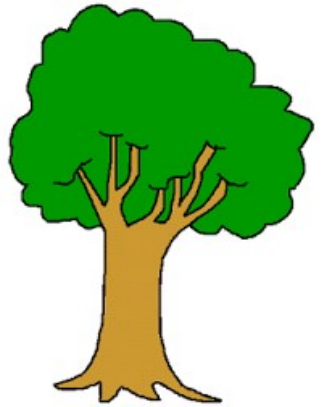
Tema 4: Árboles

Objetivo: El alumno aplicará las formas de representar y operar los grafos y listas no lineales en la computadora

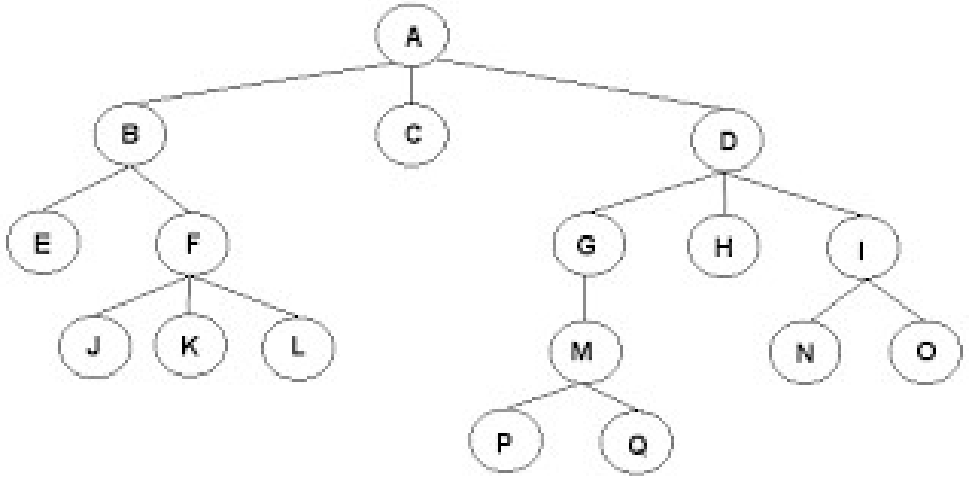
4.1 Conceptos y Definiciones

- Estructuras de datos organizadas en forma de árbol aparecen en muchas clases de información donde se requiere establecer un orden jerárquico.
- Arboles genealógicos, organigramas, clasificaciones, organización de archivos son algunos de los ejemplos donde se manejan estructuras de árboles



4.1 Conceptos y definiciones

- Un árbol es una gráfica acíclica en la que cada nodo tiene cero o más nodos hijos y máximo un nodo padre.
- Solamente un nodo no tiene padre, dicho nodo es conocido como nodo raíz.

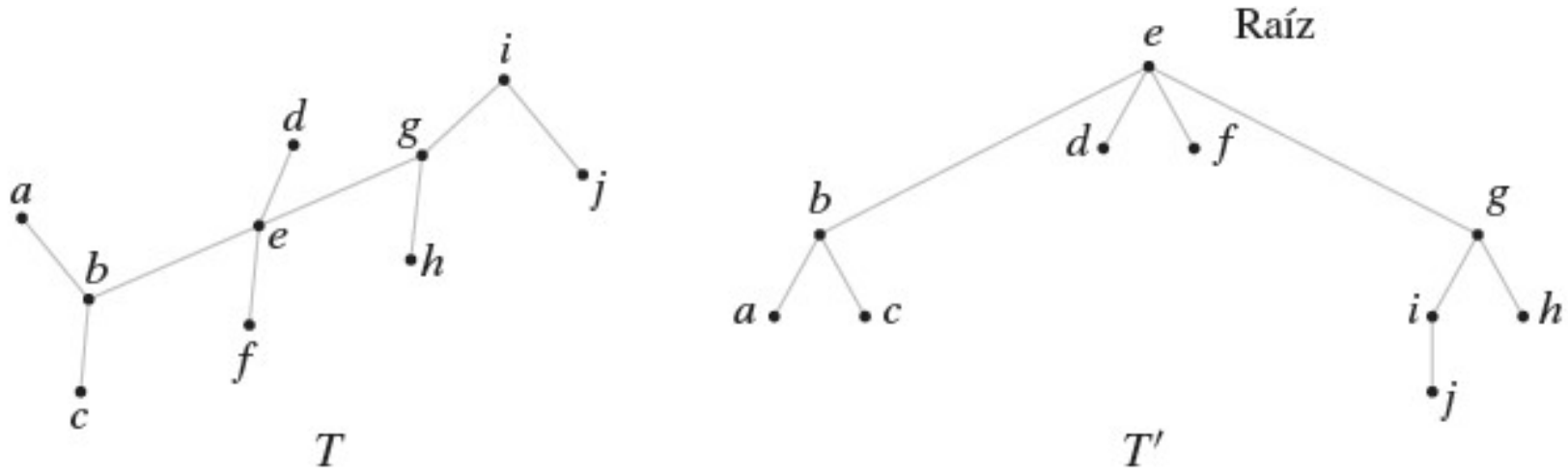


4.1 Conceptos y definiciones

- Los nodos que no tienen hijos se conocen como nodos hoja y los nodos intermedios se llaman nodos rama o sub árboles.
- El nivel de un nodo(vértice) es a longitud de la trayectoria simple de la raíz a v.
- La altura de un árbol es el número máximo de nivel

4.1 Conceptos y definiciones

- Cualquier grafo acíclico se puede ver como un árbol siempre que se defina el vértice que se establece como raíz



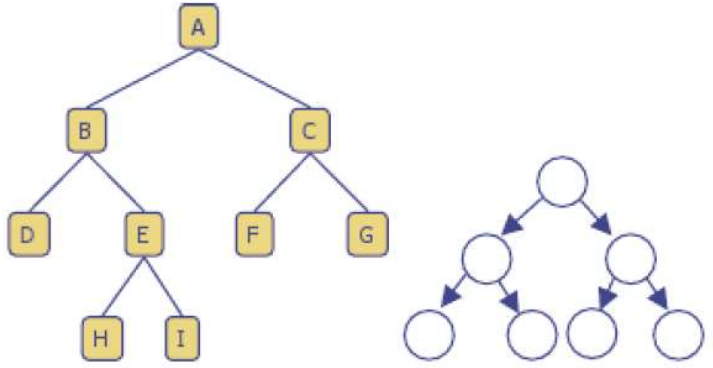
4.1 Conceptos y definiciones

Algunas Aplicaciones…

- Los sistemas operativos organizan las carpetas y los archivos usando una estructura de árbol.
- Los árboles como estructuras de definición jerárquica se usan para mostrar relaciones en registros de bases de datos.

4.2 Árboles binarios

- Es un árbol en el que todos los nodos tienen 2 hijos, aunque uno de ellos o ambos pueden ser nulos.
- Tiene múltiples aplicaciones computacionales como expresiones aritméticas, procesos de decisión, búsquedas, entre otras.



4.2.1 Transformación de árboles a árboles binarios

- Para construir un árbol binario a partir de un árbol general se realiza de la siguiente manera.
  - Se parte de la raíz del árbol.
  - El nodo izquierdo es el primer hijo izquierdo del nodo padre.
  - El nodo hijo derecho es el primer hermano del nodo
  - Se repite este proceso de manera recursiva hasta que se hayan visitado todos los nodos del árbol original.

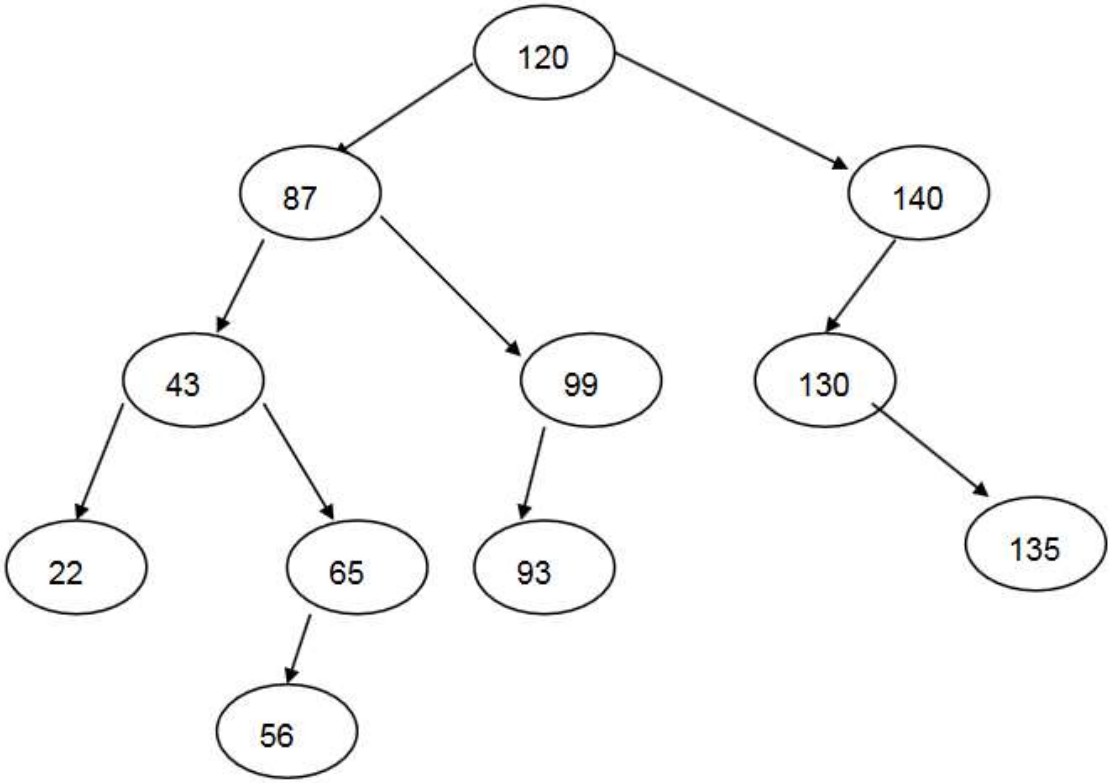
4.2.2 Árbol binario de búsqueda

- Es un tipo de árbol binario que cumple con las siguientes condiciones
  - Todos los elementos tienen un identificador único en el árbol
  - Los identificadores del subárbol izquierdo deben ser menores que la raíz.
  - Los identificadores del subárbol derecho deben ser mayores que la raíz
  - Los subárboles izquierdo y derecho deben ser árboles binarios de búsqueda

4.2.2 Árbol binario de búsqueda

- Éste tipo de árboles permiten realizar búsquedas de forma muy eficiente, si el árbol está equilibrado, la complejidad de realizar una búsqueda es de orden O(log(n))
- Un árbol binario de búsqueda tiene la misma definición y las mismas operaciones de un árbol binario con características particulares

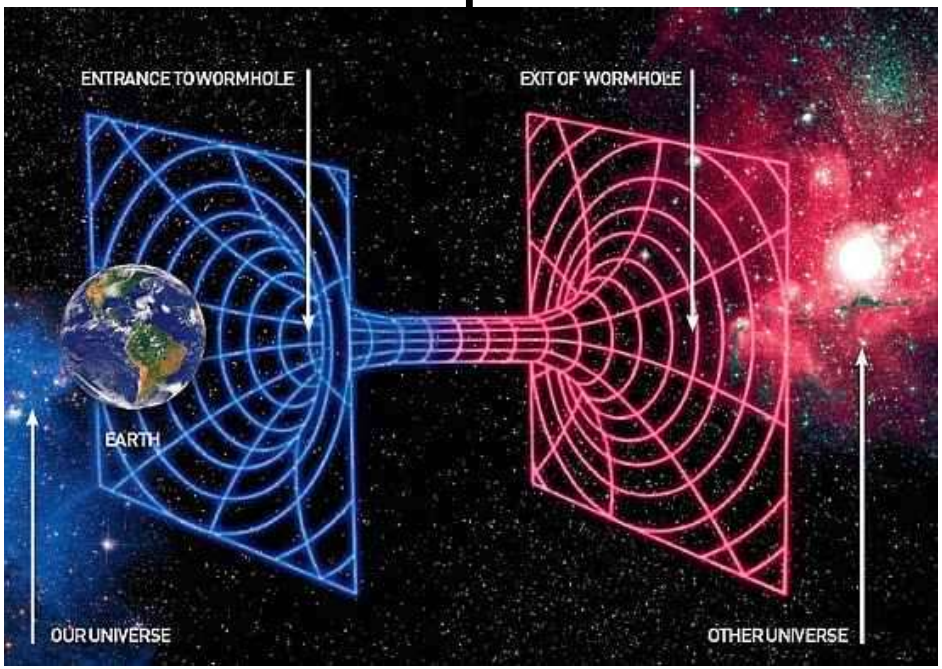
4.2.2 Árbol binario de búsqueda



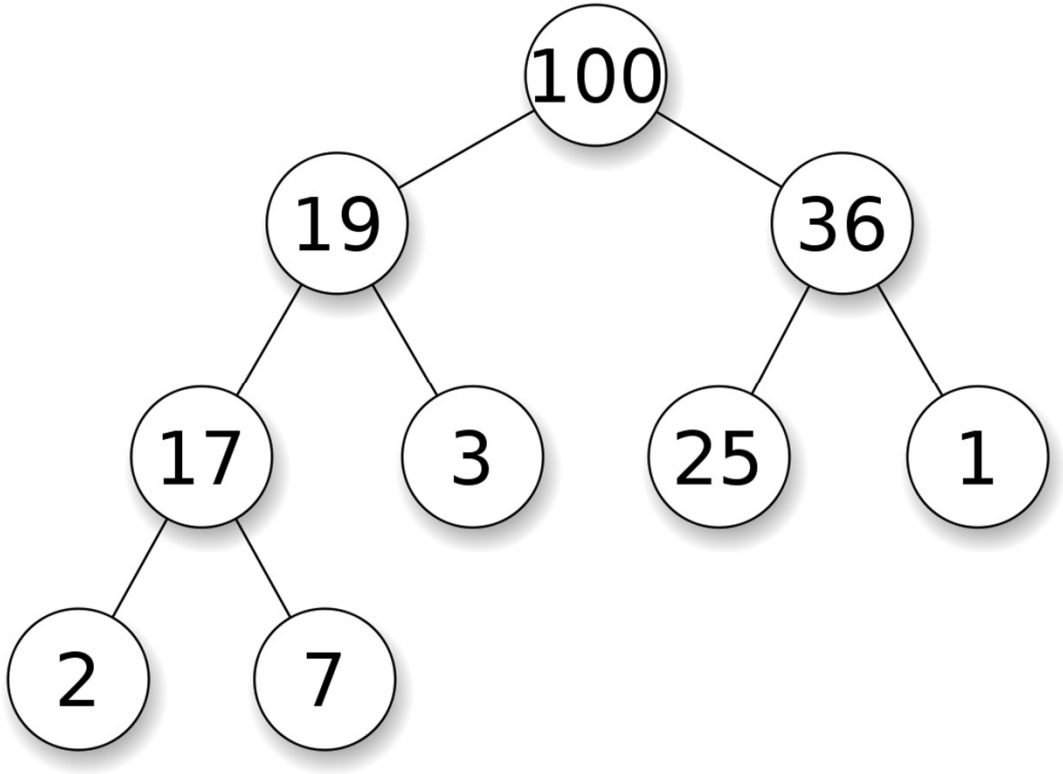
Árboles Binarios…

- La altura de un árbol binario es el nivel máximo de sus hojas (profundidad).
- La altura de un árbol nulo se define como -1
- Un árbol binario balanceado es aquel donde las alturas de los dos subárboles difiere a lo sumo en 1.
- Un árbol perfectamente balanceado es aquel donde las alturas de ambos subárboles son iguales.

4.2.3 Heaps


ENTRANCE TO WORMHOLE
EXIT OF WORMHOLE
EARTH
OUR UNIVERSE
OTHER UNIVERSE

¿Qué es un heap?



4.2.4 Recorrido de árboles

- Se denomina recorrido de un árbol al proceso que permite visitar al menos una vez a todos los nodos del árbol.
- Existen tres formas convencionales de recorrer un árbol binario.
  - Recorrido PreOrden
  - Recorrido InOrden
  - Recorrido PostOrden.

4.3.4 Recorrido de árboles

- Recorrido preOrden

  Examinar la raíz.

  Recorrer el subarbol izquierdo en preorden.

  recorrer el subarbol derecho en preorden.

4.3.4 Recorrido de árboles

- Recorrido inOrden:

  Recorrer el subarbol izquierdo en inorden.

  Examinar la raíz.

  Recorrer el subarbol derecho en inorden.

4.3.4 Recorrido de árboles

- Recorrido PostOrden:

  Recorrer el subarbol izquierdo en postorden.

  Recorrer el subarbol derecho en postorden.

  Examinar la raíz.

4.3.4 Recorrido de árboles - Ejemplo

- Árbol de expresión aritmética.
- Sea la expresión aritmética (((2+3)*((6*3))+(10/5)). El árbol que representa dicha expresión es el siguiente.

1.3.1.2 Ejemplos de uso (pila)

- Evaluación de expresiones aritméticas (2).

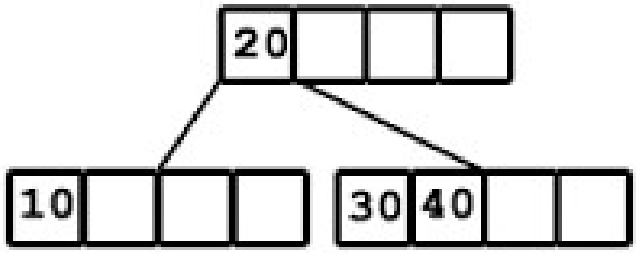
Reverse Polish Notation

La ventaja de esta forma de escribir expresiones aritméticas, es que no es necesario utilizar paréntesis ni reglas de precedencia.

En este algoritmo se utiliza una sola pila (stack).

4.4 Árboles B

- Los arboles B constituyen una categoría muy importante de estructuras de datos, que permiten una implementación eficiente de conjuntos y diccionarios, para operaciones de consulta y acceso secuencial.
- Definidos por Rudolf Bayer and Ed McCreight

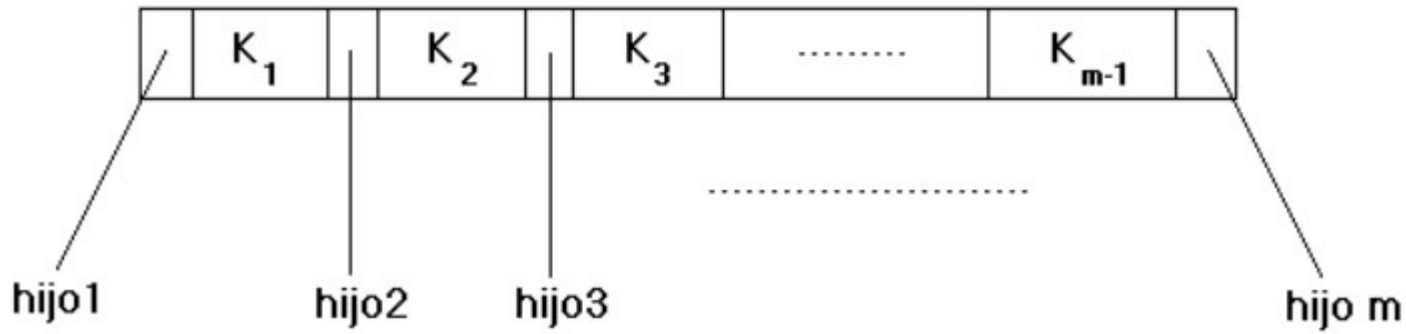


4.4.1 Árboles B

- Los árboles B son árboles de búsqueda balanceados la complejidad es O(log(n)).
- Los nodos internos deben tener un número de referencias a otros nodos y un número variable de claves y dentro de un rango predefinido.

4.4.1 Árboles B

- Estructura



4.4.1 Árboles B

- Definición y Operaciones

Un árbol-B de orden ***m*** es un árbol de búsqueda de múltiples caminos con las siguientes propiedades:

1.- La raíz tiene al menos dos subárboles. (a menos que sea una hoja única)

4.4.1 Árboles B

2.- Cada nodo intermedio (no hoja, no raíz) tiene ***k*** subárboles y ***k-1*** claves

Dónde $\left\lceil \frac{m}{2} \right\rceil \leq k \leq m$

Hasta este punto…

Un árbol-B de orden 2,3 no resulta muy practico.

Un árbol-B de orden 4 tiene máximo 3 claves y mínimo 1

Un árbol-B de orden 5 tiene máximo 4 claves y mínimo 2

4.4.1 Árboles B

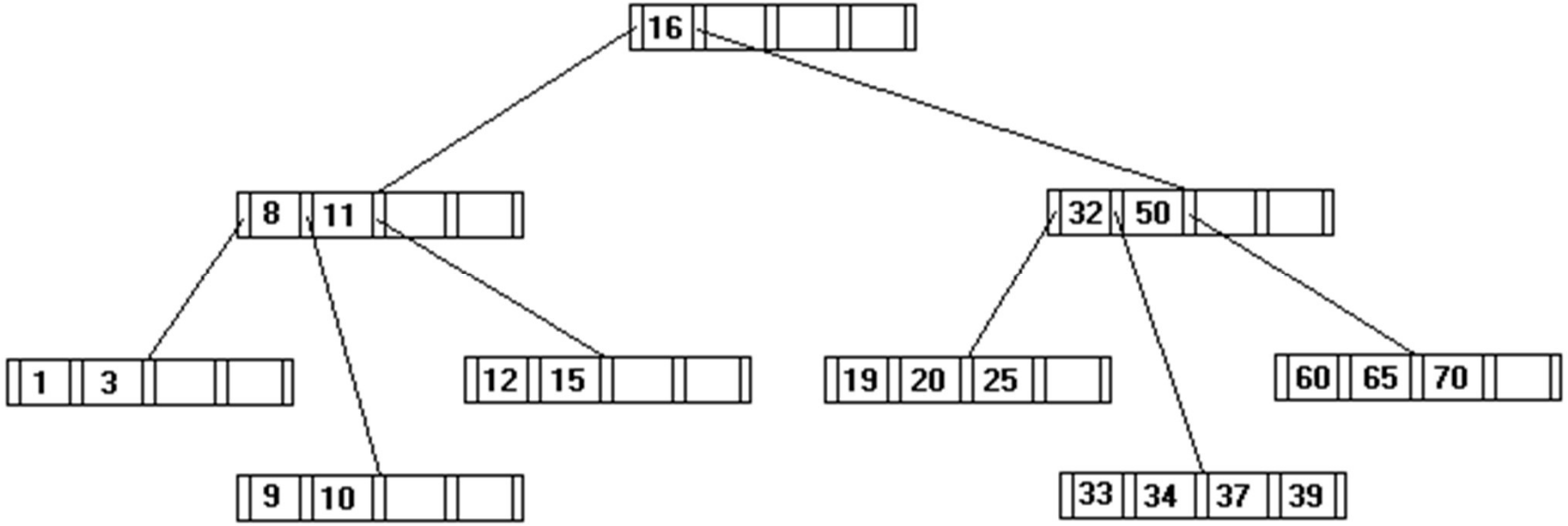
- Definición y Operaciones

3.- Todas las hojas del árbol se encuentran al mismo nivel.

4.- Un árbol B mantiene la estructura “half-full” y es perfectamente balanceado.

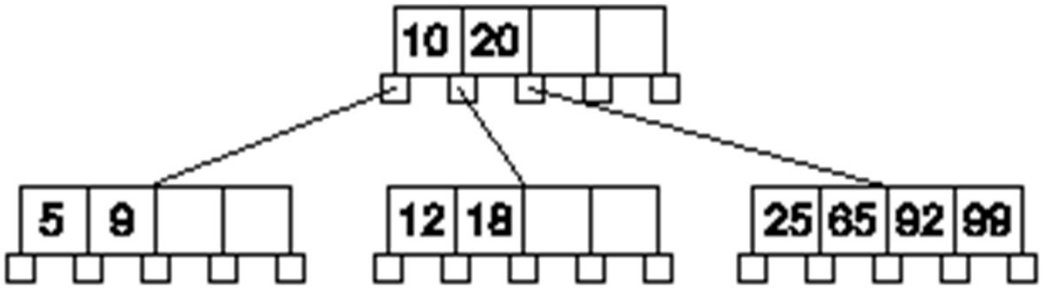
4.4.1 Árboles B

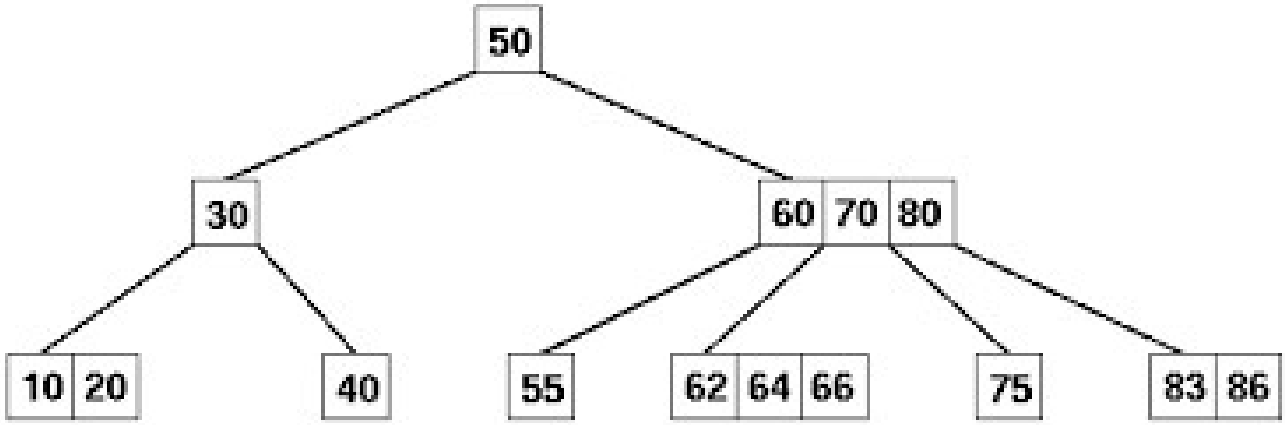
- EJEMPLOS



4.4.1 Árboles B

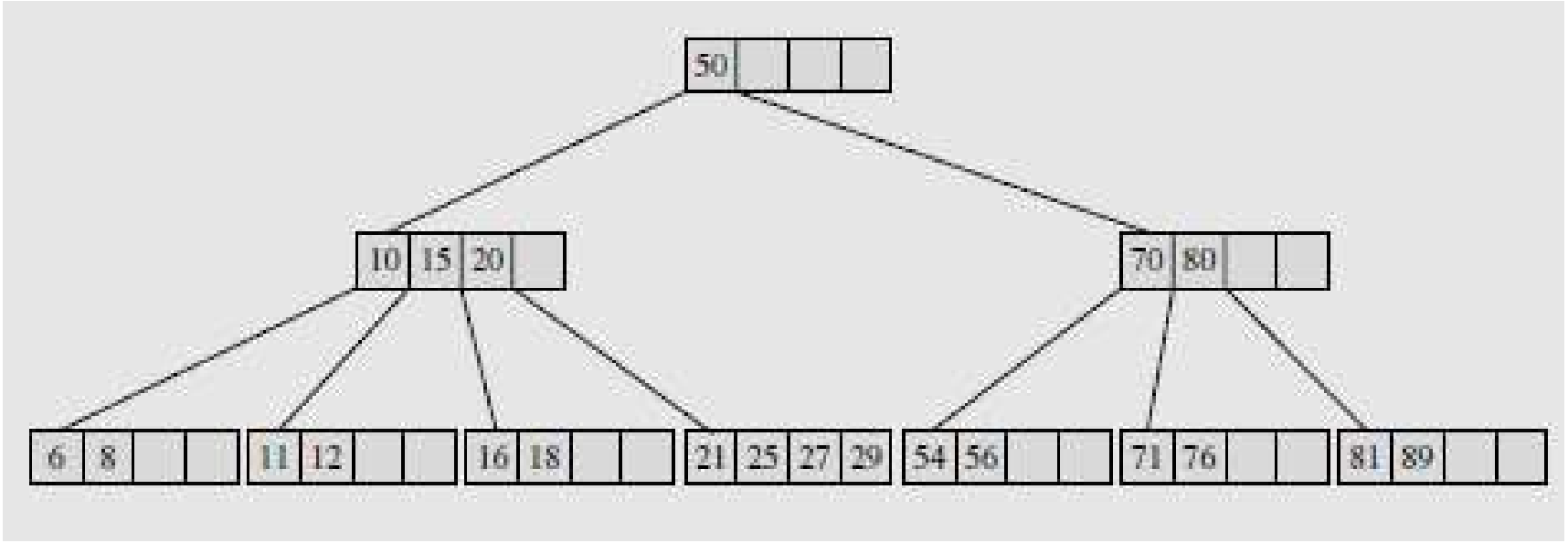
- EJEMPLOS





4.4.1 Árboles B

- EJEMPLOS



4.4.1 Árboles B

Características

- Su construcción se realiza en forma ascendente
- Se utilizan para dispositivos de almacenamiento secundario.
- Siempre que se realiza una operación de inserción, eliminación el árbol permanece perfectamente balanceado

4.4.1 Árboles B

- Inserción de CLAVES

1) Buscar el nodo correspondiente en orden a la clave a insertar.
2) Si el nodo no está lleno se inserta en ese nodo respetando el orden de las claves
3) Si el nodo está lleno, se realiza la división celular
    1) Se crea un nuevo nodo repartiendo el contenido del nodo lleno entre los dos nodos.
    2) La clave intermedia se inserta en el nodo padre.(si el nodo padre no existe, se crea)

4.4.1 Árboles B

- Ejemplos

1.- Insertar en un árbol b de orden 4 las siguientes claves:

70,50,30,40,20,80,25,90,75,10,15

2.- Insertar en un árbol b de orden 5 las siguientes claves:

20,40,10,30,15,35,7,26,18,22,5,42,13,46,27,8,32,38, 24,45,25

4.4.1 Árboles B

- Eliminación de claves

*__Consideración:__ Si el orden del árbol es ***m***, cada hoja debe tener al menos (m/2) – 1 claves.

1.- Si el elemento a borrar se encuentra en una hoja se suprime , si se rompe la consideración se realiza una redistribución

2.- Si el elemento no se encuentra en una hoja, se debe sustituir por la clave que se encuentra mas a la derecha en el sub árbol izquierdo o más a la izquierda del subárbol derecho

4.4.1 Árboles B

- Eliminación de nodos

3.- Si al sustituir la llave de la hoja respectiva se rompe la consideración previa, se debe realizar una redistribución

4.4.1 Árboles B

- Redistribución

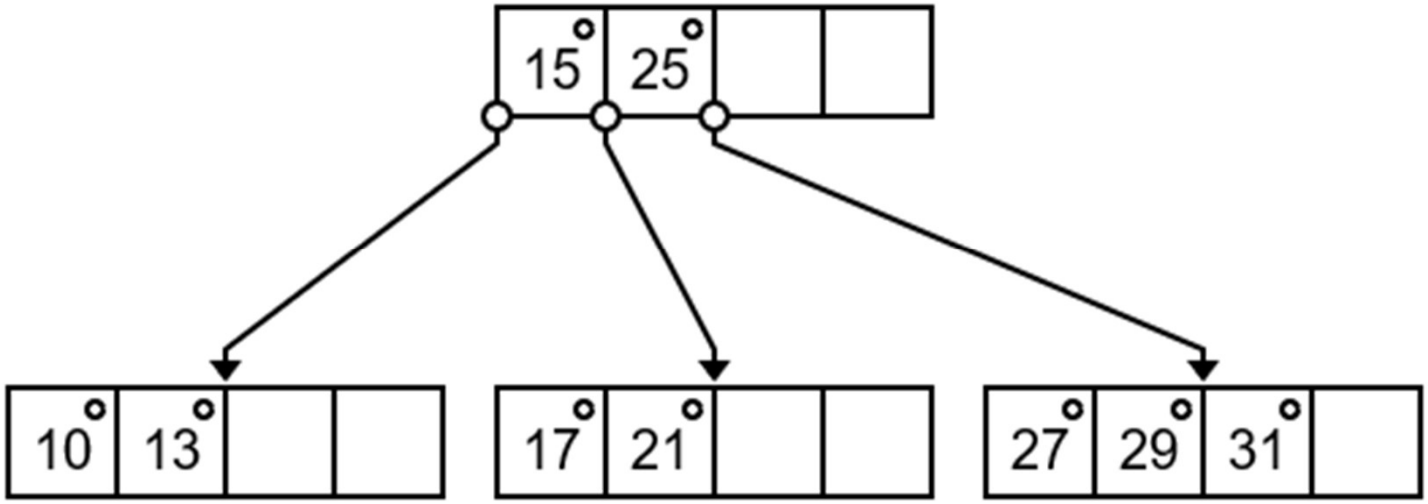
1.- Si una hoja vecina hija del mismo padre tiene suficientes claves, las claves se reparten entre los dos nodos, la clave intermedia sube donde se ha hecho el borrado y es sustituida por una etiqueta del otro nodo

2.- De lo contrario, de la hoja de la cual se ha hecho el borrado, de una hoja adyacente y la llave correspondiente del padre se hace una sola.

Si en el padre se rompe la restricción se hace el procedimiento de manera recursiva

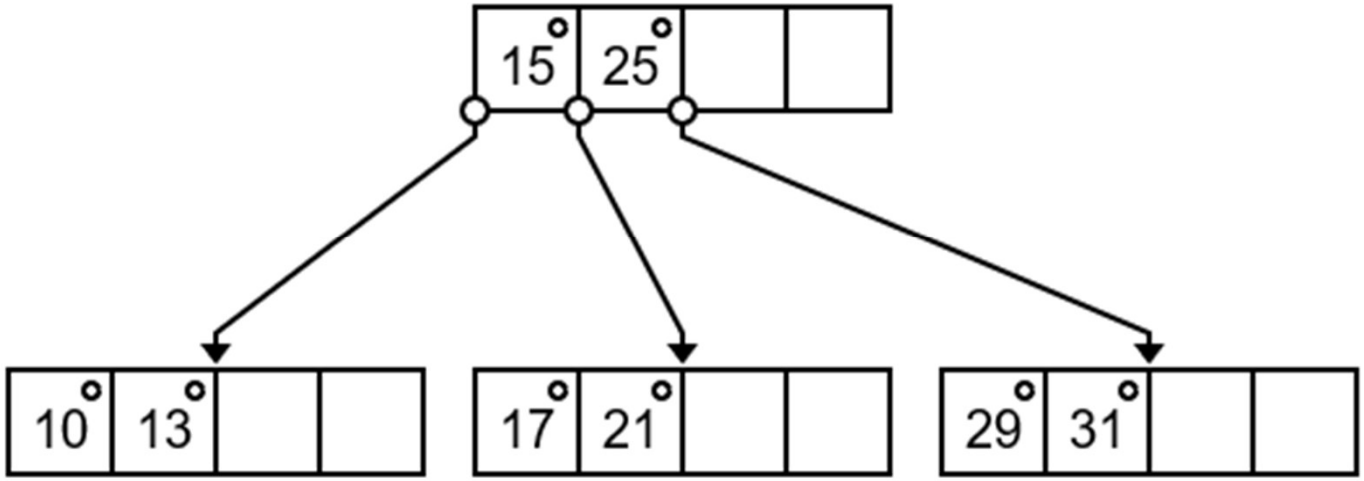
4.4.1 Árboles B

- Ejemplo 1 - Eliminación (clave 27)



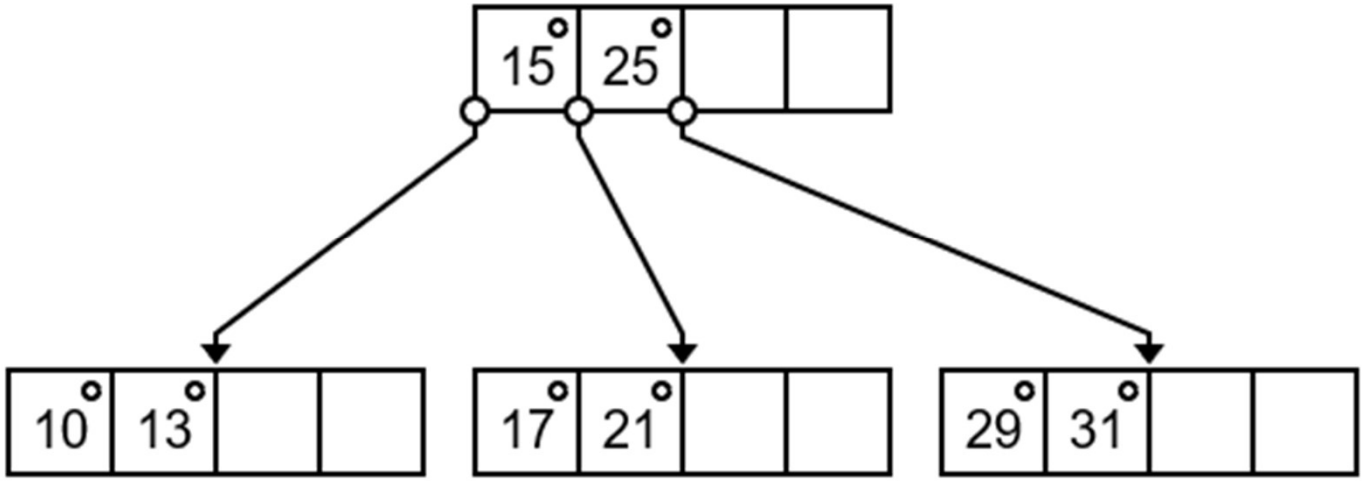
4.4.1 Árboles B

- Ejemplo 1 - Eliminación (clave 27)



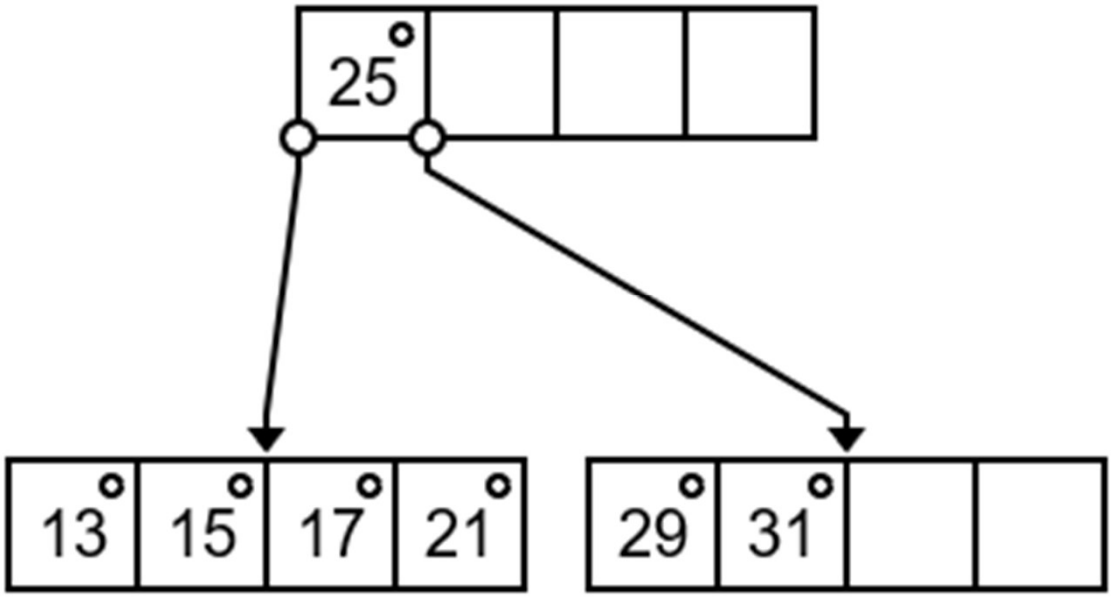
4.4.1 Árboles B

- Ejemplo 2- Eliminación(clave 10)



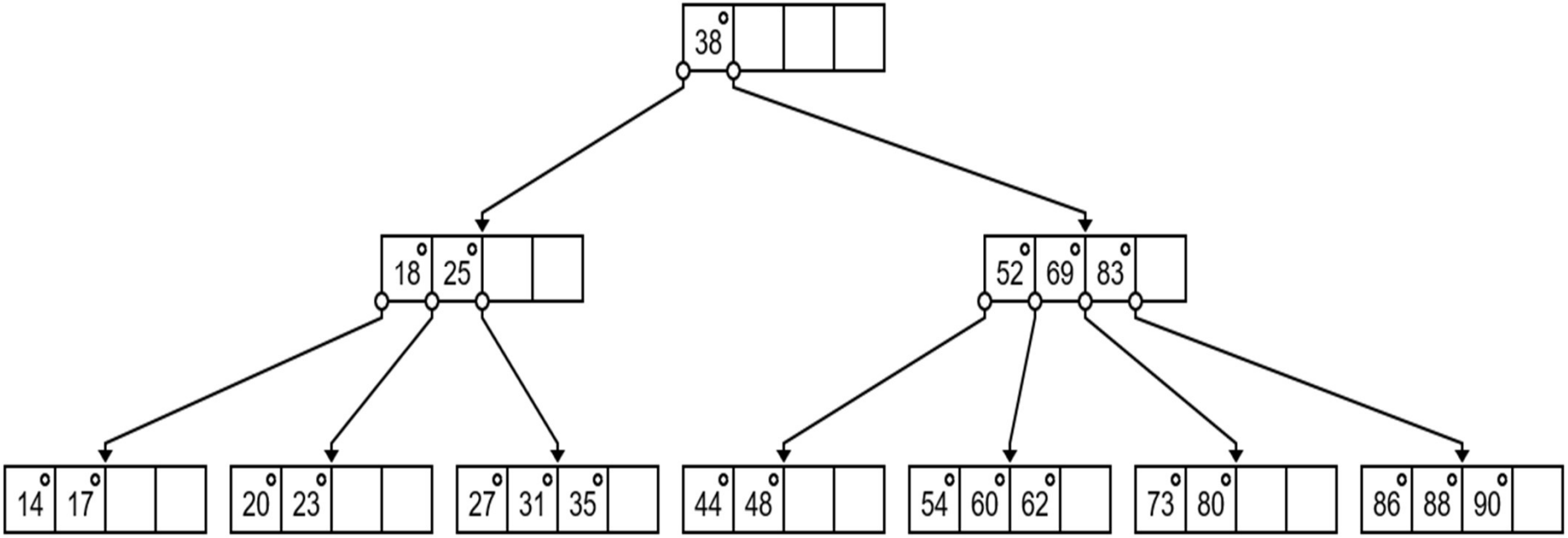
4.4.1 Árboles B

- Ejemplo 2- Eliminación(clave 10)



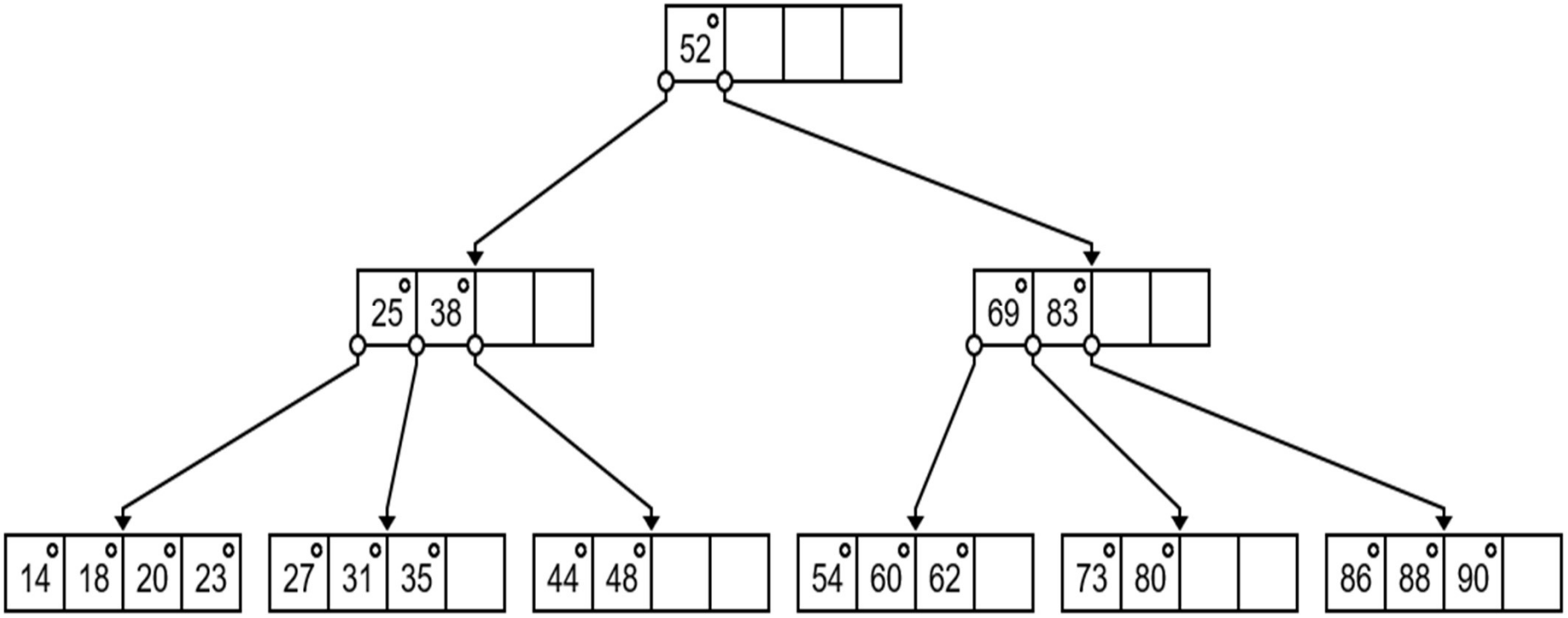
4.4.1 Árboles B

- Ejemplo 3 - Eliminación clave 17



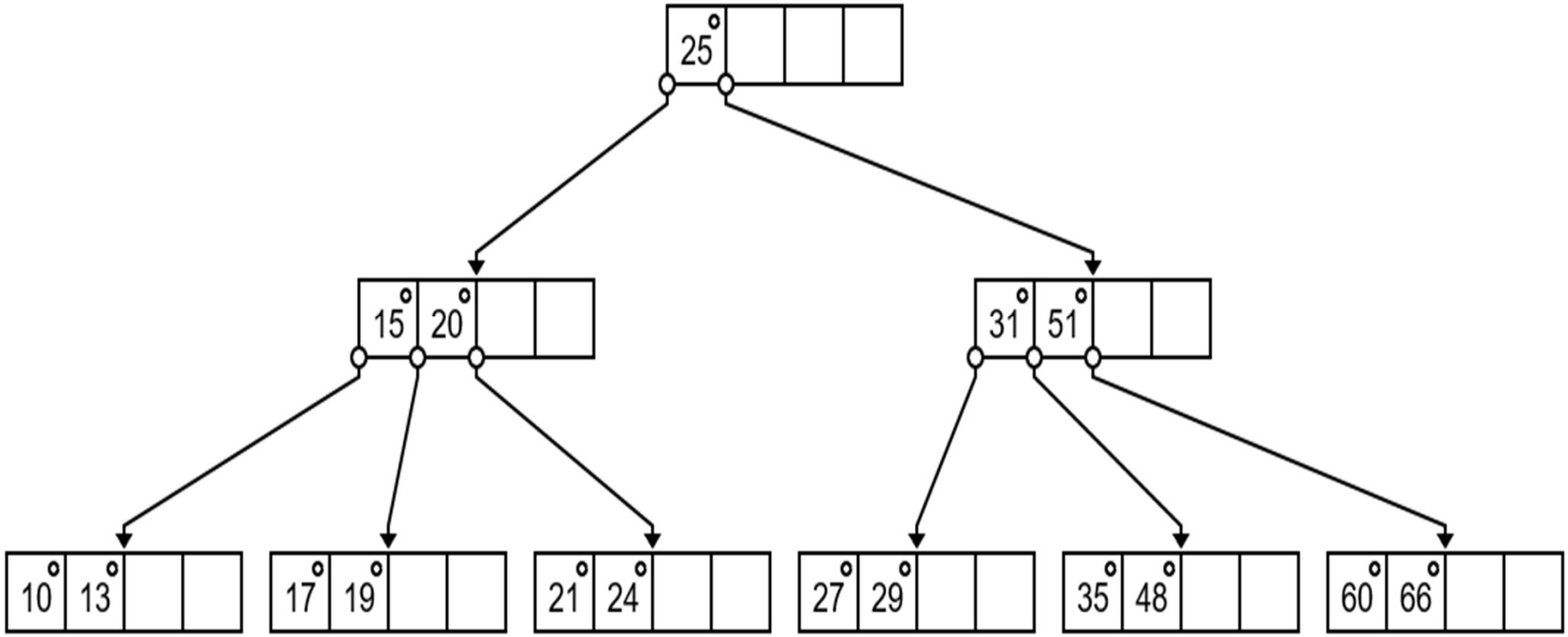
4.4.1 Árboles B

- Ejemplo 3 - Eliminación clave 17



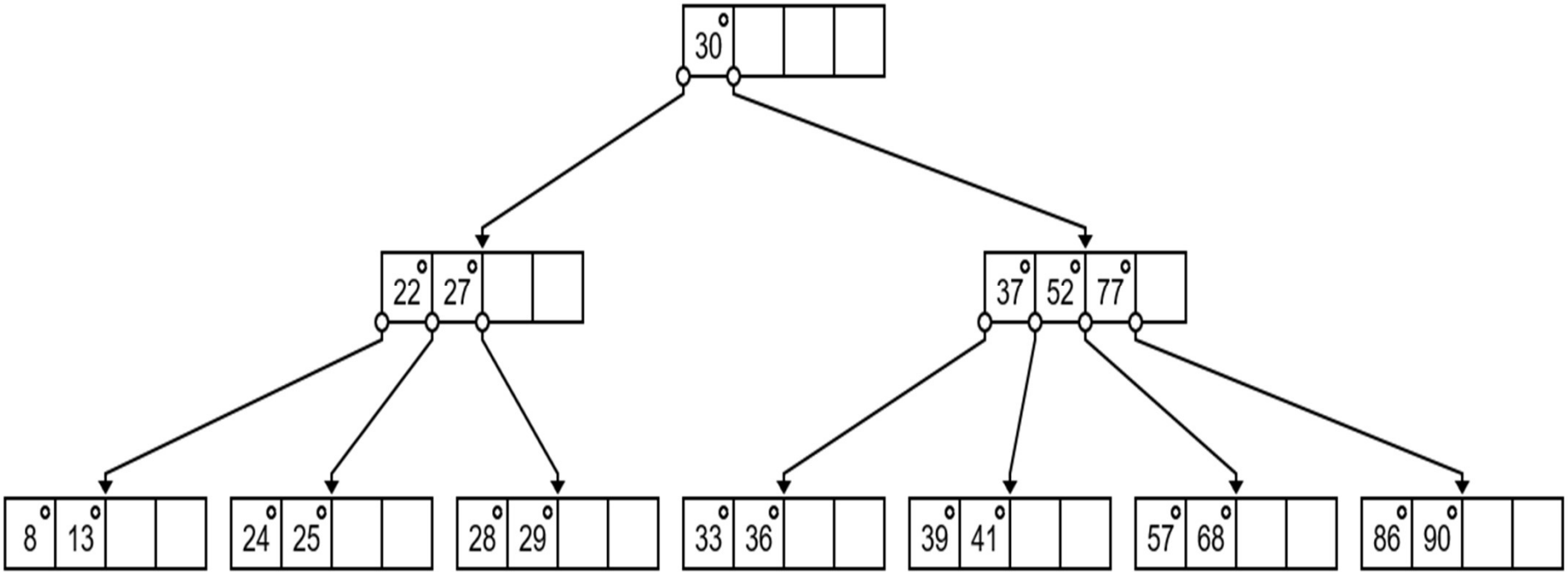
4.4.1 Árboles B

- Ejemplo 4 - Eliminación



25,24,29,27,48,19,51,21,13,15,17,66,10

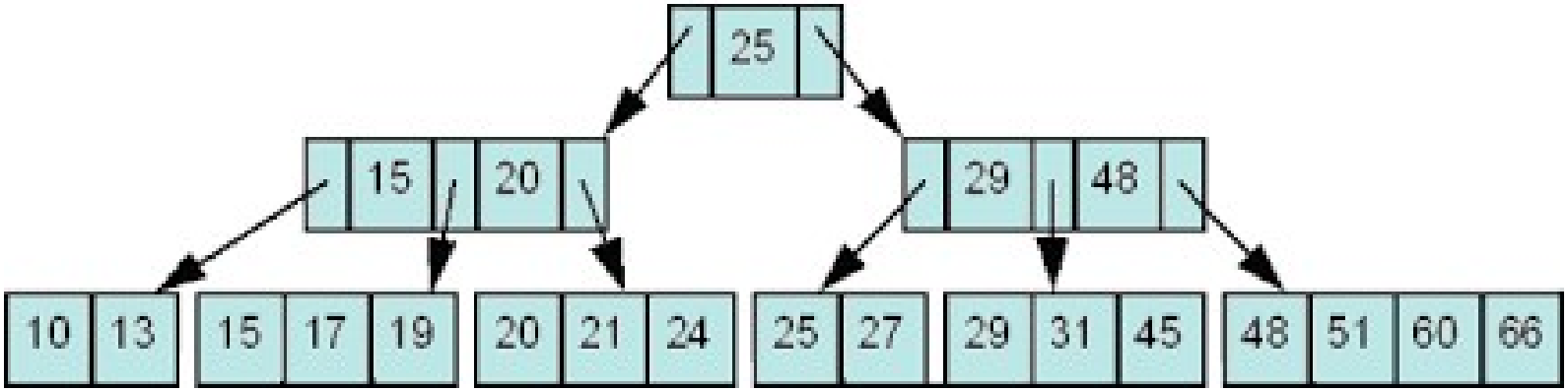
Ejercicio-Eliminación



Claves: 52,77,36,8,57,39,28,25,27

4.4.2 Árboles B+

- Los arboles B+ son una variante de los arboles B, se diferencian en que los arboles B+ toda la información se encuentra almacenada en las hojas .
- En la raíz y en las paginas internas se encuentran almacenado índices o claves para llegar a un dato.



4.4.2 Árboles B+

- Características.

Cada página excepto la raíz contiene m-1 elementos

Cada página tiene m descendientes

Las páginas hojas se encuentran al mismo nivel

Las claves almacenadas en las páginas raíz e interiores (no hoja) se utilizan como índices.

4.4.2 Árboles B+ Algoritmos

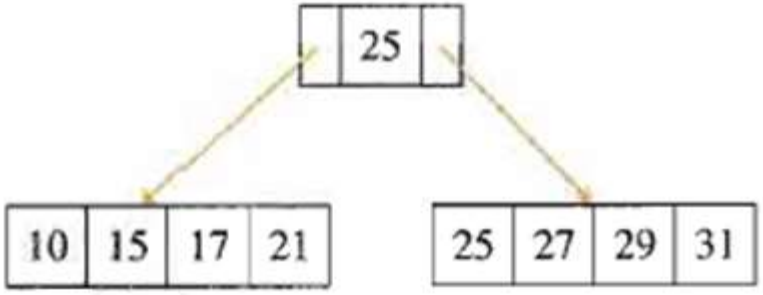
- Inserción

El proceso de inserción es similar al de los árboles B.

La diferencia radica en que cuando se inserta una clave en un nodo que se encuentra lleno, se debe realizar una copia de la clave que sube al siguiente nivel en el nodo correspondiente.

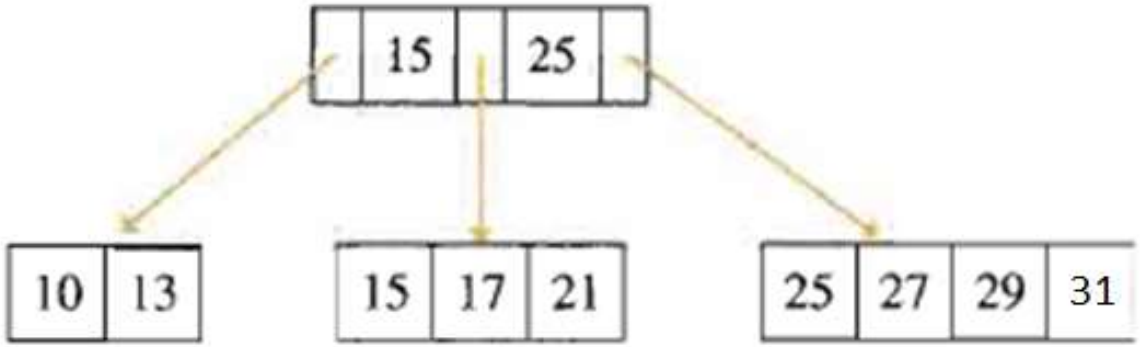
4.4.2 Árboles B+ Algoritmos

Ejemplo1. Inserción de la clave 13



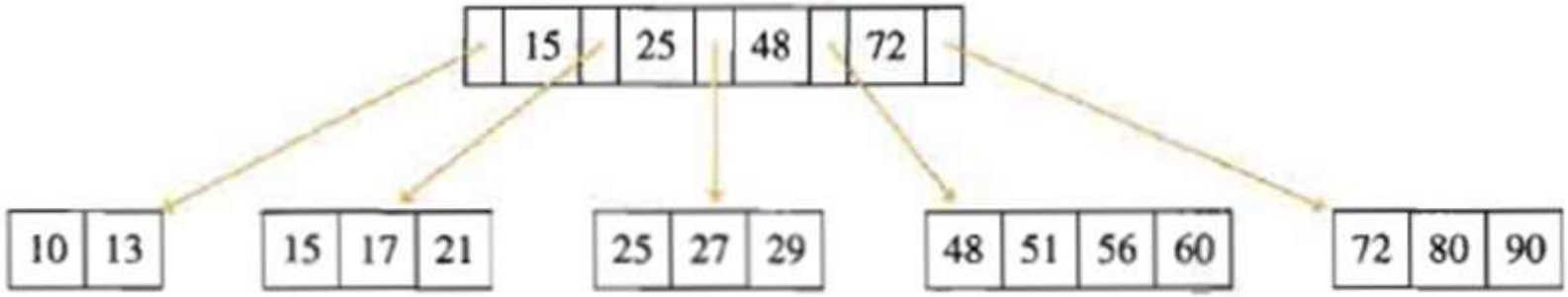
4.4.2 Árboles B+ Algoritmos

Resultado.



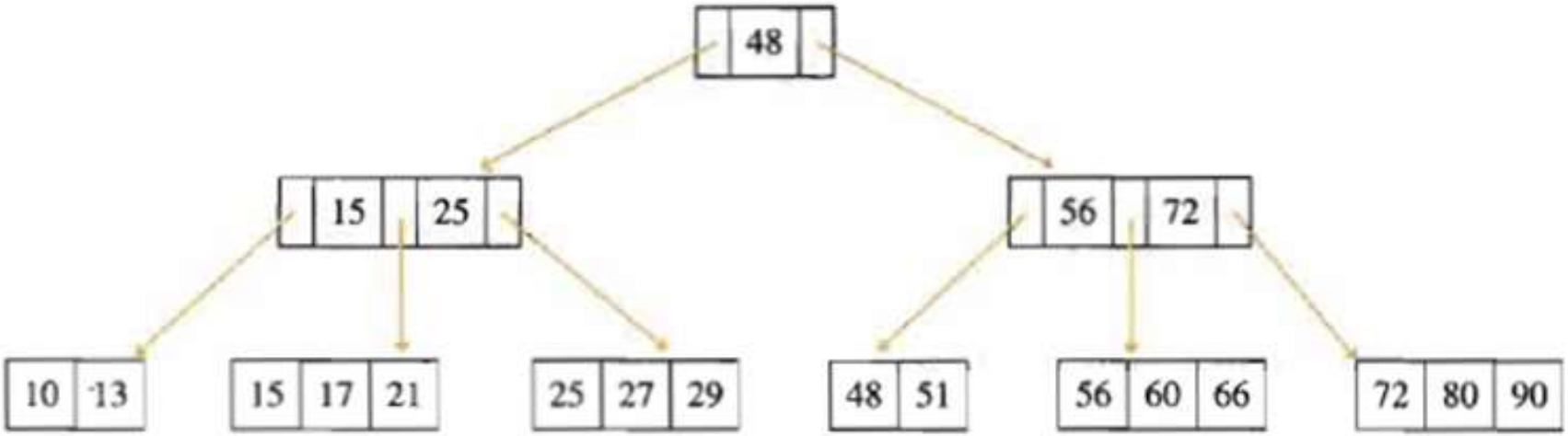
4.4.2 Árboles B+ Algoritmos

Ejemplo2. Inserción de la clave 66



4.4.2 Árboles B+ Algoritmos

Resultado.



4.4.2 Árboles B+ Algoritmos

- Eliminación

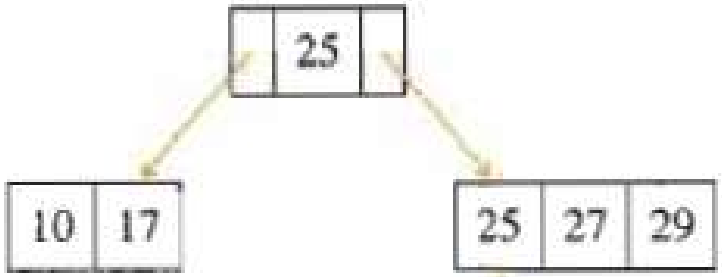
El proceso de eliminación es más sencillo que en los árboles b ya que siempre se encuentra el dato en los nodos hoja.

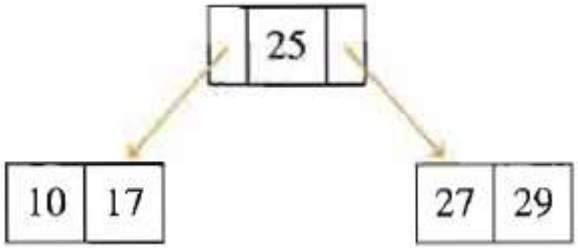
SI al eliminar un dato el número de claves se encuentra dentro del mínimo no hay nada más que hacer.

De lo contrario es necesario realizar una redistribución considerando las hojas y los índices.

4.4.2 Árboles B+ Algoritmos

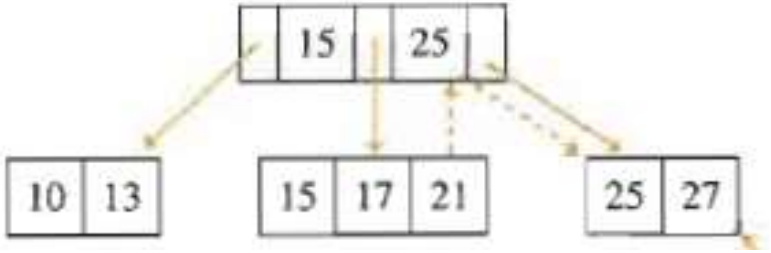
- Ejemplo 1 Eliminar la llave 25

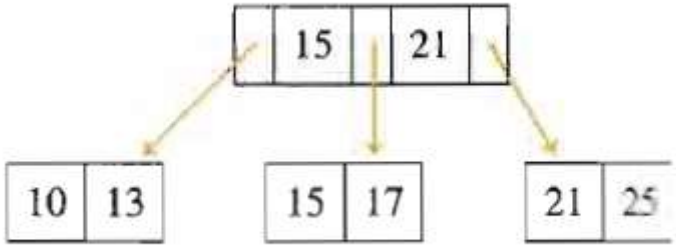




4.4.2 Árboles B+ Algoritmos

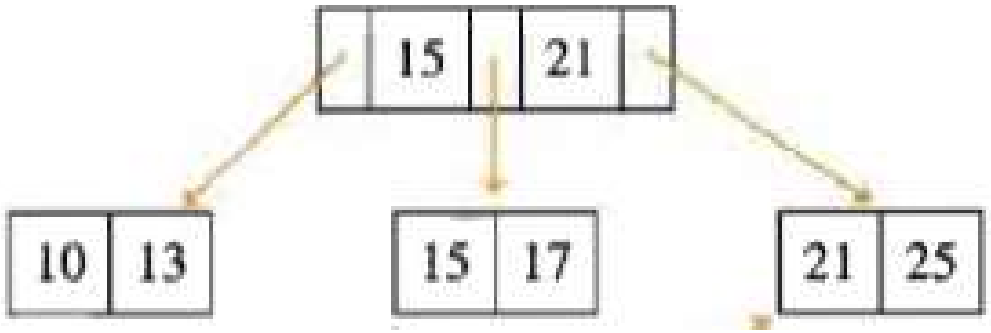
- Ejemplo 2 Eliminar la llave 27

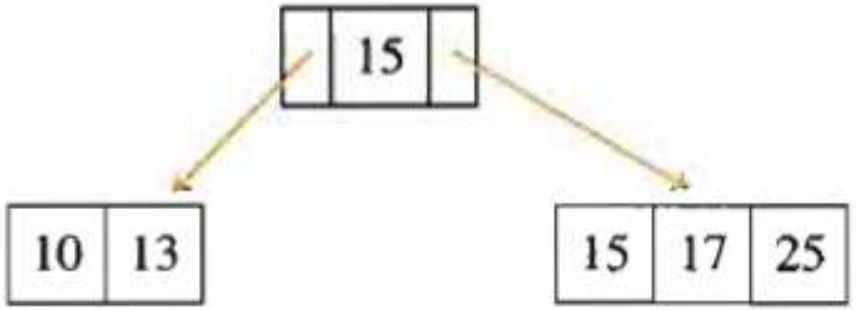




4.4.2 Árboles B+ Algoritmos

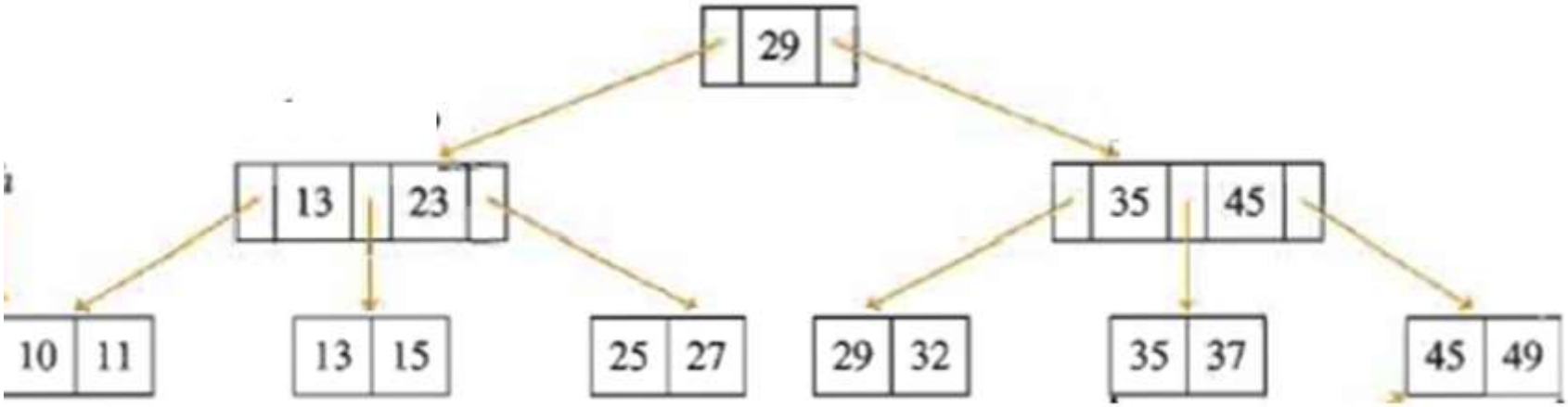
- Ejemplo 3 Eliminar la llave 21





4.4.2 Árboles B+ Algoritmos

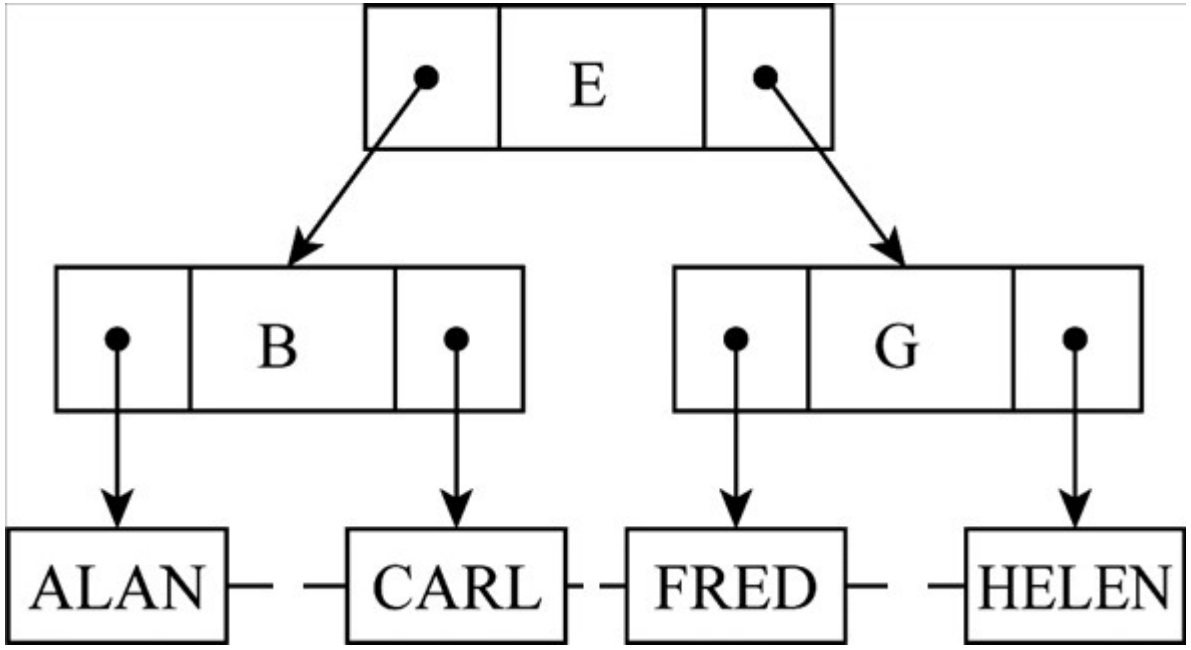
- Ejemplo 4 Eliminar la llave 37



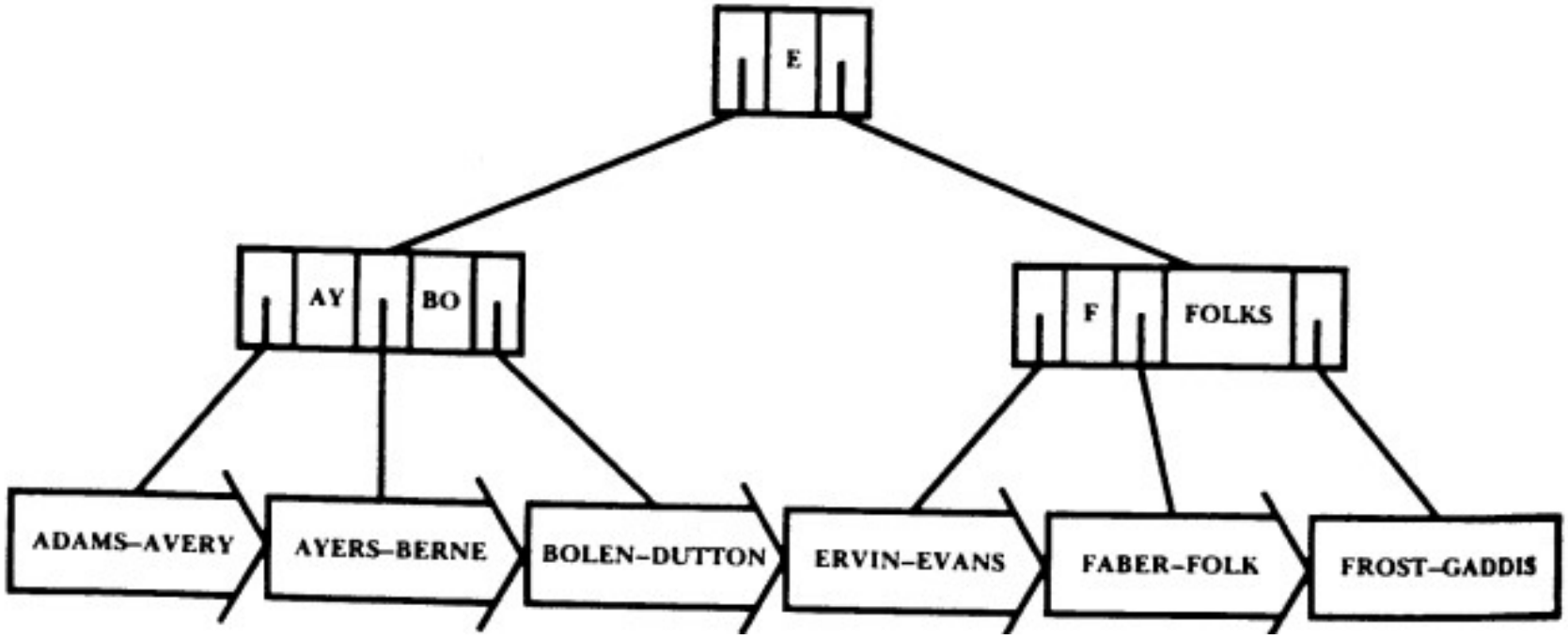
4.4.3 Árboles B+ Prefijos Simples

- Un árbol B+ de prefijos simples es un árbol B+ en el cual los separadores elegidos son los prefijos (caracteres) cortos que permiten distinguir dos llaves de índice vecinas
- La reducción del tamaño de los separadores a lo mínimo necesario no cambia el resultado de la búsqueda. Sólo vuelve los separadores más pequeños

4.4.3 Árboles B+ de prefijos Simples



4.4.3 Árboles B+ de prefijos Simples



4.4.3 Árboles B+ de prefijos simples

- En los árboles B+ de prefijos simples se reduce el factor de ramificación y el procesamiento del árbol (recorridos para búsqueda o eliminación) mejoran en cuestión de rapidez.
- Para árboles de 400-800 nodos, este tipo de árbol requieren de 20, 25% menos accesos.

Tarea

Investigar los siguientes tipos de árboles

- Árboles 2-3(2-4)
- Árboles B*
- Árboles R
- Tries

FECHA DE ENTREGA:

Lunes 5 de Noviembre