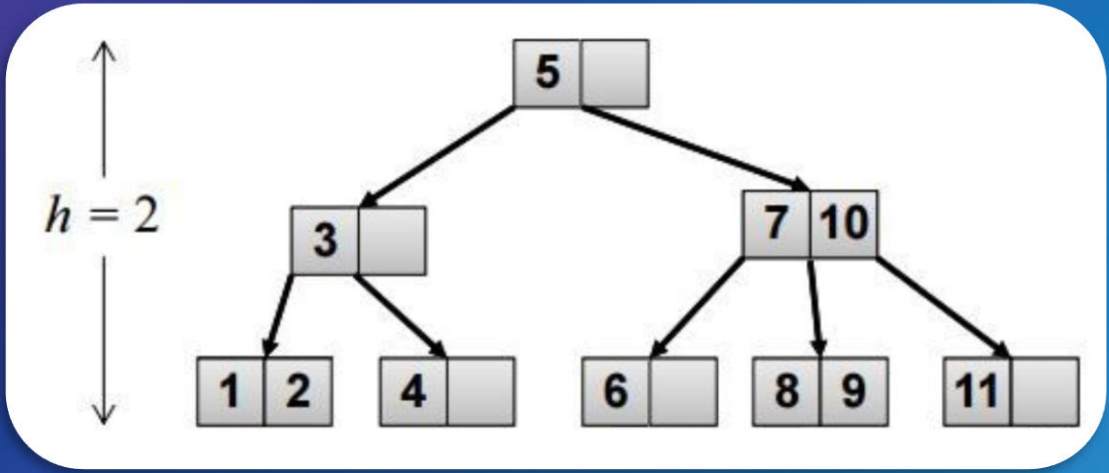
Universidad Rafael Landívar
Tradición Jesuita en Guatemala

Árboles B



Facultad de ingeniería

Árboles Multicamino

- **Definición**:
  - Se denominan árboles multicamino a aquellos árboles de grado mayor que dos.
    - Árbol que contiene nodos con más de dos ramas.
- **Aplicaciones**:
  - Muy utilizados en la construcción y mantenimiento de árboles de búsqueda con
    - Gran cantidad de nodos
    - Guardados en **memoria secundaria**,
    - En los que **se realizan con frecuencia inserciones y supresiones**.
- **Idea básica:**
  - Un árbol se subdivide en subárboles
  - Cada subárbol se representan como unidades a las que se accede simultáneamente y reciben el nombre de *páginas*.
    - Cada acceso a página requiere un único acceso a memoria secundaria.
    - 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.

Árboles B

- Un **árbol B** es un árbol de búsqueda equilibrado.
- En la mayoría de los árboles de busqueda (binarios y AVL) se supone que todo esta en la memoria principal, pero para comprender los árboles B, es necesario pensar en aplicaciones con grandes cantidades de datos que no pueden caber en la memoria principal.
- Cuando el número de claves es alto, los datos se leen del disco duro en forma de bloques
- El tiempo de acceso al disco es muy alto en comparación con el tiempo de acceso a la memoria principal. La idea principal de usar árboles B es reducir el número de accesos al disco

Árboles B

- Los **árboles B** son muy usados en Bases de Datos.
- **Necesidades propias de las aplicaciones de BD:**

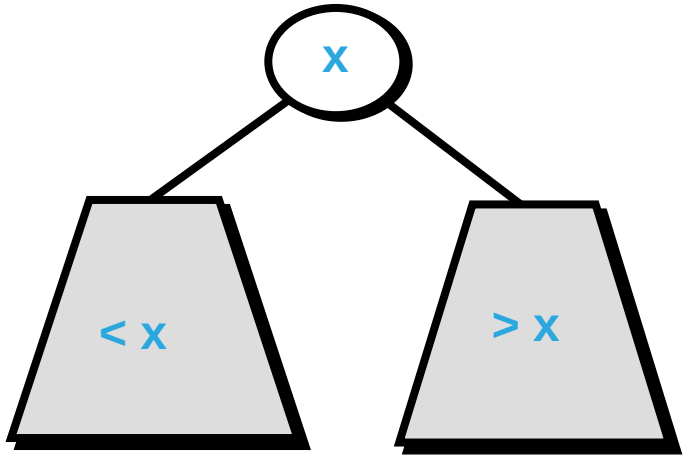
  Muchos datos, básicamente conjuntos y diccionarios.

  El acceso secuencial y directo debe ser rápido.

  Datos almacenados en memoria secundaria (disco) en bloques.

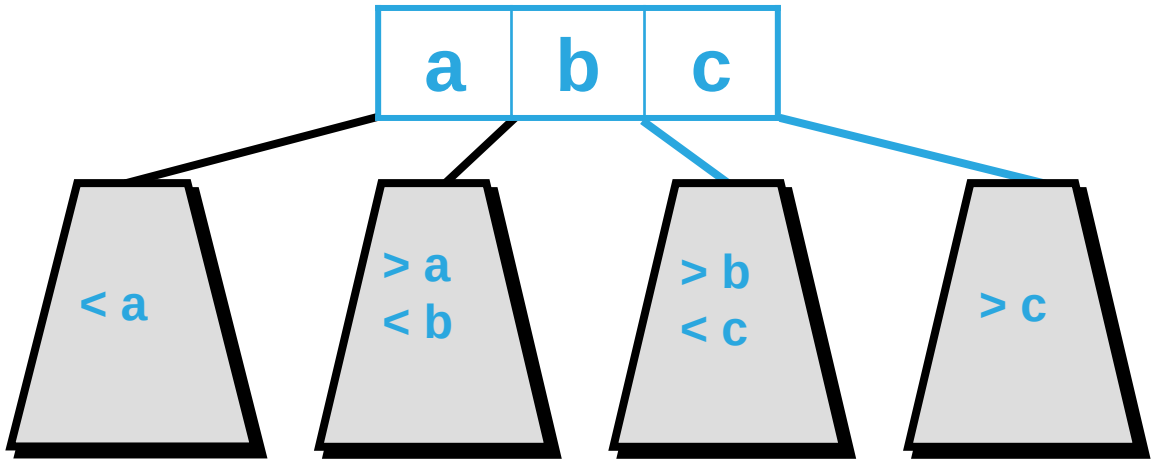
- Existen diversas variantes: árboles B, B+ y B*.

Árbol Binario de Búsqueda



Árbol B

- En cada nodo hay **n** claves y **n+1** punteros a nodos hijos.



Árboles B

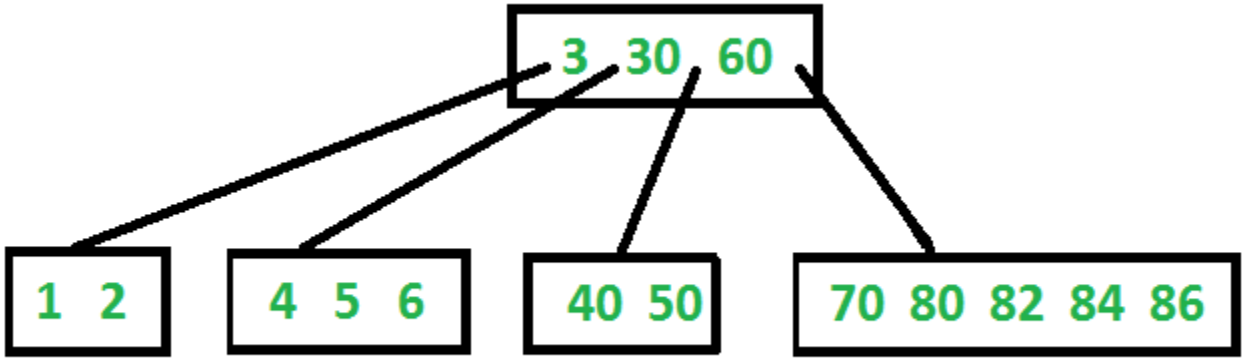
- **Definición: Un árbol B de grado mínimo t** es un árbol multicamino, que cumple las siguientes propiedades:
  1. Todas las hojas están al mismo nivel.
  2. Un árbol B se define por el término **grado mínimo 't'**. El valor de t depende del tamaño del bloque de disco.
  3. Todos los nodos, excepto la raíz, deben contener al menos **t-1** claves. La raíz puede contener un mínimo de 1 clave.
  4. Todos los nodos (incluida la raíz) pueden contener como **máximo 2t - 1 claves**.

Árboles B

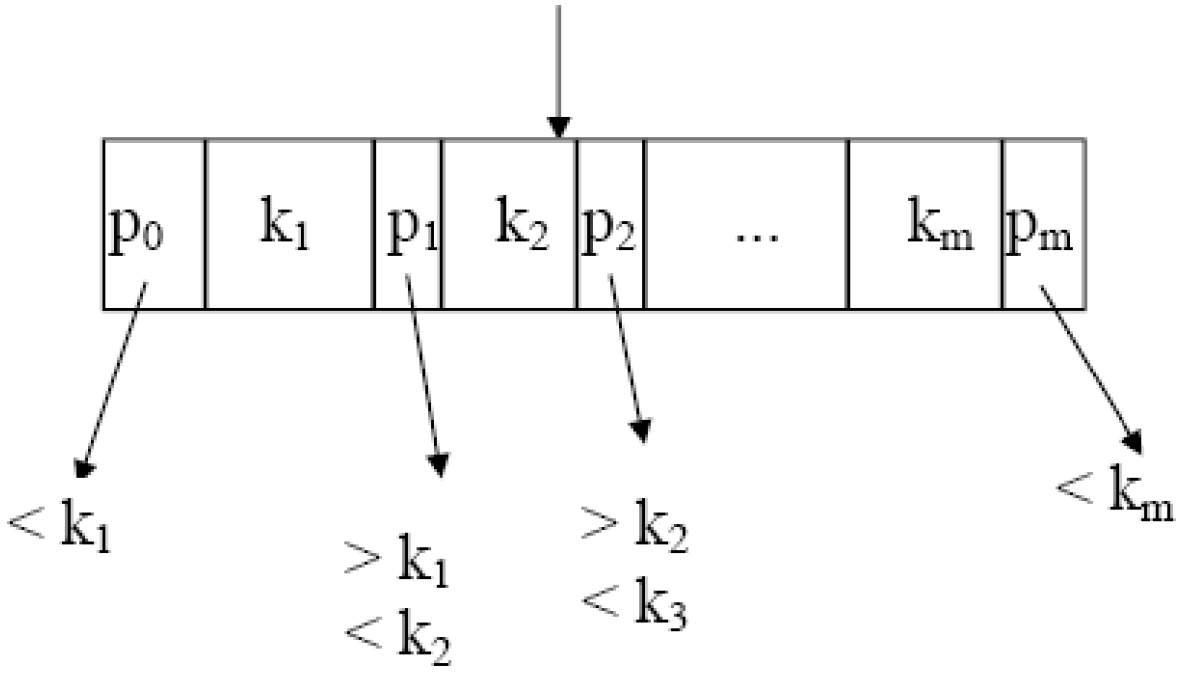
5. El **número de hijos** de un nodo es igual al **número de claves en él más 1 (2t)**.
6. Todas las claves de un nodo se ordenan en orden creciente. El hijo entre dos claves k1 y k2 contiene todas las claves en el rango de k1 y k2.
7. Un árbol B crece y se contrae desde la raíz, que es diferente a los árboles binarios de búsqueda, ya que, estos crecen hacia abajo y también se reducen desde abajo.
8. Al igual que otros árboles de búsqueda binarios equilibrados, la complejidad de tiempo para buscar, insertar y eliminar es Log(t).

Árboles B

A continuación se muestra un ejemplo de un árbol B de **grado mínimo 3**. Tenga en cuenta que, en la práctica, el valor del grado mínimo es mucho mayor que 3 (2t-1 valores y 2t hijos).



Búsqueda en Árboles B

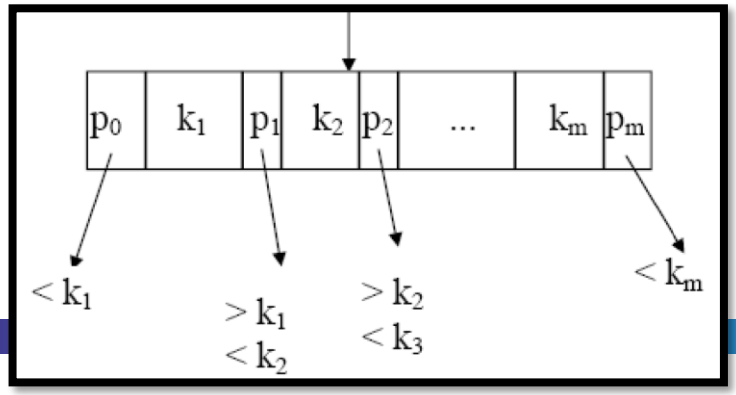


- Las llaves están ordenadas de izquierda a derecha en cada página y en cada nivel.

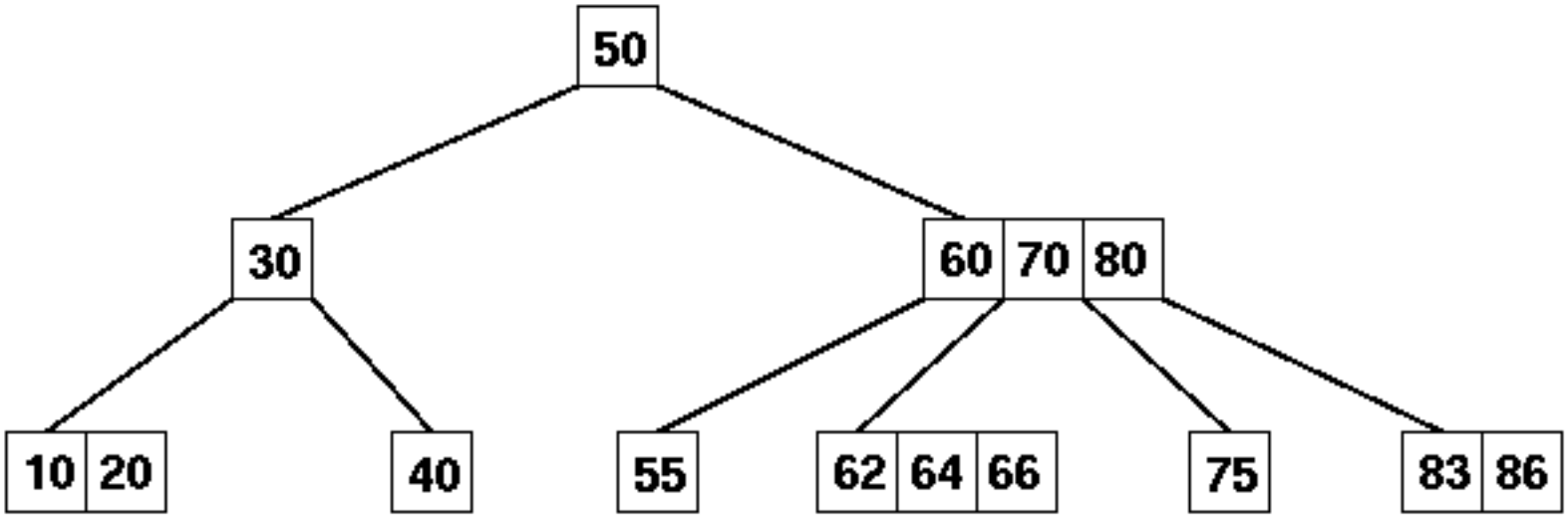
Criterios de búsqueda

Si la búsqueda en esta página fracasa, entonces:

- 1. Si el argumento de búsqueda se encuentra entre dos llaves de la página, $k_i < x < k_{i+1}$ entonces:
  - La búsqueda sigue en la página apuntada por el puntero entre ellas, p_i
- 2. Si el argumento de búsqueda es mayor que la última llave de la página, $k_m < x$, entonces:
  - Se sigue la búsqueda en la página apuntada por el último apuntador, p_m
- 3. Si el argumento de búsqueda es menor que la primera llave de la página, $x < k_1$, entonces:
  - Se sigue la búsqueda en la página apuntada por el primer apuntador, p_0
- 4. Si en cualquier caso el apuntador a la página en la que hay que seguir la búsqueda es *NIL* entonces:
  - El argumento de búsqueda *x* no está en el árbol y finaliza la búsqueda.



Búsqueda en Árboles B



- **Búsqueda**: igual que en los árboles binarios, eligiendo la rama por la que seguir.

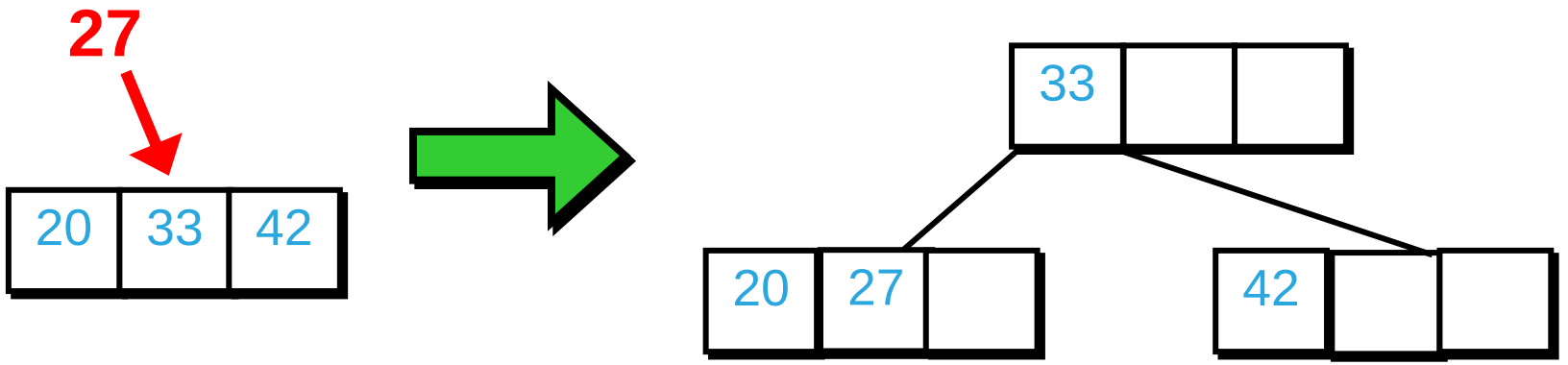
Inserción en Árboles B

- Buscar el nodo hoja donde se debería colocar la entrada.

  Si quedan sitios libres en esa hoja, insertarlo (en el orden adecuado).

  Si no quedan sitios (la hoja tiene **2t-1** valores) partir la hoja en **2** hojas (con $\lceil$ **(2t-1)/2** $\rceil$ y $\lfloor$ **(2t-1)/2** $\rfloor$ nodos cada una) y añadir la mediana al nodo padre.

  Si en el padre no caben más elementos, repetir recursivamente la partición de las hojas.

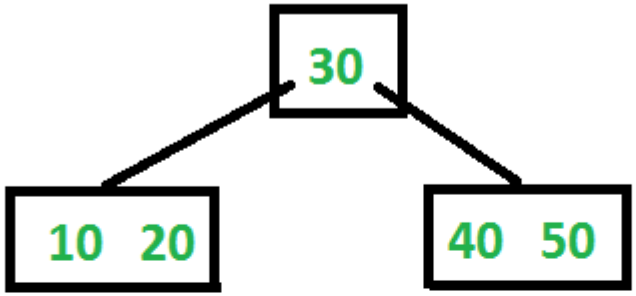


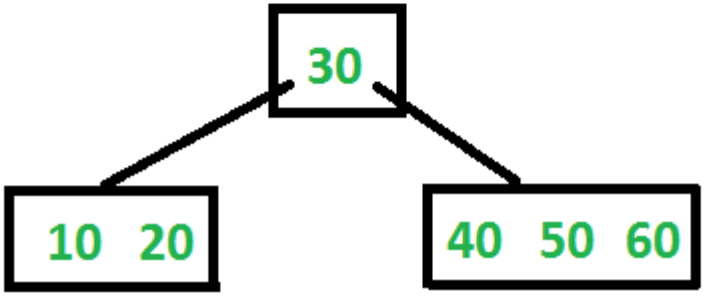
Árboles B de grado 3

Insert 10

Insert 20, 30, 40 and 50

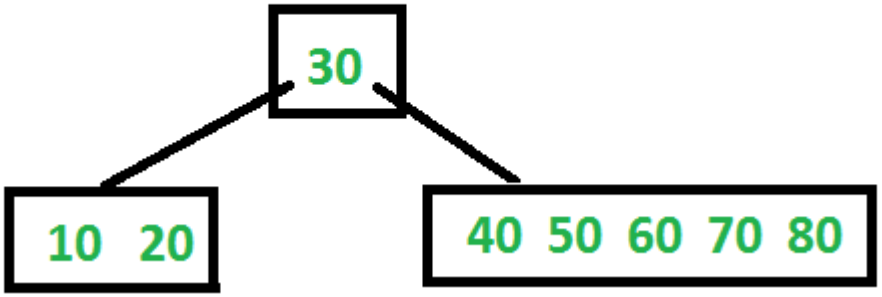
Insert 60



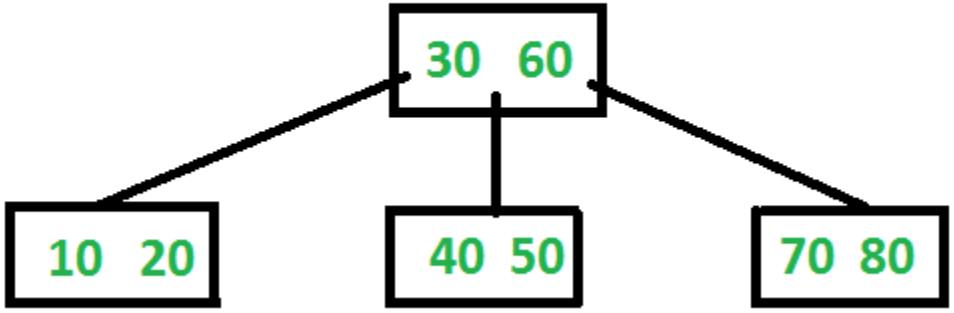


Árboles B de grado 3

Insert 70 and 80

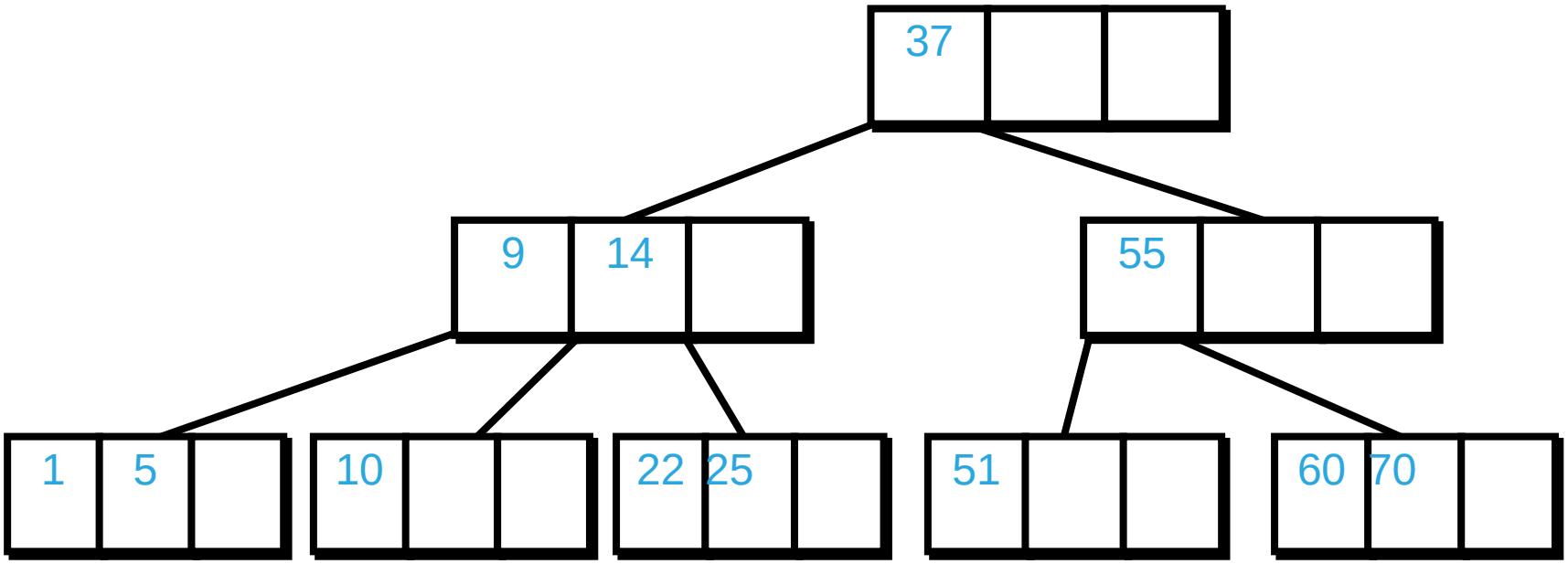


Insert 90



Inserción en Árboles B

- **Ejemplo:** En un árbol B de grado **t**=2, insertar las claves: 37, 14, 60, 9, 22, 51, 10, 5, 55, 70, 1, 25.



- ¿Cuál es el resultado en un árbol B?

Inserción en Árboles B

- **Ejercicio:** En un árbol B de grado **t**=3 inicialmente vacío, introduzca las siguientes claves: 30, 60, 45, 8, 22, 35, 50, 38, 61, 54, 110, 105, 200, 1, 15, 23, 70, 4, 3, 100, 99

- ¿Cuál es el resultado en un árbol B?

Inserción en Árboles B

- **Ejercicio:** En un árbol B de grado **t**=3 inicialmente vacío, introduzca las siguientes claves: 30, 60, 45, 8, 22, 35, 50, 38, 61, 54, 110, 105, 200, 1, 15, 23, 70, 4, 3, 100, 99, 101, 102

- ¿Cuál es el resultado en un árbol B?

3.4. Árboles B

- **Eliminación de entradas en un árbol B:** Buscar la clave en el árbol.
    1. **Nodo interno (no hoja):** Sustituirla por la siguiente (o la anterior) en el orden. Es decir, por la mayor de la rama izquierda, o la menor de la rama derecha.
    2. **Nodo hoja:** Eliminar la entrada de la hoja.

- **Casos de eliminación en nodo hoja. d = (t-1)**

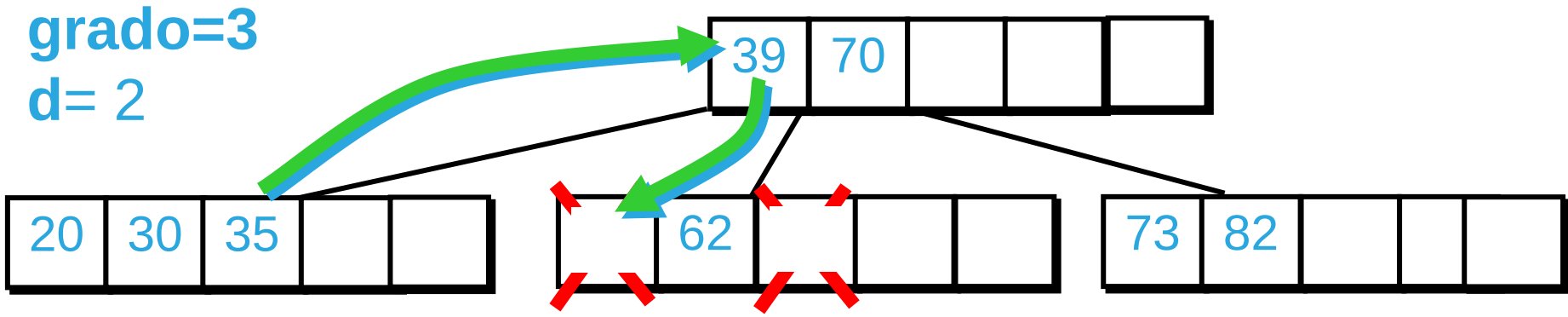
    Nodo con más de **d** entradas: suprimir la entrada.

    Nodo con **d** entradas (el mínimo posible): reequilibrar el árbol.

3.4. Árboles B

- **Eliminación en nodo con d entradas**:

  Nodo hermano con más de d entradas: Se produce un proceso de **préstamo** de entradas: Se suprime la entrada, la entrada del padre (más próximo) pasa a la hoja de supresión y la vecina cede una entrada al nodo padre.



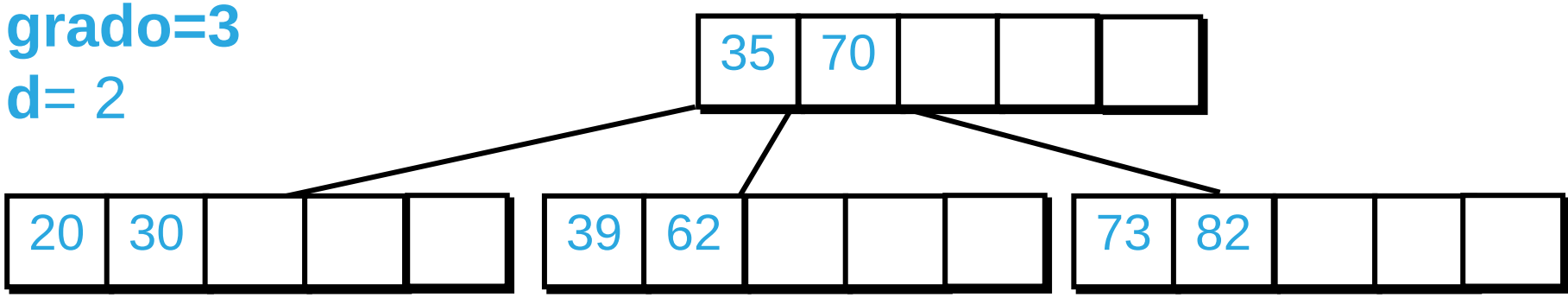
- **Ejemplo.** Eliminar 67, 45.

3.4. Árboles B

- **Eliminación en nodo con d entradas**:

  Nodo hermano con más de d entradas: Se produce un proceso de **préstamo** de entradas: Se suprime la entrada, la entrada del padre (más próximo) pasa a la hoja de supresión y la vecina cede una entrada al nodo padre.

grado=3
d= 2

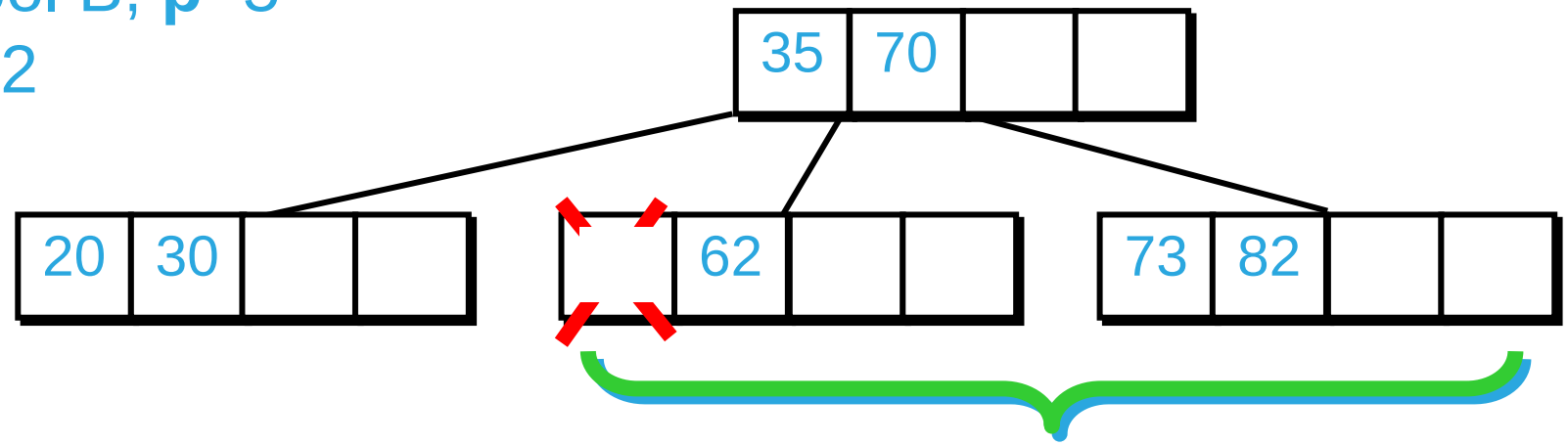


- **Ejemplo.** Eliminar 67, 45.

3.4. Árboles B

Ningún hermano con más de d entradas: Con la hoja donde se hace la supresión (**d-1** entradas) más una hoja hermana (**d** entradas), más la entrada del padre, se hace una nueva hoja con **2d** entradas.

Árbol B, **p**=5
d= 2

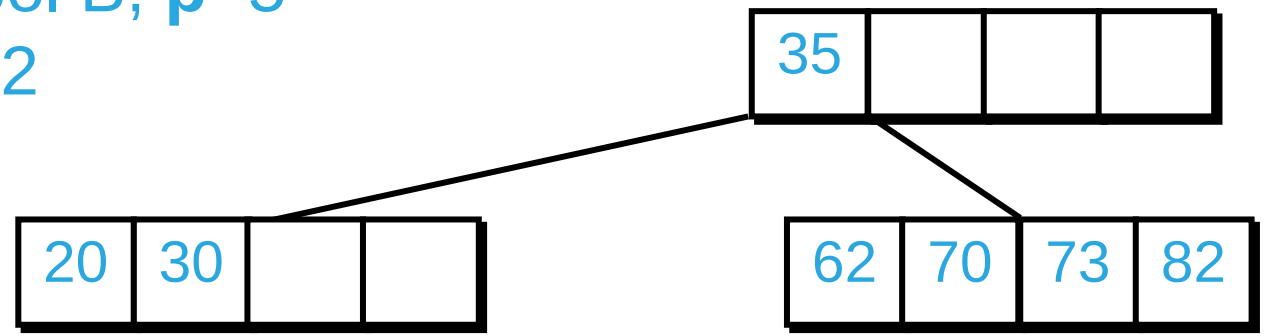


- **Ejemplo.** Eliminar 39.

3.4. Árboles B

Ningún hermano con más de d entradas: Con la hoja donde se hace la supresión (**d-1** entradas) más una hoja hermana (**d** entradas) más la entrada del padre, se hace una nueva hoja con **2d** entradas.

Árbol B, **p**=5
d= 2



- **Ejemplo.** Eliminar 39.
- **Ojo:** se suprime una entrada en el padre. Se repite el proceso de eliminación en el nivel superior.