

Tema 6

ÁRBOLES AVANZADOS

Tutor: Enrique Carmona
Asignatura: Estructuras de Datos y Algoritmos

TEMA VI : ÁRBOLES AVANZADOS

6.1 Árboles multicamino

6.2 Árboles B

6.3 Árboles B Binarios (árboles BB)

6.4 Árboles B Binarios Simétricos (árboles BBS)

Árboles Multicamino:

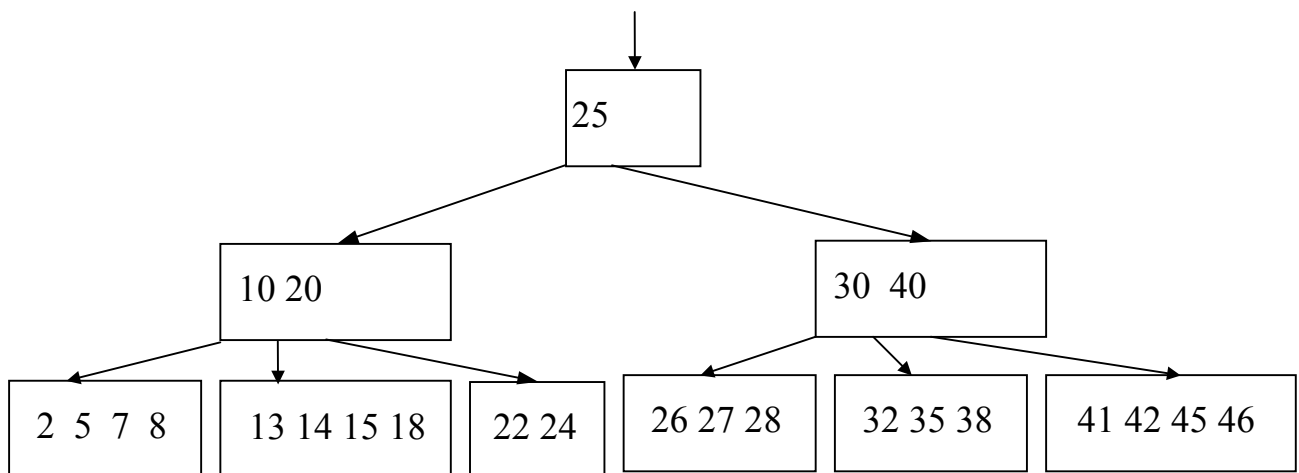
- **Definición:** Árbol que contiene nodos con más de dos ramas.
- **Aplicaciones:** Muy utilizados en la construcción y mantenimiento de árboles de búsqueda a gran escala guardados en *memoria secundaria*, en los que *se realizan con frecuencia inserciones y supresiones*.
- **Idea básica:**
 - Un árbol se subdivide en subárboles (páginas) y éstos se representan como unidades a las que se accede simultáneamente.
 - Se aprovecha las características de direccionamiento mediante paginación de la memoria secundaria y se consigue un ahorro en el número de accesos a la misma.
- **Problemas:**
 - (1) ¿De cuántos nodos máximo considero cada página?
 - (2) Es necesario establecer un plan de crecimiento controlado del árbol (mínima profundidad) para evitar que este crezca aleatoriamente (podría, en el peor de los casos, haber un nodo por página)
- **Solución:** Árboles B multicamino o, simplemente, árboles B.

Árboles B

➤ **Definición:** Un árbol B de *orden n* es aquel árbol de búsqueda que satisface las siguientes propiedades:

1. Cada *página* contiene *máximo $2n$ llaves* y *mínimo n llaves*, excepto la raíz (que puede contener sólo una).
2. Cada página o bien es una página hoja (no tiene descendientes) o bien tiene $m+1$ páginas descendientes, siendo m el nº de llaves en esta página.
3. Todas las páginas hoja aparecen al mismo nivel.

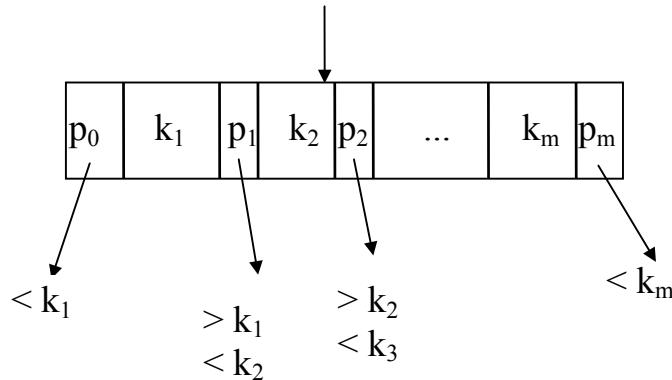
- Ejemplo:



(Árbol B de orden 2 con 3 niveles)

Búsqueda en árboles B:

- Representación de una página (m llaves y $m+1$ punteros):

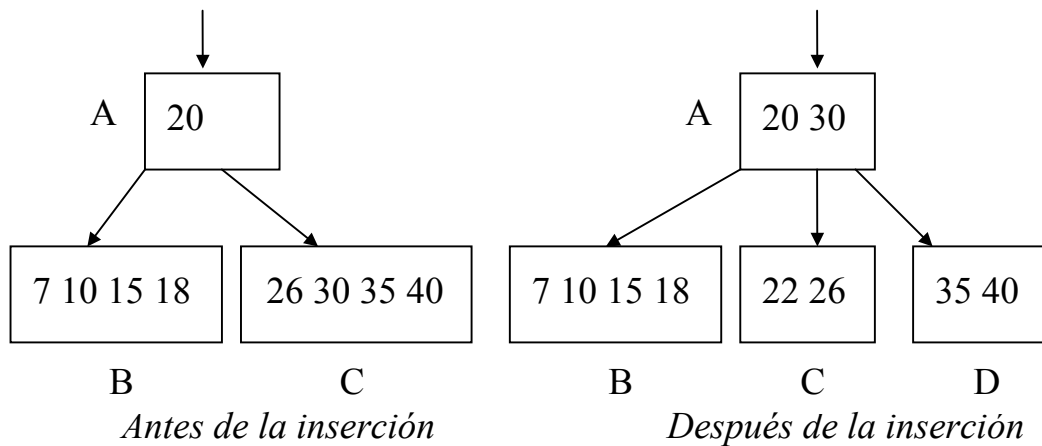


- Las llaves están ordenadas de izquierda a derecha en cada página y en cada nivel.
- Sea x : argumento de búsqueda, la búsqueda en una página se realiza como sigue:
- Una vez cargada en memoria principal la página:
 - Si m es grande: búsqueda binaria
 - Si m es pequeña: búsqueda secuencial
 - Si la búsqueda en esta página fracasa, entonces:
 - Si $k_i < x < k_{i+1}$, para $1 \leq i \leq m \Rightarrow$ Proseguimos la búsqueda en la página apuntada por p_i
 - Si $k_m < x \Rightarrow$ Búsqueda prosigue en pág. apuntada por p_m
 - Si $x < k_1 \Rightarrow$ Búsqueda prosigue en pág. apuntada por p_0
 - Si apuntador $p_i = \text{NIL} \Rightarrow x$ no está en el árbol \Rightarrow fin de la búsqueda.

Inserción en árboles B:

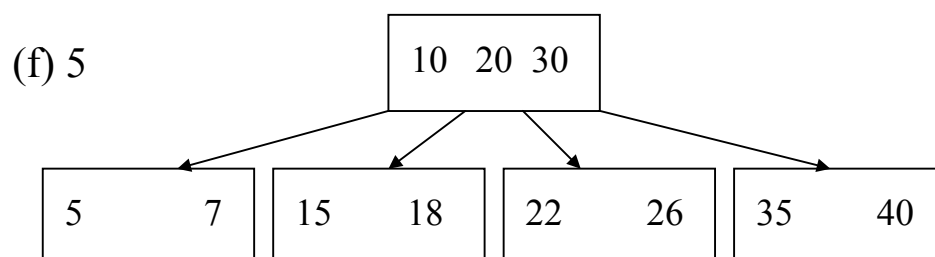
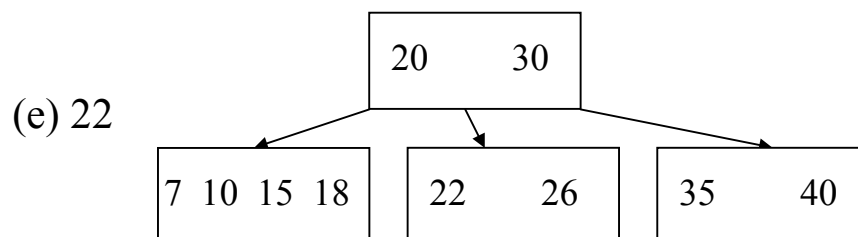
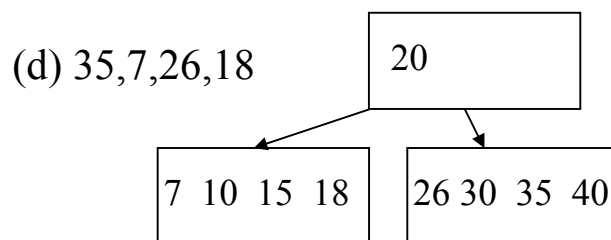
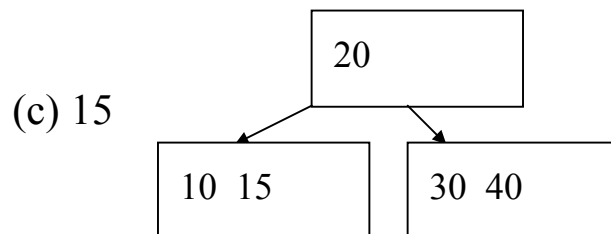
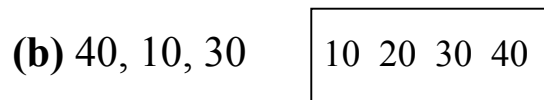
- Si se inserta en una página con $m < 2n$ llaves, el proceso de inserción queda limitado a esa página.
- La inserción en una página ya llena es la que puede ocasionar modificaciones en la estructura del árbol \Rightarrow creación de páginas nuevas.

➤ ***Ejemplo de inserción:*** Inserción de la llave 22 en un árbol B

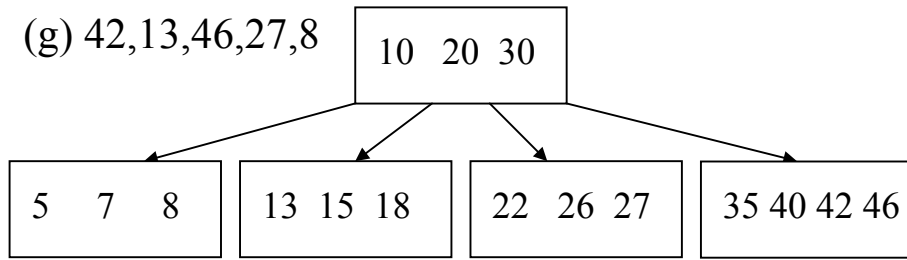


1. La inserción le correspondería en la página C pero es imposible porque C está llena.
 2. La página C se divide en 2 páginas (C y la nueva pág. D).
 3. Las $2n+1$ llaves se distribuyen uniformemente en C y D, y la llave que ocupa la posición intermedia se sube hacia la página padre A.
- Este método de inserción tiene que conservar siempre las propiedades de un árbol B \Rightarrow Puede ocurrir que esta división de páginas tenga que propagarse hasta la raíz.
- Manera peculiar de crecer: de las hojas a la raíz.

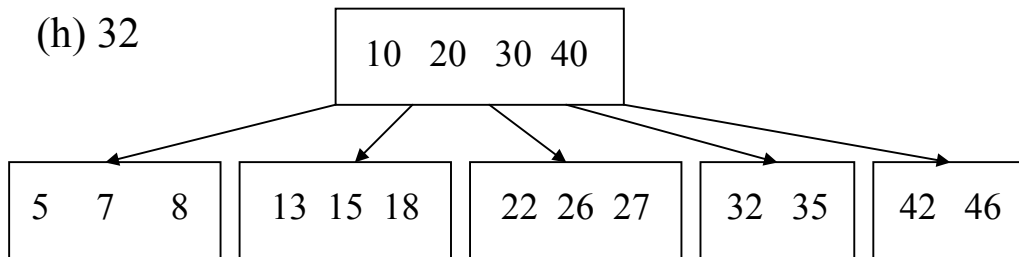
- **Ejemplo.** Construcción de árbol B de orden dos ($n=2$) con la siguiente inserción de llaves:
 20; 40 10 30; 15; 35 7 26 18; 22; 5; 42 13 46 27 8; 32; 38 24 45; 25



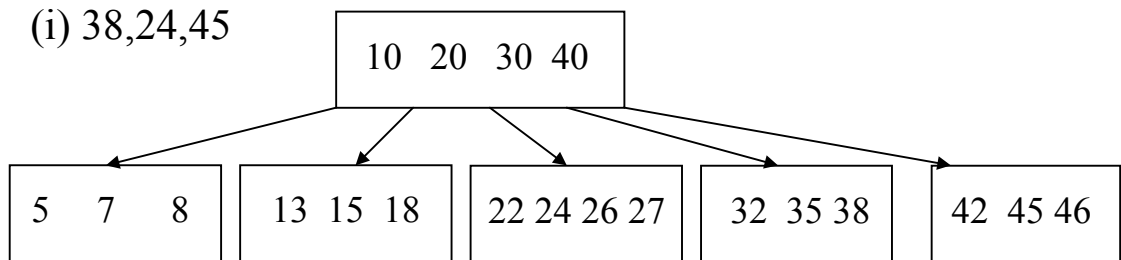
(g) 42,13,46,27,8



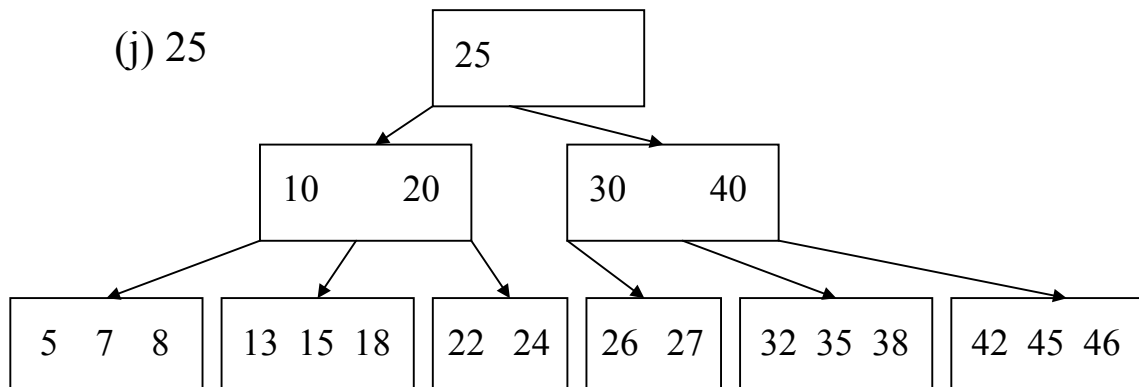
(h) 32



(i) 38,24,45



(j) 25



Eliminación en un árboles B:

➤ Se pueden distinguir 2 casos:

1. El elemento a suprimir se encuentra en una página hoja:
Proceso de eliminación inmediato.
2. El elemento no se encuentra en una página hoja => Se usa la misma estrategia que en los árboles binarios de búsqueda: tomar el elemento mas a la derecha de la página hoja del subárbol izquierdo, o el de más a la izquierda de la página hoja del subárbol derecho.

➤ Para resolver el caso 2:

Igual que
en los
árboles de
búsqueda

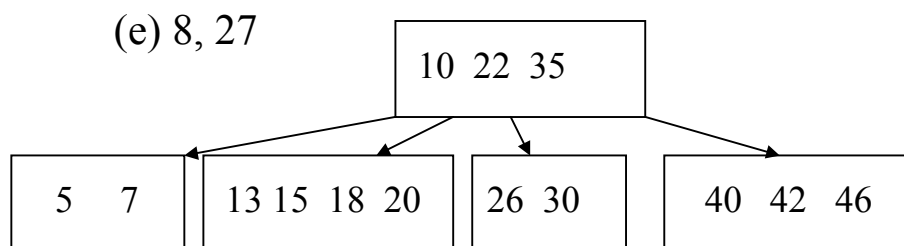
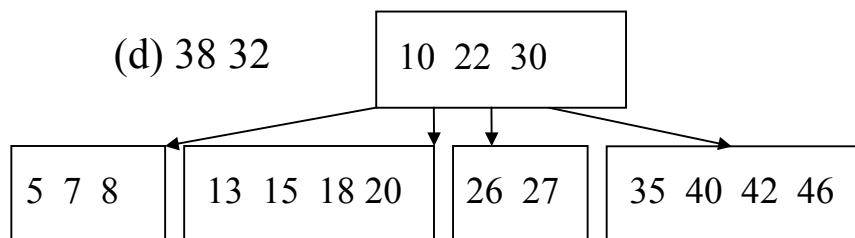
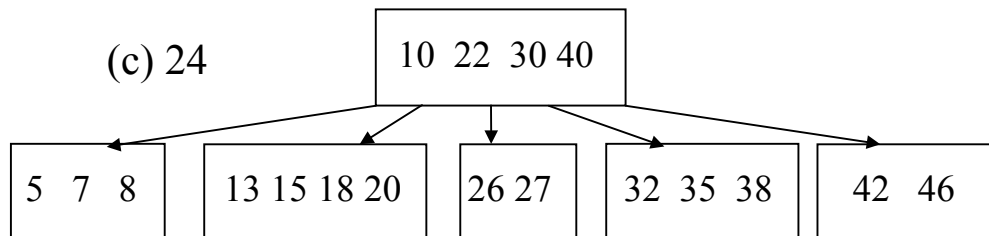
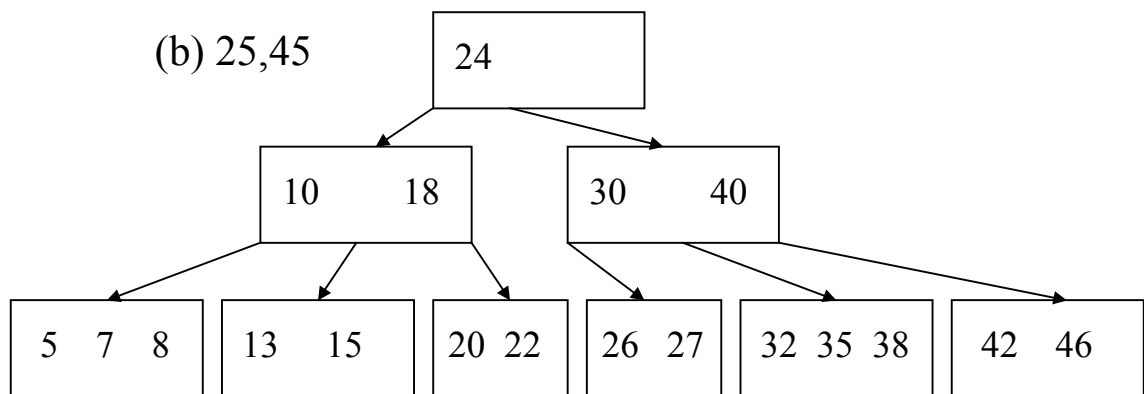
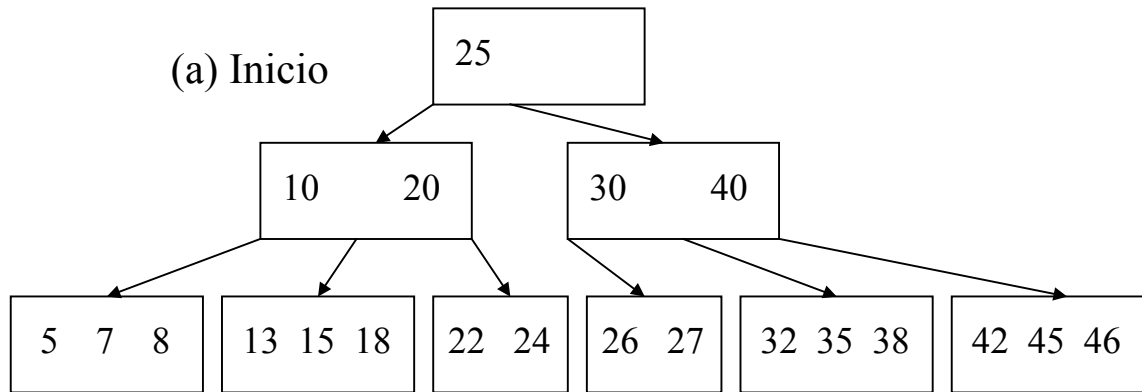
- 2.1. Bajar por los apuntadores situados al extremo derecho hasta la página hoja P.
- 2.2. Reemplazar el elemento a eliminar con el extremo derecho de P y reducir en 1 el tamaño de P.
- 2.3. Verificar el n° de elementos de la página: no puede ser que $m < n$ por ser árbol B (***Subocupación***).

➤ Estrategia habitual para solucionar la ***Subocupación***: ***Balanceo de Páginas***, distribuir los elementos de la página en cuestión P y de su página vecina Q de manera uniforme en ambas (ojo con el elemento pivote de la página madre de P y Q).

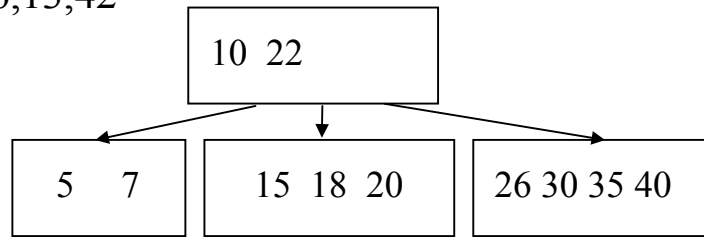
➤ Problema: Si Q ya tuviera tamaño mínimo n, el n° total de elementos en P y Q es $2n-1$ => Solución: *Combinar* P y Q en una sola página, agregando el elemento intermedio de la página madre de P y Q. De la página Q se prescinde. Esta combinación de páginas puede propagarse hasta llegar a la raíz => única forma de que un árbol B disminuya de altura.

➤ **Ejemplo:** Eliminación secuencial de las llaves:

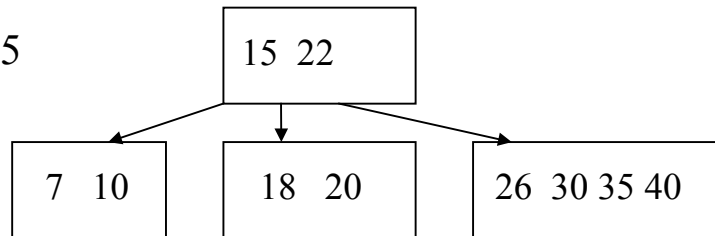
25, 45, 24, 38, 32, 8, 27, 46, 13, 42, 5, 22, 18, 26, 7, 35, 15



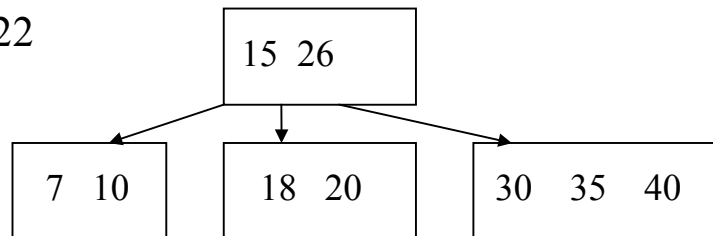
(d) 46,13,42



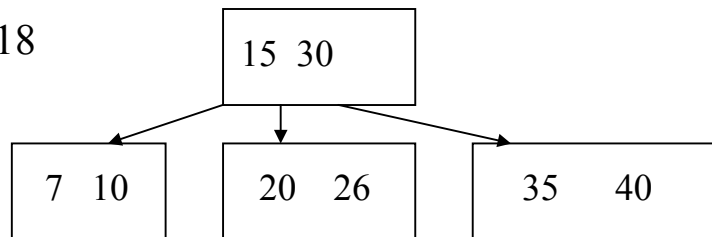
(e) 5



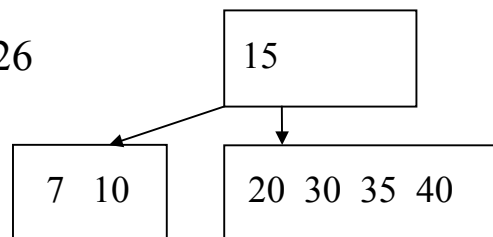
(f) 22



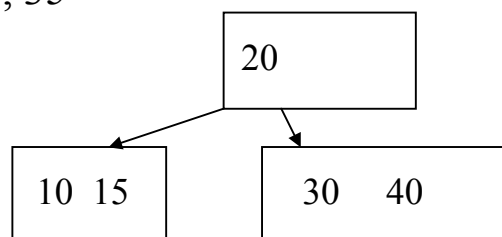
(g) 18



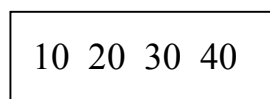
(h) 26



(e) 7, 35

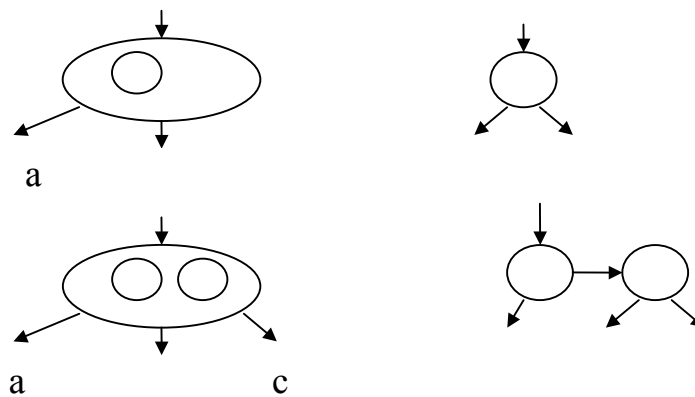


(f) 15



Árboles B binarios

- Un **árbol B binario** (*árbol BB*) es un árbol B de orden 1. Consta de nodos (páginas) con uno o dos elementos => cada página contiene 2 o 3 punteros a los descendientes (*árbol 2-3*).
- Todas las páginas hoja aparecen en el mismo nivel (árbol B) y todas las páginas no hoja tienen 2 ó 3 descendientes (incluida la raíz).
- Se utilizan fundamentalmente en memoria primaria. En memoria secundaria tendrían una alta penalización de acceso.
- Representación de nodos de árbol BB:

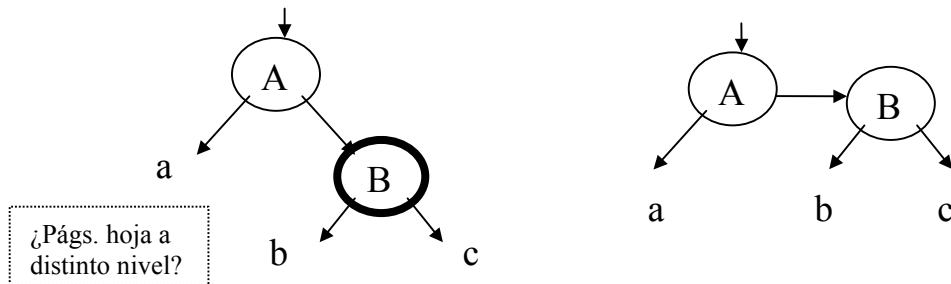


- Hay dos tipos de apuntadores:
 - 1) A los descendientes => verticales (izq. y dcho.)
 - 2) A los hermanos => horizontales (siempre a la derecha).
- Definición de un nodo de árbol:

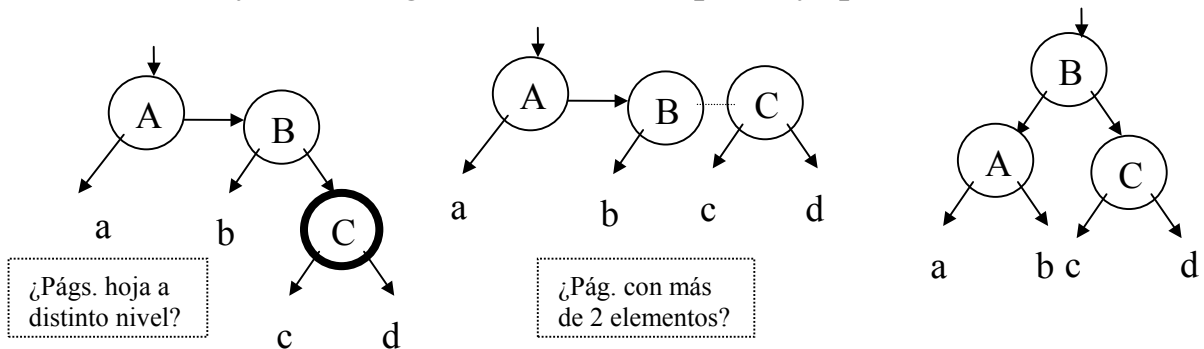
```
TYPE Ptr_Nodo = POINTER TO Nodo;  
TYPE Node = RECORD  
    llave: INTEGER;  
    izq, dch: Ptr_Nodo;  
    h: BOOLEAN (*rama horiz. a la derch.*)  
END
```

Inserción de llaves en un árbol BB (4 posibilidades)

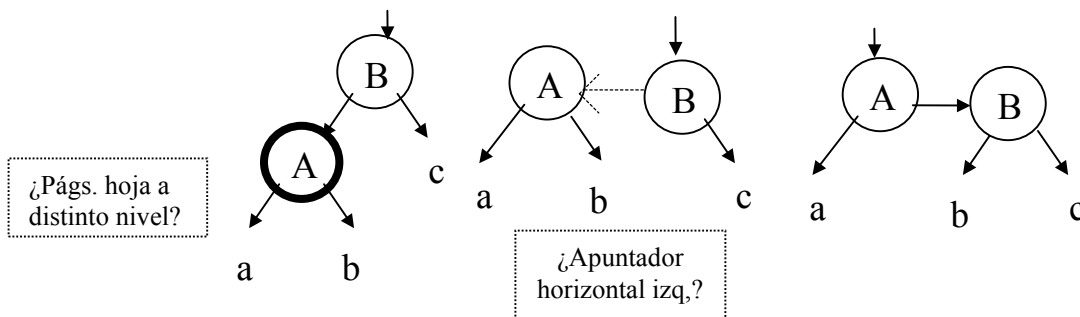
1) Crece el subárbol a la derecha de un nodo A, siendo A la única llave en su página:



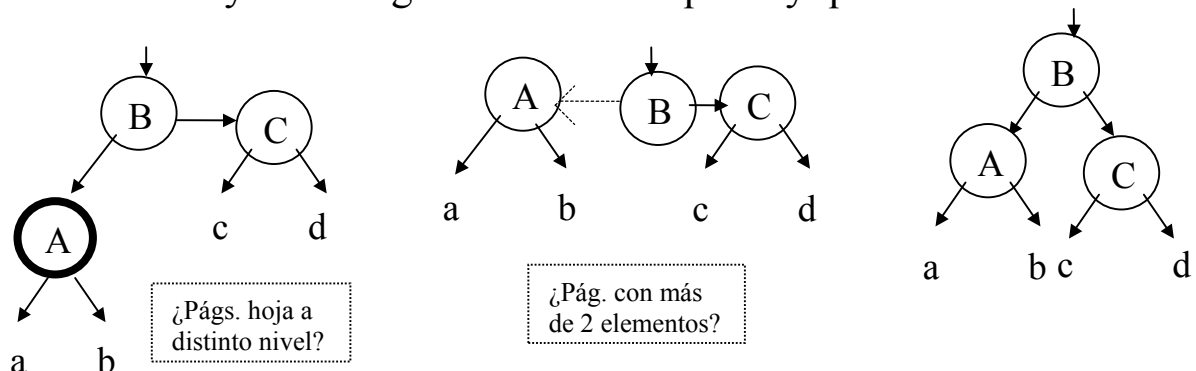
2) Crece el subárbol a la derecha de un nodo que posee dos hermanos A y B \Rightarrow Página de 3 nodos que hay que dividir.



3) Crece el subárbol a la izquierda de un nodo B, siendo B la única llave en su página \Rightarrow Rotación de apuntadores



4) Crece el subárbol a la izquierda de un nodo que tiene dos hermanos B y C \Rightarrow Página de 3 nodos que hay que dividir.



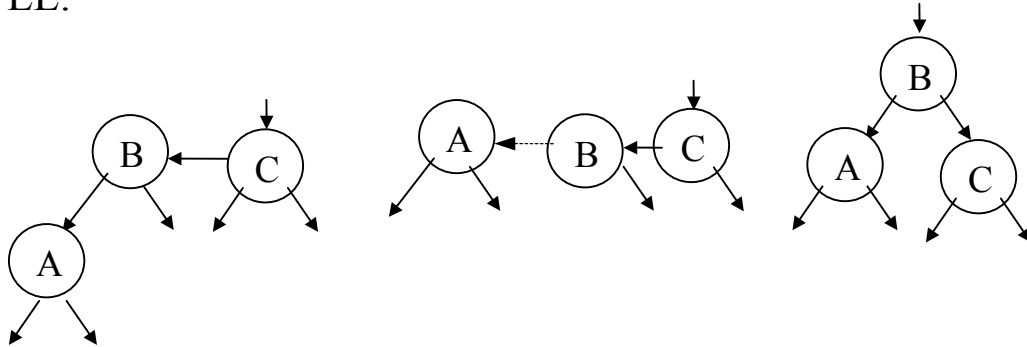
Arbol B binario Simétrico (árbol BBS)

- Surgen con la idea de garantizar un crecimiento simétrico de los subárboles derecho e izquierdo del árbol. En general conducen a árboles de búsqueda ligeramente más eficientes pero los algoritmos de inserción y eliminación son más complejos.
- Se definen como árboles de búsqueda que satisfacen las siguientes propiedades:
 1. Todo nodo contiene una llave, y como máximo dos apuntadores a subárboles.
 2. Todo apuntador es horizontal (*derecho o izquierdo*) o vertical. No existen dos apuntadores consecutivos horizontales (en el mismo sentido) en ninguna trayectoria de búsqueda. *En cambio, si se permite que un nodo presente un apuntador derecho e izquierdo.*
 3. Todos los nodos terminales aparecen al mismo nivel.
- En los árboles BBS se cumple que la trayectoria de búsqueda más larga no es mayor que el doble de la altura del árbol. Si la altura máxima en este tipo de árboles no puede ser mayor de $\log n$ (siendo n el número de nodos del árbol), entonces la longitud de trayectoria de búsqueda es como máximo:

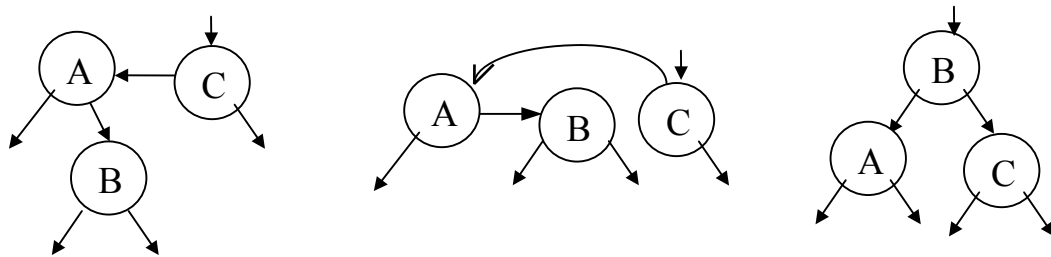
$$2^{\lceil \log n \rceil}$$

- En la inserción en árboles BBS pueden darse cuatro casos en los que es necesario rebalanceo para que se sigan cumpliendo sus propiedades:

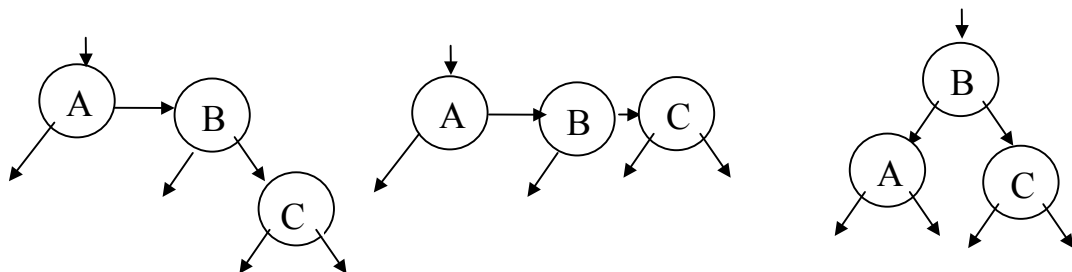
1) LL:



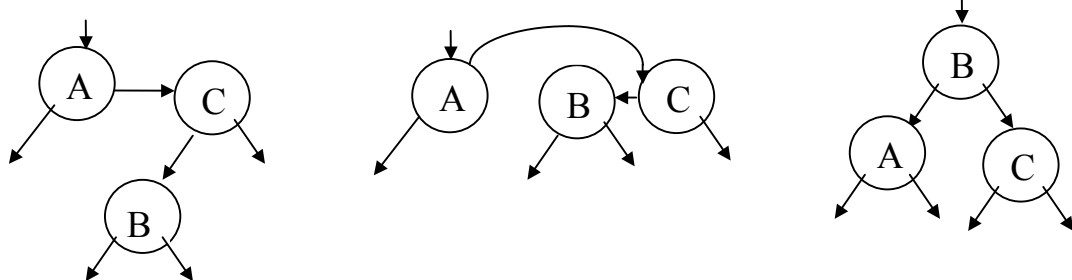
2) LR



3) RR



4) RL



➤ Declaración del nodo de un árbol BBS:

```
TYPE Ptr_Nodo=POINTER TO Nodo;  
TYPE Nodo = RECORD  
    llave: INTEGER;  
    cont: CARDINAL;  
    izq, dch: Ptr_Nodo;  
    h_izq, h_dch: BOOLEAN  
END;
```

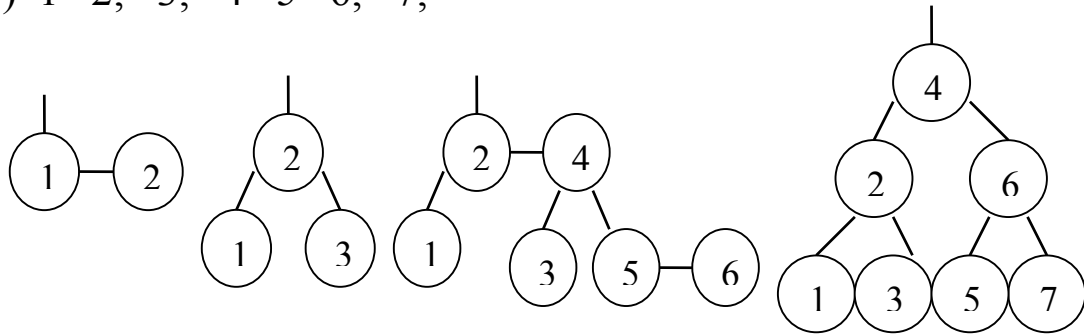
- La variable h se utiliza con el siguiente significado:
 1. Si $h=0$, el subárbol p no requiere cambios en su estructura
 2. Si $h=1$, el nodo p ha obtenido un hermano
 3. Si $h=2$, el subárbol p ha aumentado de altura
- Las variables booleanas h_izq y h_dch se utilizan para indicar la naturaleza del enlace, horizontal (*true*) o vertical (*false*), de los apuntadores izquierdo y derecho respectivamente.

Procedimiento para inserción en árboles BBS

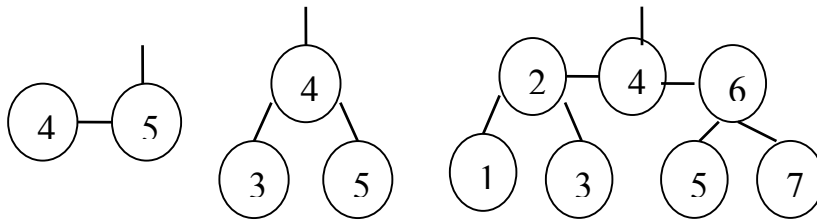
```
PROCEDURE BBS(x:INTEGER;VAR p:Ptr_Nodo;VAR h:CARDINAL);
  VAR p1, p2: Ptr_Nodo;(* h=0 *)
  BEGIN
    IF p=NIL THEN (* insertar *)
      ALLOCATE(p,SIZE(Nodo)); h:=2;
      WITH p^ DO
        llave:=x;cont:=1;izq:=NIL;dch:=NIL;h_izq:=FALSE;h_dch:=FALSE
      END
    ELSIF p^.llave>x THEN BBS(x, p^.izq, h);
      IF h>0 THEN (* la rama izquierda ha crecido *)
        IF p^.h_izq THEN
          p1:=p^.izq; h:=2; p^.h_izq:=FALSE;
          IF p1^.h_izq THEN (* LL *)
            p^.izq:=p1^.dch; p1^.dch:=p; p:=p1; p^.h_izq:=FALSE
          ELSIF p1^.h_dch THEN (* LR *)
            p2:=p1^.dch; p1^.dch:=p2^.izq; p2^.izq:=p1;
            p^.izq:=p2^.dch; p2^.dch:=p; p:=p2;
            p1^.h_dch:=FALSE
          END
        ELSE h:=h-1;
          IF h>0 THEN p^.h_izq:=TRUE END
        END
      END
    ELSIF p^.llave<x THEN BBS(x,p^.dch,h);
      IF h>0 THEN (* la rama derecha ha crecido *)
        IF p^.h_dch THEN
          p1:=p^.dch; h:=2; p^.h_dch:=FALSE;
          IF p1^.h_dch THEN (* RR *)
            p^.dch:=p1^.izq; p1^.izq:=p; p:=p1;p^.h_dch:=FALSE
          ELSIF p1^.h_izq THEN (* RL *)
            p2:=p1^.izq; p1^.izq:=p2^.dch; p2^.dch:=p1;
            p^.dch:=p2^.izq; p2^.izq:=p; p:=p2;
            p1^.h_izq:=FALSE
          END
        ELSE h:=h-1;
          IF h>0 THEN p^.h_dch:=TRUE END
        END
      END
    ELSE p^.cont:=p^.cont+1; (*encontrado*)
  END
END BBS;
```

Ejemplos de crecimiento de árboles BBS

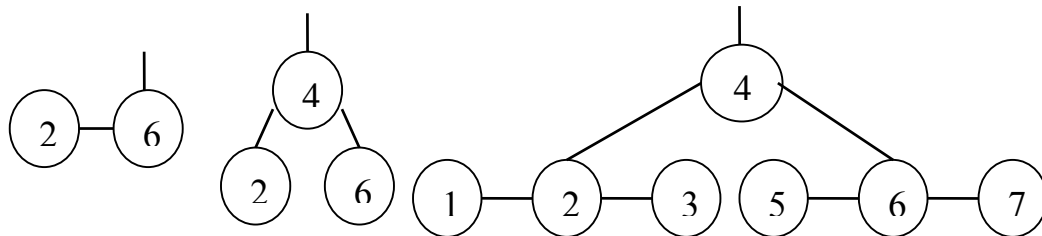
1) 1 2; 3; 4 5 6; 7;



2) 5 4; 3; 1 2 7 6;



3) 6 2; 4; 1 7 3 5;



4) 4 2 6; 1 7; 3 5;

