7.5 Arboles equilibrados Red-Black

Es posible la construcción de árboles balanceados en los que cada nodo pueda tener más de dos hijos. Obviamente, este caso implica que el árbol ya no es binario. En esta sección vamos a ver los árboles red-black, que son de tipo binario pero que pueden interpretarse como la representación de árboles con nodos conteniendo la posibilidad de más de dos hijos, más concretamente de árboles 2-4. . Son la base de los tipos <set> y <map> de la STL. En un árbol 2-4 todos los nodos tienen entre 2 y 4 hijos, y entre 1 y 3 llaves respectivamente (una llave por cada par de hijos). Por tanto existen 3 tipos de nodos:

1. Con una llave K y 2 hijos H_1 y H_2.
   a) Todas las claves en H_1 son menores o iguales que K y
   b) todas las de H_2 son mayores que K.
2. Con dos llaves K_1, K_2 y 3 hijos H_1, H_2, H_3.
   a) Todas las llaves en H_1 son menores o iguales que K_1
   b) todas las llaves en H_2 son mayores que K_1 y menores o iguales que K_2.
   c) Todas las llaves en H_3 con mayores que K_2.
3. Con tres llaves K_1, K_2, K_3 y 4 hijos H_1, H_2, H_3, H_4.
   La figura 7.24 muestra un ejemplo de descripción de los nodos. Los nodos a son nodos del tipo 1 los nodos b son del tipo 2 y los nodos c son del tipo 3

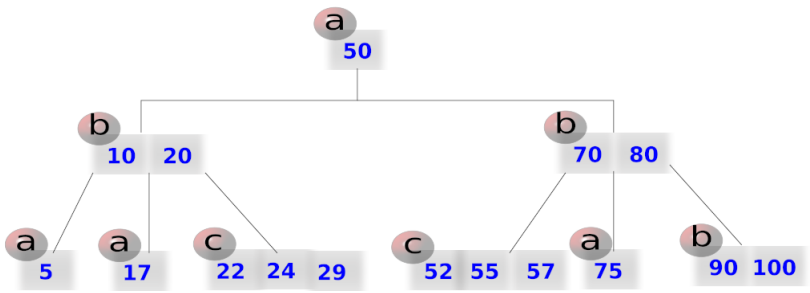

Figura 7.24: Nodos en un árbol 2-4

   a) Todas las llaves en H_1 son menores o iguales que K_1
   b) todas las llaves en H_2 son mayores que K_1 y menores o iguales que K_2.
   c) Todas las llaves en H_3 con mayores que K_2 y menores o iguales que K_3.
   d) Todas las llaves en H_4 son mayores que K_3.

Los elementos insertados en el árbol 2-4 están en las hojas y éstas se encuentran todas en un mismo nivel. Veamos cómo se realizan las operaciones de inserción y borrado:

- **Inserción:** Para insertar en un árbol 2-4: se busca el nodo en el que insertar el nuevo hijo. Si el nodo tiene 2 ó 3 hijos se añade uno más construyendo un nodo nuevo con 3 ó 4 hijos respectivamente. Si el nodo tiene 4 hijos primero es dividido de forma que la llave K_1 forma un nodo con 2 hijos, K_3 otro de dos hijos y finalmente K_2 sube para ser insertada en el nodo padre (esto puede provocar sucesivas divisiones en los padres hasta llegar al nodo raiz, el cual si se divide, provoca un aumento en el número de niveles, es decir, aumente la altura). El elemento que se pretendía insertar queda insertado en el correspondiente 2-nodo de entre los dos creados. Un ejemplo gráfico de la inserción lo podemos ver en la figura 7.25

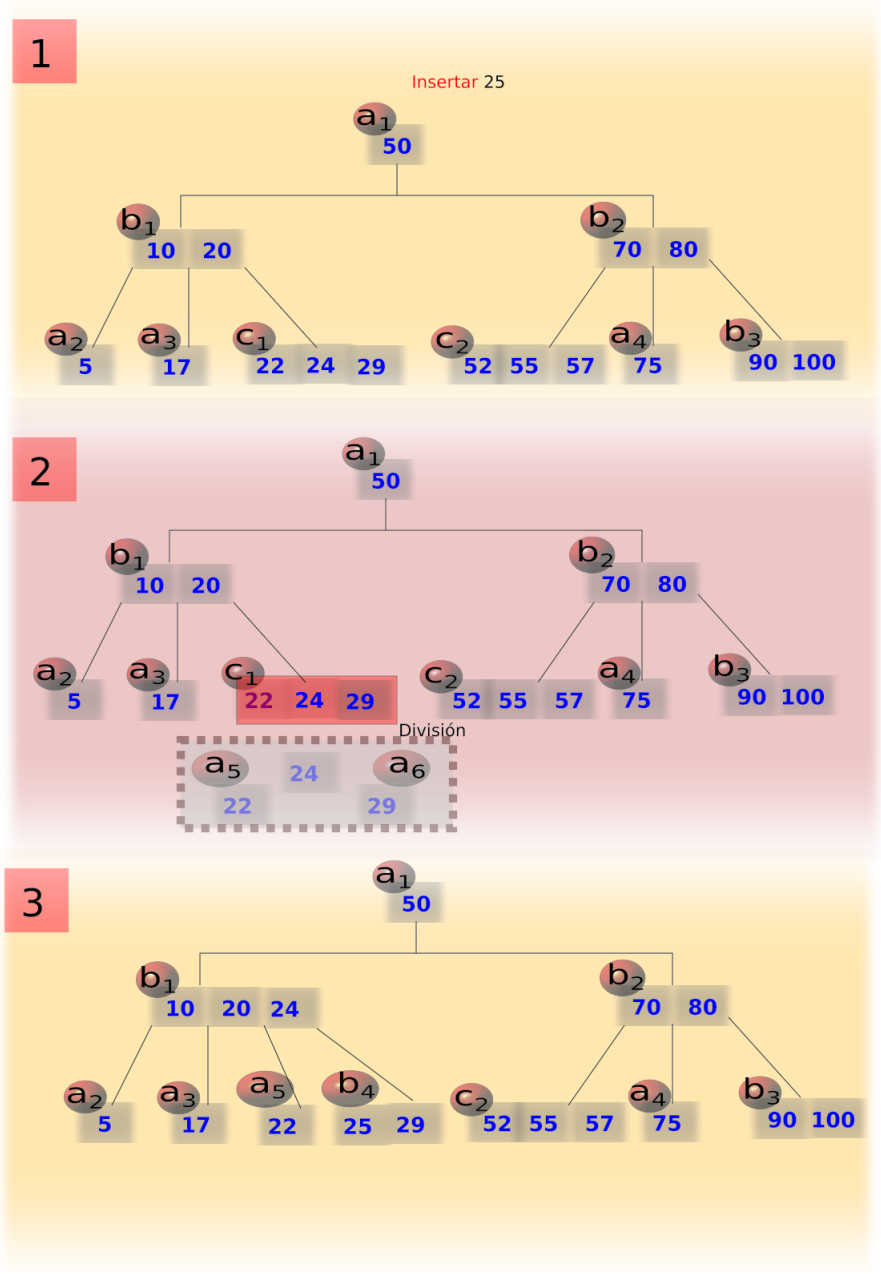

Figura 7.25: Nodos en un árbol 2-4

- **Borrado:** Para borrar en un árbol 2-4 se realiza de forma paralela. En este caso, si un nodo se queda con un sólo hijo debe producirse un reagrupamiento o incluso una unión de nodos, la cual también se propagará hacia la raíz si es necesario (si los hijos de la raíz se reagrupan en un solo nodo la altura del árbol se ve disminuída).
  Un ejemplo gráfico de la borrado lo podemos ver en la figura 7.26

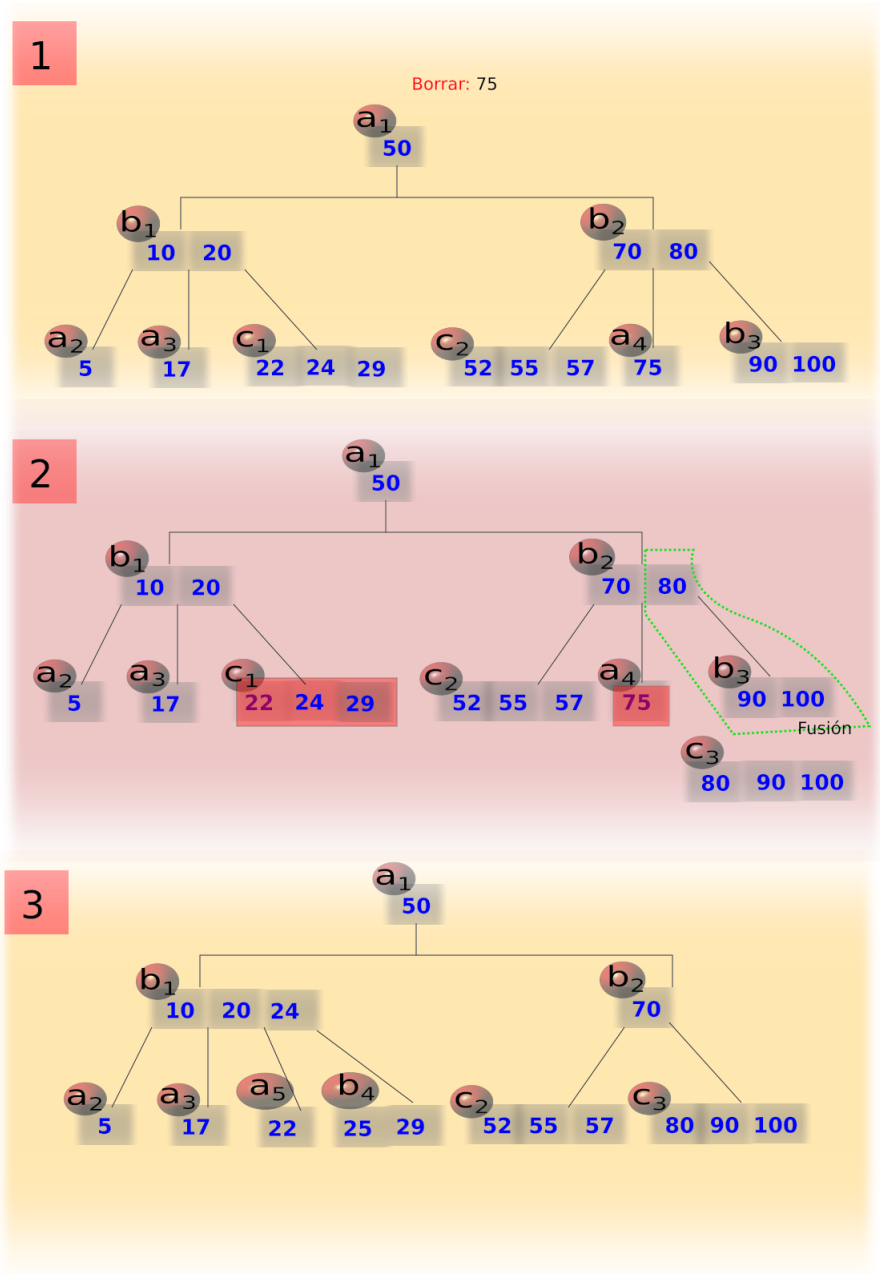

Figura 7.26: Nodos en un árbol 2-4

Veamos como podemos utilizar los árboles binarios para representar estos árboles. Obviamente, cada nodo en un árbol binario sólo puede tener 2 hijos y por consiguiente los nodos de este tipo en el árbol 2-4 que aparece a continuación son directamente representables. Pero, ¿qué hacer con 3 ó 4 hijos?. Sólo es posible representar dichos nodos con varios nodos del árbol binario tal como muestra la figura 7.27

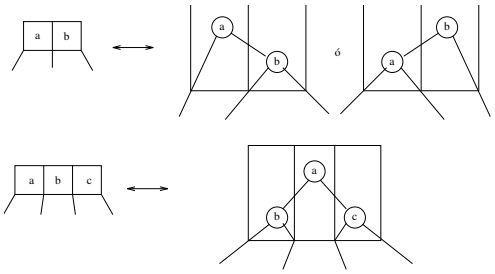

Figura 7.27: Representación del árbol 2-4 mediante árboles binarios.

Como se puede observar, la solución se representa usando nodos con dos hijos y:

- Si el nodo a representar tiene 3 hijos coger una de las dos ramas y añadirle otro *nodo auxiliar* de forma que de la estructura final terminan colgando 3 ramas al igual que el 3-nodo (ver figura 7.27 arriba).
- Si el nodo a representar tiene 4 hijos, al nodo binario se cuelgan 2 *nodos auxiliares* en cada rama de forma que de la estructura final cuelgan 4 ramas al igual que el 4-nodo (ver figura 7.27 abajo).

En definitiva, para conseguir la representación bastará con señalar en el árbol binario los nodos que se considerarán auxiliares, es decir, aquellos que junto con sus padres forman una estructura correspondiente a un 3-nodo o a un 4-nodo. Los árboles red-black realizan esta tarea coloreando de rojo (red) los *nodos auxiliares* y de negro (black) el resto. Podemos derivar así la definición de los árboles red-black: es un árbol binario de búsqueda cuyos nodos son coloreados rojos o negros según las 3 reglas siguientes:

1. Todas las hojas y la raíz son negras.
2. Los hijos de cualquier nodo rojo son negros.
3. Todo camino desda la raíz a cualquier hoja pasa por el mismo número de nodos negros. Si este número es $h+1$, el árbol tiene una altura negra de h.

Se deja al lector el ejercicio de interpretar estas reglas según el paralelismo que existe entre este tipo de árboles y los árboles 2-4. Lógicamente, cualquier árbol red-black puede ser dibujado como su equivalente 2-4 y viceversa. Por ejemplo, el árbol 2-4 de la figura 7.28 (a) tendría como árbol red-black el que se muestra en la figura 7.28(b)

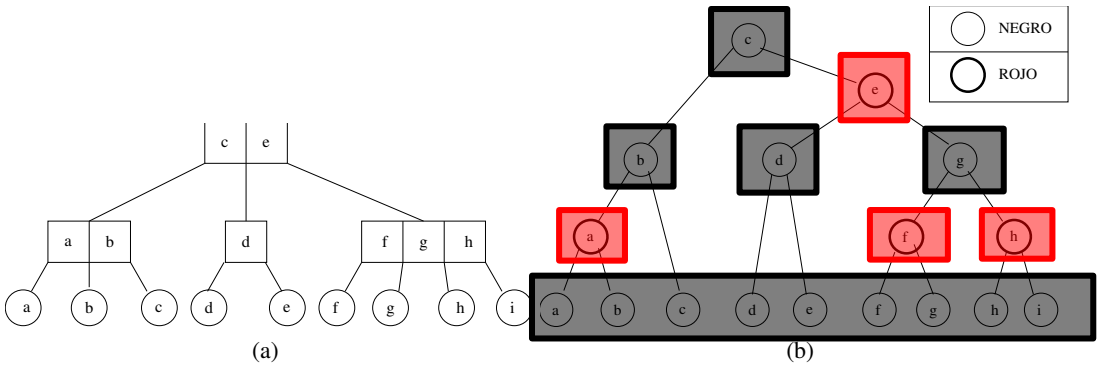

Figura 7.28: (a) Ejemplo de árbol 2-4. (b) Arbol red-black asociado.

Otro ejemplo de transformación de árbol 2-4 a red-black puede verse en la figura 7.29

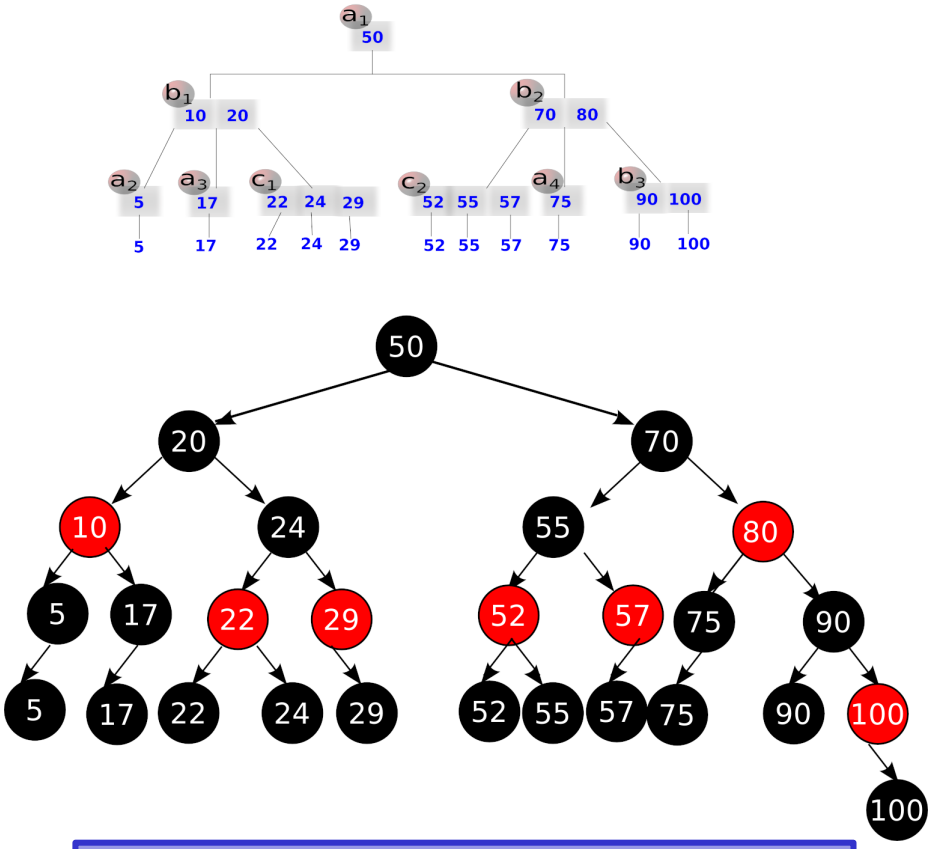

Figura 7.29: Ejemplo de transformación de árbol 2-4 a red-black

Las operaciones más complejas de implementar son la inserción y el borrado ya que la búsqueda de una clave resulta trivial. Las herramientas usadas para el mantenimiento del árbol son el recoloreado de nodos y las rotaciones estudiadas en la sección de árboles AVL. El árbol que se muestra en la figura 7.29 puede ser simplificado ya que como se puede observar las etiquetas o claves de búsqueda se encuentran duplicadas en los nodos interiores y en las hojas al igual que en el árbol 2-4 equivalente. Si en dichos árboles guardamos en cada uno de los nodos interiores las claves y la información deseada sin necesidad por tanto de duplicar información nos encontramos con una representación de árbol red-black que aunque no tiene todas sus hojas etiquetadas como negras es equivalente.

Veamos un ejemplo. Vamos a obtener el árbol red-black resultante de la inserción del conjunto de elementos {1,9,2,8,3,7} mediante la escritura paralela de los árboles 2-4 equivalentes.

La secuencia de pasos para construir el árbol se muestra en la figura 7.30.

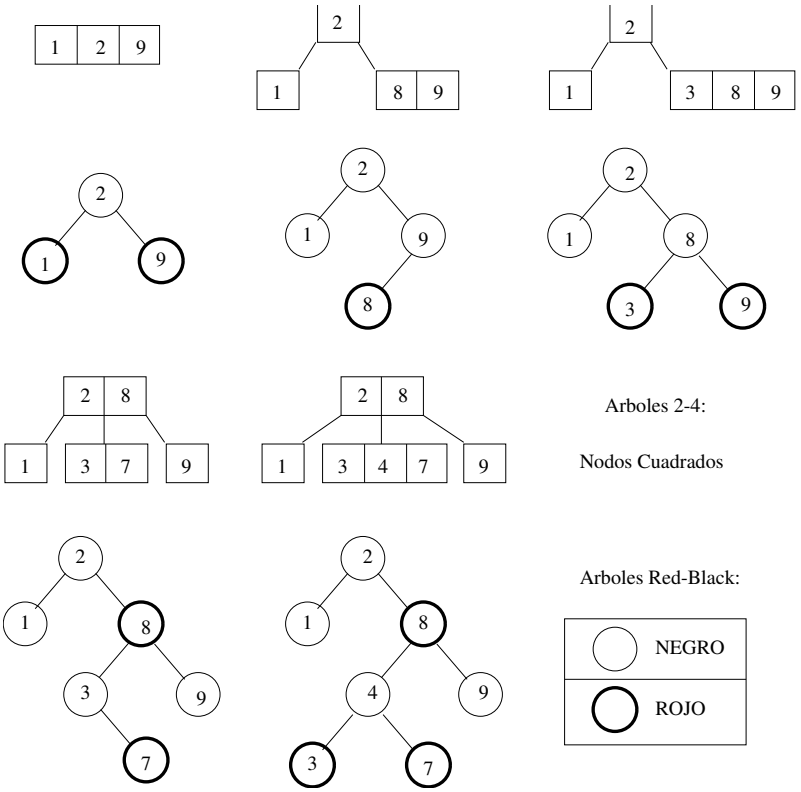

Figura 7.30: Inserción de 1,9,2,8,3,7 en un Red-Black.

7.6 Otros tipos de árboles equilibrados

7.6.1 Introducción

Los B-árboles surgieron en 1972 creados por R. Bayer y E. McCreight. El problema original comienza con la necesidad de mantener índices en almacenamiento externo para acceso a bases de datos, es decir, con el grave problema de la lentitud de estos dispositivos 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.

Como se ha visto anteriormente existen métodos y estructuras de datos que permiten realizar una búsqueda dentro de un conjunto alto de datos en un tiempo de orden $O(log_2 n)$. Así tenemos el caso de los árboles binarios AVL. ¿Por qué no usarlos para organizar a través de ellos el índice de una base de datos? la respuesta aparece si estudiamos más de cerca el problema de acceso a memoria externa. Mientras que en memoria interna el tiempo de acceso a n datos situados en distintas partes de la memoria es independiente de las direcciones que estos ocupen ($n \cdot cte$ donde cte es el tiempo de acceso a 1 dato), en memoria externa es fundamental el acceder a datos situados en el mismo bloque para hacer que el tiempo de ejecución disminuya debido a que el tiempo depende fuertemente del tiempo de acceso del dispositivo externo, si disminuimos el número de accesos a disco lógicamente el tiempo resultante de ejecución de nuestra búsqueda se ve fuertemente recortado. Por consiguiente, si tratamos de construir una estructura de datos sobre disco es fundamental tener en cuenta que uno de los factores