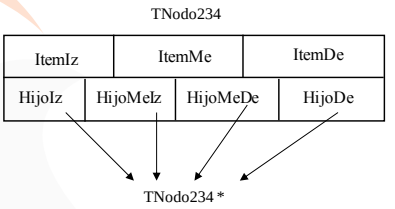
3.4. Árboles 2-3-4

DEFINICIONES

- Un árbol 2-3-4 es un árbol que está vacío o satisface las siguientes propiedades:
  - Los nodos pueden tener 2, 3 ó 4 hijos (2-nodo, 3-nodo ó 4-nodo)
  - Cumple las propiedades de árbol multicamino de búsqueda
  - Todas las hojas están en el mismo nivel
- Representación




```
class TArb234 {
    public:
    . . . . .
    private:
        TNodo234 * farb;
};
```


3.4. Árboles 2-3-4

OPERACIONES BÁSICAS. PROPIEDADES

- Operaciones básicas:
  - Búsqueda (similar a los árboles multicamino de búsqueda)
  - Inserción (se realiza en las hojas. Se pueden producir reestructuraciones del árbol)
  - Borrado (se realiza en las hojas. Se pueden producir reestructuraciones del árbol)
- Propiedades:
  - En un árbol 2-3-4 de altura h tenemos:

    2^h - 1 elementos si todos los nodos son del tipo 2-nodo

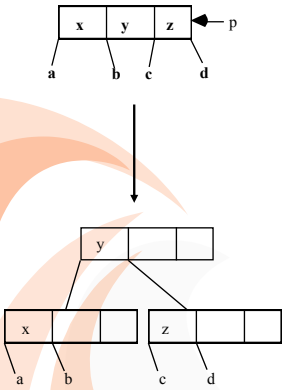
    4^h - 1 elementos si todos los nodos son del tipo 4-nodo

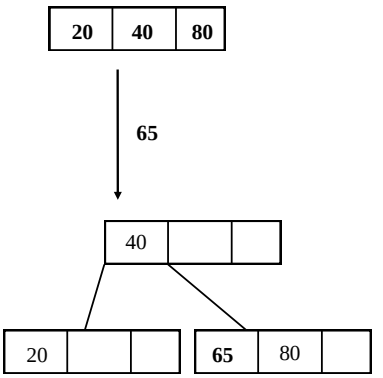
    por lo que la altura de un árbol 2-3-4 con *n* elementos se encuentra entre los límites: $\log_4 (n+1)$ y $\log_2 (n+1)$
  - Las reestructuraciones se realizan desde la raíz hacia las hojas

3.4. Árboles 2-3-4

OPERACIONES BÁSICAS. INSERCIÓN (I)

- Existen 3 situaciones en las que se puede encontrar un 4-nodo: (1)



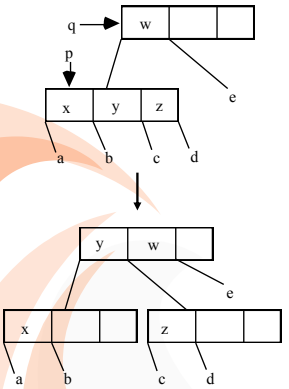


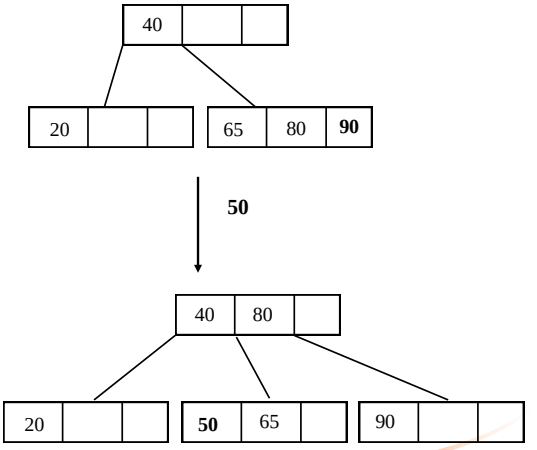
Es la raíz de un árbol 2-3-4:
(DIVIDERAIZ (p))

3.4. Árboles 2-3-4

OPERACIONES BÁSICAS. INSERCIÓN (II)

- Existen 3 situaciones en las que se puede encontrar un 4-nodo: (2)



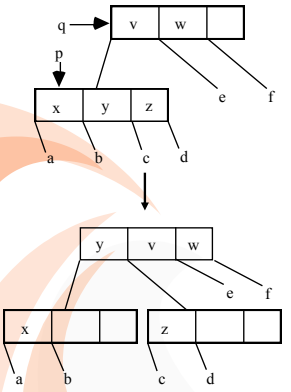


Su padre (q) es un 2-nodo:
(DIVIDEHIJODE2 (p, q))

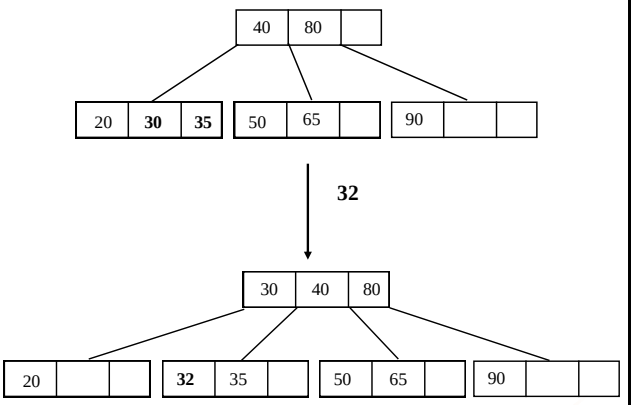
3.4. Árboles 2-3-4

OPERACIONES BÁSICAS. INSERCIÓN (III)

- Existen 3 situaciones en las que se puede encontrar un 4-nodo: (3)



Su padre (q) es un 3-nodo:
(DIVIDEHIJODE3 (p, q))



3.4. Árboles 2-3-4

OPERACIONES BÁSICAS. INSERCIÓN (IV)

ALGORITMO insertar (A: TArb234, y: item)

si raíz es 4-nodo → DIVIDERAIZ

si en el camino hasta la hoja me encuentro un 4-nodo → DIVIDEHIJO

3.4. Árboles 2-3-4

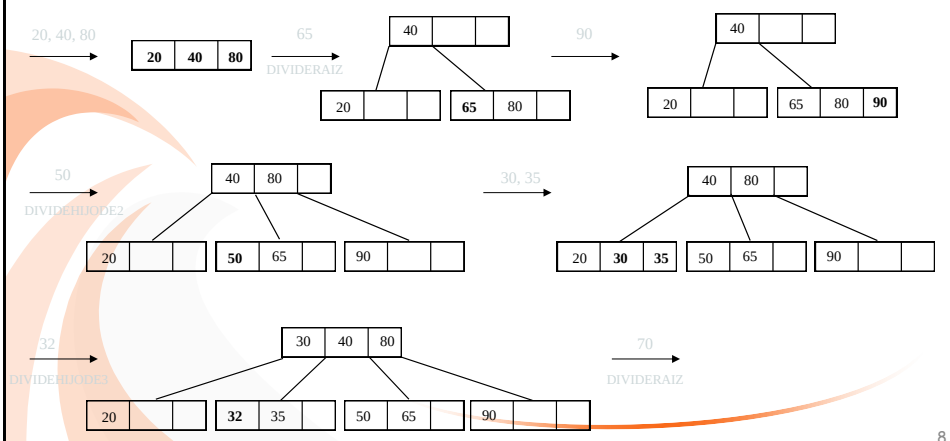
OPERACIONES BÁSICAS. INSERCIÓN (V)

```
ALGORITMO insertar (A: TArb234, y: item)
VAR p, q: TNodo234*; noencontrado: Boolean; B: TArb234; FVAR
      p = A.farb; q = p;
      si EsVacío( A ) entonces A = ENRAIZAR(A, y, B)
      sino
          si p es 4-nodo entonces DIVIDERAIZ( A ) fsi
          noencontrado = VERDADERO;
          mientras noencontrado hacer
            si p es 4-nodo entonces
              si q es 2-nodo entonces DIVIDEHIJODE2( p, q );
              sino DIVIDEHIJODE3( p, q ); fsi
              p = q;
            fsi
            caso de COMPARAR( y, p ):
              0:// Clave de y coincide con clave en p
                ERROR, ETIQUETA EXISTENTE;
              1:// p apunta a un nodo hoja
                PONER( y, p ); noencontrado = FALSO;
              2:// clave( y ) < ItemIz.clave( p )
                q = p; p = p → HiIz;
              3:// ItemIz.clave(p)<clave(y)<ItemMe.clave(p)
                q = p; p = p → HiMeIz;
              4://ItemMe.clave(p)<clave(y)<ItemDe.clave(p)
                q = p; p = p → HiMeDe;
              5:// clave(y) > ItemDe.clave(p)
                q = p; p = p → HiDe;
            fcaso
          fmientras
      fsi
FALGORITMO
```


3.4. Árboles 2-3-4

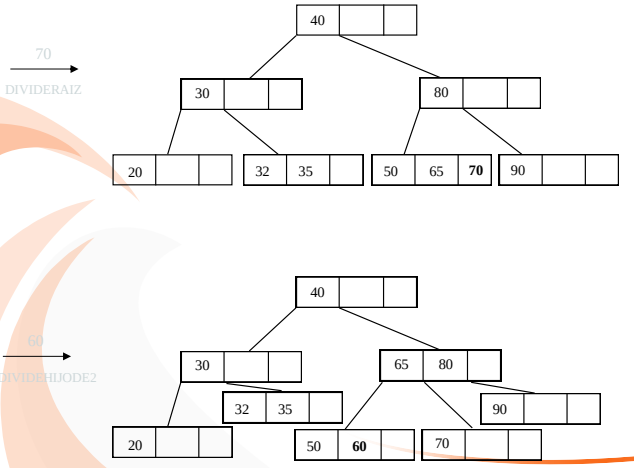
OPERACIONES BÁSICAS. INSERCIÓN. EJEMPLO (VI)

- **Ejemplo**. Insertar en un árbol 2-3-4 inicialmente vacío los siguientes items: 20, 40, 80, 65, 90, 50, 30, 35, 32, 70, 60



3.4. Árboles 2-3-4

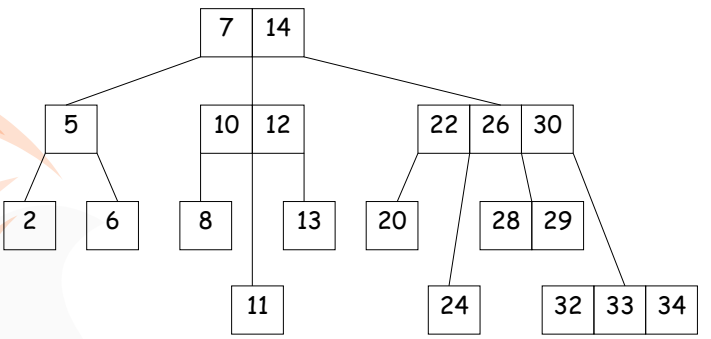
OPERACIONES BÁSICAS. INSERCIÓN. EJEMPLO (VII)



3.4. Árboles 2-3-4

EJERCICIOS *inserción*

1) Dado el siguiente árbol 2-3-4, insertar los elementos 21 y 35



3.4. Árboles 2-3-4

OPERACIONES BÁSICAS. BORRADO (I)

- Se reduce al borrado de un elemento en una hoja
- En el movimiento de búsqueda, cuando pasemos a un nodo en el siguiente nivel, éste nodo debe ser 3-nodo ó 4-nodo; si no es así (es 2-nodo) hay que reestructurar
  - **p =** nodo donde estamos
  - **q =** siguiente nodo en la búsqueda
  - **r =** uno de los nodos adyacentes a **q** (si hay dos adyacentes, escogemos **r** según criterio –hermano de la izquierda o hermano de la derecha–)

3.4. Árboles 2-3-4

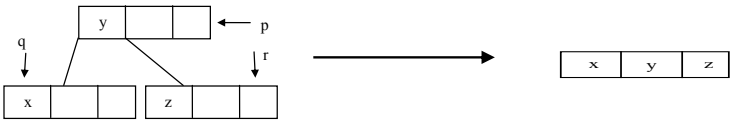
OPERACIONES BÁSICAS. BORRADO (II)

- Algoritmo de Borrado de 1 elemento
  - Comenzar ***p*** = raíz
  - Escoger ***q*** = siguiente en el camino de búsqueda
  - Reestructuraciones si procede //***q*** es 2-nodo
  - Mientras ***q*** no sea un nodo hoja ó ***q*** sea 2-nodo
    - ***p*** = ***q***
    - ***q*** = siguiente en el camino de búsqueda
    - Reestructuraciones si procede //***q*** es 2-nodo
  - Borrar el elemento //***q*** está en un nodo hoja y ***q*** no es 2-nodo
- Casos:
  1. **p es una hoja:** p sólo puede ser 2-nodo si es la raíz
  2. **q es 3-nodo ó 4-nodo:** la búsqueda continúa en q sin reestructurar

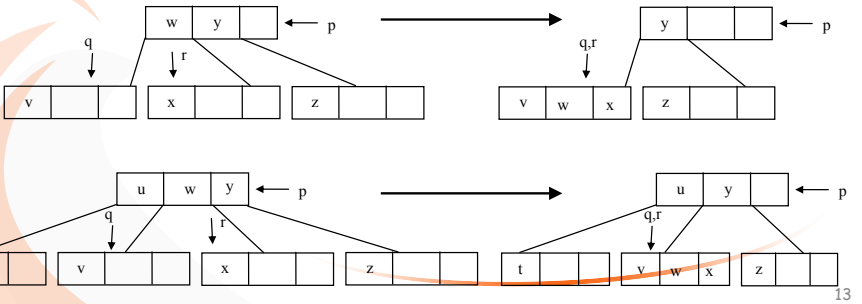
3.4. Árboles 2-3-4

OPERACIONES BÁSICAS. BORRADO (III)

3. **q es 2-nodo y r es 2-nodo (COMBINACIÓN):**
   1. **p es 2-nodo:** es la raíz



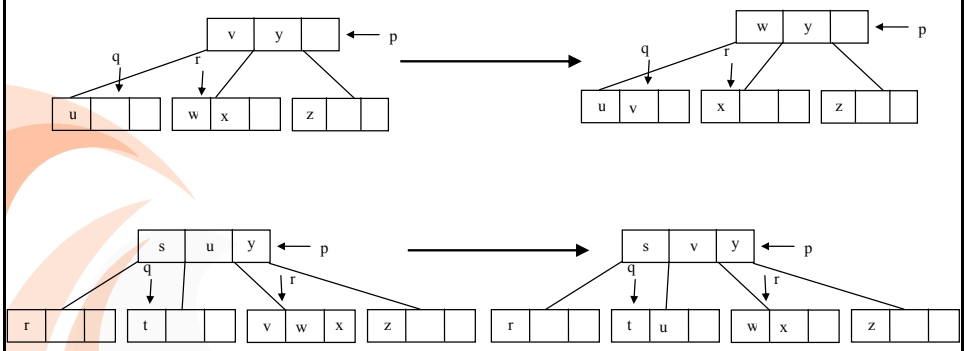
   2. **p es 3-nodo ó 4-nodo**



3.4. Árboles 2-3-4

OPERACIONES BÁSICAS. BORRADO (IV)

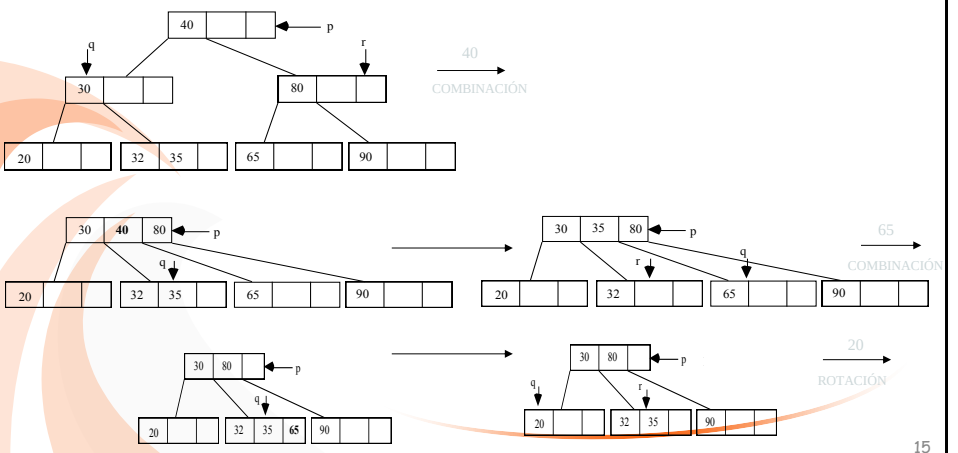
4. **q es 2-nodo y r es 3-nodo ó 4-nodo (ROTACIÓN):**



3.4. Árboles 2-3-4

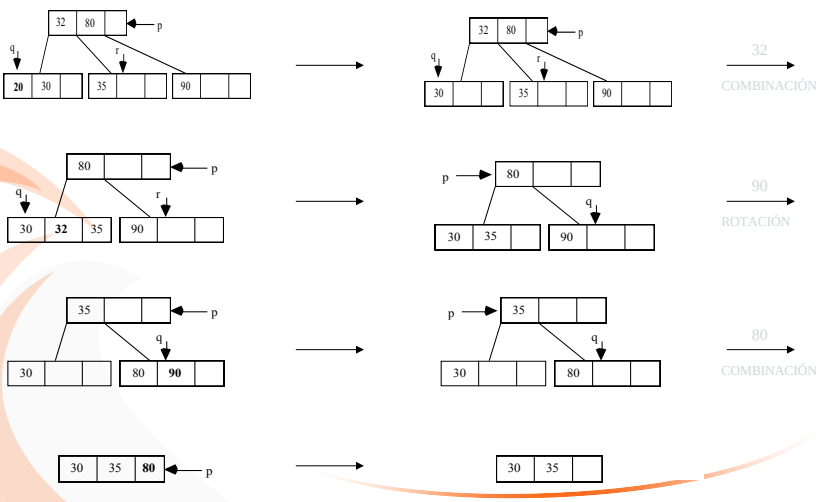
OPERACIONES BÁSICAS. BORRADO. EJEMPLO (V)

- **Ejemplo**. Borrar en el siguiente árbol 2-3-4 los siguientes items: 40, 65, 20, 32, 90, 80. (Criterios: (1) si el nodo tiene dos hijos hay que sustituir por el mayor de la izquierda, (2) Si hay dos nodos adyacentes a q, entonces r será el hermano de la izquierda)



3.4. Árboles 2-3-4

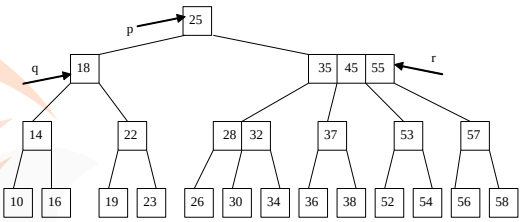
OPERACIONES BÁSICAS. BORRADO. EJEMPLO (VI)



3.4. Árboles 2-3-4

EJERCICIOS *borrado*

1) Borrar en el siguiente árbol 2-3-4: 25,18,55 y 35. (Criterios: (1) si el nodo tiene dos hijos hay que sustituir por el mayor de la izquierda, (2) Si hay dos nodos adyacentes a q, entonces r será el hermano de la izquierda)



3.4. Árboles 2-3-4

Preguntas de tipo test: Verdadero vs. Falso

- El árbol 2-3-4 no vacío tiene como mínimo una clave en cada nodo
- La complejidad temporal en el peor caso de la operación inserción en un árbol 2-3-4 es $\log_2(n+1)$
- Un árbol 2-3-4 es un árbol binario completo

Árboles de búsqueda

Aplicaciones

- **Acceso a grandes ficheros de datos. Organización interna de una BD.**
  - Restricción: Los índices residen en disco.
  - Problema: Accesos a disco muy costoso.
  - 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 (Árbol Binario).
    - El índice puede diseñarse para que el tamaño de cada nodo coincida con el de un bloque de disco.