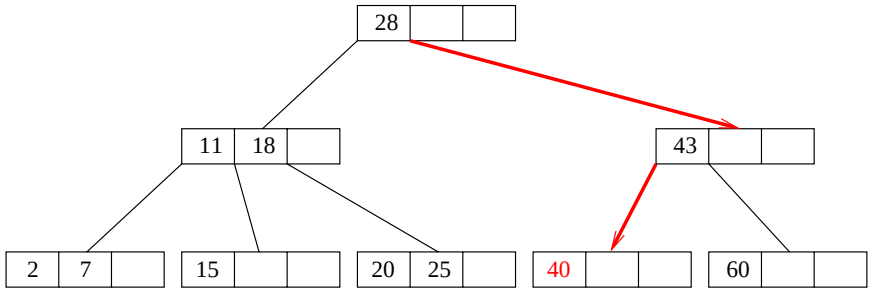


Arbol después de borrar la clave 13



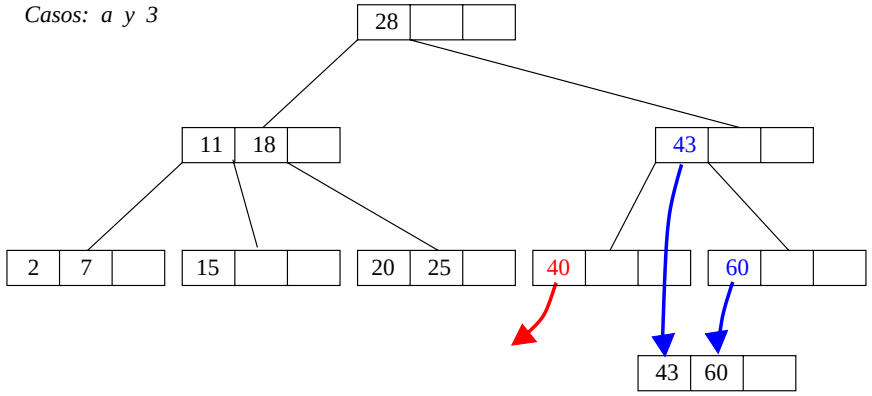
Arbol después de borrar la clave 13 (alternativa)

5. Borrado de 40.

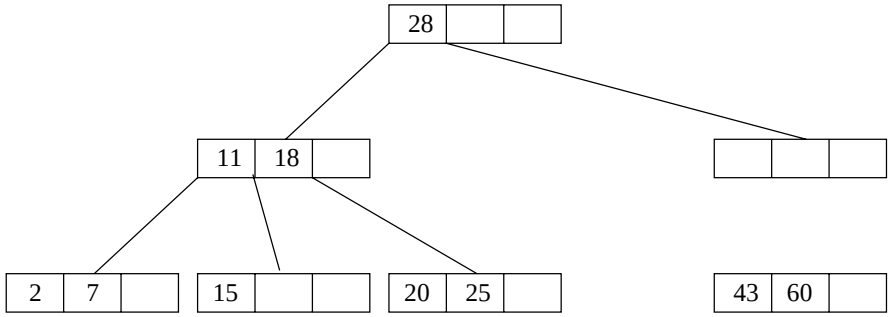


Búsqueda del nodo en el que borrar 40

Casos: a y 3



Operaciones para la unión de claves en el borrado de la clave 40



Resultado de la unión de claves

Caso: 2

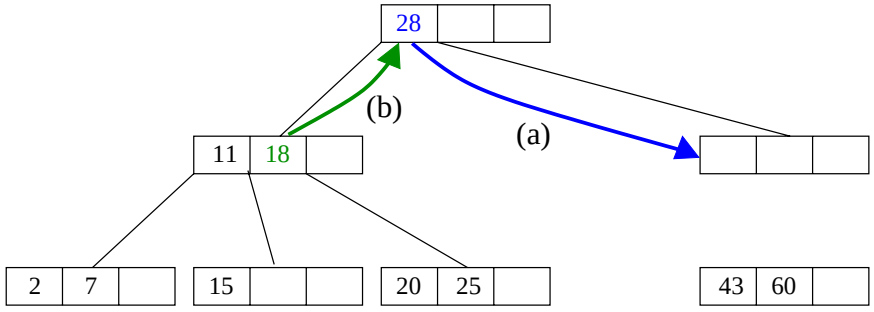
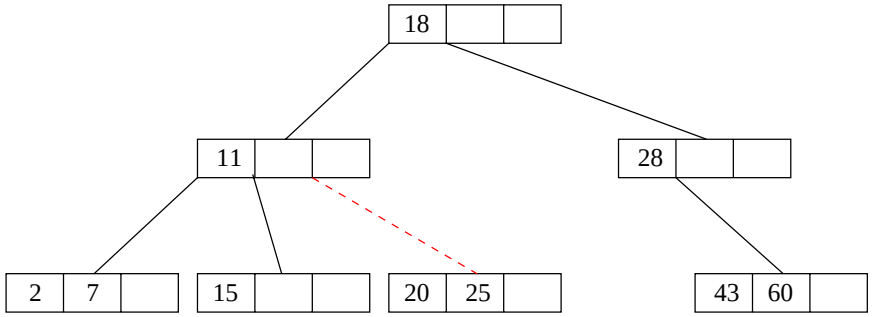


Figura 20: Distribución de claves para ajustar el nodo vacío

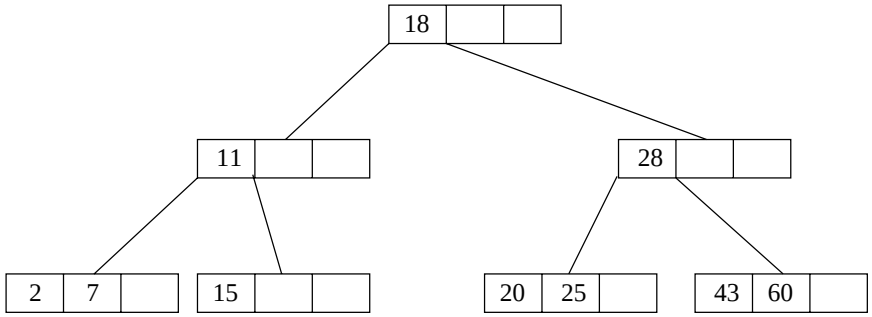
Distribución de claves para ajustar el nodo vacío

(a) La clave 28, que está en el padre de los dos nodos involucrados, y que los separa, pasa al nodo que ha quedado vacío.

(b) La clave mayor del hermano a la izquierda, 18, pasa al nodo padre, ocupando el lugar que tenía la clave 28.



Resultado de la distribución de claves

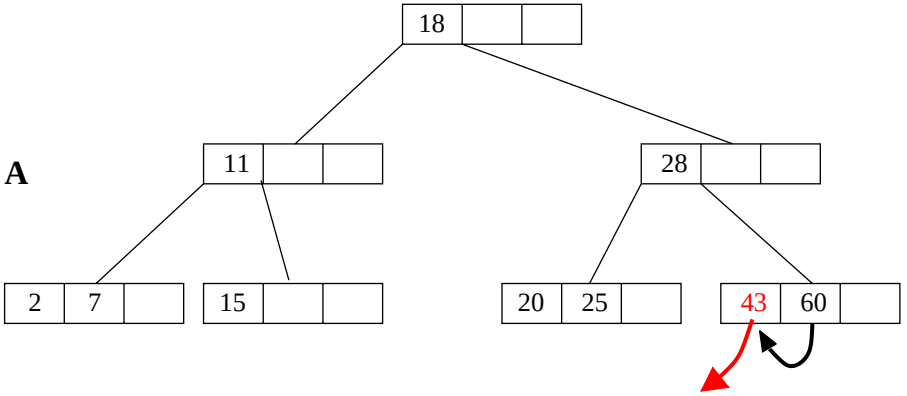


Arbol final resultante del borrado de 40

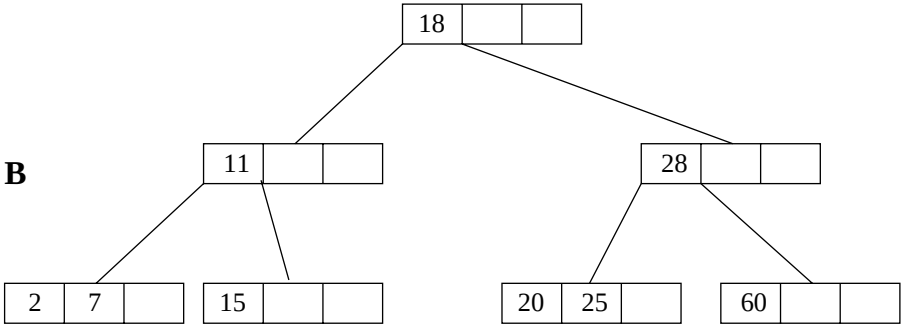
6. Borrado de 43.

Casos: a y 1

A

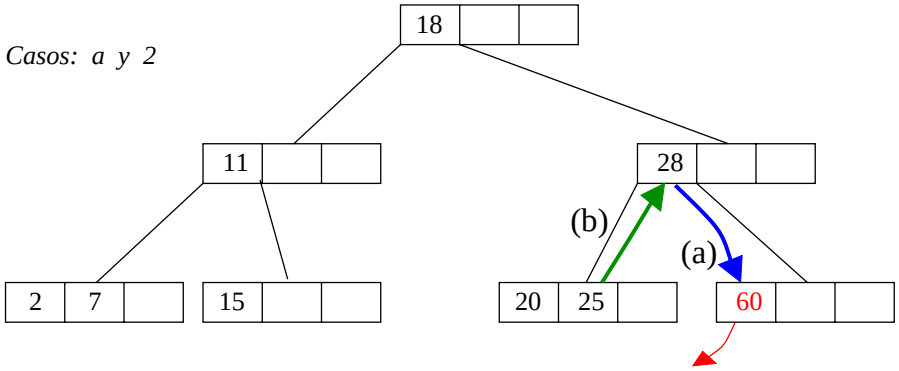


B



A) Operaciones asociadas al borrado de la clave 43.
B) Arbol final resultante del borrado de 43

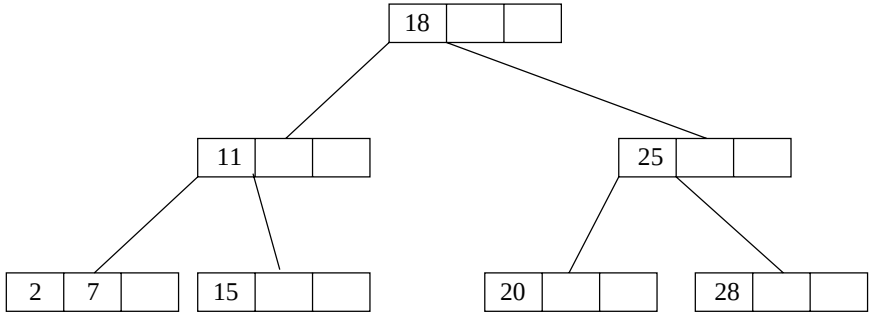
7. Borrado de 60.



Distribución de claves para el borrado de la clave 60

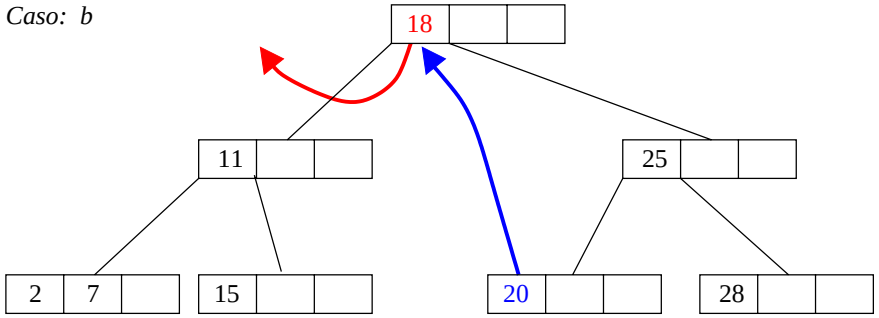
(a) La clave 28, que está en el padre de los dos nodos involucrados, y que los separa, pasa al nodo que ha quedado vacío.

(b) La clave mayor del hermano a la izquierda, 25, pasa al nodo padre, ocupando el lugar que tenía la clave 28.

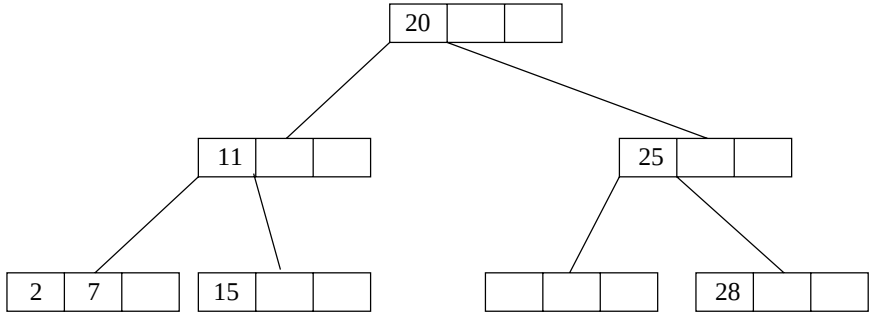


Arbol final tras el borrado de la clave 60

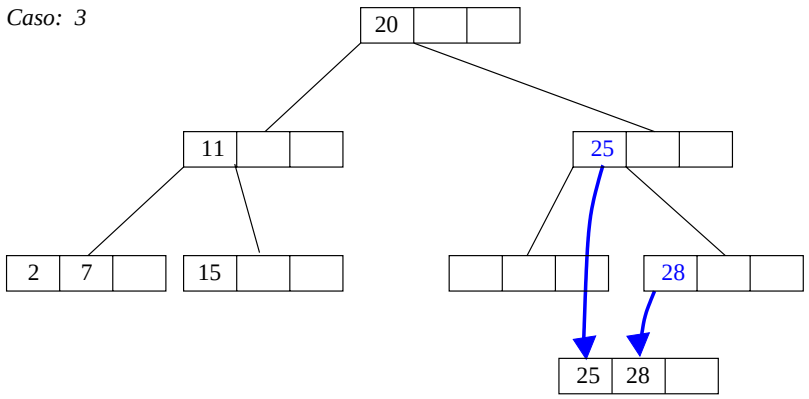
8. Borrado de 18.



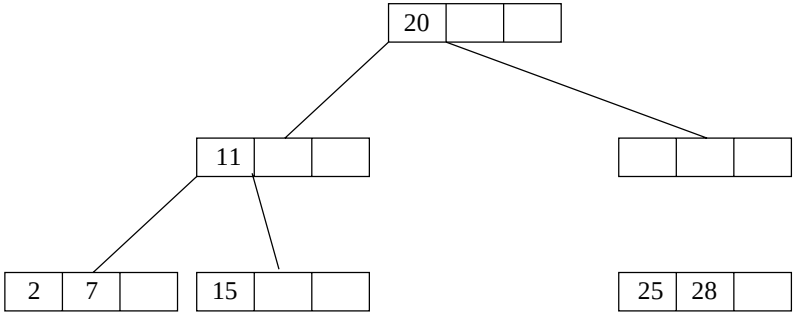
Operaciones requeridas para borrar la clave 18



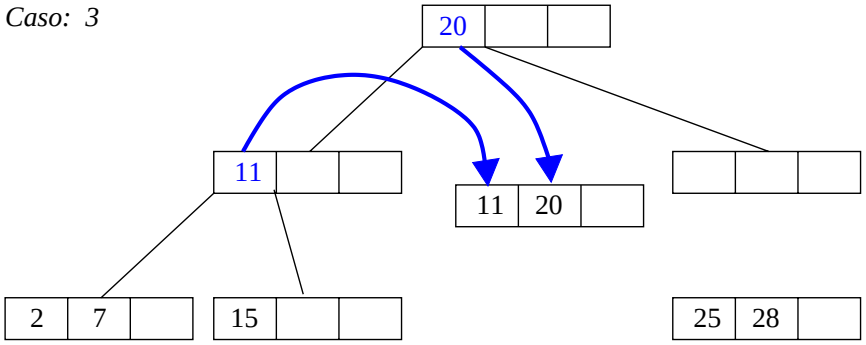
Arbol después de borrar la clave 18



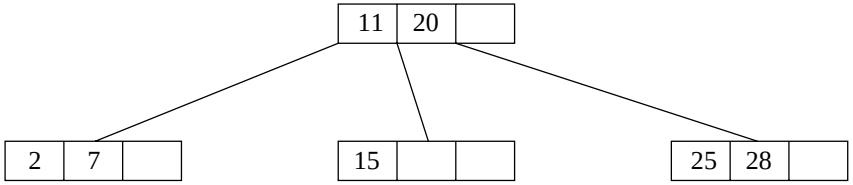
Operaciones para la unión tras el borrado de la clave 18



Resultado de la unión de claves



Operaciones para la unión por el desajuste creado en el borrado de la clave 18



Arbol final tras el borrado de la clave 18

4.5. Rendimiento de los árboles B

1. Mínimo número de nodos que tiene un árbol B de orden n y altura m.

   - *Nivel 0:*
     El número de nodos de un árbol de altura 0 es 1.
   - *Nivel 1:*
     Propiedad 2: *la raiz tiene al menos dos descendientes o es una hoja.*
     No es una hoja: el mínimo número de nodos en el nivel 1 es 2.
   - *Nivel 2:*
     Tiene una raiz a nivel 0 y al menos dos nodos internos a nivel 1.
     Propiedad 3: *cada nodo interno tiene, al menos,* $\lceil \frac{n}{2} \rceil$ *descendientes.*
     En el nivel 2 habrán, al menos, $2 \lceil \frac{n}{2} \rceil$ nodos.
   - *Nivel 3:*
     En el nivel 3 habrán, al menos,

     $$(2 \lceil \frac{n}{2} \rceil) \lceil \frac{n}{2} \rceil \;=\; 2 (\lceil \frac{n}{2} \rceil)^2 \text{ nodos}$$

   - *Nivel 4:*
     En el nivel 4 habrán, al menos,

     $$2 (\lceil \frac{n}{2} \rceil)^2 \lceil \frac{n}{2} \rceil) \;=\; 2 (\lceil \frac{n}{2} \rceil)^3 \text{ nodos}$$

De forma general, el **número mínimo de nodos en el nivel** i:

$$2 (\lceil \frac{n}{2} \rceil)^{i-1}$$

Número mínimo de nodos de un árbol de orden n **y de altura** m:

$$1 \;+\; \sum_{j=1}^{m} 2 \left(\left\lceil \frac{n}{2} \right\rceil \right)^{j-1} \qquad (1)$$

Ejemplos.

- Un árbol B de orden $n = 5$ y altura $m = 2$ contiene un mínimo de:
  $1 + 2 \times 3^0 = 3$ nodos interiores y $2 \times 3^1 = 6$ nodos hoja.
- Un árbol B de orden $n = 10$ y altura $m = 4$ contiene un mínimo de:

  $$1 \;+\; 2 \times 5^0 \;+\; 2 \times 5^1 \;+\; 2 \times 5^2 \;=\; 63 \text{ nodos interiores}$$
  $$2 \times 5^3 \;=\; 250 \text{ nodos hoja}$$

- **Incremento en altura**: Aumento del mínimo número de nodos.
  Un árbol B de orden $n = 10$ y altura $m = 5$ contiene un mínimo de:

  $$1 \;+\; 2 \times 5^0 \;+\; 2 \times 5^1 \;+\; 2 \times 5^2 \;+\; 2 \times 5^3 \;=\; 313 \text{ nodos interiores}$$
  $$2 \times 5^4 \;=\; 625 \text{ nodos hoja}$$

---

2. Mínimo número de registros indexados por un árbol B.

Debe ser una función de:

1. el mínimo número de nodos del árbol (ecuación 1), y
2. del mínimo número de claves por nodo (depende de n):
   - Nodos intermedios: $\lceil \frac{n}{2} \rceil - 1$ claves.
   - Nodo raiz y hojas: 1 clave.

Mínimo número de registros indexados por un árbol B de orden n y altura m:

$$k \;=\; 1 \;+\; \sum_{j=1}^{m-1} 2 \left(\left\lceil \frac{n}{2} \right\rceil \right)^{j-1} \left(\left\lceil \frac{n}{2} \right\rceil - 1 \right) \;+\; 2 \left(\left\lceil \frac{n}{2} \right\rceil \right)^{m-1} \qquad (2)$$

$$k \;=\; 4 \left(\left\lceil \frac{n}{2} \right\rceil \right)^{m-1} - 1 \qquad (3)$$

Ejemplos.

- Un árbol B de orden $n = 5$ y altura $m = 2$ contiene 2 nodos interiores (no contamos la raiz) y un mínimo de 6 nodos hoja y la raiz. Así, puede indexar un mínimo de:

$$k \;=\; 4 \left(\left\lceil \frac{5}{2} \right\rceil \right)^{2-1} \;-\; 1 \;=\; 4 \times 3^1 \;-\; 1 \;=\; 11 \text{ registros}$$

- Un árbol B de orden $n = 10$ y altura $m = 4$ contiene un mínimo de 62 nodos interiores (excluyendo a la raiz), 250 nodos hoja y la raiz. Este árbol puede indexar

$$k \;=\; 4 \left(\left\lceil \frac{10}{2} \right\rceil \right)^{4-1} \;-\; 1 \;=\; 4 \times 5^3 \;-\; 1 \;=\; 4 \times 125 \;-\; 1 \;=\; 499 \text{ registros}$$

- Finalmente, un árbol B de orden $n = 10$ y altura $m = 5$ puede indexar un mínimo de:

$$k \;=\; 4 \left(\left\lceil \frac{10}{2} \right\rceil \right)^{5-1} \;-\; 1 \;=\; 4 \times 5^4 \;-\; 1 \;=\; 4 \times 625 \;-\; 1 \;=\; 2499 \text{ registros}$$

- En realidad, la variable es m (se conoce k).
- El valor de n se fija de antemano para ajustar el nodo al tamaño del bloque.

$$m \;\leq\; 1 \;+\; \log_{\lceil \frac{n}{2} \rceil} \left(\frac{k+1}{4} \right) \qquad (4)$$

Ejemplos.

- La altura de un árbol B de orden $n = 50$ que debe direccionar $k = 39062499$ registros se calcula de la siguiente forma:

$$m \;=\; 1 \;+\; \log_{\lceil \frac{50}{2} \rceil} \left(\frac{39062499 + 1}{4} \right) \;=\; 6$$

- La altura de un árbol B de orden $n = 200$ para direccionar $k = 3999999$ registros se calcula de la siguiente forma:

$$m \;=\; 1 \;+\; \log_{\lceil \frac{200}{2} \rceil} \left(\frac{3999999 + 1}{4} \right) \;=\; 4$$

Frente a este tipo de índices, el uso de un árbol AVL hubiera empleado, en el peor de los casos, $\log_2 3999999 \approx 22$ consultas.

5. Árboles B$^+$

5.1. Definición y estructura

Un árbol B$^+$ puede definirse a partir de un árbol B.

1. El índice se mantiene en los nodos no terminales.
   Los nodos interiores **no** contienen referencias a dirs. en el fichero.
   - Los nodos interiores y las hojas tienen diferente estructura.
   - Se reduce la altura del árbol, por lo que la cantidad de accesos al índice es menor.

2. Las direcciones asociadas a las claves están únicamente en las hojas.
   Así, las búsquedas siempre terminan en las hojas.

3. Las claves aparecen en varios niveles del árbol.

4. Las hojas están encadenadas.
   Permite el procesamiento secuencial según el orden impuesto por la clave sobre la que se monta el índice de una forma simple.

5. El número de claves en un nodo no terminal coincide con el número de indicadores.

Un *árbol B$^+$ de orden n* se caracteriza por tener un número máximo de n descendientes (resp. claves).

Por ejemplo, un árbol B$^+$ de orden 5 se caracteriza porque la raiz puede tener entre 1 y 5 claves y los nodos intermedios y las hojas de 2 a 5 claves.

Estructura de los nodos no terminales:

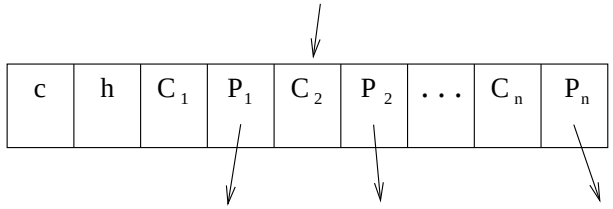


Figura 21: Estructura de un nodo no terminal de un árbol B$^+$

- $P_i \quad i = 1, 2, \ldots, n$ son **indicadores** a otro nodo en el árbol.
- $C_i \quad i = 1, 2, \ldots, n$ son las **claves** asociadas al nodo.
- c es el número de claves en el nodo.
- h es la altura del nodo en el árbol según el convenio:
  hoja: $h = 1$, padre (de una hoja): $h = 2$, ...

$$C_1 < C_2 < \ldots < C_n \quad \text{y} \quad \overline{P_i} \leq C_i \quad i = 1, 2, \ldots, c$$

Además, C_i **es el valor de la mayor clave del nodo referenciado por** P_i.

Estructura de los nodos hoja:

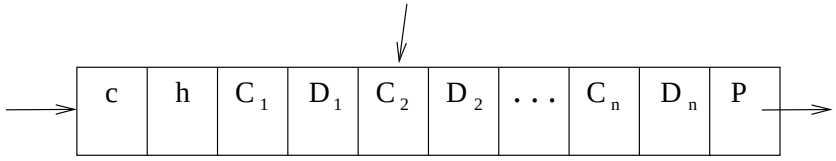


Figura 22: Estructura de un nodo hoja de un árbol B$^+$

- $D_i \quad i = 1, 2, \ldots, n$ son **direcciones** sobre el fichero de datos.
- P: Indicador para acceder a la siguiente hoja.
- c es el número de claves en la hoja.
- h es la altura del nodo ($h = 1$).

Ejemplo

Arbol B^+ de orden 4 que indexa sobre un fichero de 13 registros.

Raiz: entre 1 y 4 claves. Nodos intermedios y hojas: entre 2 y 4 claves.

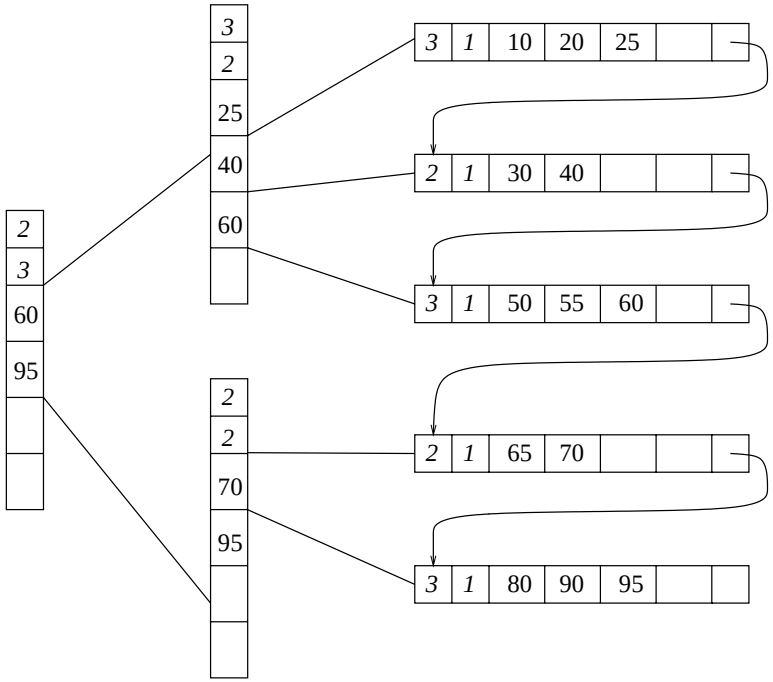


Figura 23: árbol B^+ de orden 4 que indexa un fichero de 13 registros

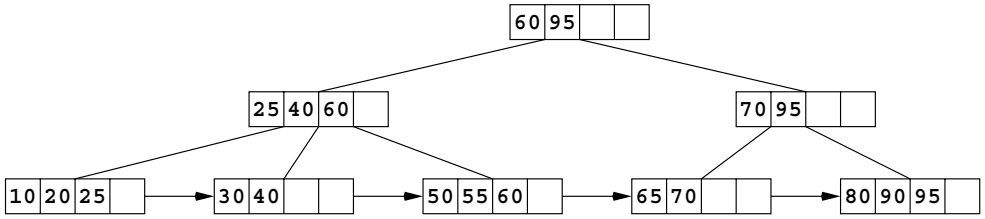


Figura 24: Representación simplificada de un árbol B^+

5.2. Inserción de claves en árboles B^+

Algoritmo de inserción en un árbol B^+

1. Buscar la hoja donde debería insertarse el valor x.

2. Comprobar si la hoja está llena.

   (a) Si no está llena,

   1. Insertar la clave de forma ordenada.

   2. Si la clave insertada no es la mayor de la hoja (**Caso 1**),
      Terminar.

      Si es la mayor (**Caso 2**),
      Actualizar la clave más a la derecha del nodo padre, fijándola al valor x y repetir este proceso hasta donde sea necesario.

      Este proceso puede propagarse al nodo raiz, pero en ningún caso se modifica la altura del árbol.

   (b) Si está llena,

   1. Crear un nuevo nodo.

   2. Redistribuir las claves entre los dos nodos.

   3. Actualizar las claves del nodo padre (tener en cuenta que hay que insertar una nueva clave en éste).

   4. Si el nodo padre no está lleno (**Caso 3**),
      Terminar.

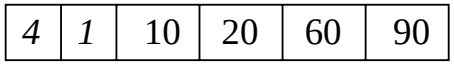
      Si está lleno (**Caso 4**),
      Repetir los pasos anteriores para el nodo padre, dividiendo el nodo, distribuyendo las claves y actualizando las claves de su inmediato antecesor.

      Este proceso puede afectar al nodo raiz, de forma que si está lleno se divide y se crea una nueva raiz, aumentando la altura del árbol en una unidad.

Ejemplo.

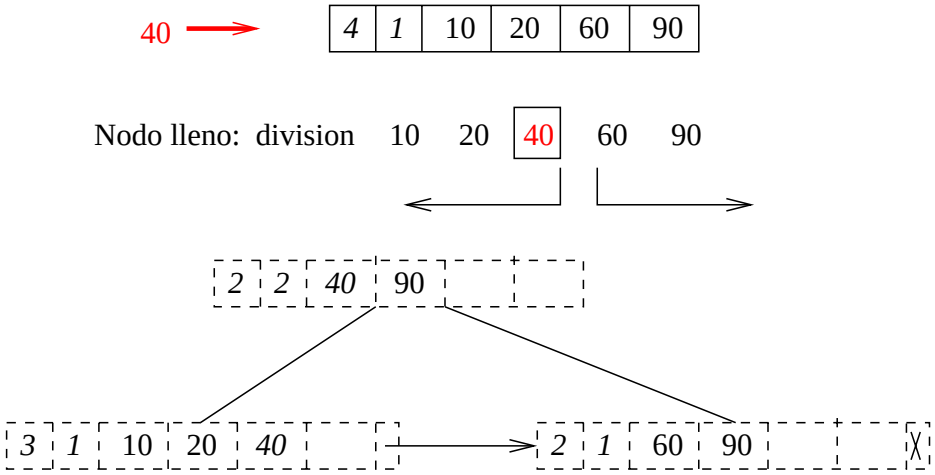
Se utilizará un árbol B^+ de orden 4.

1. Inserción de 20, 10, 90 y 60.



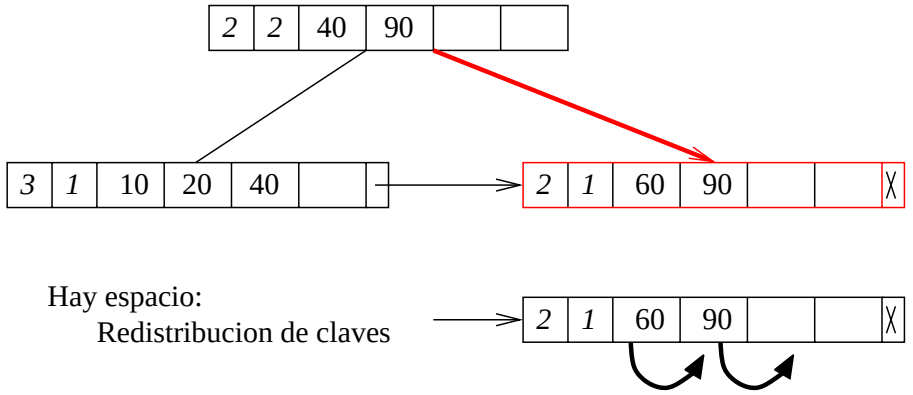
Arbol B^+ tras la inserción de 20, 10, 90 y 60

2. Inserción de 40.

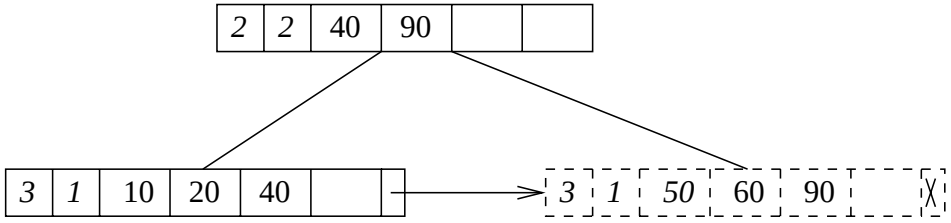


Arbol B^+ tras la inserción de 40

3. Inserción de 50.

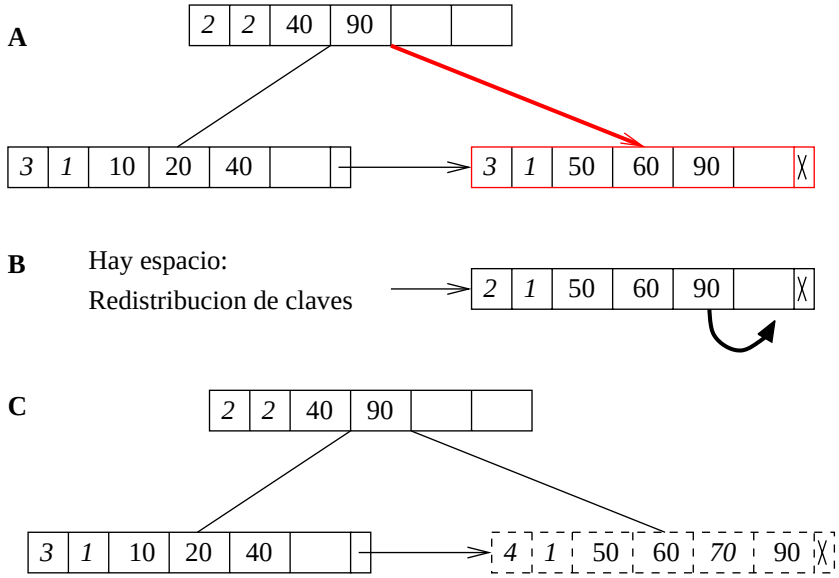


Procedimiento de inserción de la clave 50



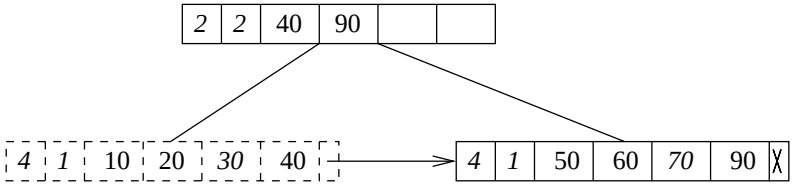
Arbol B^+ tras la inserción de 50

4. Inserción de 70.



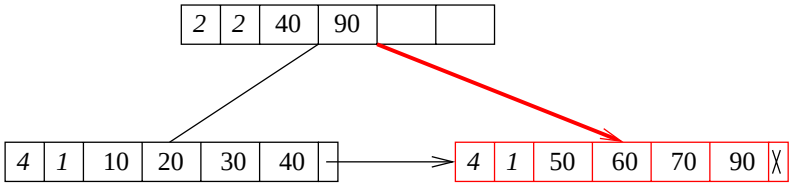
Inserción de la clave 70. A) Camino de búsqueda. B) Redistribución de claves. C) Arbol final tras insertar 70.

5. Inserción de 30.

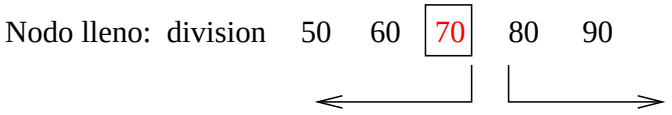


Arbol B$^+$ tras la inserción de 30

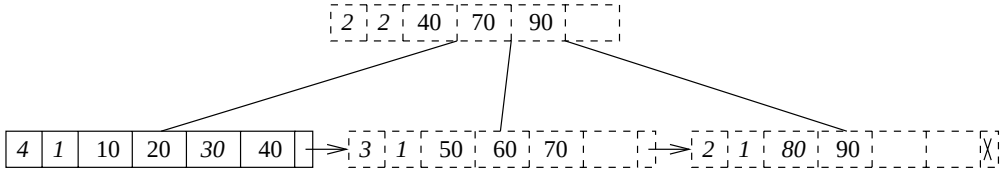
6. Inserción de 80.



Camino de búsqueda seguido para insertar la clave 80

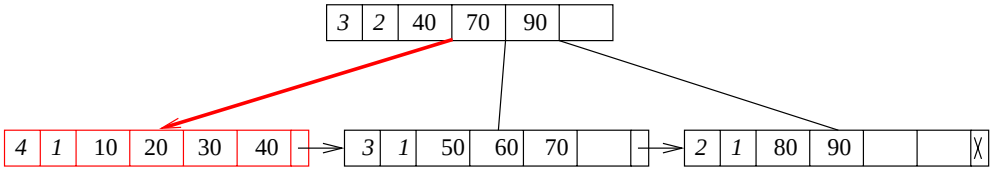


División del nodo hoja

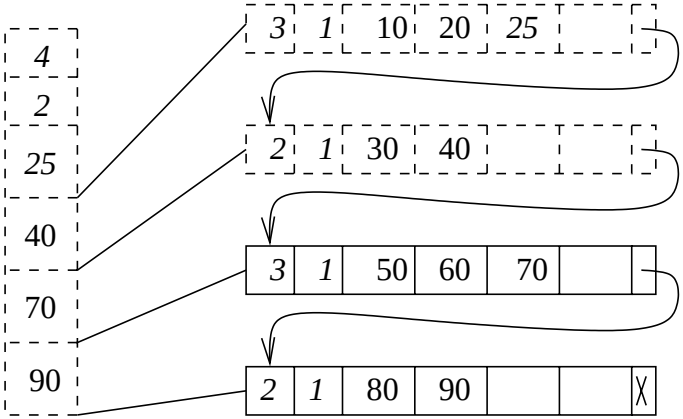


Arbol B$^+$ tras la inserción de 80

7. Inserción de 25.

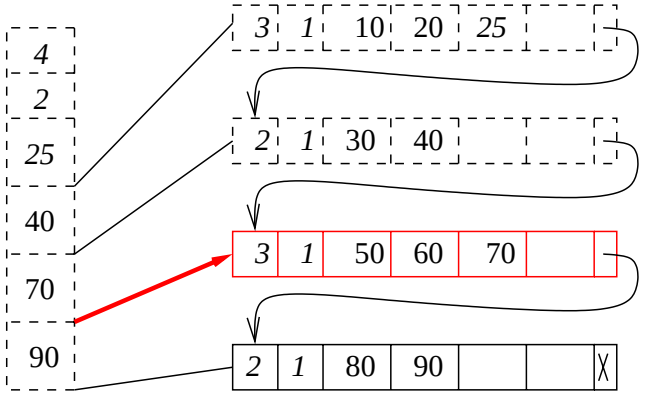


Camino de búsqueda seguido para insertar la clave 25

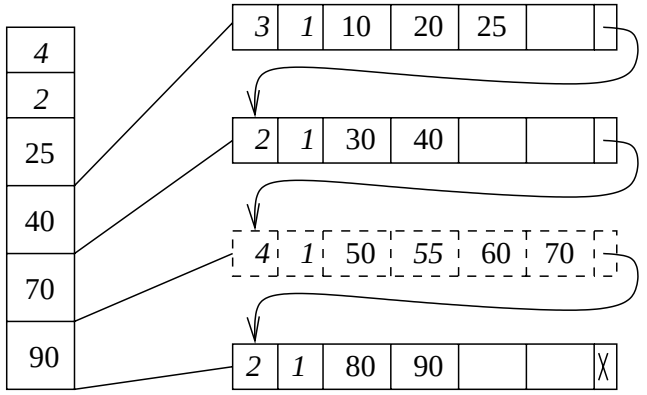


Arbol B^+ tras la inserción de 25

8. Inserción de 55.

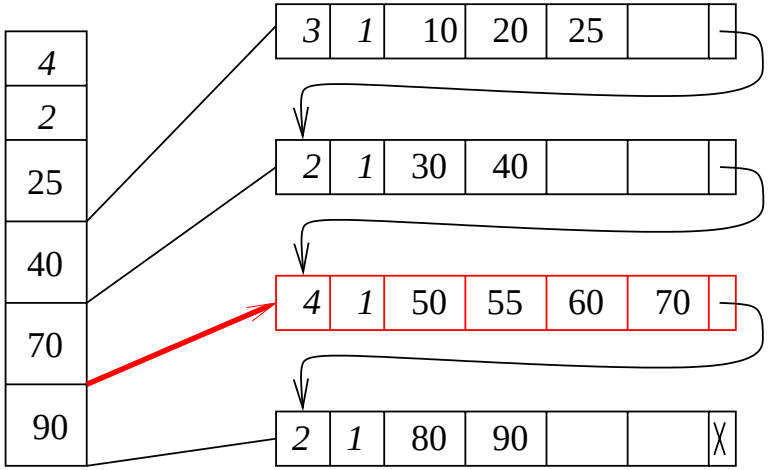


Camino de búsqueda seguido para insertar la clave 55

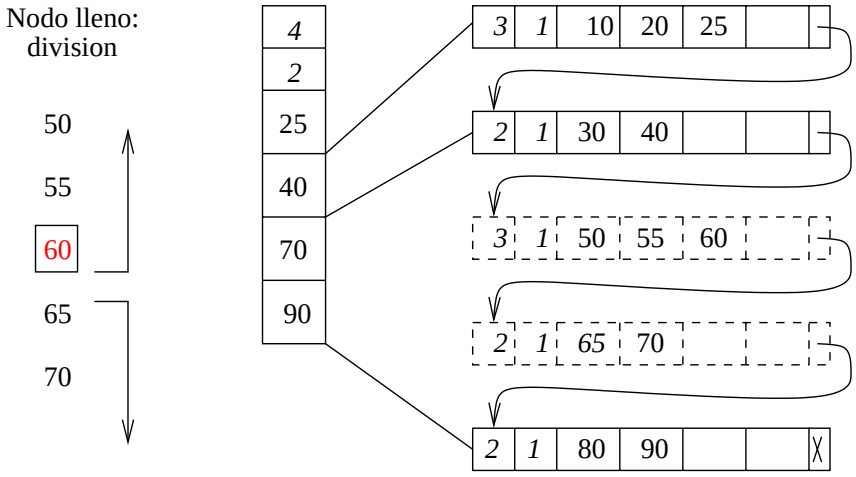


Arbol B^+ tras la inserción de 55

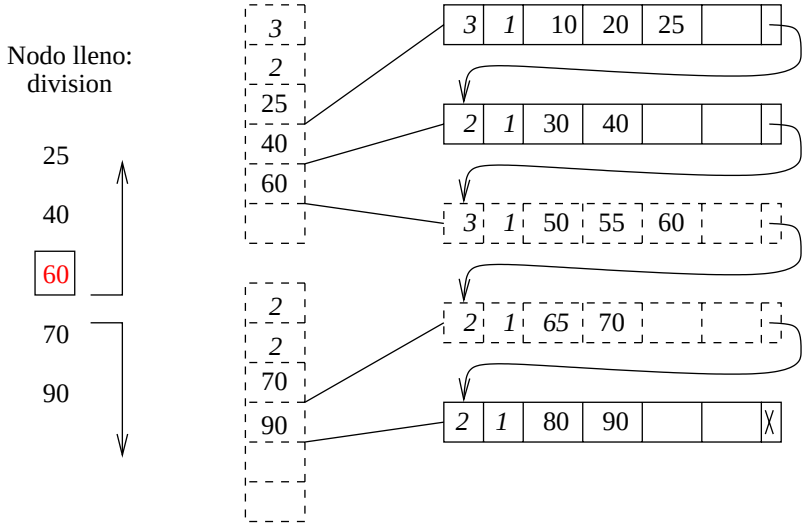
9. Inserción de 65.



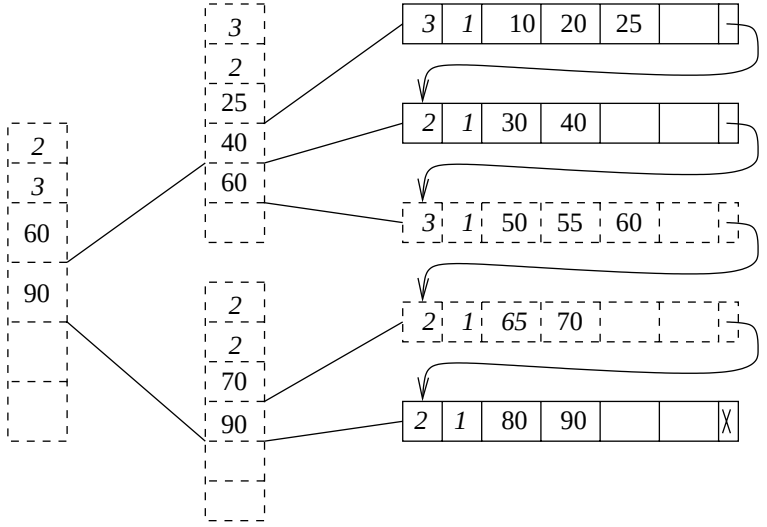
Camino de búsqueda seguido para insertar la clave 65



Partición del nodo hoja y redistribución de claves

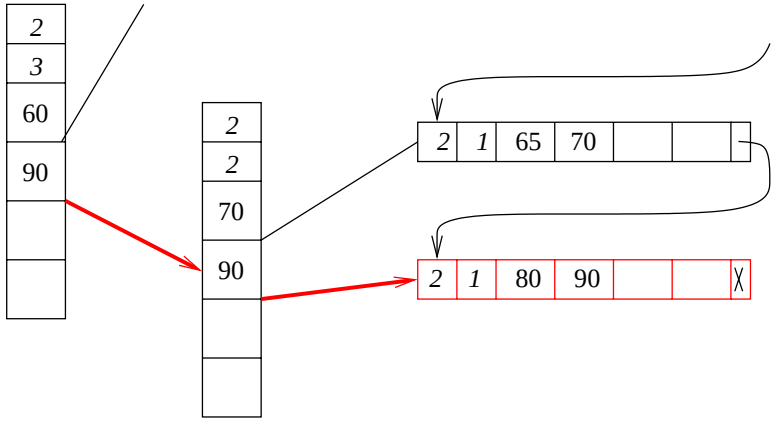


Partición del nodo padre (raiz) y redistribución de claves

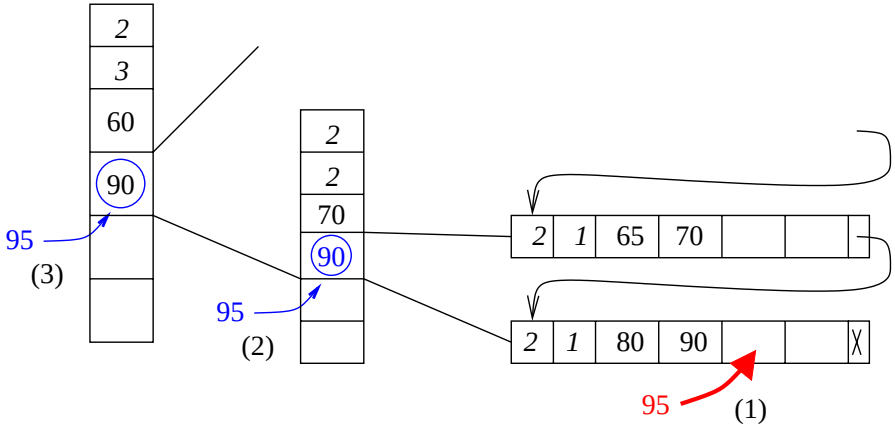


Arbol B^+ resultante de la inserción de 65

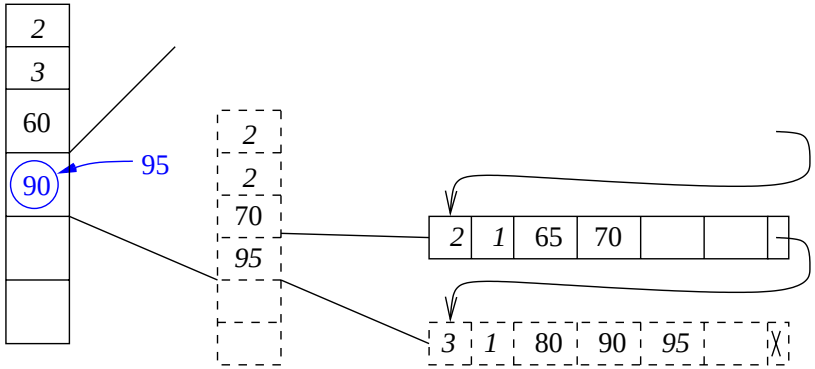
10. Inserción de 95.



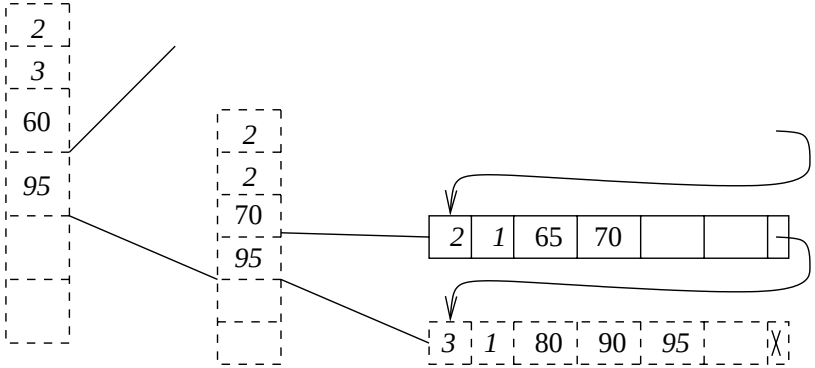
Camino de búsqueda seguido para insertar la clave 95



Inserción de la clave 95 y modificación requerida en el nodo padre



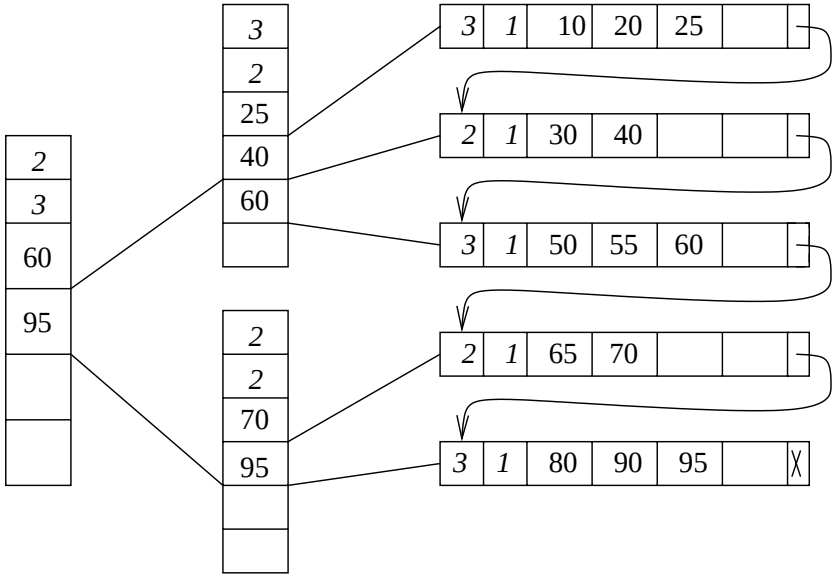
Modificación requerida en el nodo raiz



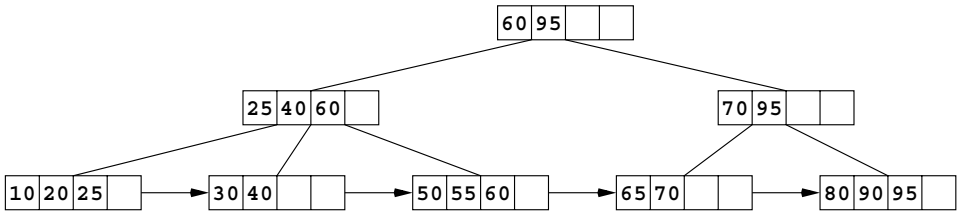
Subárbol final, tras realizar todas las modificaciones

Arbol B^+ final.

Creado partiendo de un árbol vacío, tras la inserción de las claves 20, 10, 90, 60, 40, 50, 70, 30, 80, 25, 55, 65 y 95, en este orden.



Arbol B^+ final



Representación simplificada del árbol B^+ final