Árboles binarios equilibrados
Árboles B

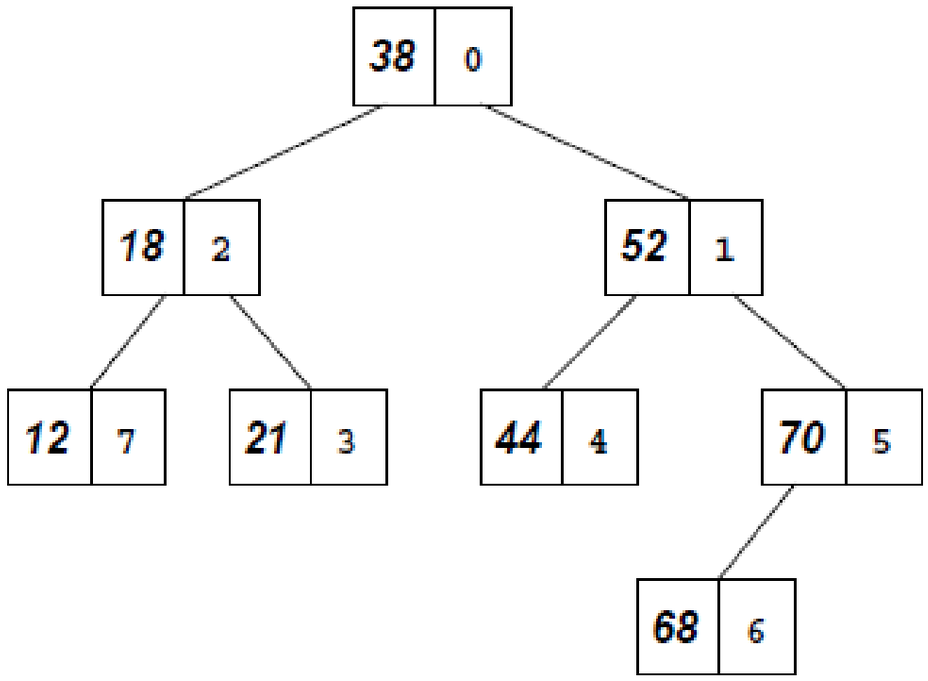
Joaquín Fernández-Valdivia
Javier Abad
Dpto. de Ciencias de la Computación e Inteligencia Artificial
Universidad de Granada

DECSAI

Árboles B

- Surgieron en 1972, creados por R. Bayer y E. McCreight
- Se originaron para aliviar el problema de la lentitud de los discos de la época y las grandes bases de datos
- Se pretende aprovechar la gran capacidad de almacenamiento para mantener una cantidad de información muy alta organizada de forma que el acceso a una clave sea lo más rápido posible

Árboles B



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

Árboles B

- ¿Por qué no usar árboles AVL?

- En memoria interna el tiempo de acceso a *n* datos situados en distintas partes de la memoria es independiente de las direcciones que ocupen (*n* * *cte*, siendo *cte* el tiempo de acceso a 1 dato)
- En memoria externa es fundamental acceder a datos situados en un mismo bloque para que el tiempo de ejecución disminuya
- Si disminuimos el número de accesos a disco, el tiempo de ejecución se verá reducido
- Uno de los factores fundamentales a la hora de construir una ED sobre disco es el número total de accesos
- Solo tenemos 2 caminos

Árboles B

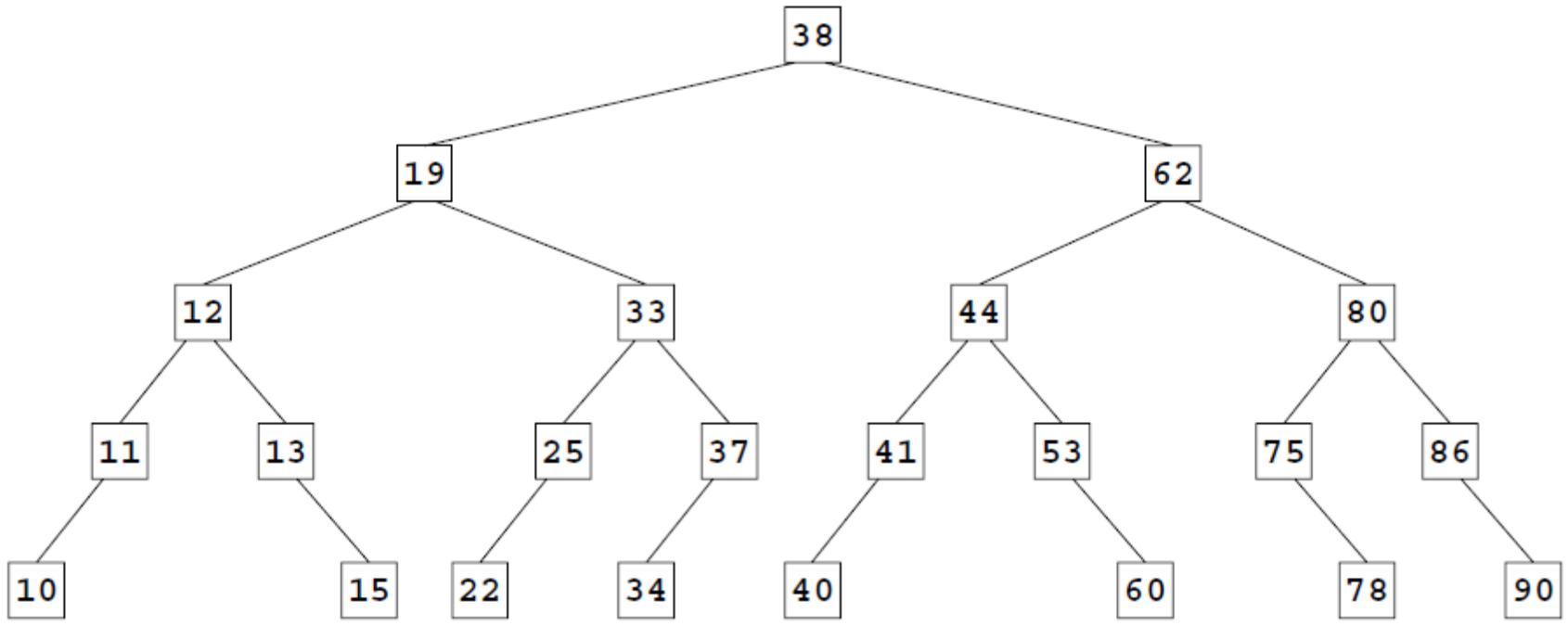
- Si construimos un ABB equilibrado en disco, los accesos serán para cargar en memoria uno de los nodos, es decir, para poder llevar a memoria una cantidad de información suficiente como para poder decidir entre dos ramas

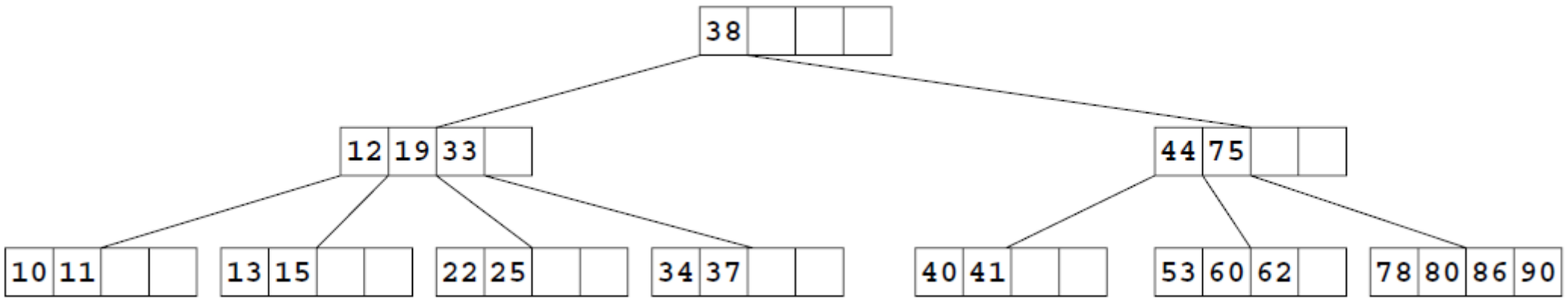
- Los árboles de múltiples ramas tienen una altura menor que los ABB, pues pueden contener más de dos hijos por nodo, además de que puede hacerse corresponder los nodos con las *páginas* en disco, de forma que al realizar un único acceso, se lee un alto número de datos que permiten elegir un camino de búsqueda no entre dos ramas, sino en un número mucho mayor

- Además, es más fácil y menos costoso equilibrar el árbol

Definición

- Árboles cuyos nodos pueden tener un número múltiple de hijos





Definición

- Un árbol B es de orden *m* si sus nodos pueden contener hasta un máximo de *m* hijos

- El conjunto de claves que se sitúan en un nodo cumplen la condición:

$$K_1 < K_2 < \ldots < K_{m-1}$$

- Los elementos que cuelgan del primer hijo tienen una clave menor que K_1 y menor que K_2 , etc…

Definición

- Para que un árbol sea B deberá cumplir lo siguiente:
  - Cada nodo tiene como máximo *m* descendientes
  - La raíz tiene al menos dos descendientes o es una hoja
  - Cada nodo interno tiene al menos $\left\lceil \frac{m}{2} \right\rceil$ descendientes
  - Un nodo no hoja con *k* descendientes tiene *k-1* claves
  - 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:

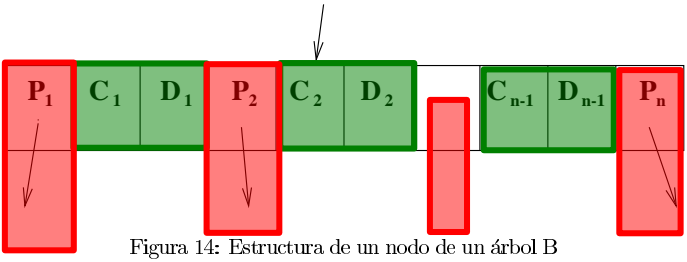


Figura 14: Estructura de un nodo de un árbol B

- $P_i \quad i = 1, 2, \ldots, n$ 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.

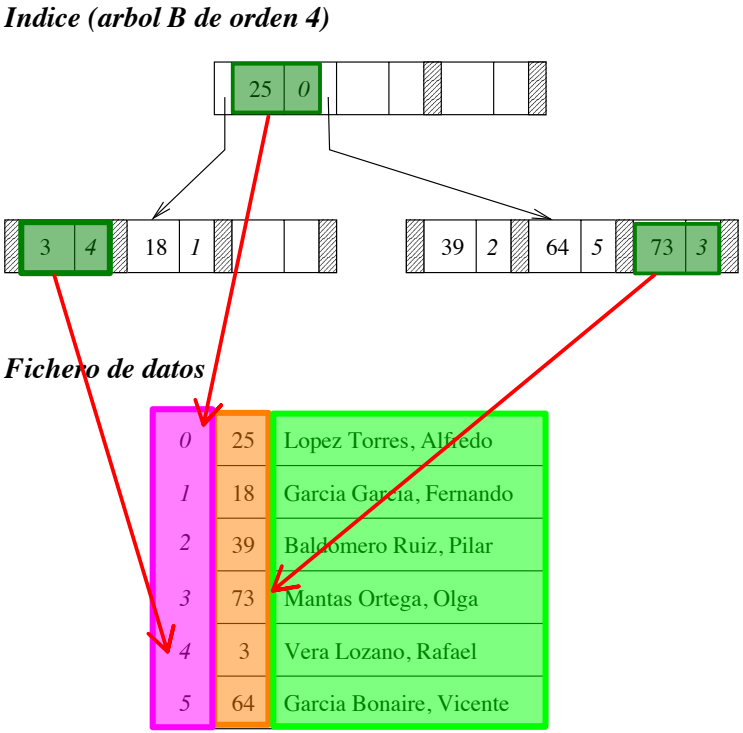


Figura 15: árbol B de orden 4 que indexa un fichero de 6 registros

Ejemplo 2

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

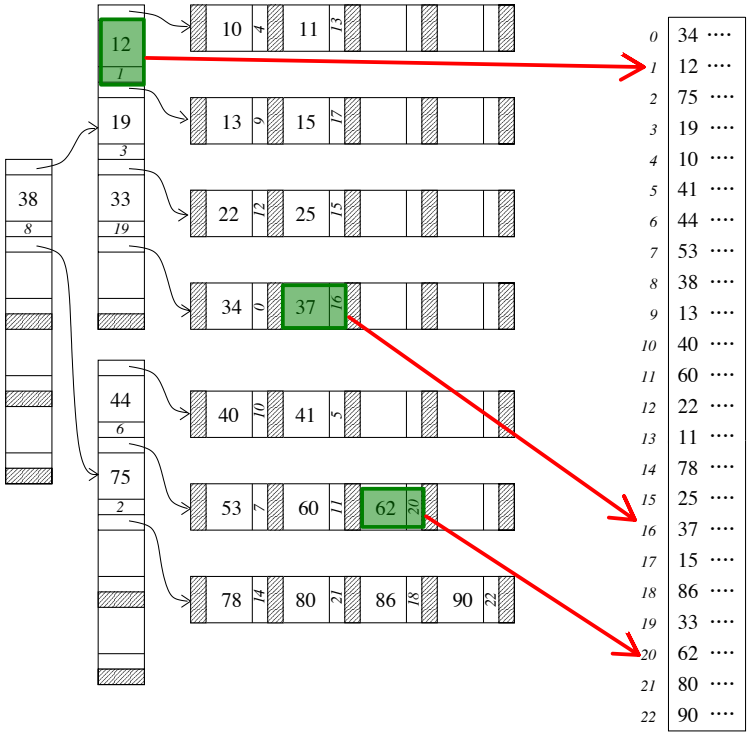


Figura 16: árbol B de orden 5 que indexa un fichero de 23 registros

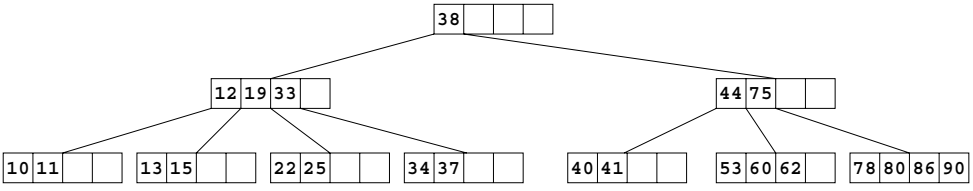


Figura 17: Representación simplificada del árbol B anterior

Búsqueda

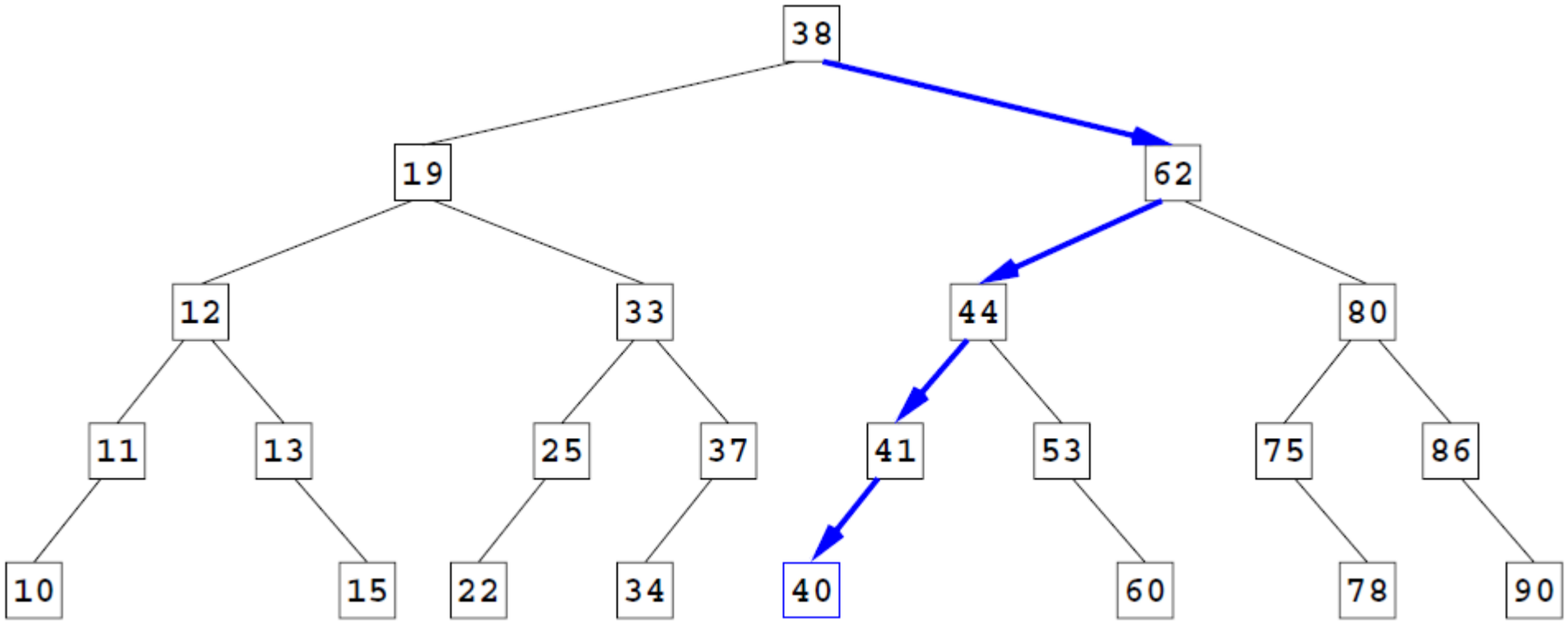


Figura 11: Búsqueda de la clave 40 en un AVL

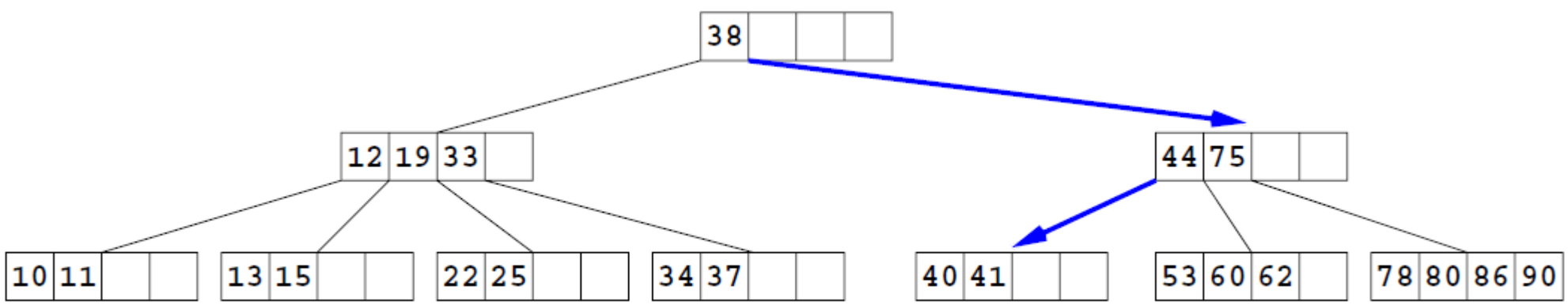


Figura 12: Búsqueda de la clave 40 en un árbol B de orden 5

Búsqueda

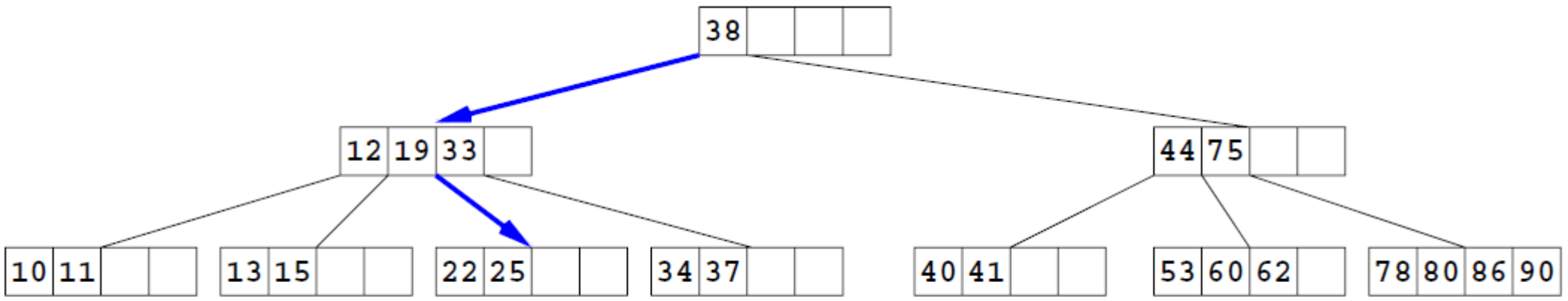


Figura 13: Búsqueda de la clave 25 en un árbol B de orden 5

Búsqueda

- Localizar una clave es una operación simple en un árbol B
- Proceso:
  1. Seleccionar como nodo actual la raíz del árbol
  2. Comprobar si la clave se encuentra en el nodo actual:
     a) Si la clave está, fin
     b) Si la clave no está:
        - Si estamos en una hoja, no se encuentra la clave, fin
        - Si no estamos en una hoja, hacer nodo actual al hijo que corresponde según el valor de la clave a buscar y los valores de las claves del nodo actual* y volver al segundo paso

* Si buscamos la clave K en un nodo con n claves: el hijo izquierdo si $K < K_1$, el hijo derecho si $K > K_n$ y el hijo i-ésimo si $K_i < K < K_{i+1}$

Inserción

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
      i. Crear un nuevo nodo
      ii. Redistribuir las claves entre los dos nodos y el nodo padre, de forma que la mediana se promociona al nodo padre
      iii. 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 raíz, de forma que si está lleno se divide y se crea una nueva raíz con una sola clave, aumentando la altura del árbol en una unidad*

Inserción

- Podemos realizar una modificación al algoritmo de forma que se retrase al máximo el momento de romper un nodos en dos

- Esto conllevaría dos beneficios:

  - Los nodos del árbol están más llenos, lo que tiene un menor gasto de memoria para mantener la estructura

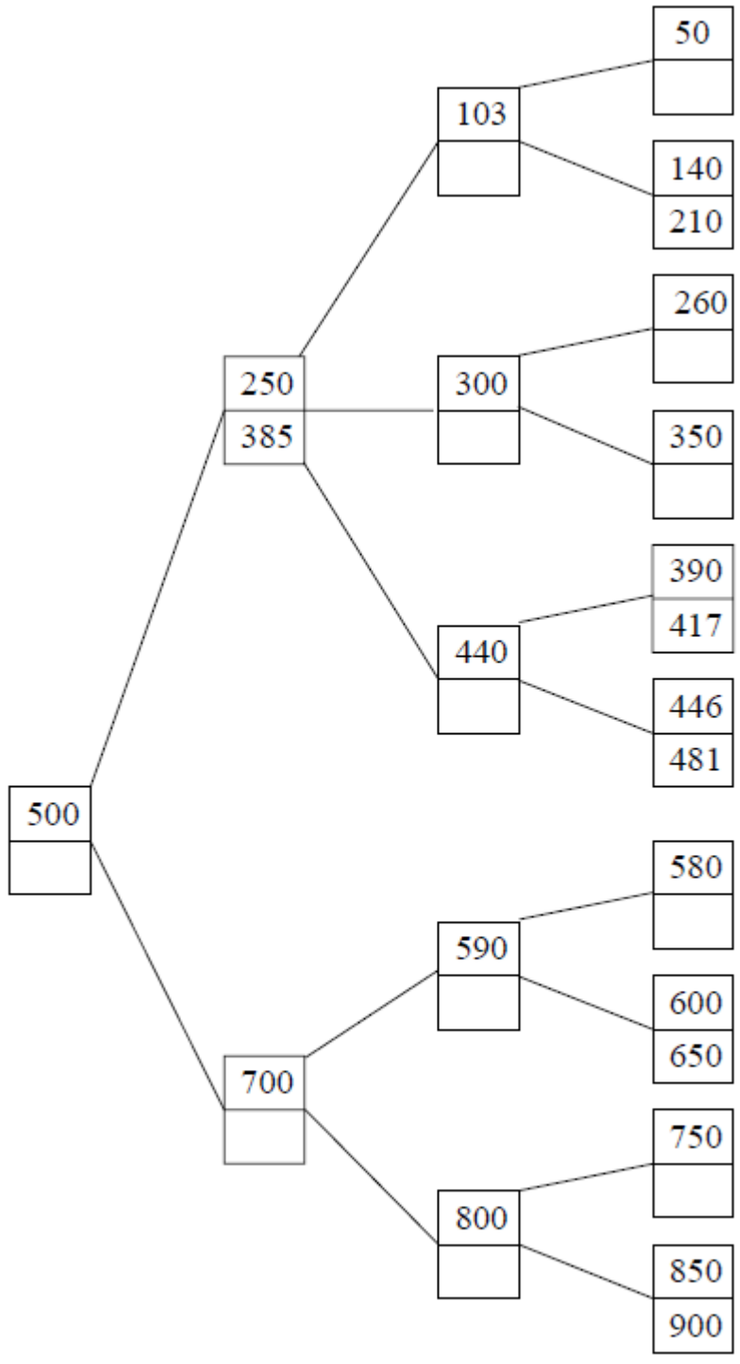
  - Retrasamos el momento en que la raíz llega a dividirse y, por consiguiente, el momento en que la altura del árbol aumenta

Inserción

Proceso:

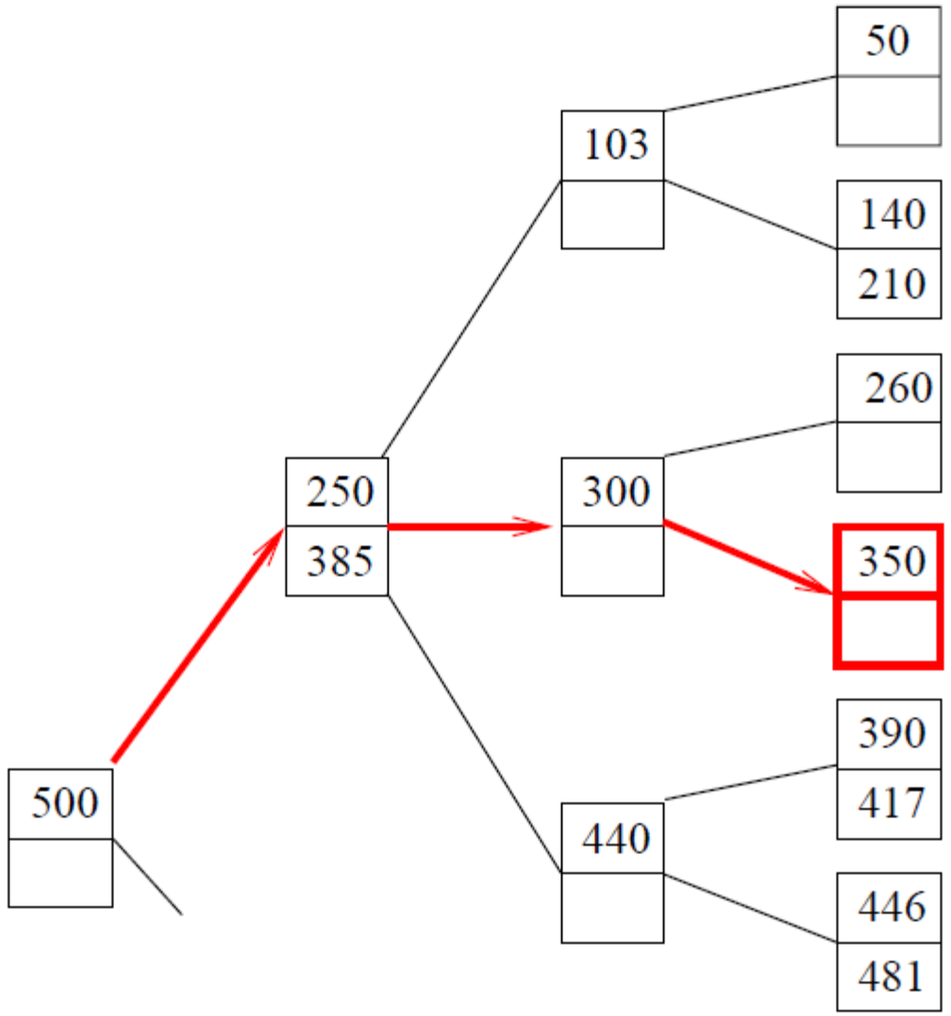
- Comprobar, antes de dividir un nodo, si algún hermano adyacente tiene huecos
- Si los tiene, se trata de redistribuir las claves de éste, más la clave del padre que las separa, más la clave a insertar
- La redistribución consiste en promocionar al nodo padre la mediana de todas las claves involucradas

Inserción



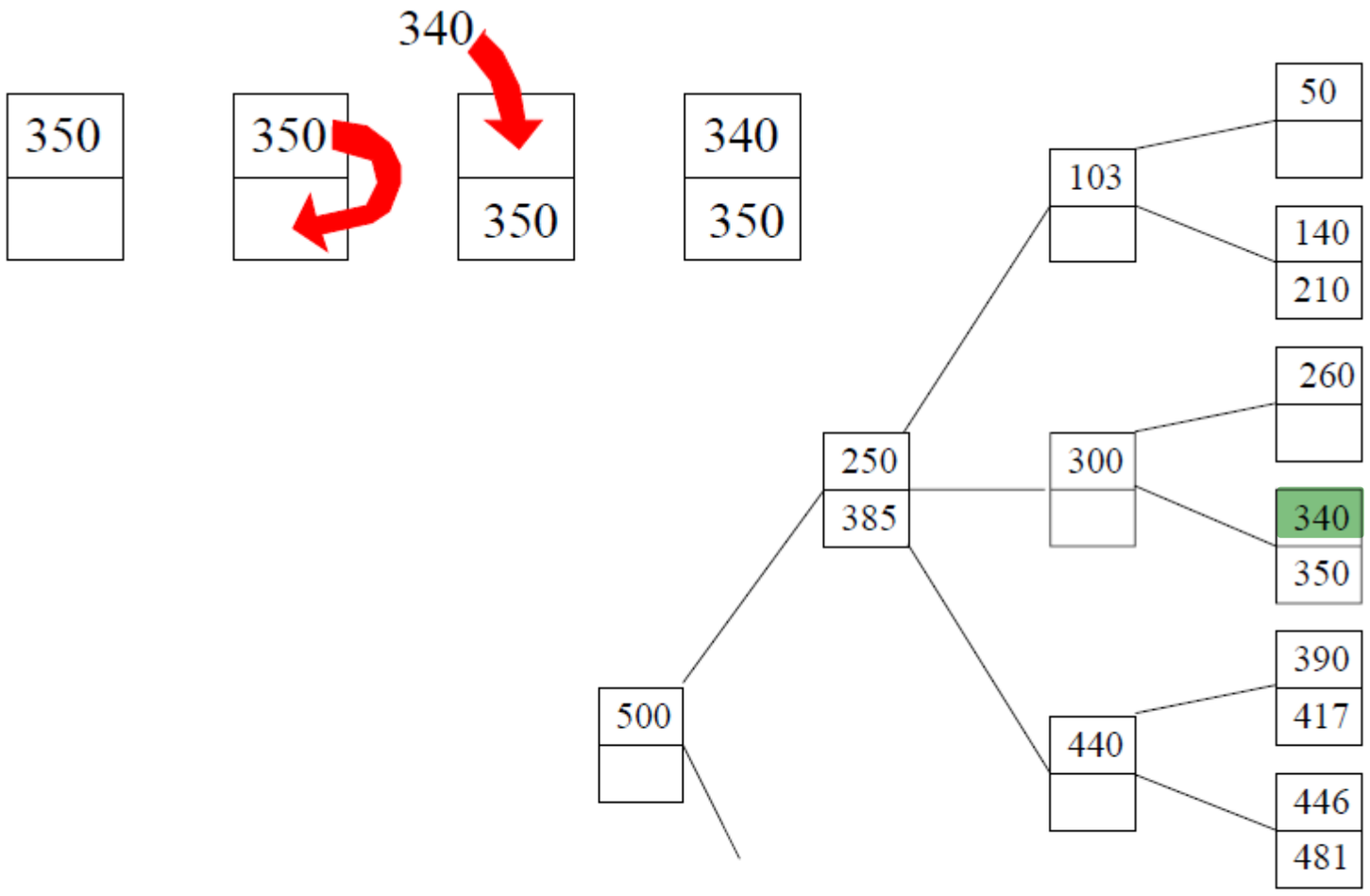
ARBOL B DE ORDEN 3 INICIAL

Inserción



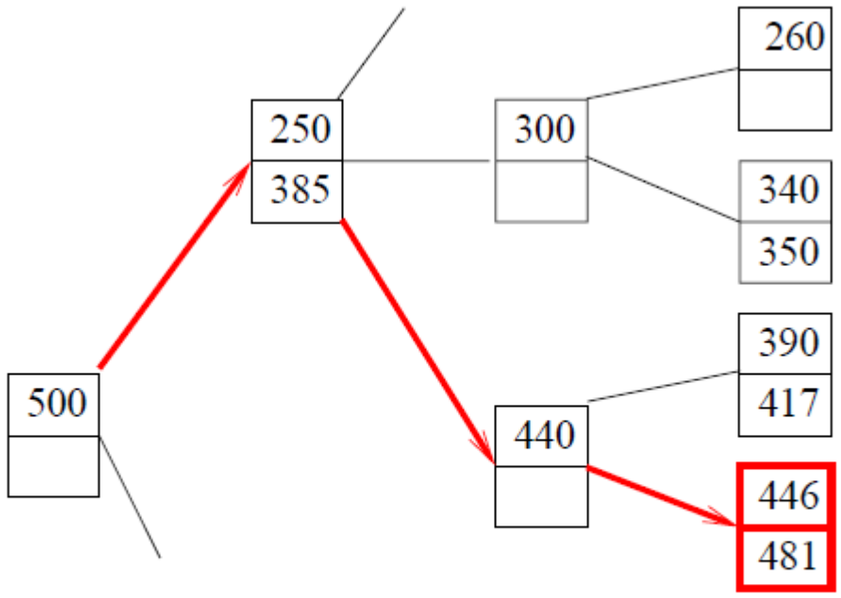
Búsqueda de la hoja en la que insertar 340

Inserción



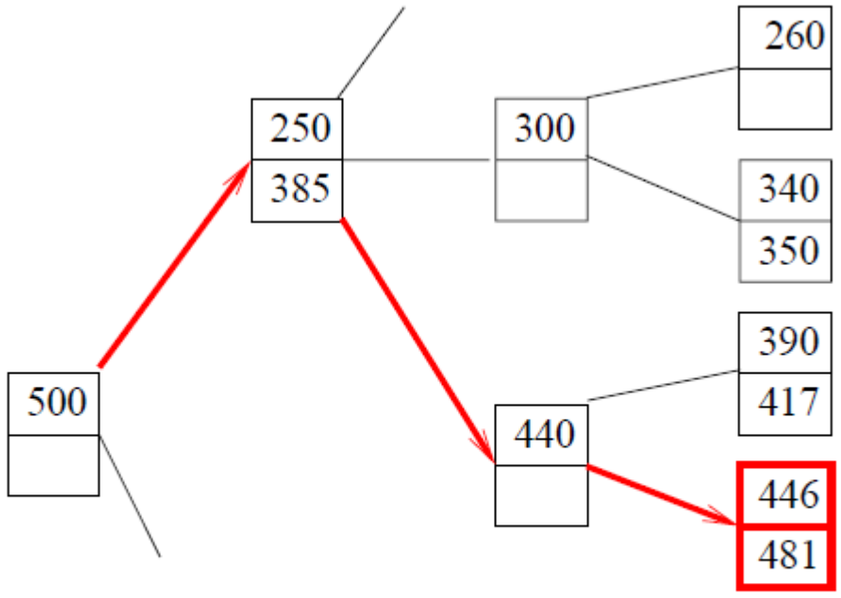
INSERCIÓN ORDENADA EN LA HOJA Y ESTADO FINAL DEL SUBÁRBOL

Inserción

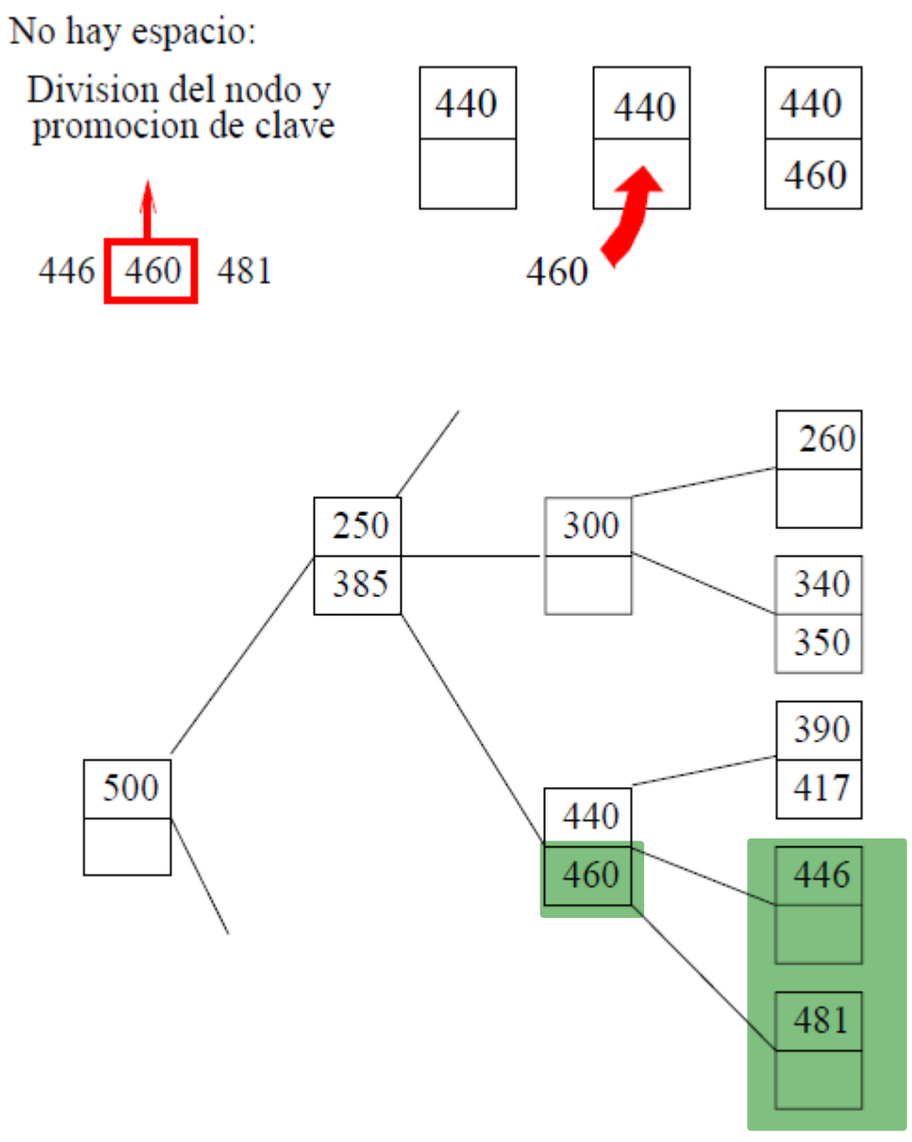


Búsqueda de la hoja en la que insertar 460

Inserción

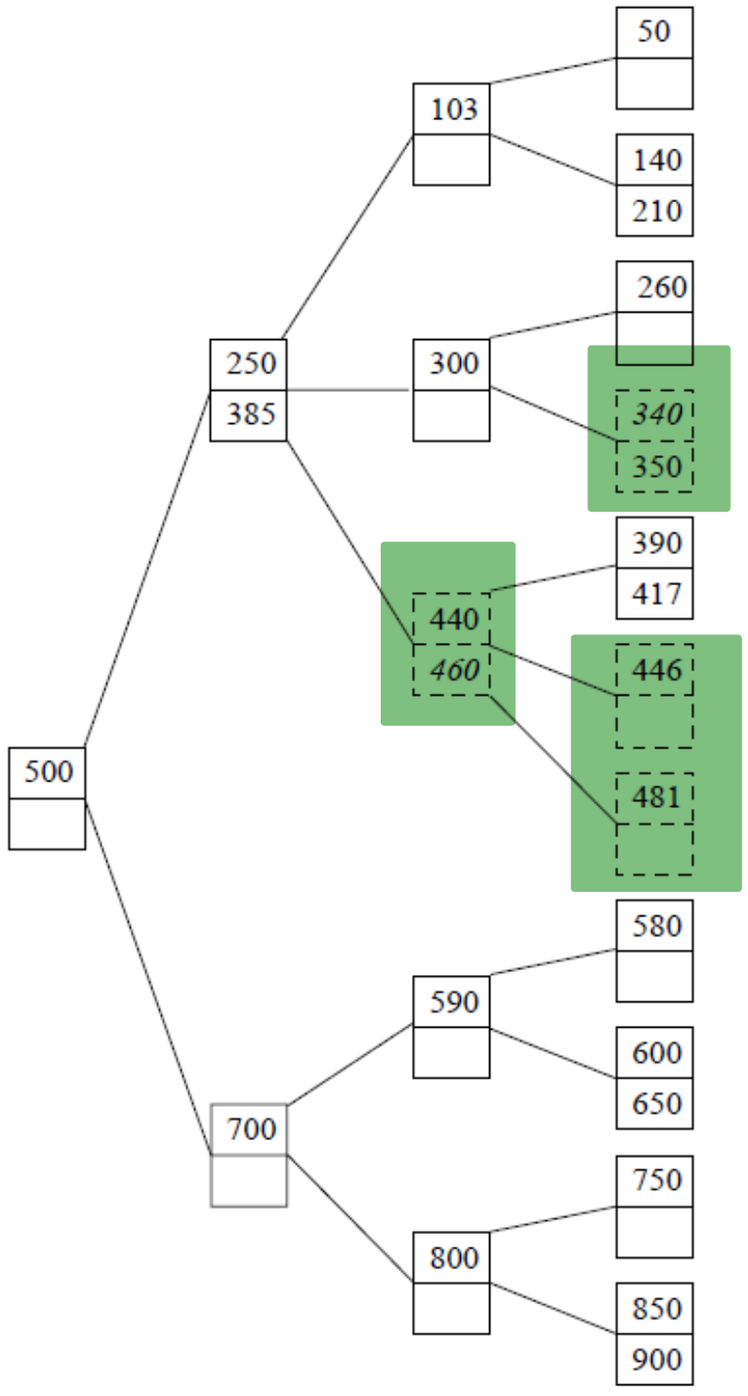


Búsqueda de la hoja en la que insertar 460



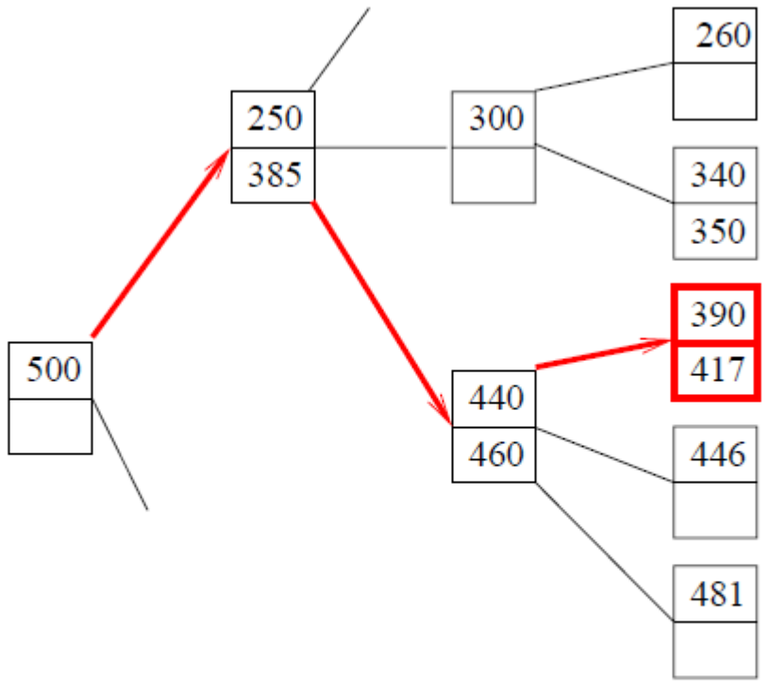
Inserción ordenada en el padre y estado final del subárbol

Inserción



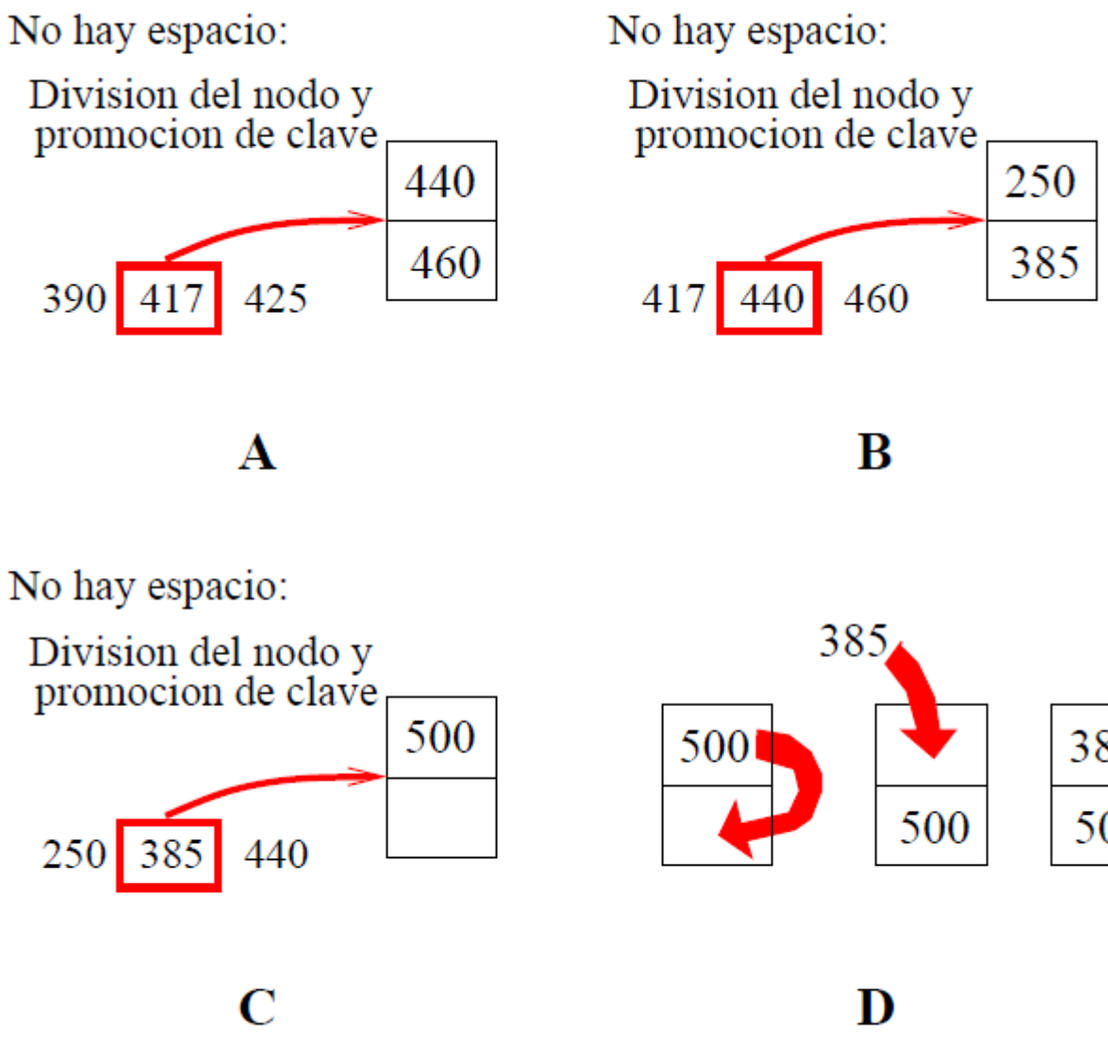
Arbol tras la inserción de 340 y 460

Inserción



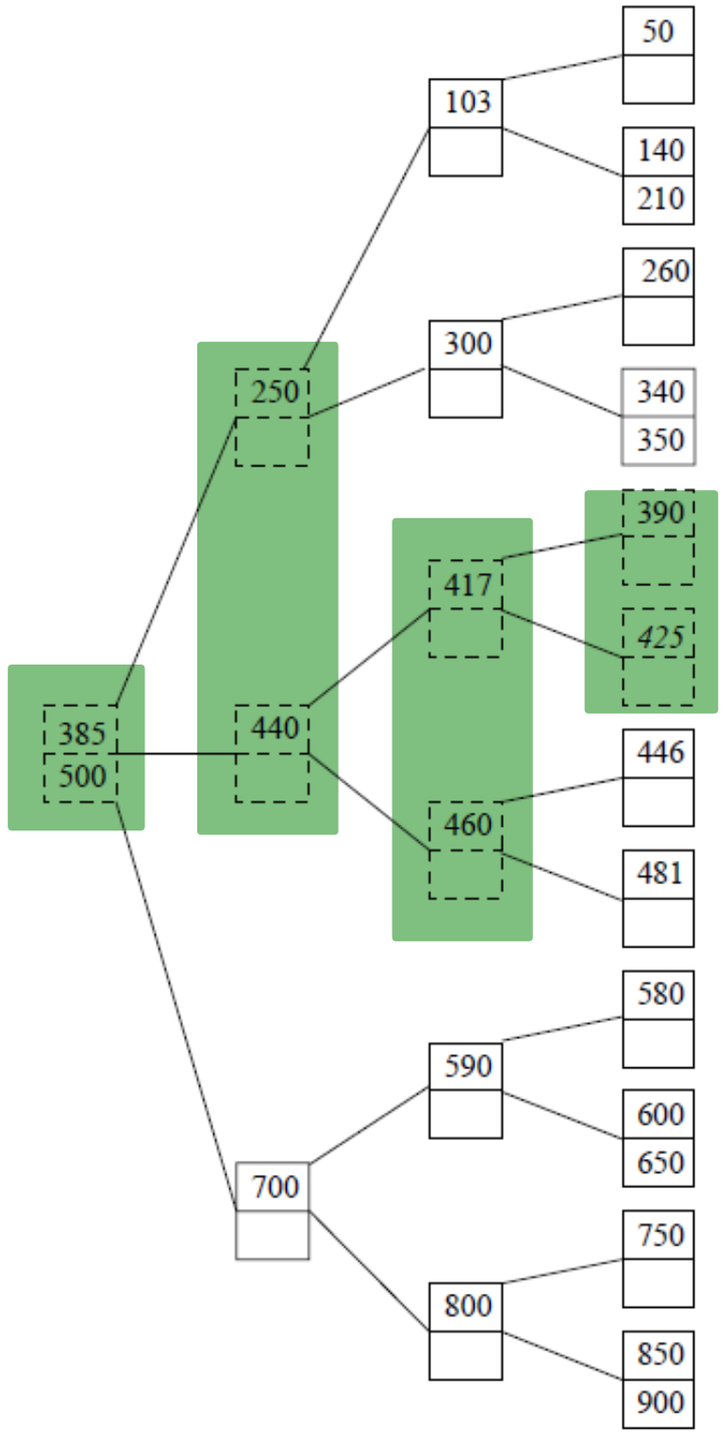
Búsqueda de la hoja en la que insertar 425

Inserción



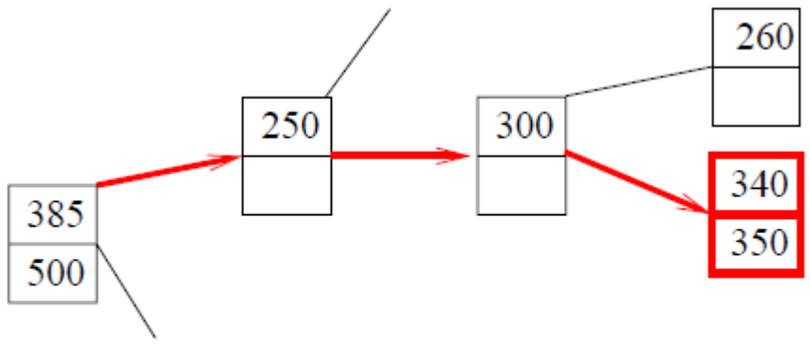
PROCESO DE ACTUALIZACIÓN DEL ÁRBOL PARA INSERTAR 425

Inserción

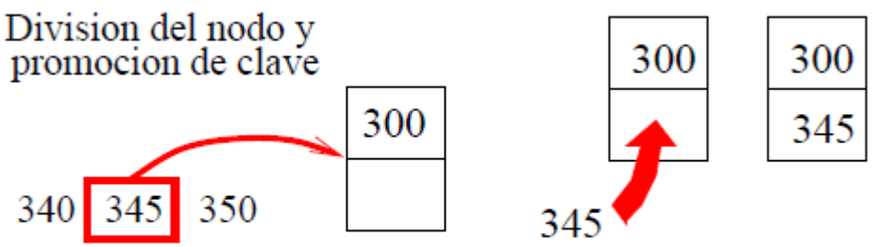


ARBOL TRAS LA INSERCIÓN DE 425

Inserción

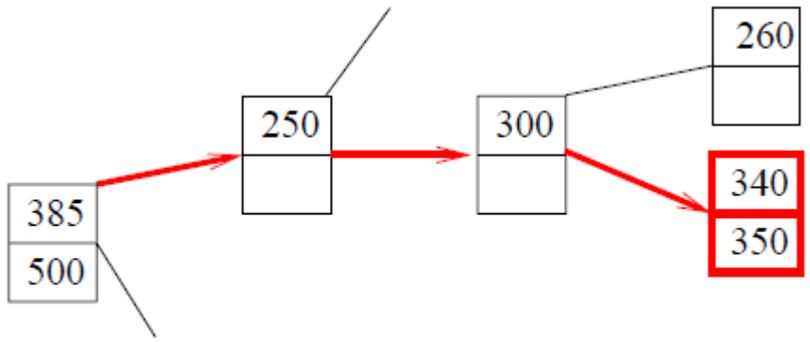


No hay espacio:

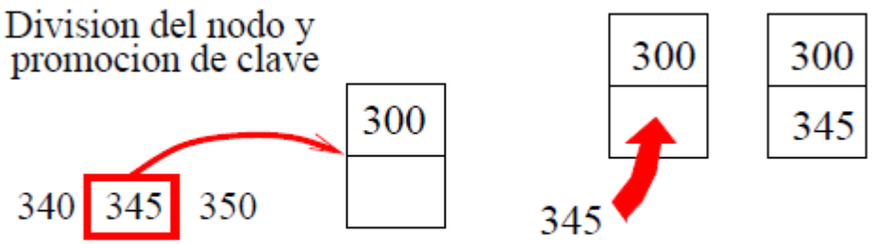


INSERCIÓN DE LA CLAVE 345

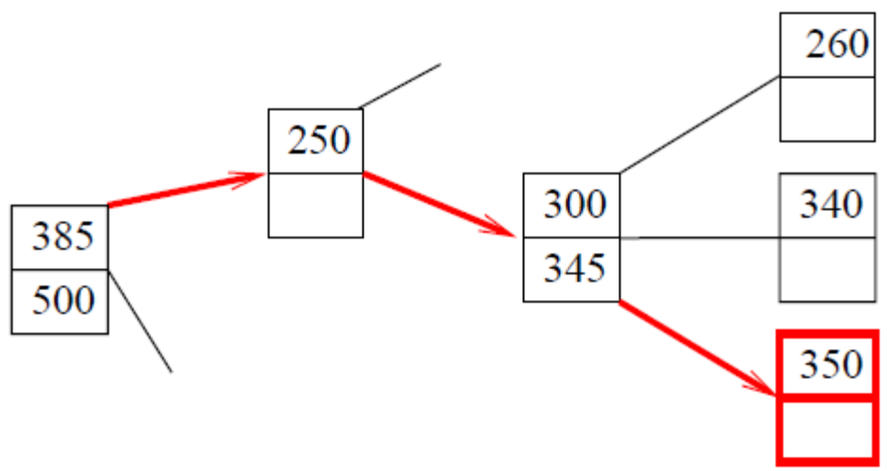
Inserción

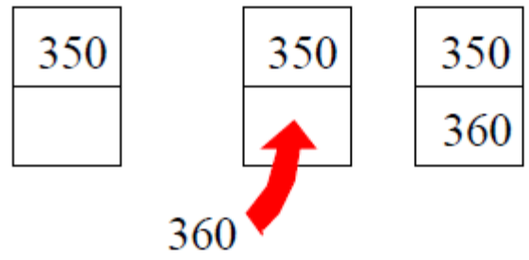


No hay espacio:



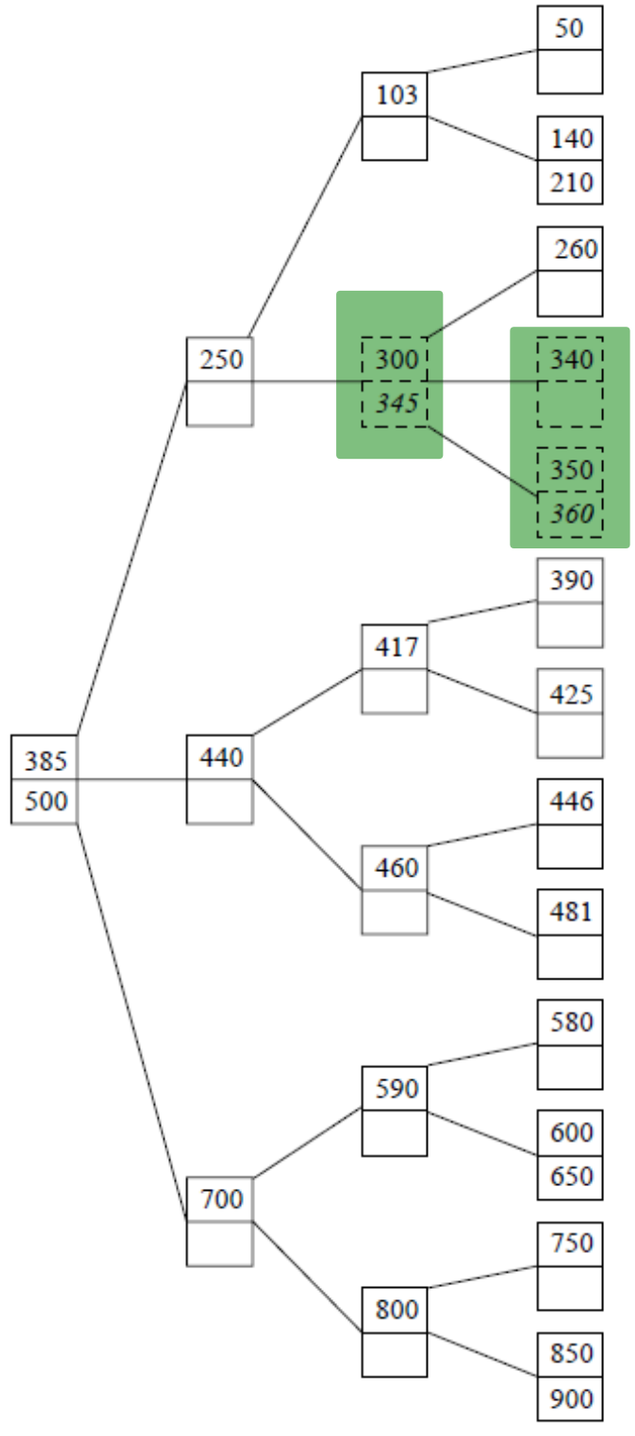
INSERCIÓN DE LA CLAVE 345





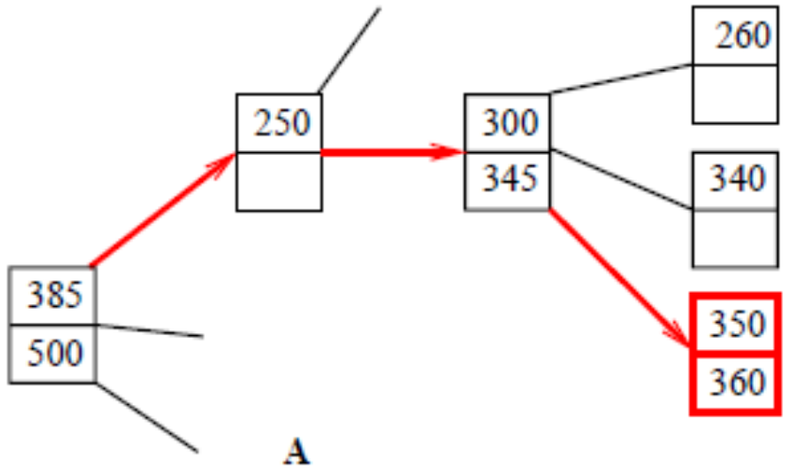
INSERCIÓN DE LA CLAVE 360

Inserción



Arbol tras la inserción de 345 y 360

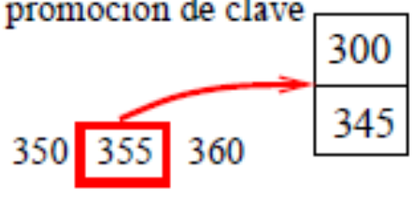
Inserción



A

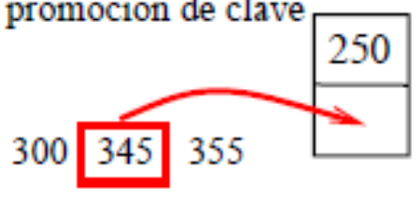
No hay espacio:

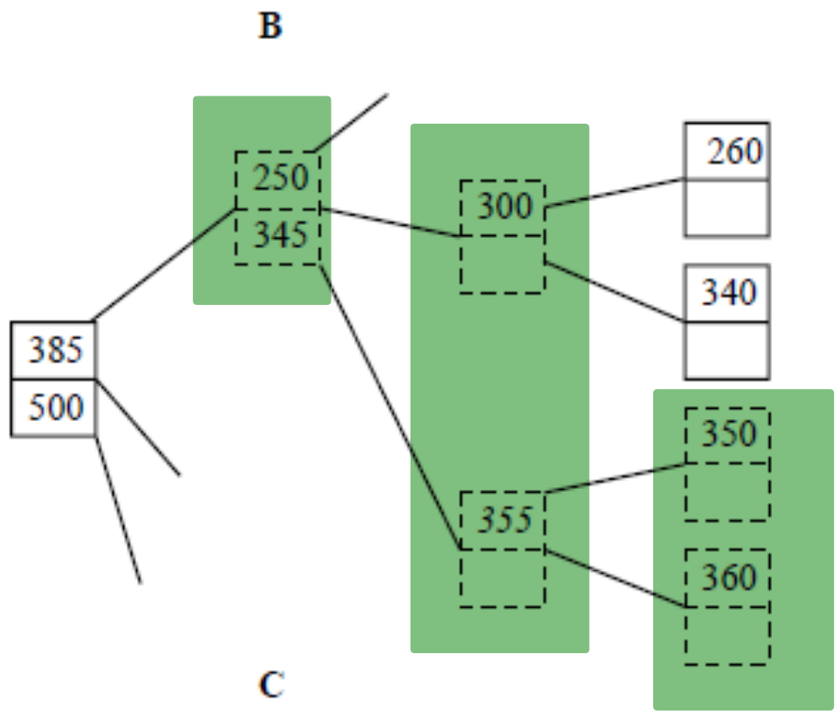
Division del nodo y promocion de clave



No hay espacio:

Division del nodo y promocion de clave

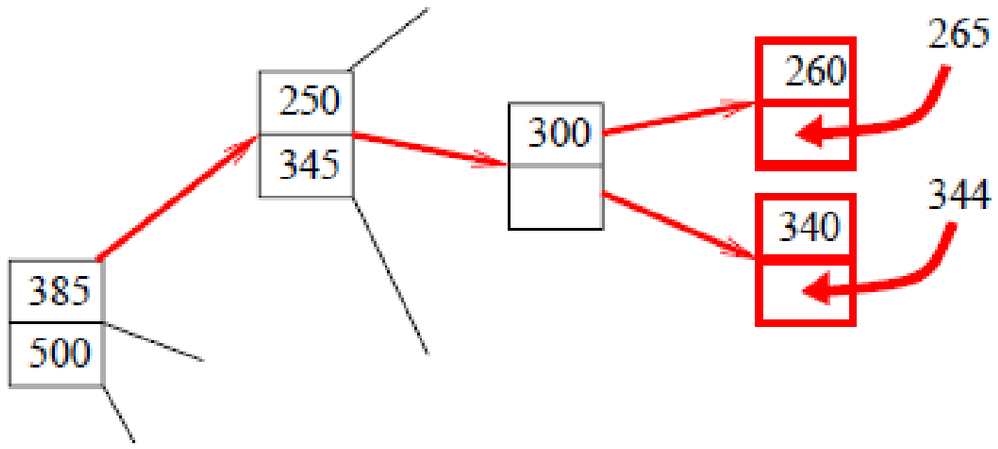




INSERCIÓN DE LA CLAVE 355

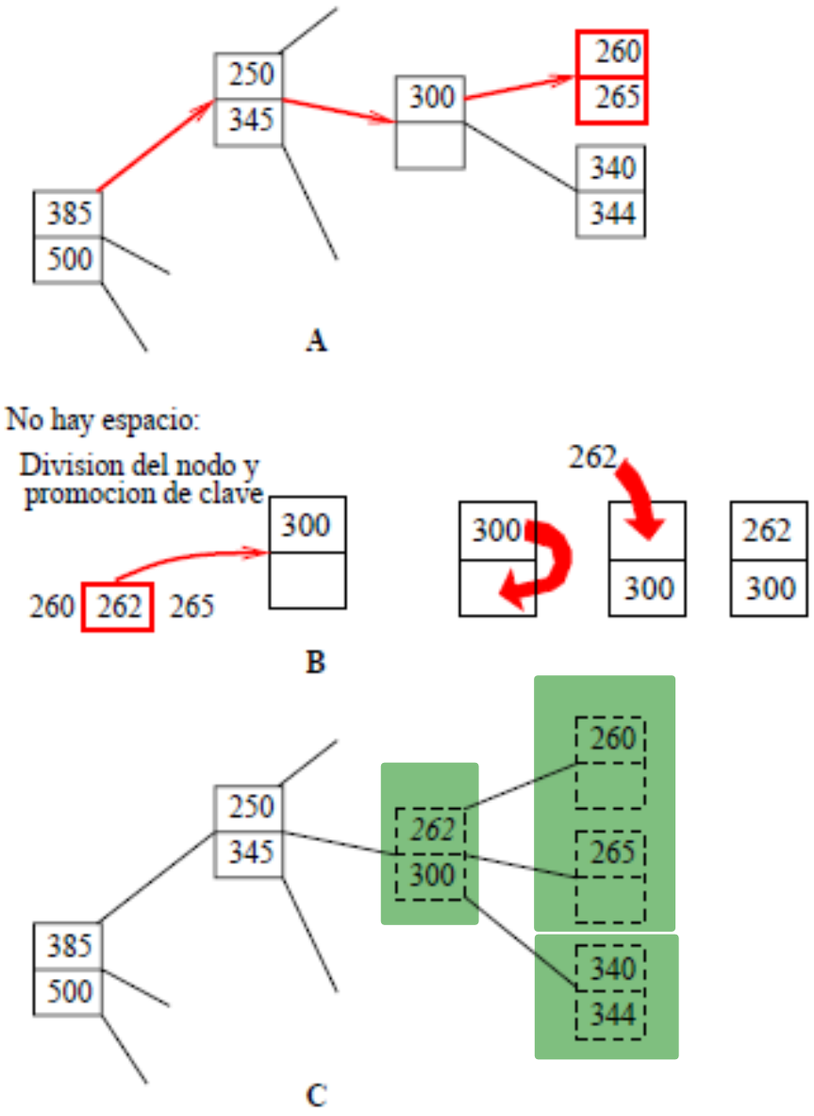
Inserción

7. Inserción de 265 y 344.



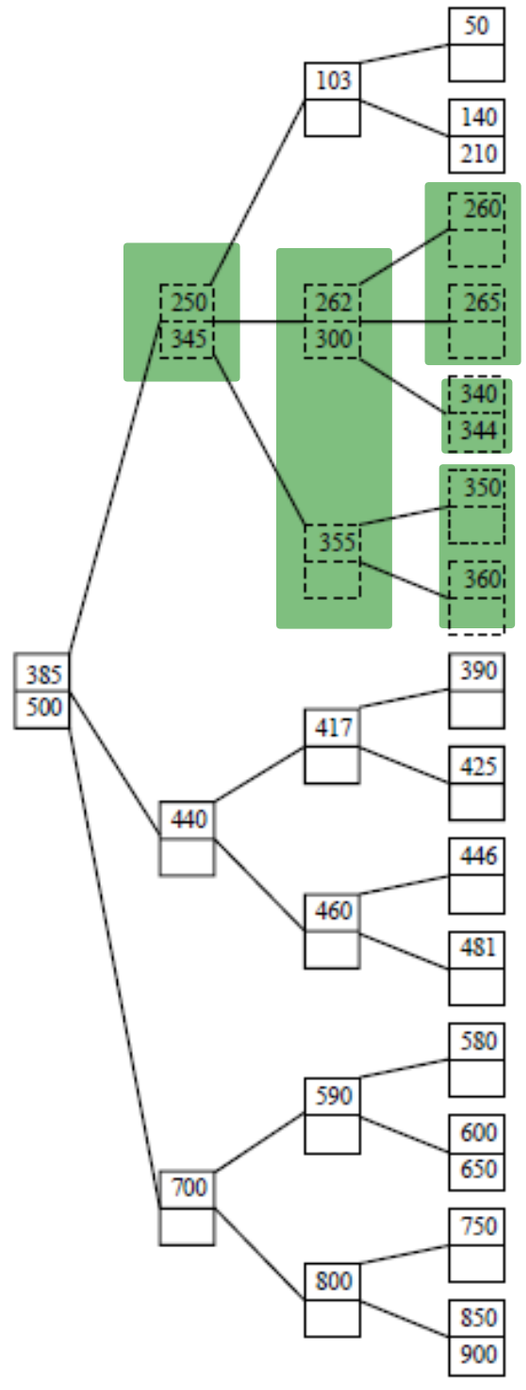
Inserción trivial de las claves 265 y 344

8. Inserción de 262.



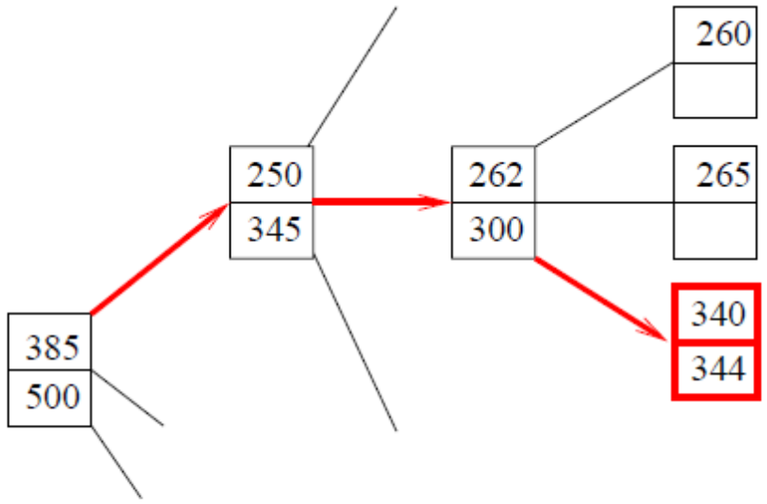
Inserción de la clave 262

Inserción

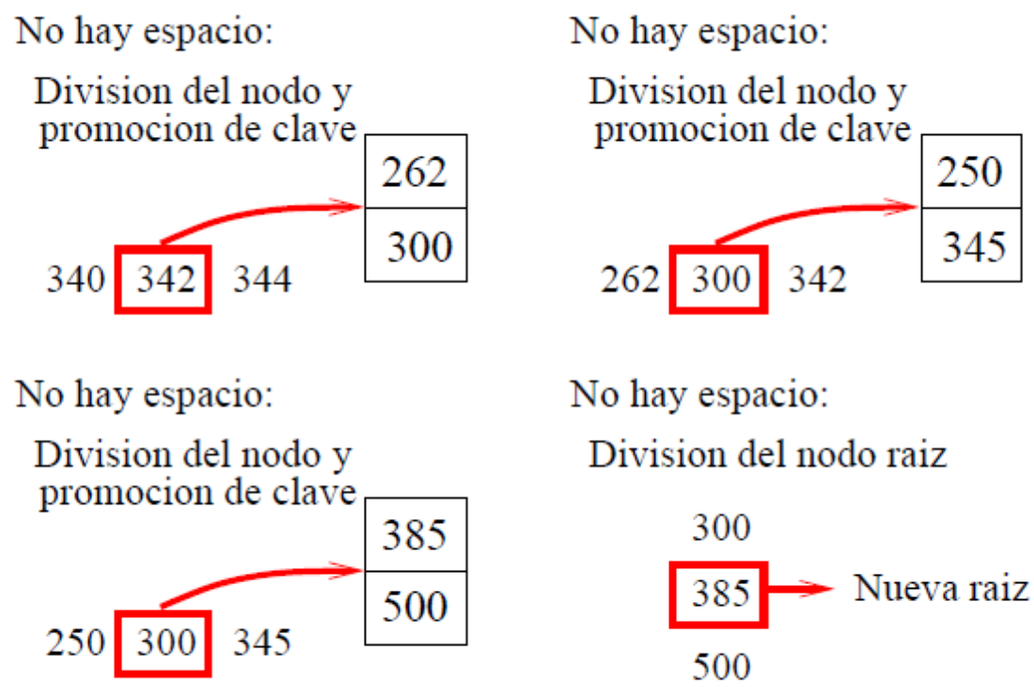


Arbol tras la inserción de 355, 265, 344 y 262

Inserción

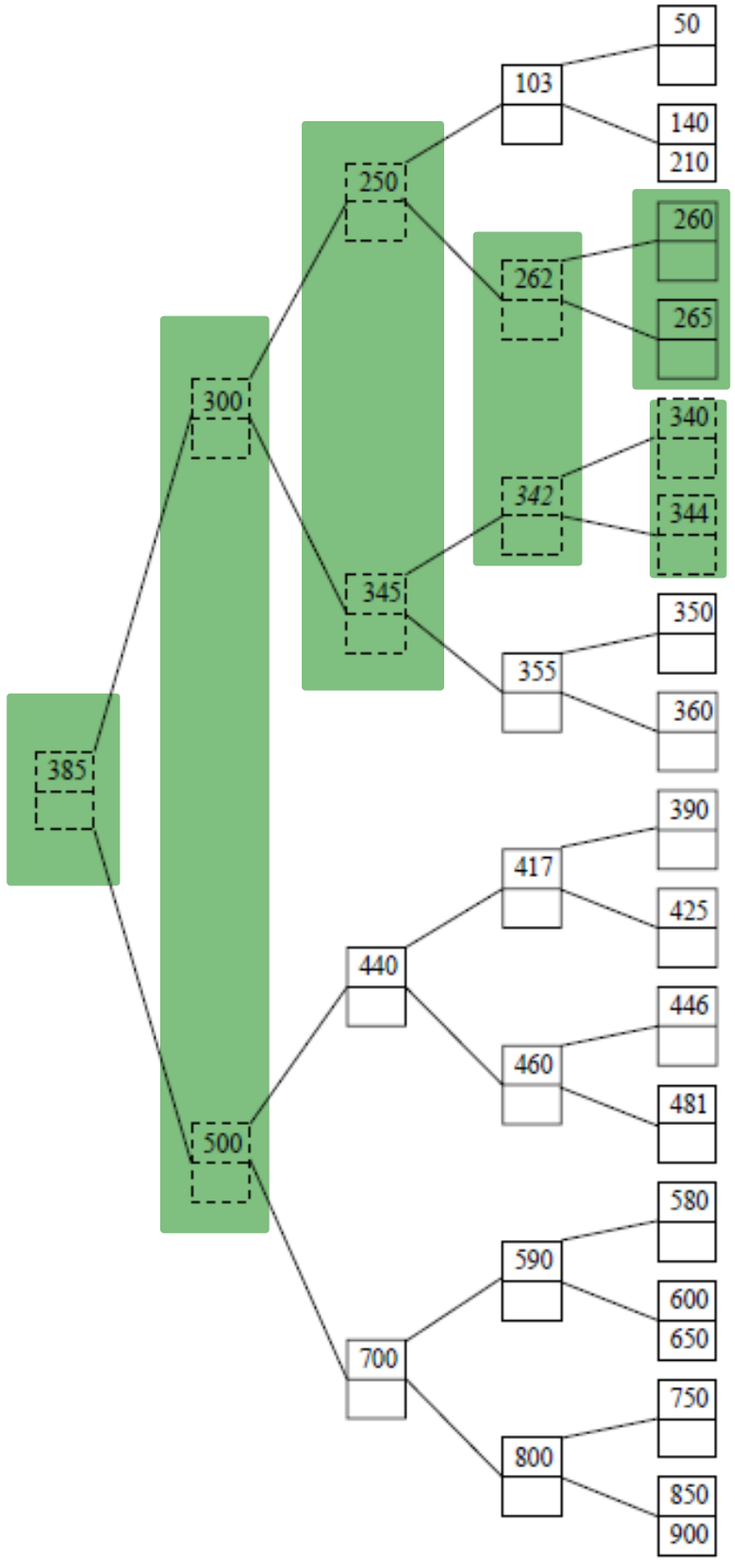


Búsqueda de la hoja en la que insertar 342



Proceso de actualización del árbol para insertar 342

Inserción



ARBOL TRAS LA INSERCIÓN DE 342

Borrado

1. Buscar el nodo donde está la clave *x*
2. Si es un nodo hoja
   a) Borrar la clave y reordenar las restantes
   b) *Verificar y ajustar el árbol*
3. Si es un nodo interior
   a) Sustituir la clave a borrar por la inmediatamente superior, que ha de estar forzosamente en un nodo hoja
   b) *Verificar y ajustar el árbol*

Borrado

Verificar y ajustar el árbol

1. Si el número de claves del nodo modificado es válido
   - Terminar
2. Si no
   a) Si algún hermano adyacente tiene más claves que el mínimo
      - *Redistribución*
   b) Si no
      - *Unión*
      - Volver a verificar y ajustar el árbol para el nodo padre

Borrado

Redistribución

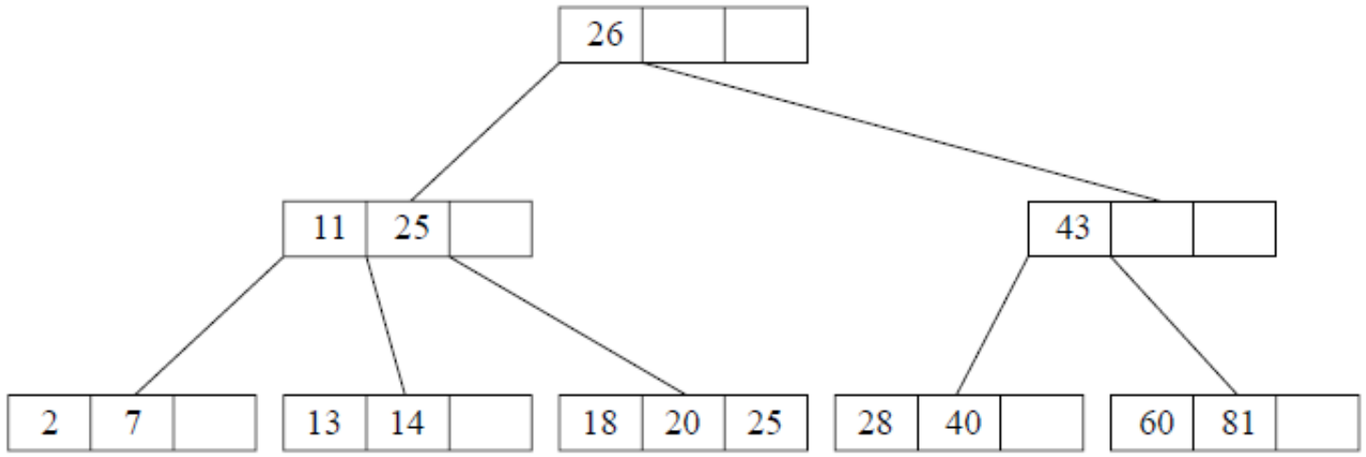
- Se aplica cuando el nodo afectado queda con menos claves que el mínimo permitido, y algún hermano adyacente tiene más claves que el mínimo y puede "prestar" una clave al nodo afectado
- La clave del padre de ambos nodos pasa al nodo donde se ha borrado y ésta se sustituye por una clave del otro nodo
  - La menor, si el hermano está a su derecha
  - La mayor, si el hermano está a su izquierda

Borrado

Unión

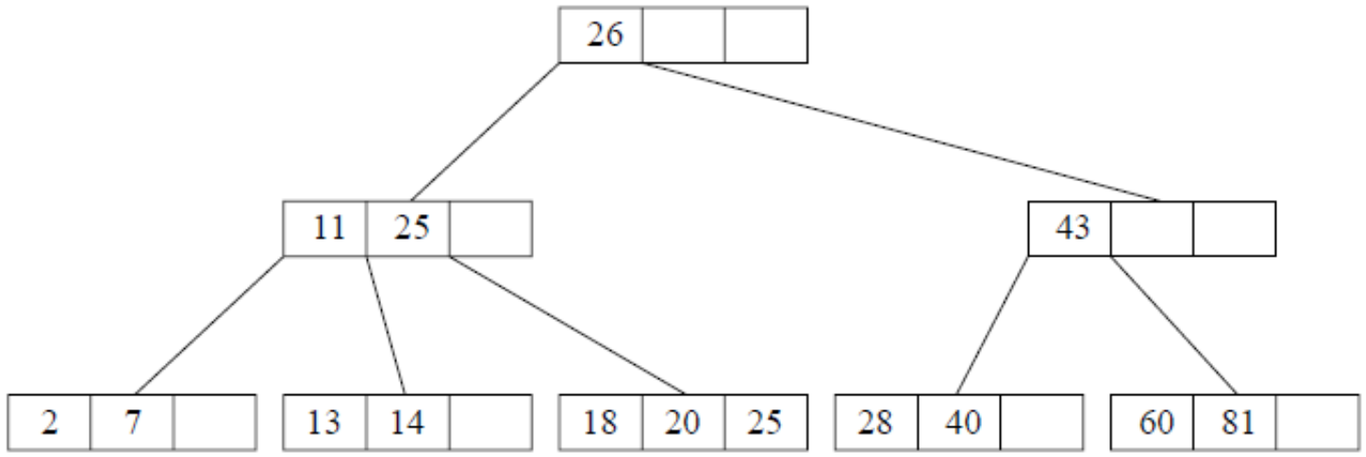
- Se aplica cuando no sea posible la redistribución y la única opción es compactar dos hermanos adyacentes en uno
- Se hace un solo nodo entre el nodo de se ha borrado, un hermano, y la clave que las separa del nodo padre
- La unión puede obligar a ajustar el árbol, tomando ahora como referencia el nodo padre
- Este efecto puede propagarse hacia la raíz, produciendo, en algunos casos, el decrecimiento de altura del árbol

Borrado

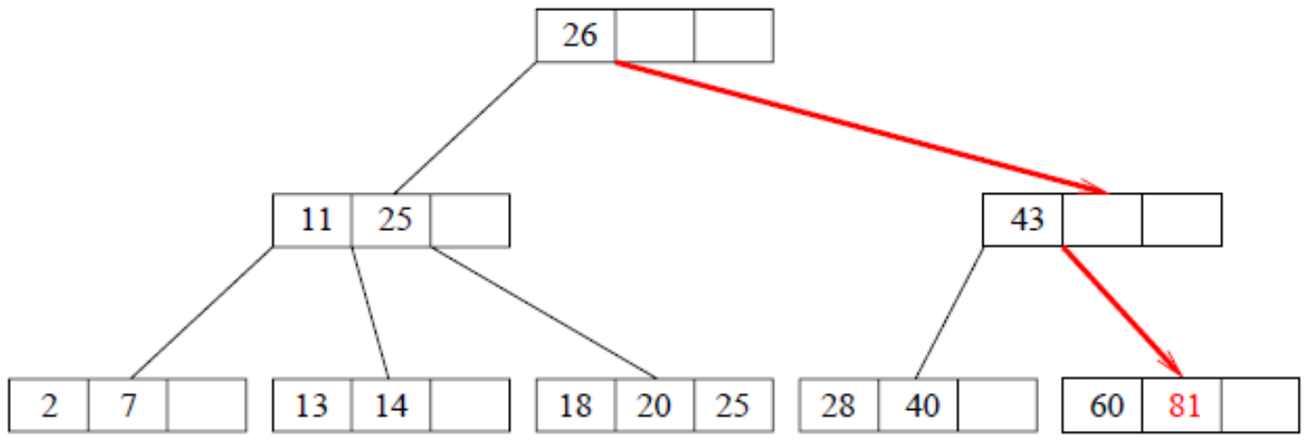


ARBOL B DE ORDEN 4 INICIAL

Borrado

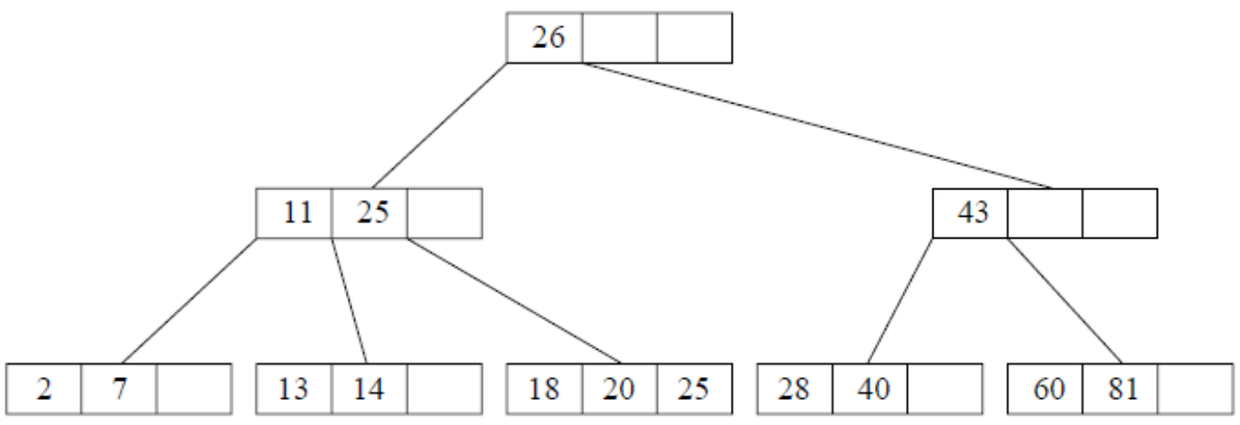


Arbol B de orden 4 inicial

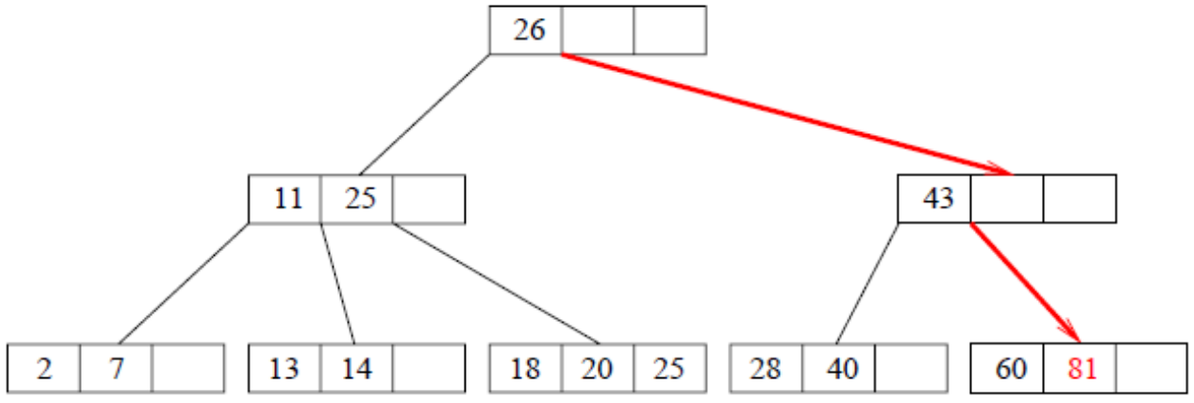


Búsqueda del nodo en el que borrar 81

Borrado

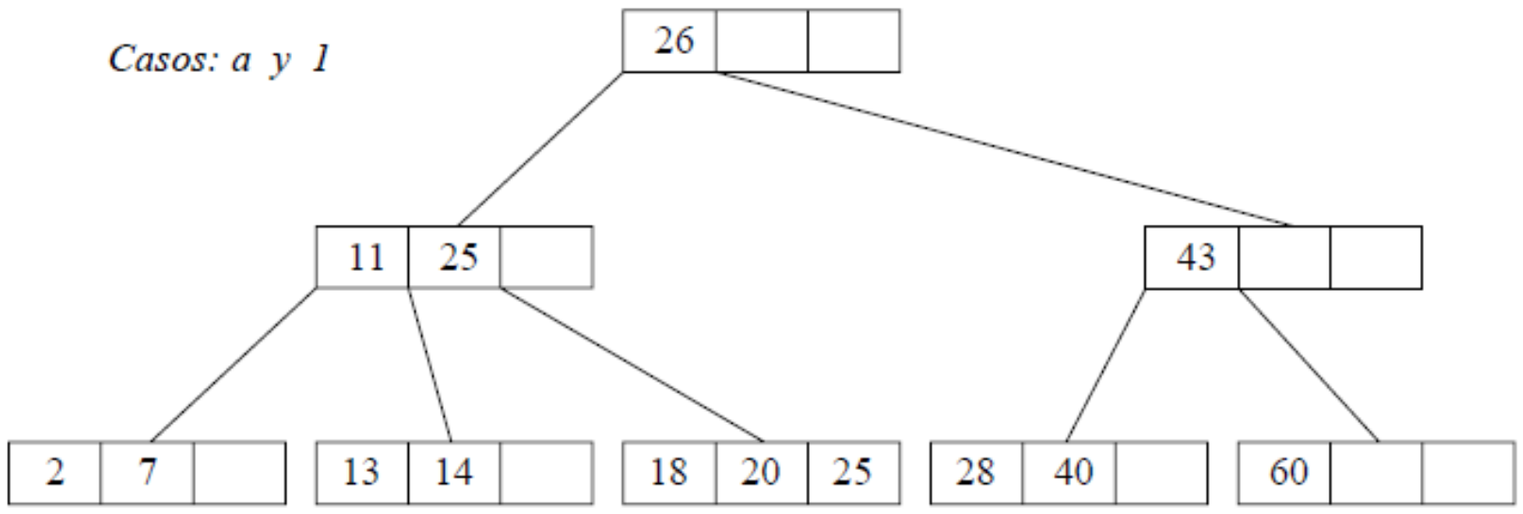


Arbol B de orden 4 inicial



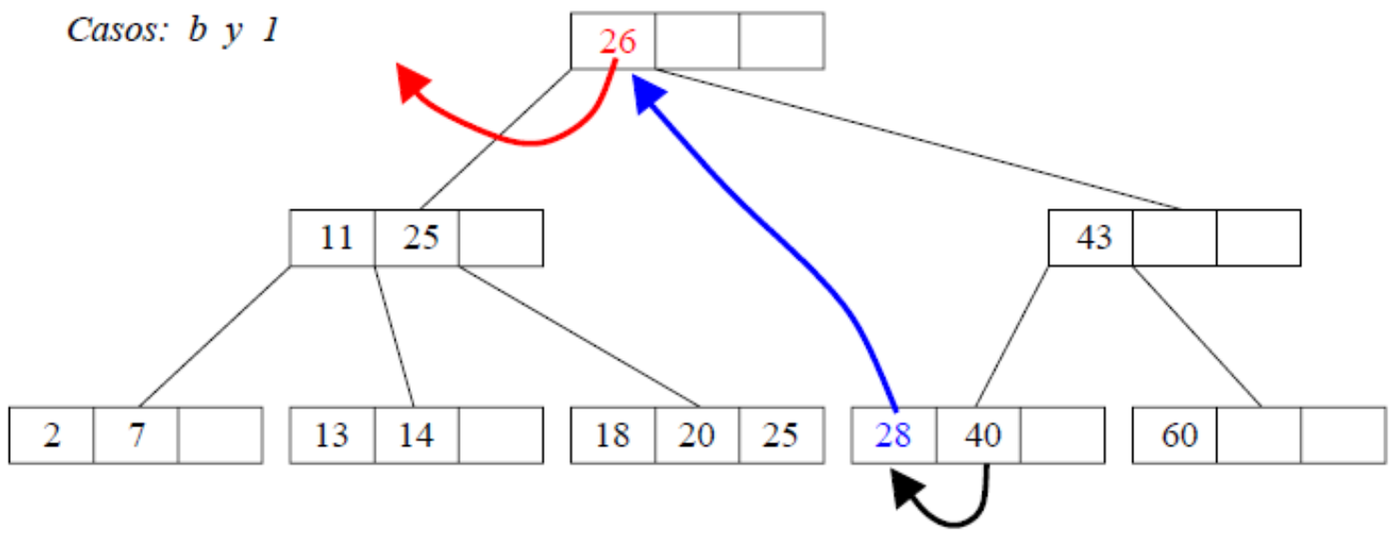
Búsqueda del nodo en el que borrar 81

Casos: a y 1



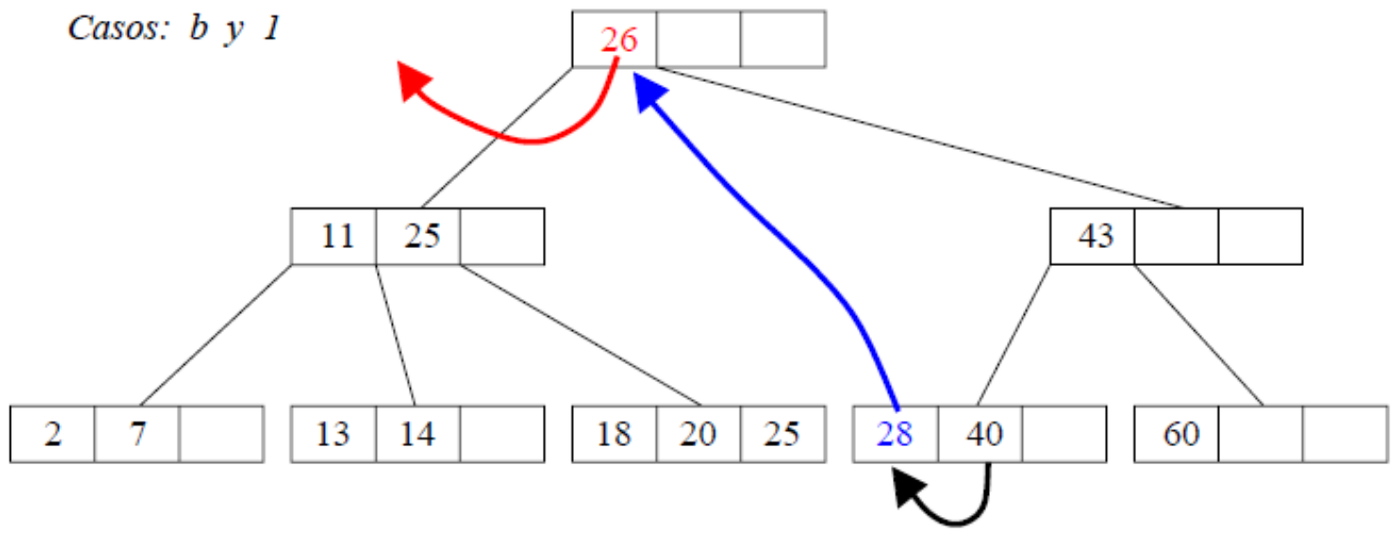
Arbol después de borrar la clave 81

Borrado

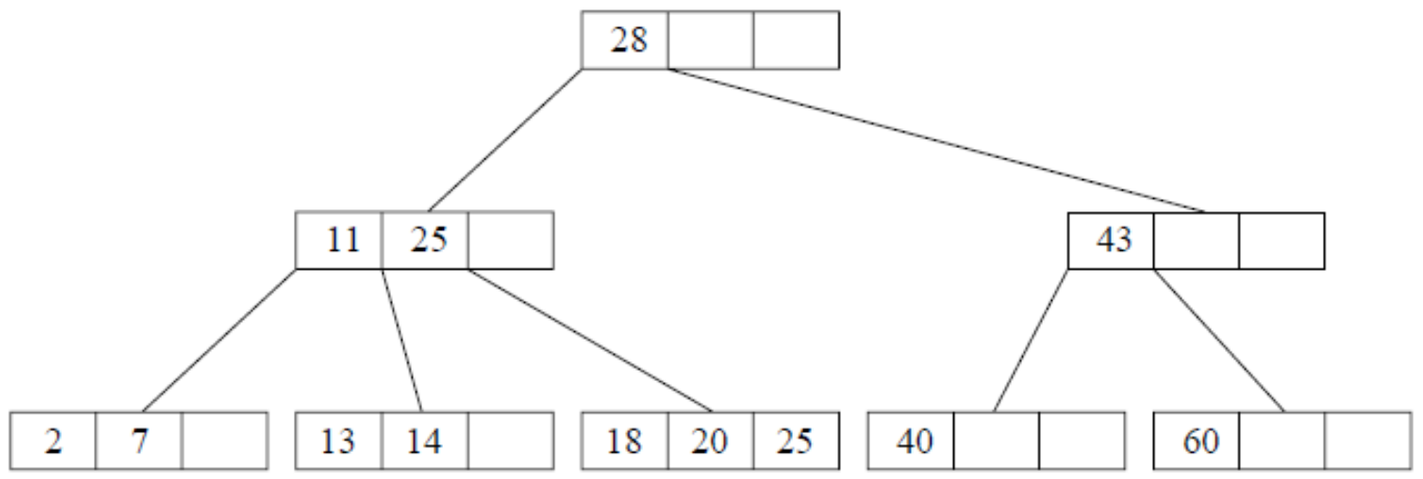


OPERACIONES REQUERIDAS PARA BORRAR LA CLAVE 26

Borrado

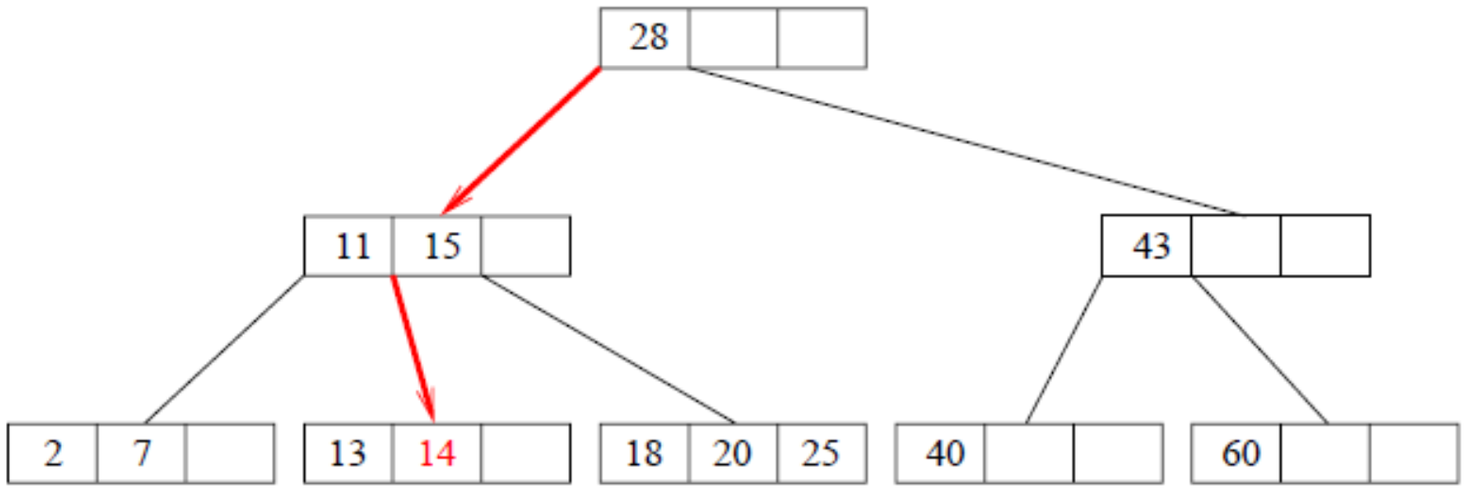


Operaciones requeridas para borrar la clave 26



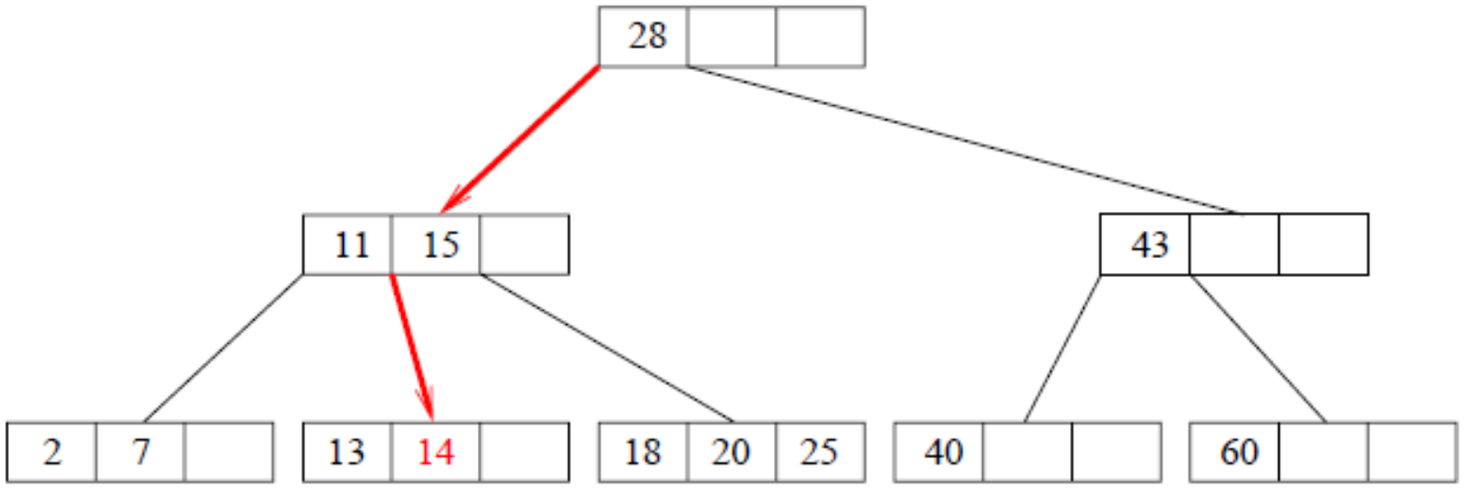
Arbol después de borrar la clave 26

Borrado

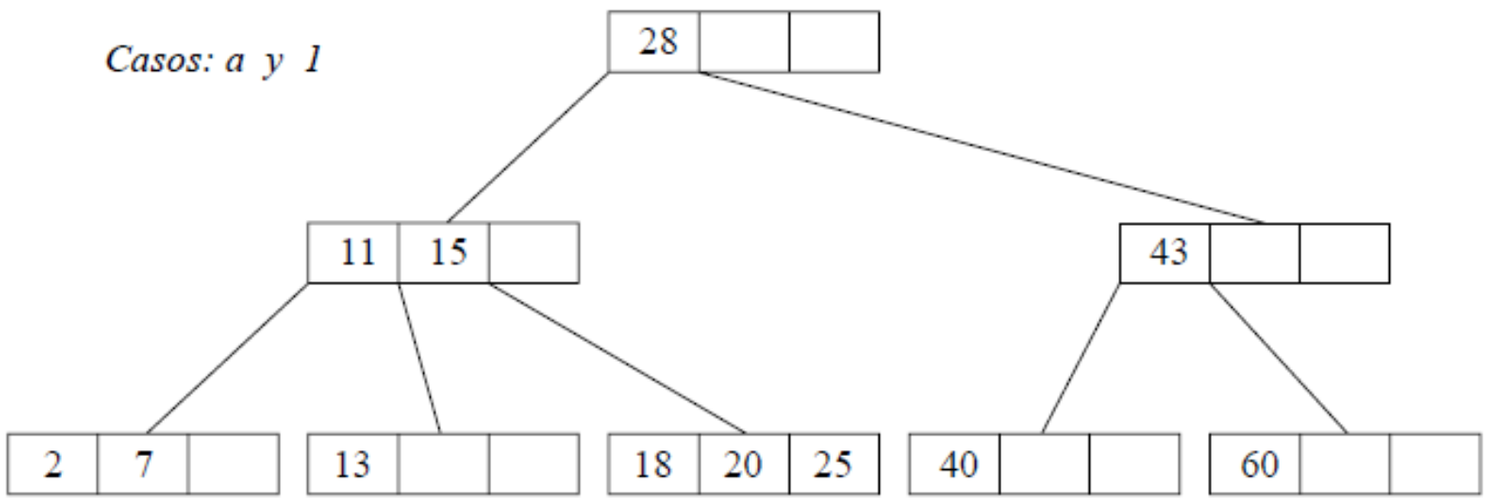


Búsqueda del nodo en el que borrar 14

Borrado

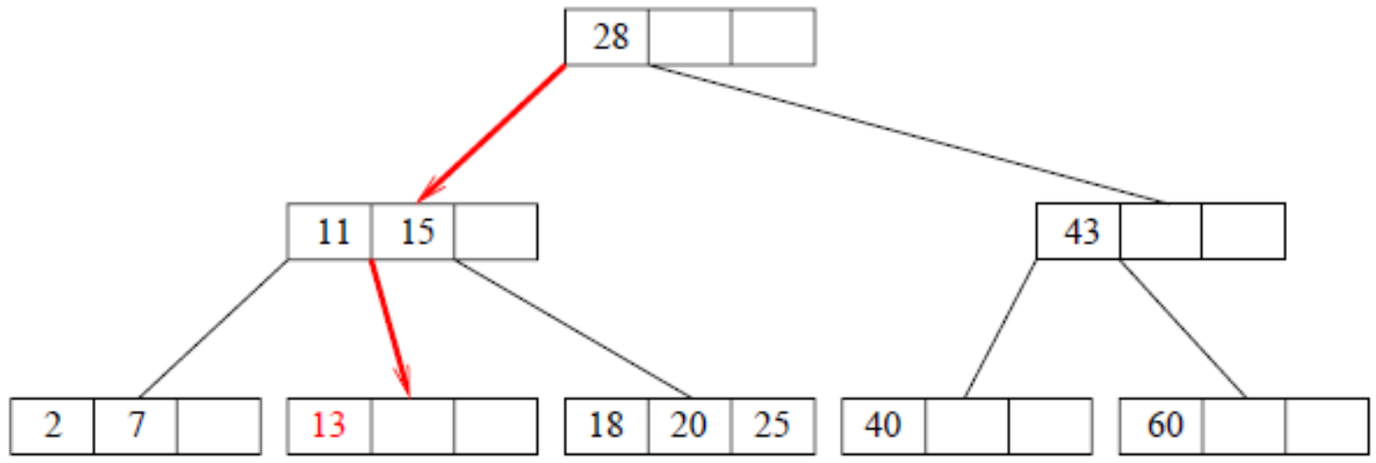


Búsqueda del nodo en el que borrar 14



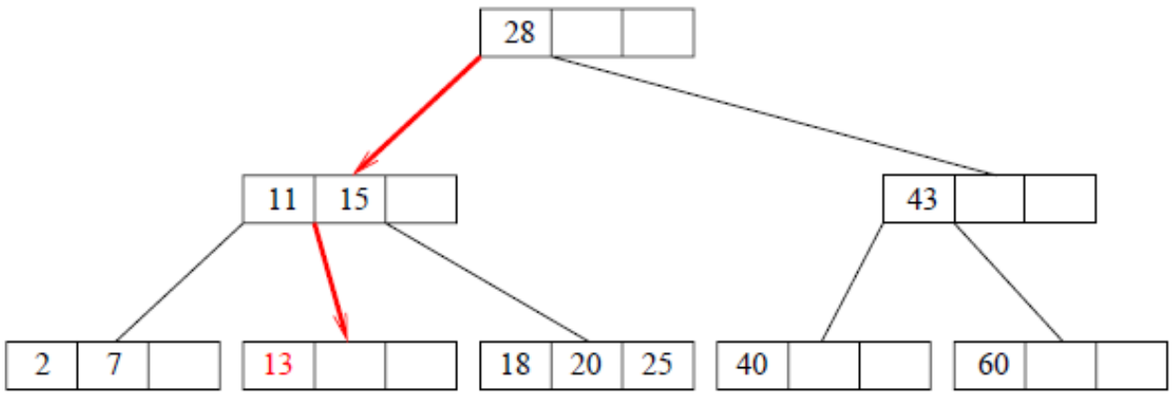
Arbol después de borrar la clave 14

Borrado

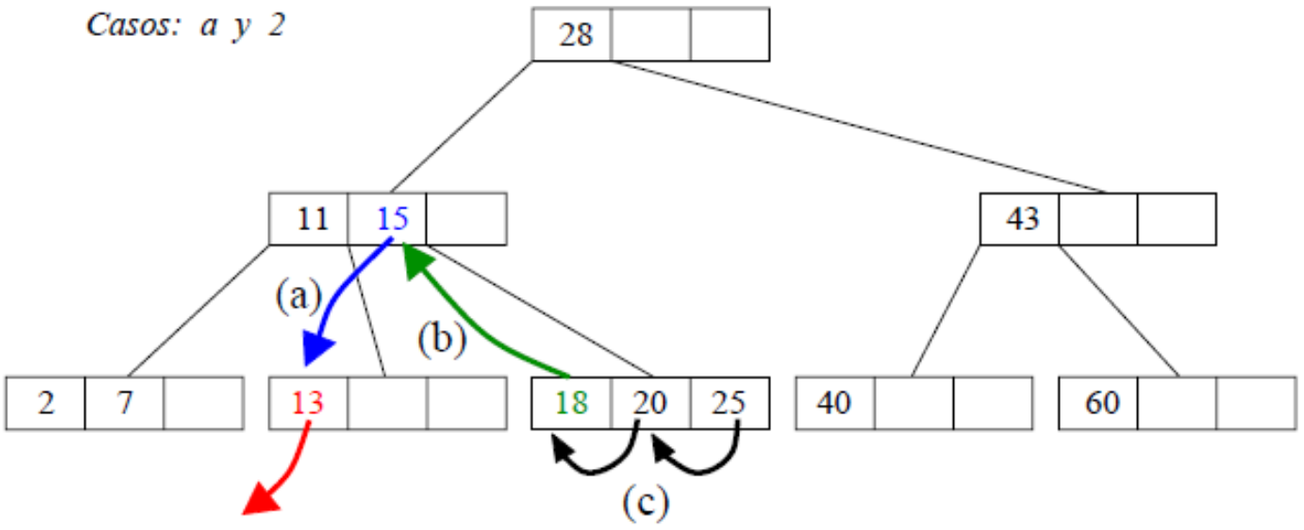


BÚSQUEDA DEL NODO EN EL QUE BORRAR 13

Borrado



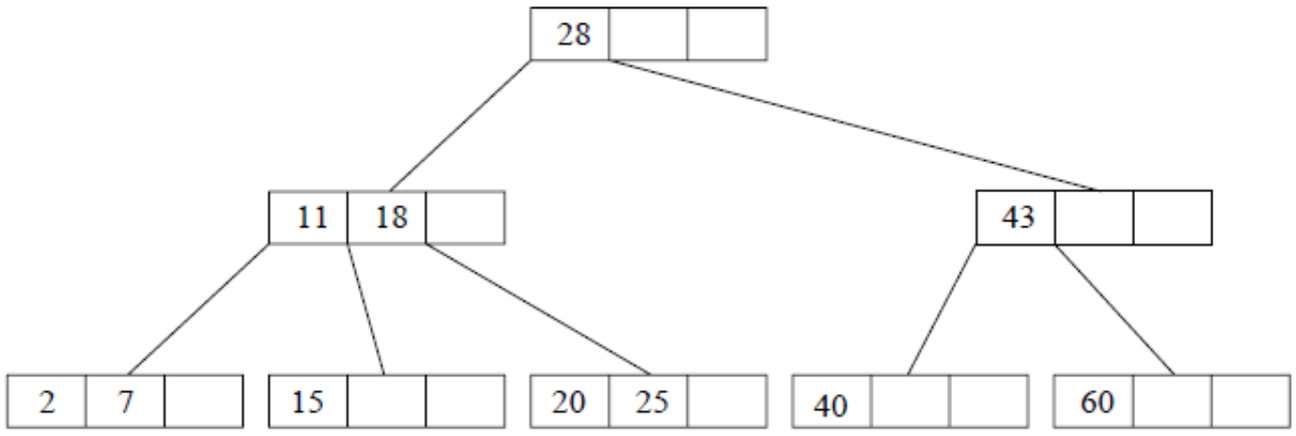
Búsqueda del nodo en el que borrar 13



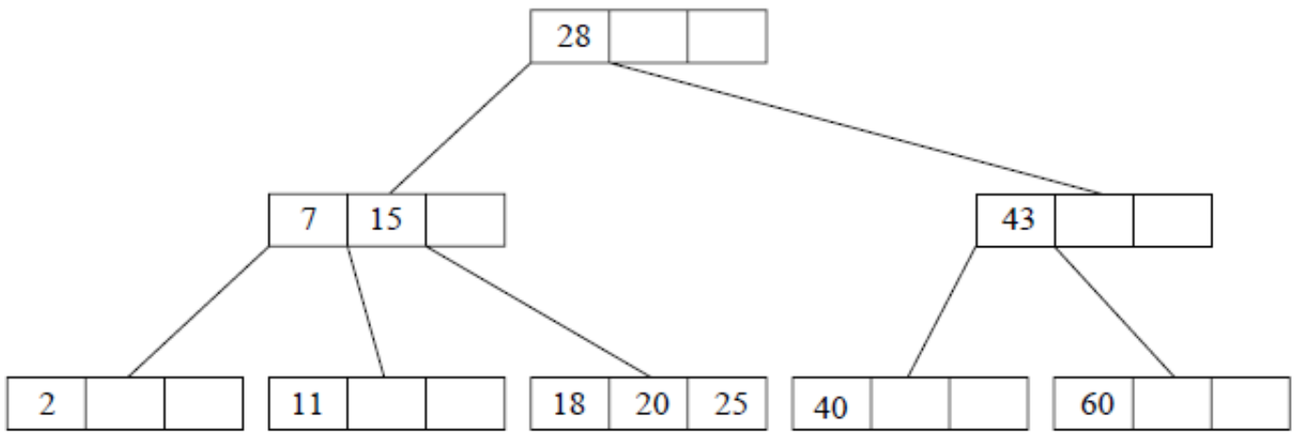
Distribución de claves en el borrado de la clave 13

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

Borrado

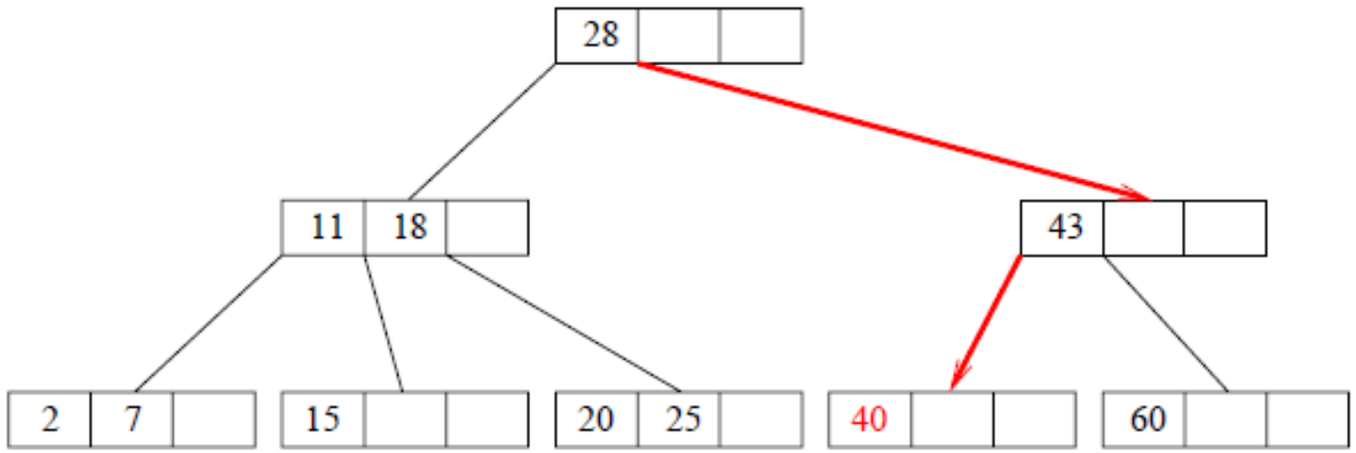


Arbol después de borrar la clave 13



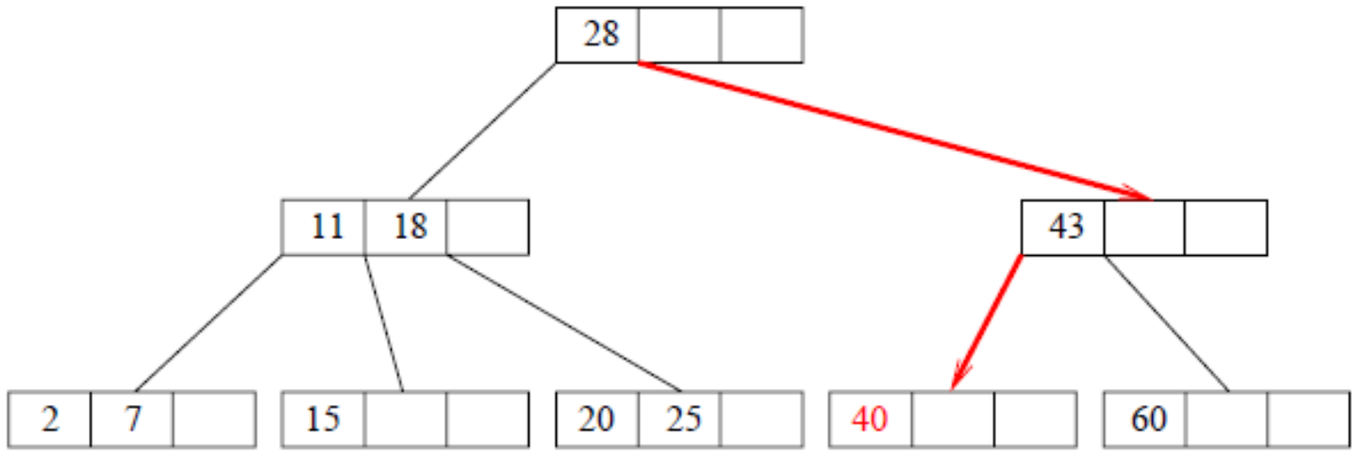
Arbol después de borrar la clave 13 (alternativa)

Borrado



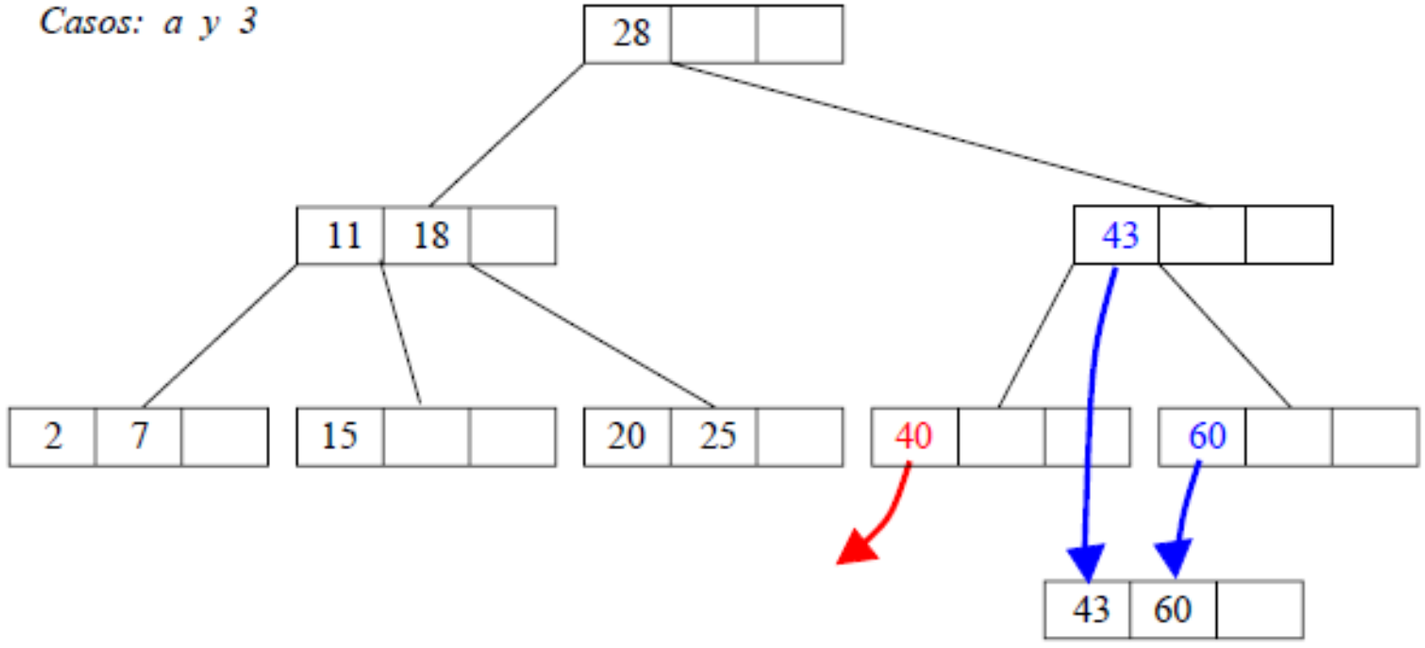
Búsqueda del nodo en el que borrar 40

Borrado



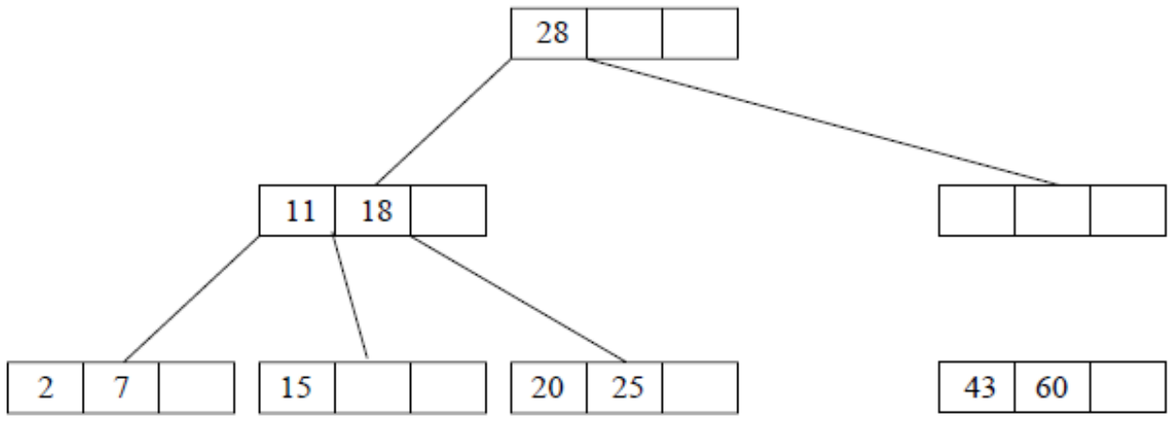
Búsqueda del nodo en el que borrar 40

Casos: a y 3



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

Borrado



Resultado de la unión de claves

Caso: 2

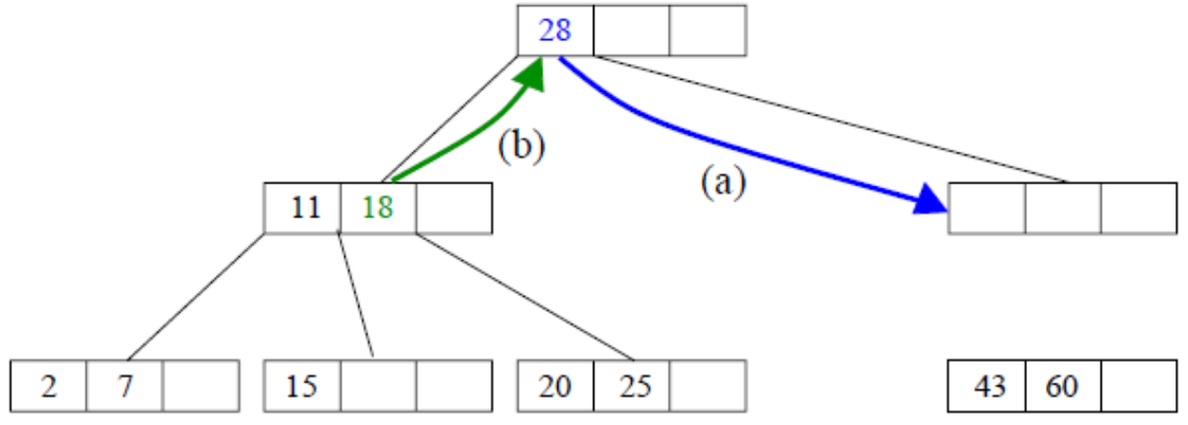
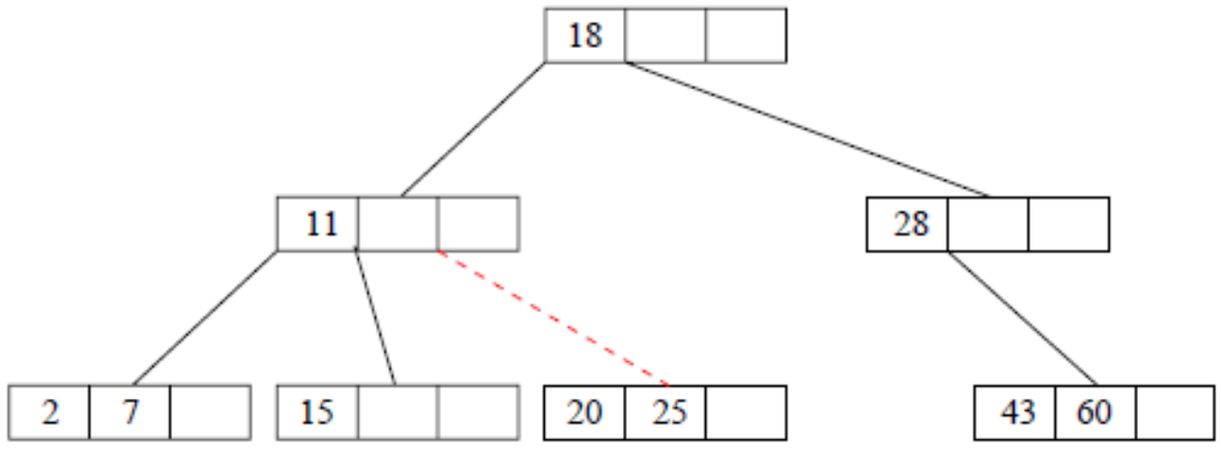


Figura 20: Distribución de claves para ajustar el nodo vacío

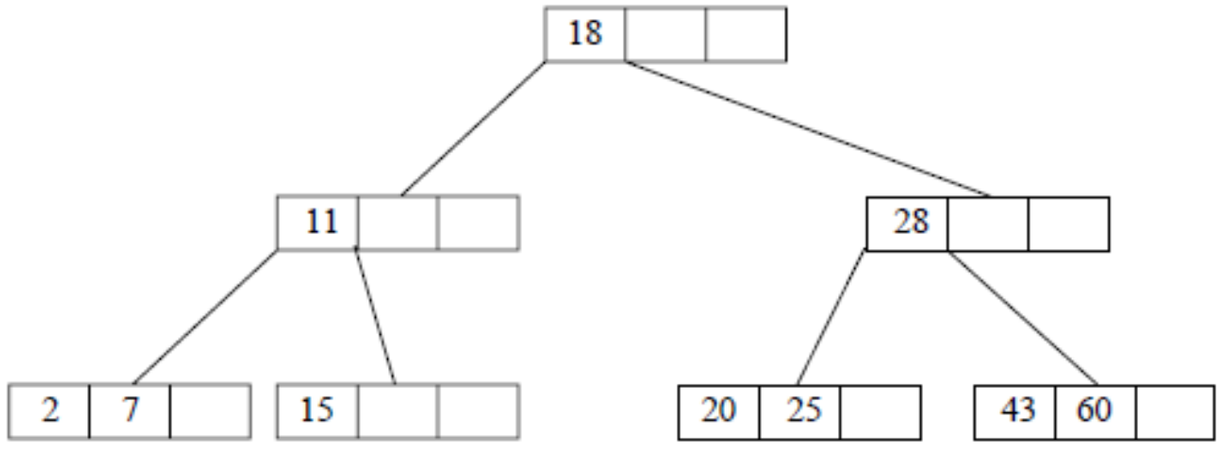
Distribución de claves para ajustar el nodo vacío

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

Borrado



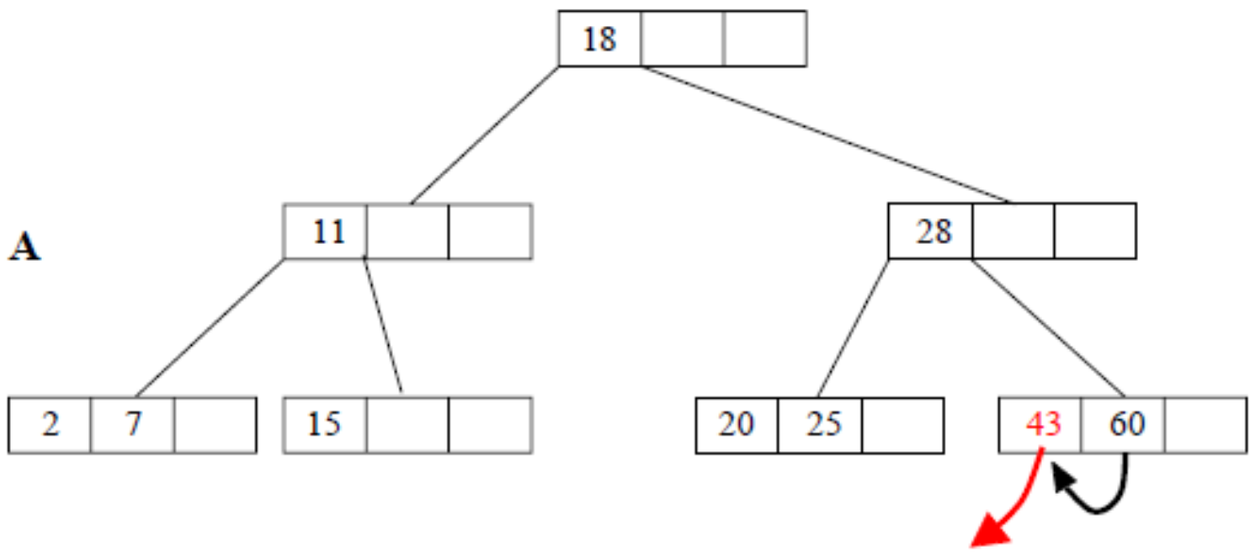
Resultado de la distribución de claves

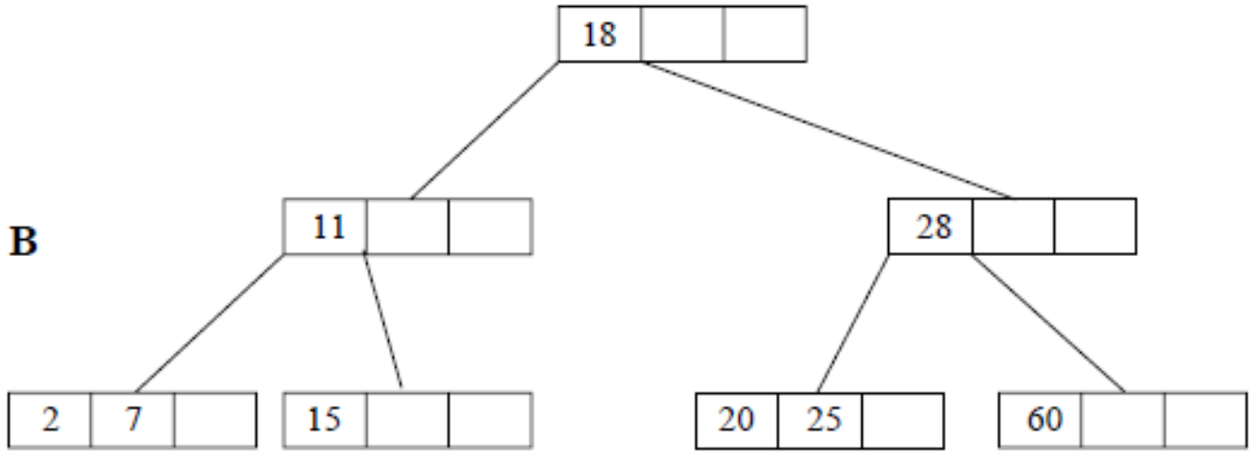


Arbol final resultante del borrado de 40

Borrado

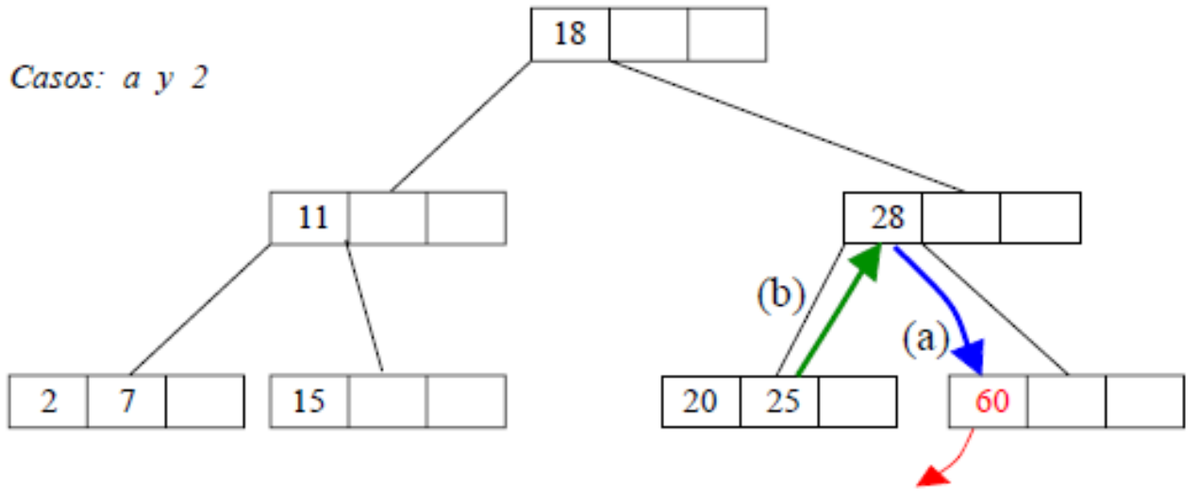
Casos: a y 1





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

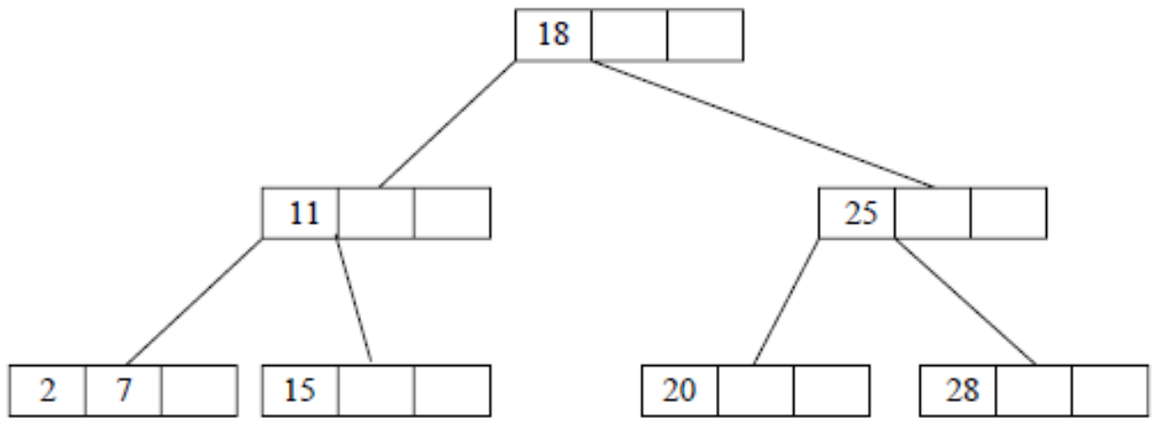
Borrado



Distribución de claves para el borrado de la clave 60

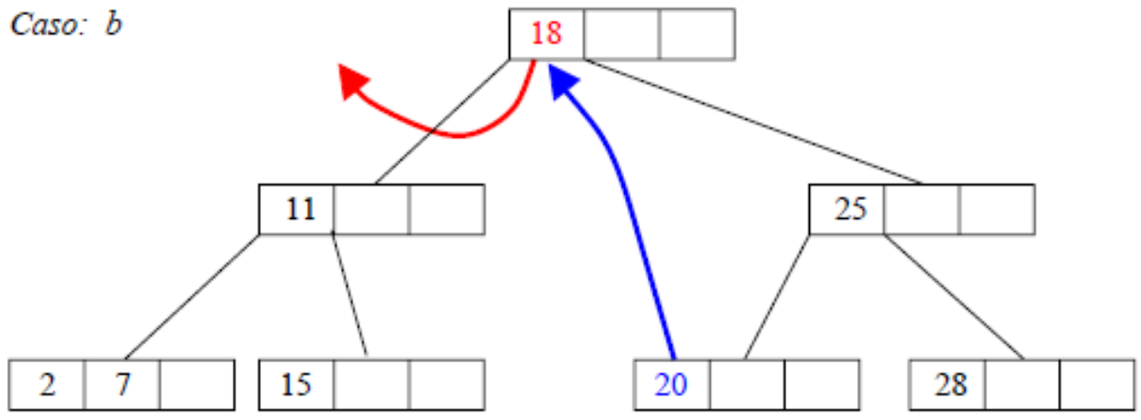
(a) La clave 28, que está en el padre de los dos nodos involucrados, y que los separa, pasa al nodo que ha quedado vacío.

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

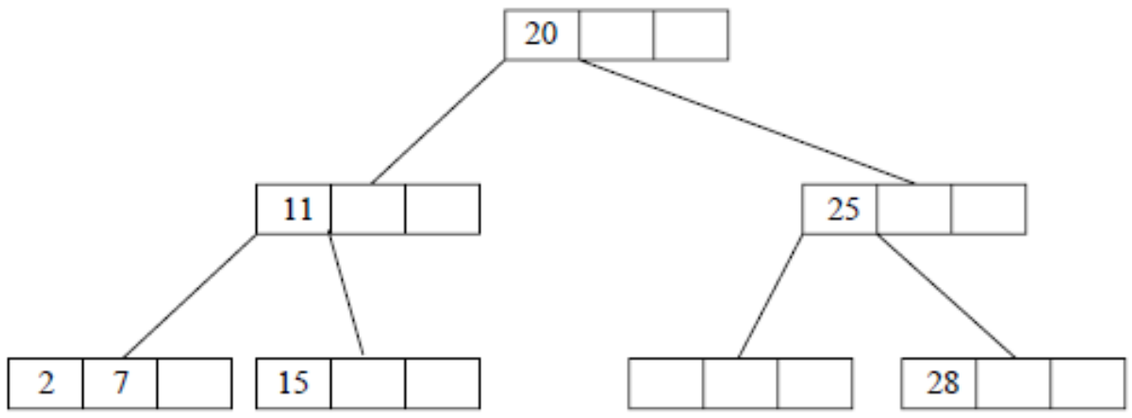


Arbol final tras el borrado de la clave 60

Borrado

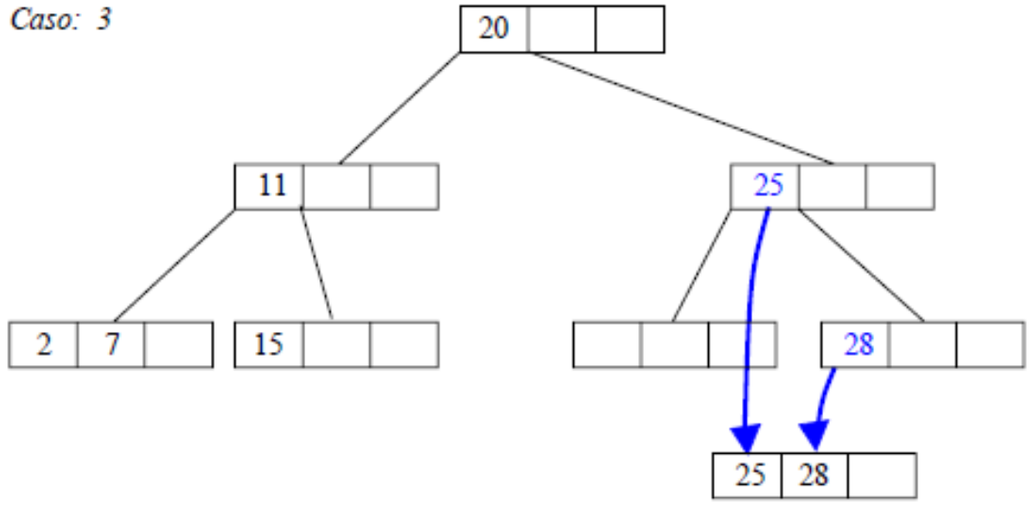


Operaciones requeridas para borrar la clave 18

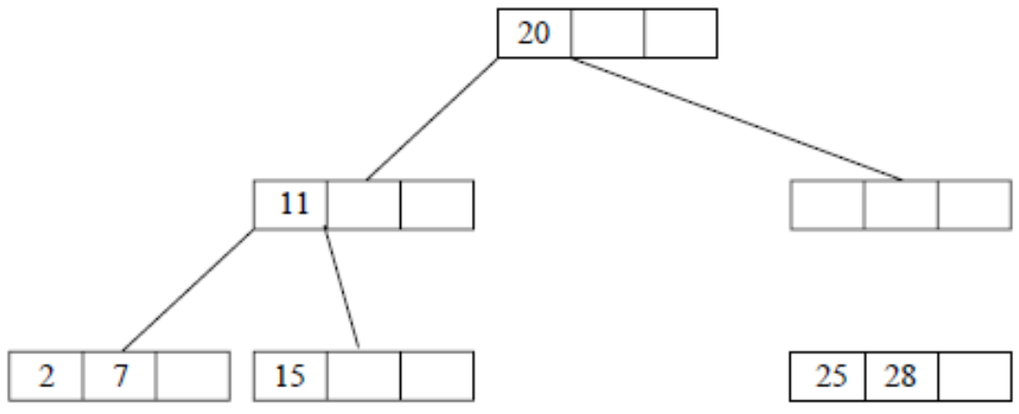


Arbol después de borrar la clave 18

Borrado



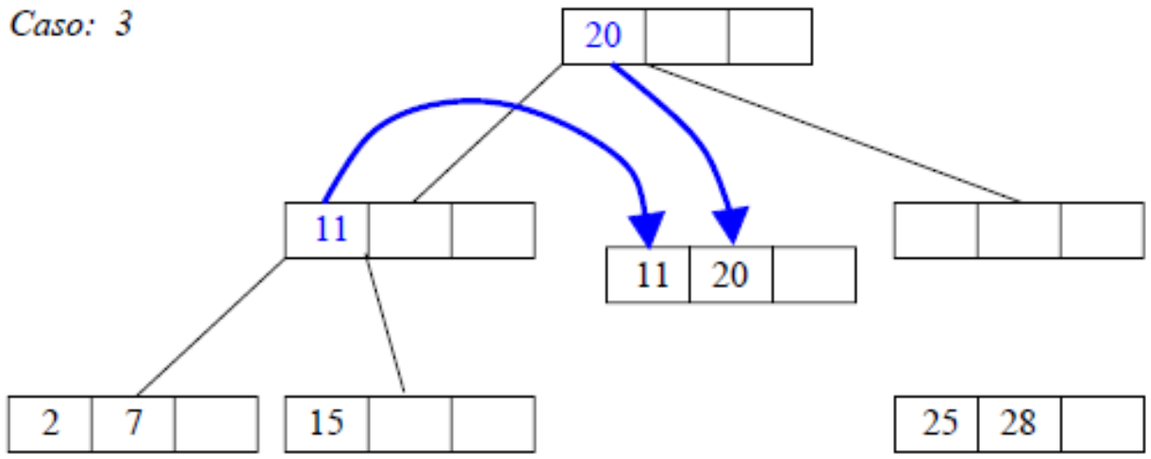
Operaciones para la unión tras el borrado de la clave 18



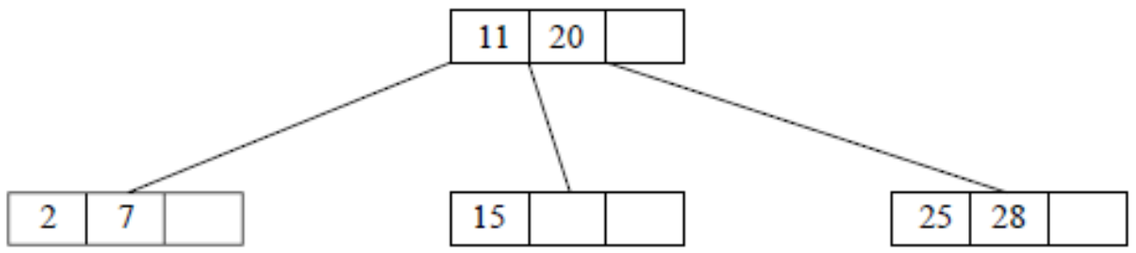
Resultado de la unión de claves

Borrado

Caso: 3



OPERACIONES PARA LA UNIÓN POR EL DESAJUSTE CREADO EN EL BORRADO DE LA CLAVE 18



ARBOL FINAL TRAS EL BORRADO DE LA CLAVE 18