Analista Programador Universitario

Estructura de Datos

Unidad VI: Árboles B

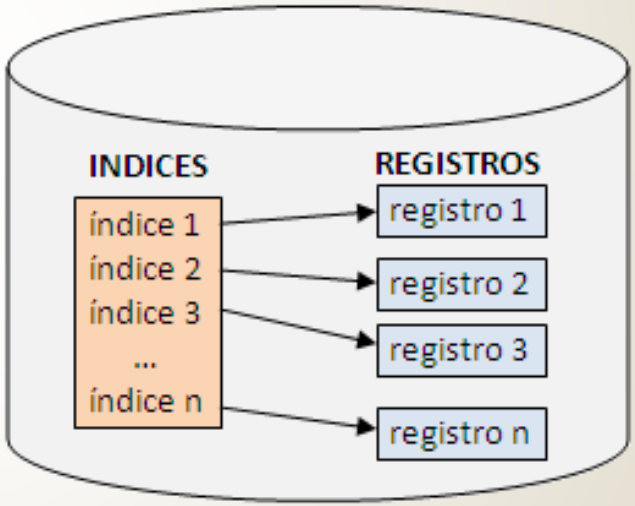
Facultad de Ingeniería
Universidad Nacional de Jujuy

Árboles B (1)

- En el manejo de grandes volúmenes de información, siempre estuvo presente la necesidad de HACER EFICIENTE EL PROCESO DE BÚSQUEDA.

- *Problema*: acceso rápido y eficiente a grandes volúmenes de datos indizados almacenados en memoria secundaria.

  Por ejemplo, las bases de datos de los buscadores como *Google*.



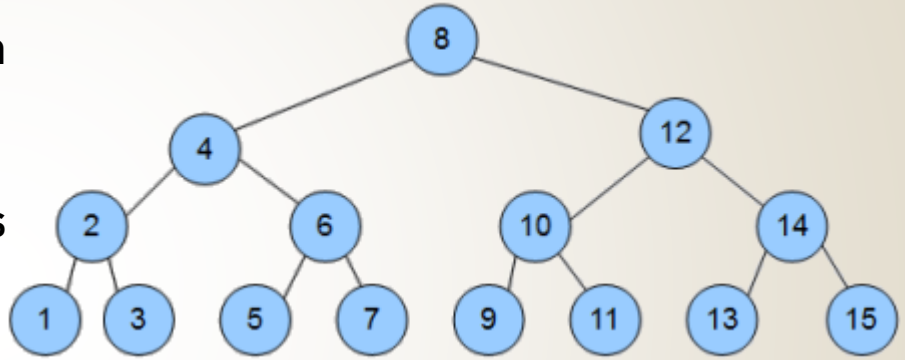
Árboles B (2)

- *Solución 1:* los árboles binarios de búsqueda estructuran la información de modo que la búsqueda se realice rápidamente *¿qué ocurre si el volumen de datos es muy grande?*

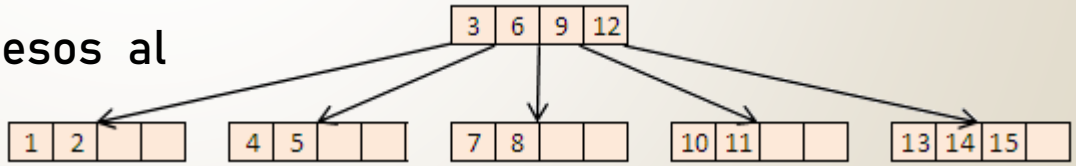
- *Solución 2*: a principios de los años 70, Rudolf Bayer y Edward McCreight propusieron los árboles B (estructuras no lineales) como solución al mantenimiento de índices en almacenamiento secundario.

Árboles B (3)

- *Ventajas*
  - Poseen una estructura que agiliza la búsqueda.
  - Tienen menor profundidad que sus equivalentes binarios.
  - Aprovechan mejor el espacio de almacenamiento secundario.
  - Reducen el número de accesos al dispositivo.



Árbol binario de búsqueda con 4 niveles

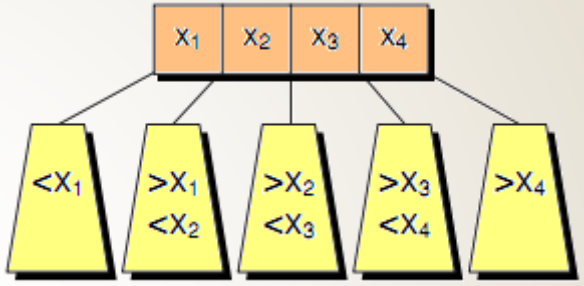


Árbol B de orden 2 con 2 niveles

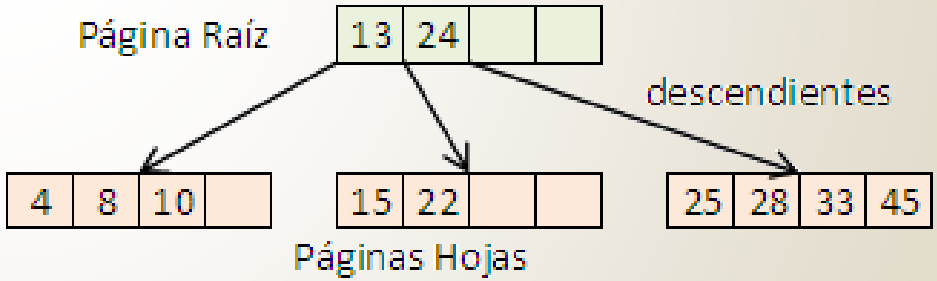
Definición (1)

Un árbol B de orden *n* es un *árbol de búsqueda* que satisface las siguientes propiedades:

- Cada página tiene como máximo *2 n* claves.
- Cada página tiene como mínimo *n* claves, salvo la raíz que puede tener sólo 1.
- Una página con *m* claves tiene *m+1* descendientes o *ninguno* (página hoja).
- Todas las páginas hojas aparecen al mismo nivel (condición de balance).

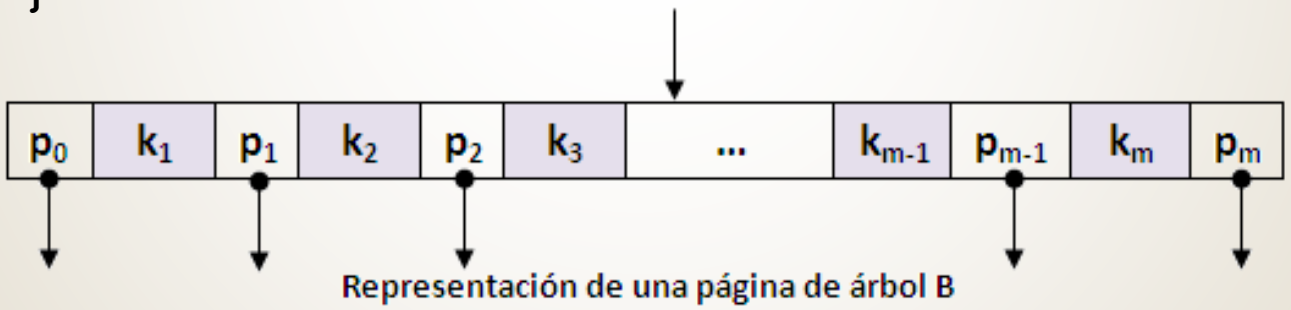


ÁRBOL B DE ORDEN 2



Definición (2)

```
typedef struct pagina *ppagina
typedef struct pagina
        {
         tipoClave claves[k]; //claves (k=2*n)
         ppagina ramas[k+1]; //punteros a páginas
         int cuenta;  //número de claves de la página
        }
```



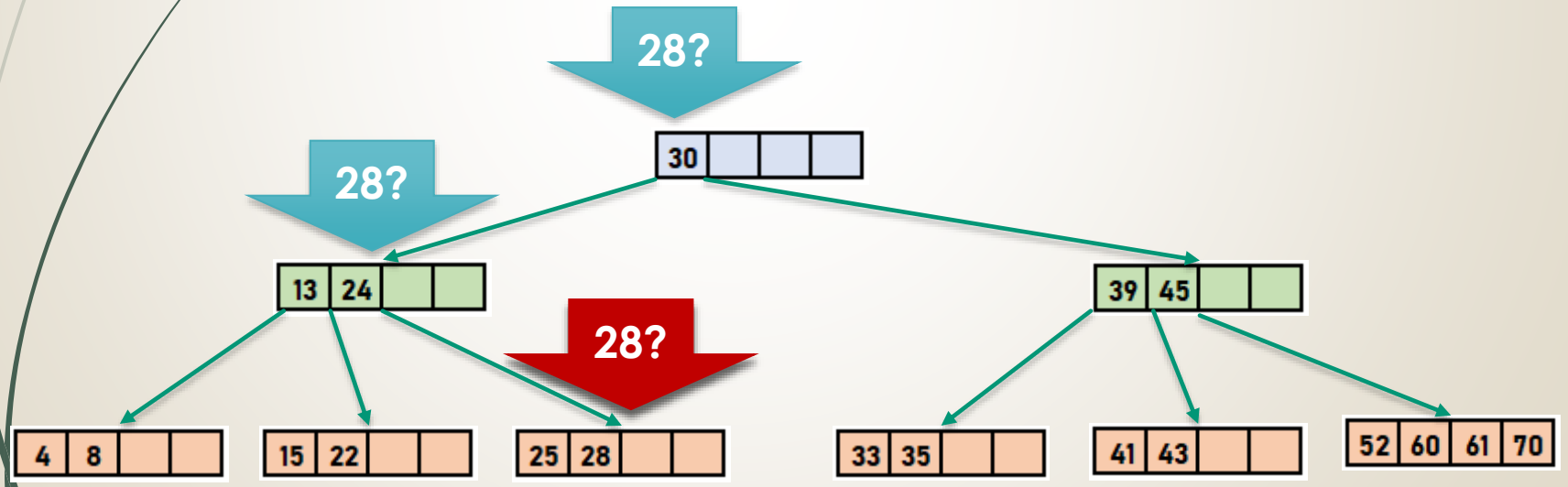
Representación de una página de árbol B

Operaciones Básicas

- Búsqueda
- Inserción
- Eliminación
  - Caso 1: Eliminar un elemento de una página hoja.
    A. La hoja tiene más de *n* claves
    B. La hoja tiene *n* claves
  - Caso 2: Eliminar un elemento de una página interna.
    - ✓ Estrategia Mayor de los Menores
    - ✓ Estrategia Menor de los Mayores

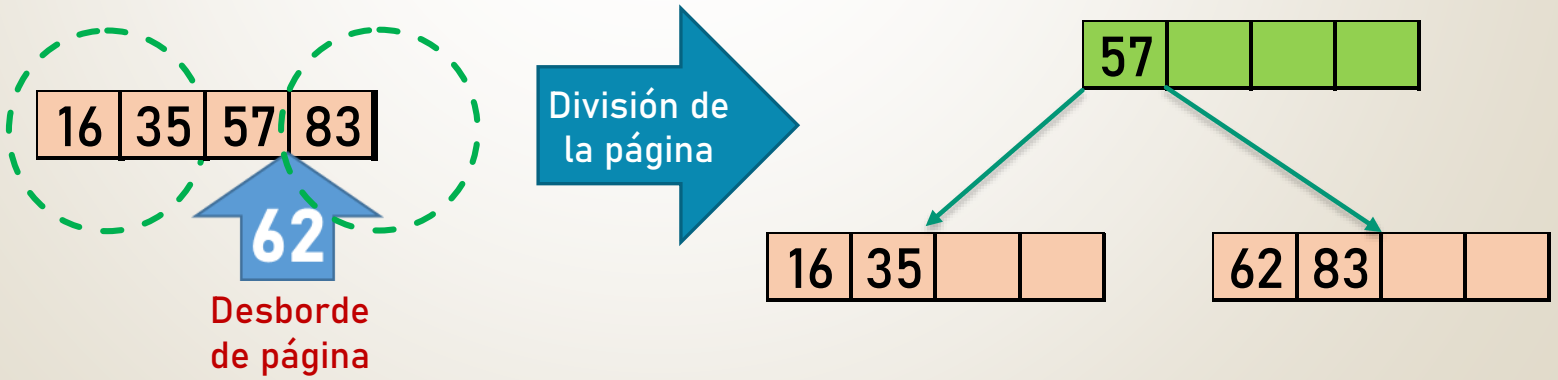
Búsqueda

- El proceso de búsqueda se inicia en la raíz del árbol, explorando las páginas de los siguientes niveles según el valor de las claves almacenadas. La búsqueda finaliza en una página hoja.
- Por ej., si se busca el valor 28 en el siguiente árbol B.

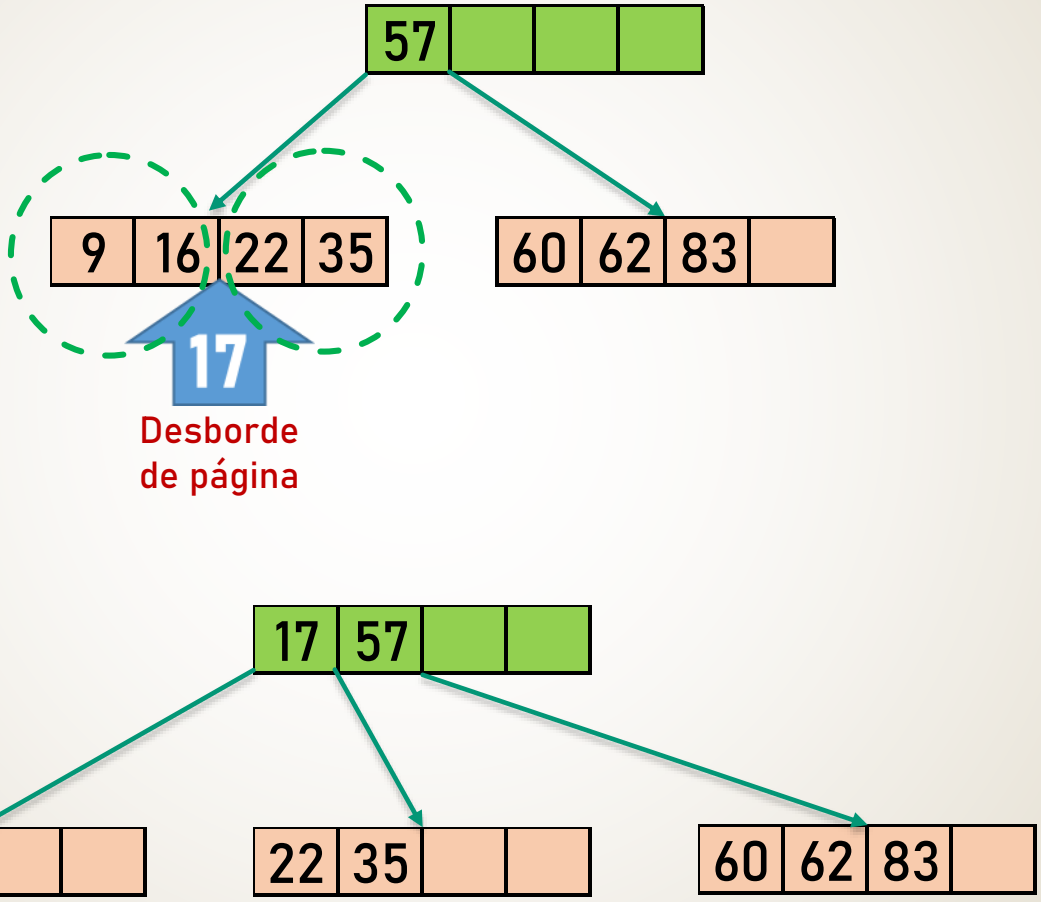


Inserción (1)

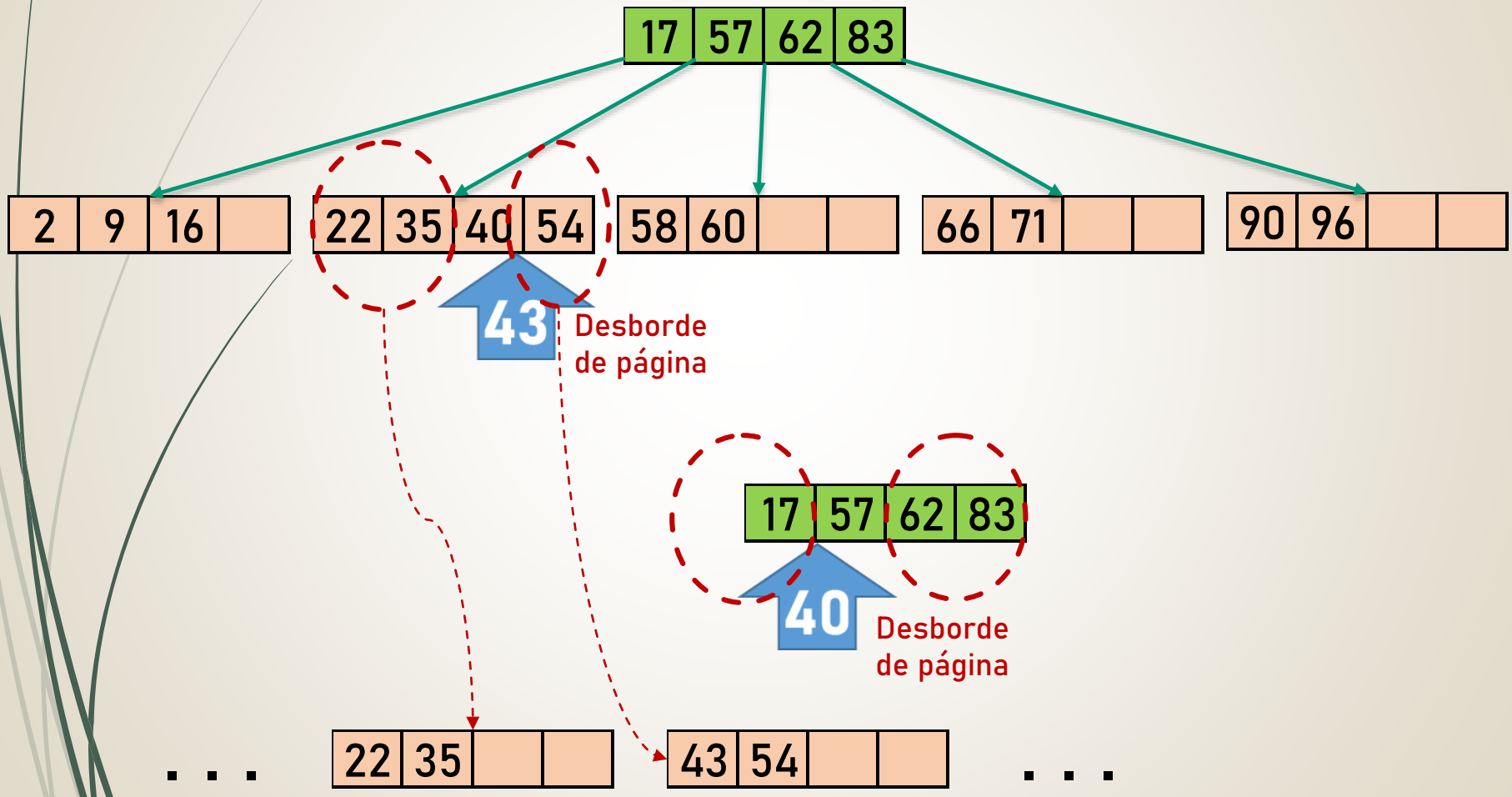
- Para insertar un nuevo elemento en un árbol B, éste se recorre hasta encontrar la página hoja en la que debe almacenarse.
- Si la página está completa se produce un desborde de página y es necesario dividir ésta en 2 nuevas páginas, llevando la clave central a la página inmediata de nivel superior.



Inserción (2)

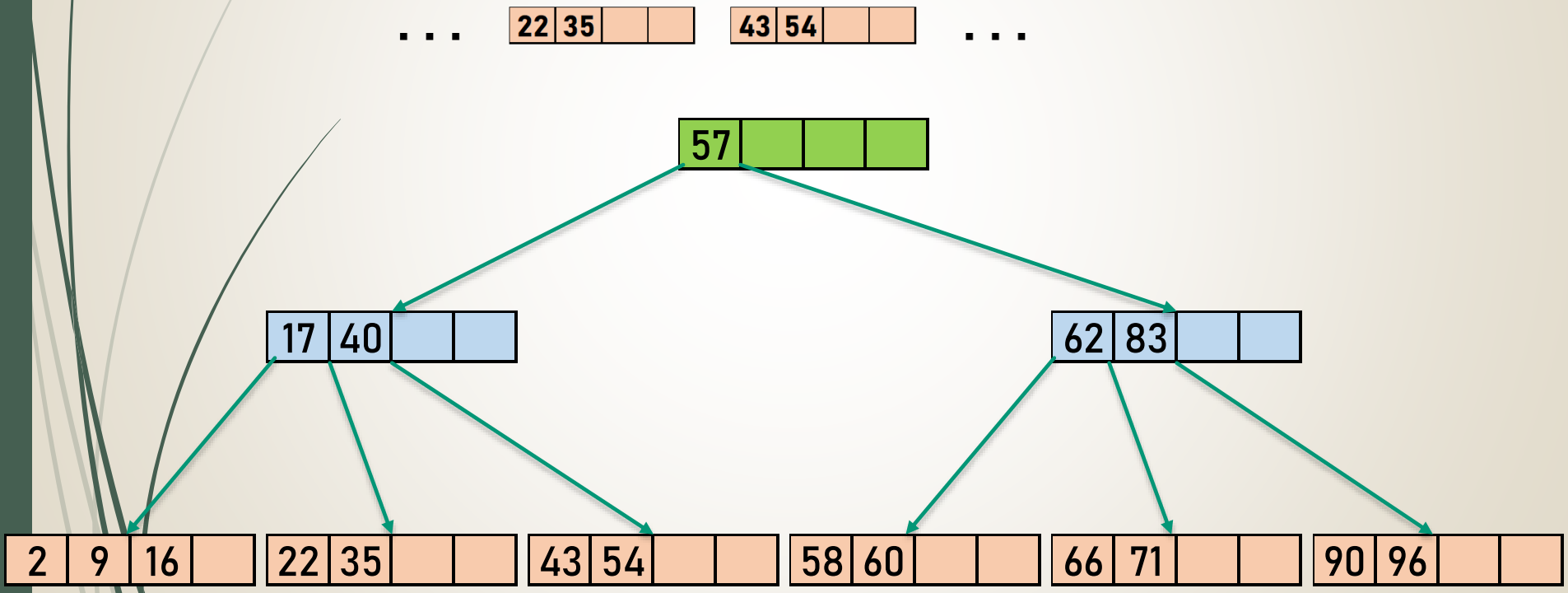

57
9
16
22
35
60
62
83
17
Desborde
de página
17
57
9
16
22
35
60
62
83

Inserción (3)


17 57 62 83
2 9 16
22 35 40 54
58 60
66 71
90 96
43
Desborde de página
17 57 62 83
40
Desborde de página
22 35
43 54

Inserción (4)


17
57
62
83
40
Desborde
de página

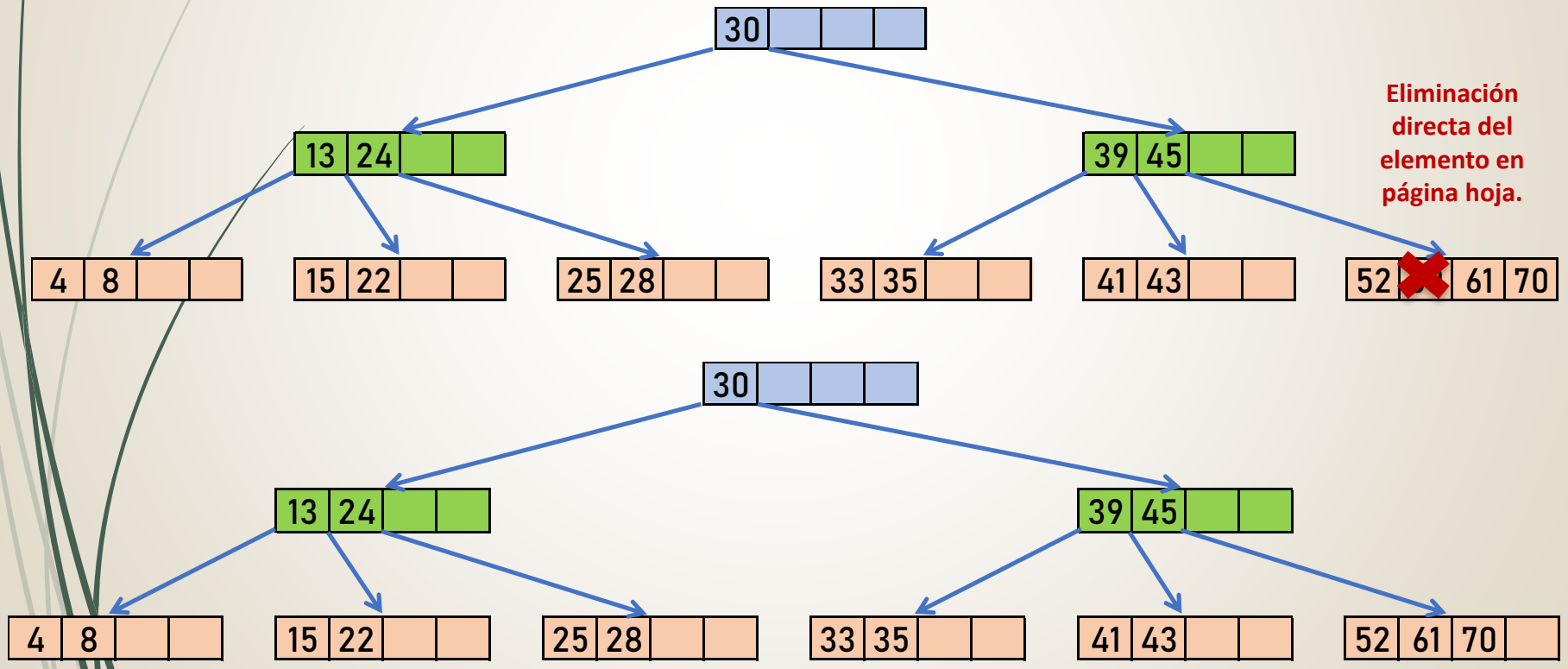

. . .
22
35
43
54
. . .
57
17
40
62
83
2
9
16
22
35
43
54
58
60
66
71
90
96

Eliminación

- Eliminación
  - Caso 1: Eliminar un elemento de una página hoja.
    A. La hoja tiene más de *n* claves
    B. La hoja tiene *n* claves
  - Caso 2: Eliminar un elemento de una página interna.
    - ✓ Estrategia Mayor de los Menores
    - ✓ Estrategia Menor de los Mayores

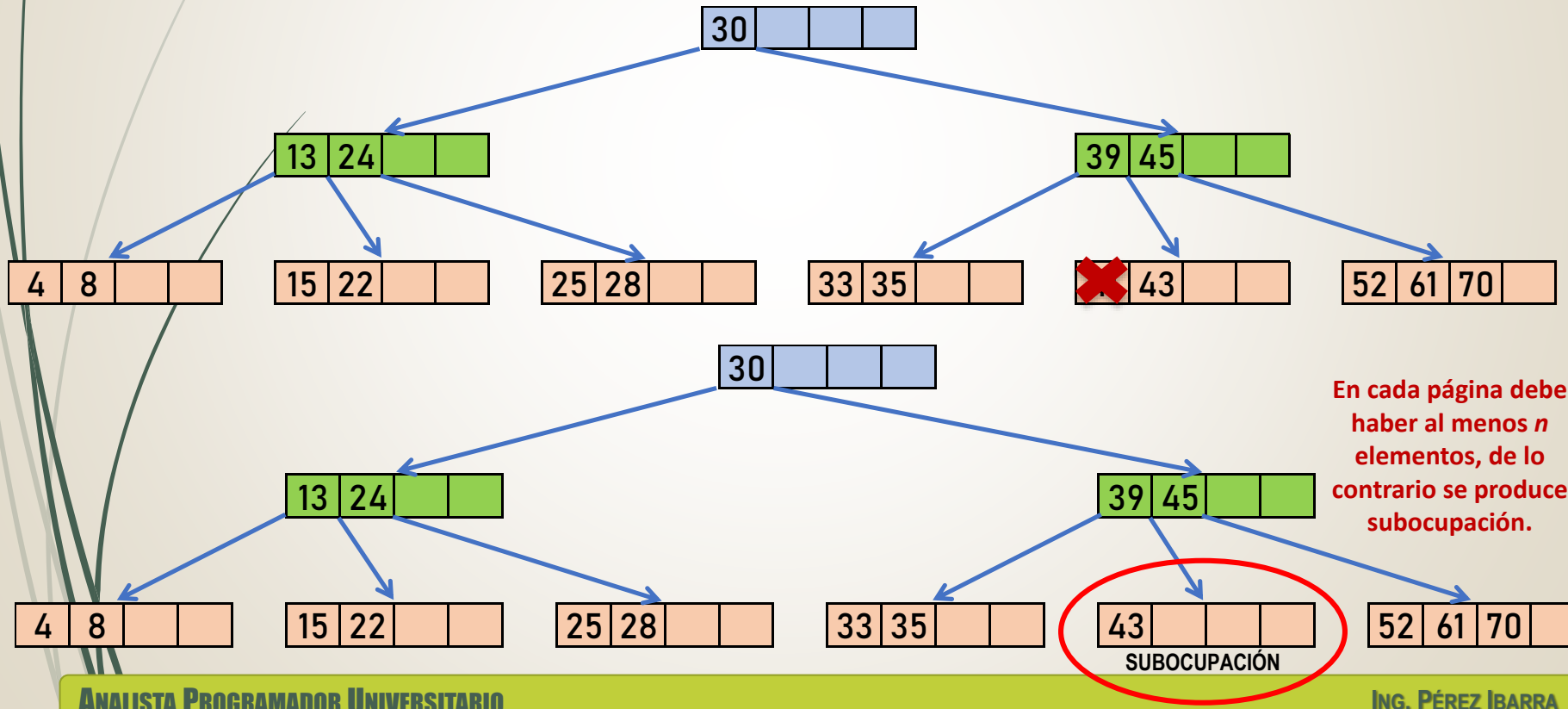
Eliminación. Caso 1A

- Caso 1.A: Eliminar un elemento que pertenece a una página hoja, con más de *n* elementos.



Eliminación. Caso 1B (1)

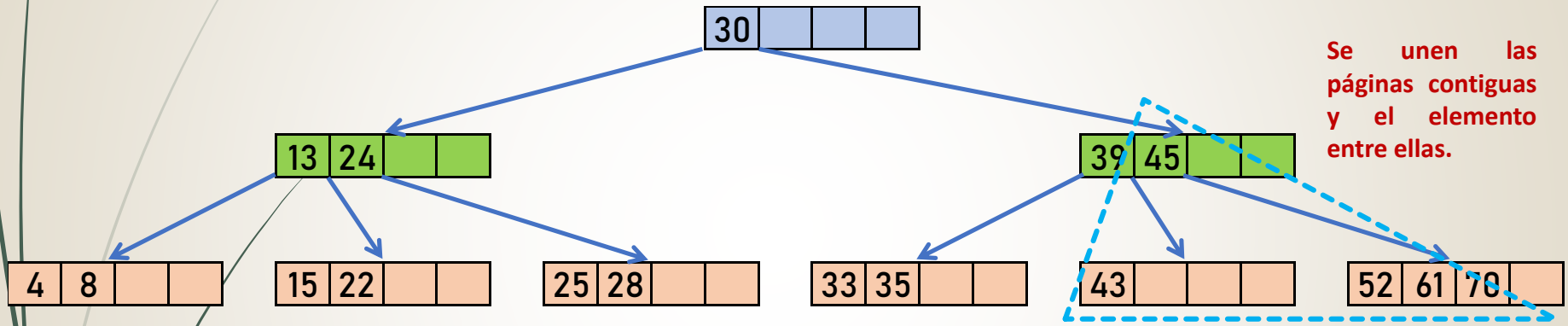
- Caso 1.B: Eliminar un elemento que pertenece a una página hoja, con *n* elementos.



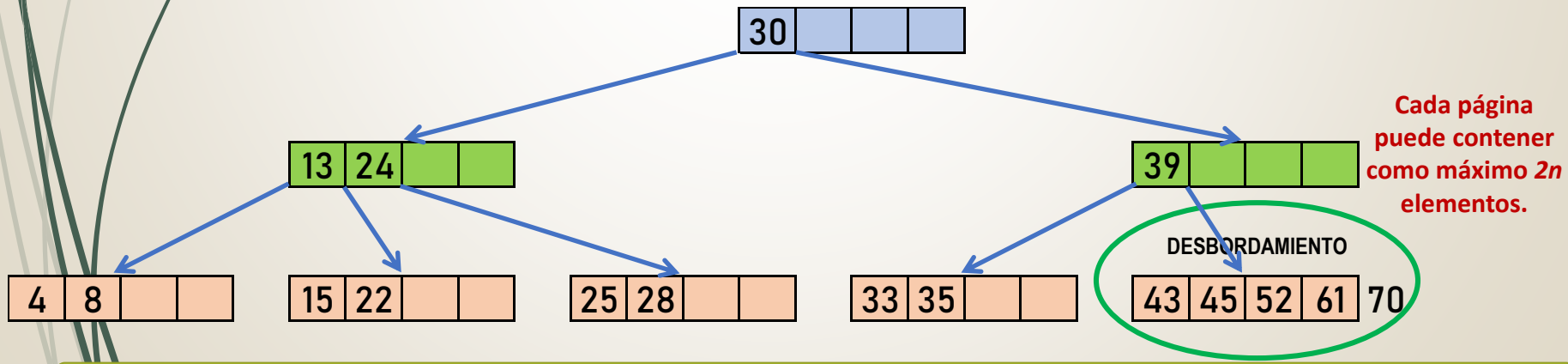
Eliminación. Caso 1B (2)

ALTERNATIVA 1

- ¿Cómo se resuelve la subocupación?



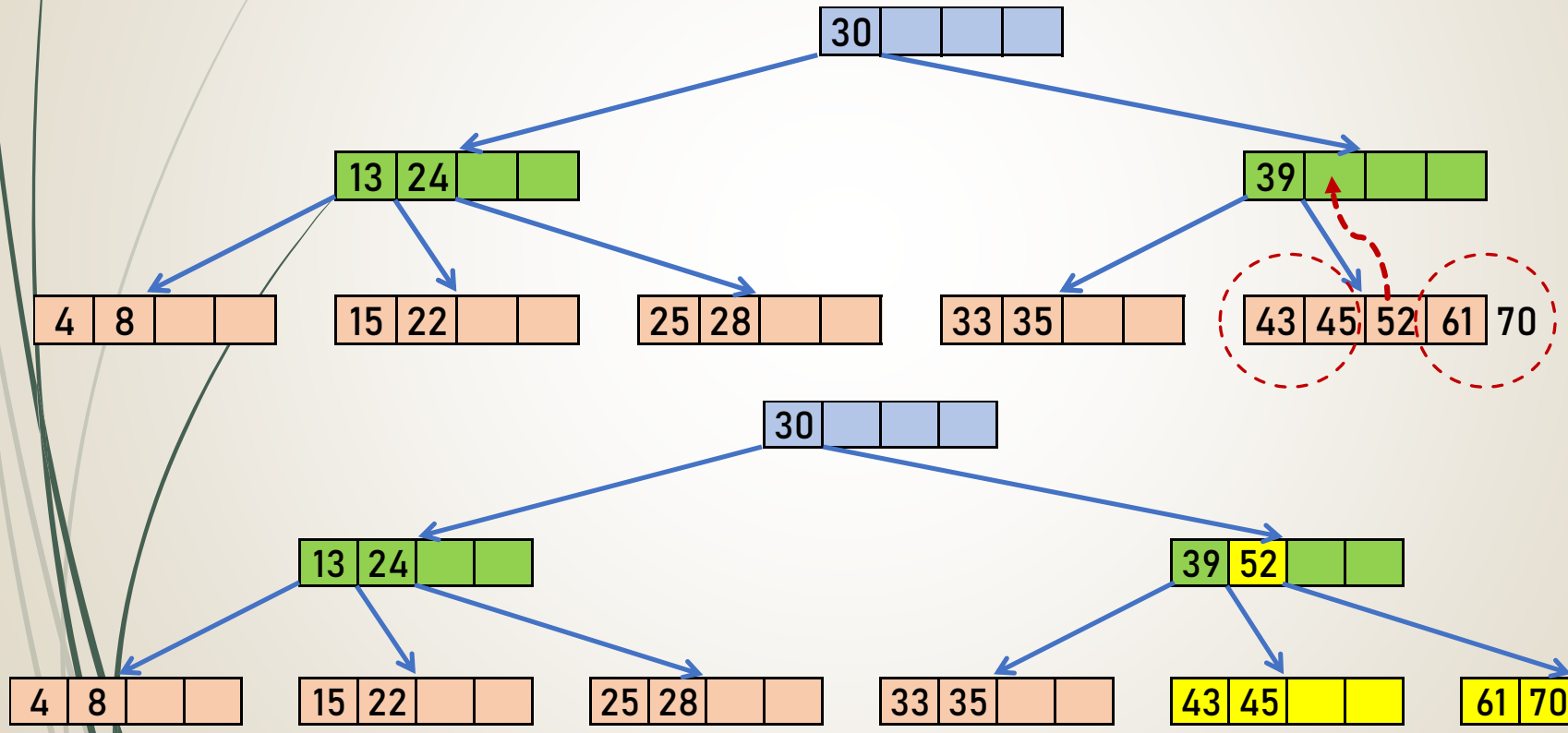
Se unen las páginas contiguas y el elemento entre ellas.



Cada página puede contener como máximo *2n* elementos.

Eliminación. Caso 1B (3)

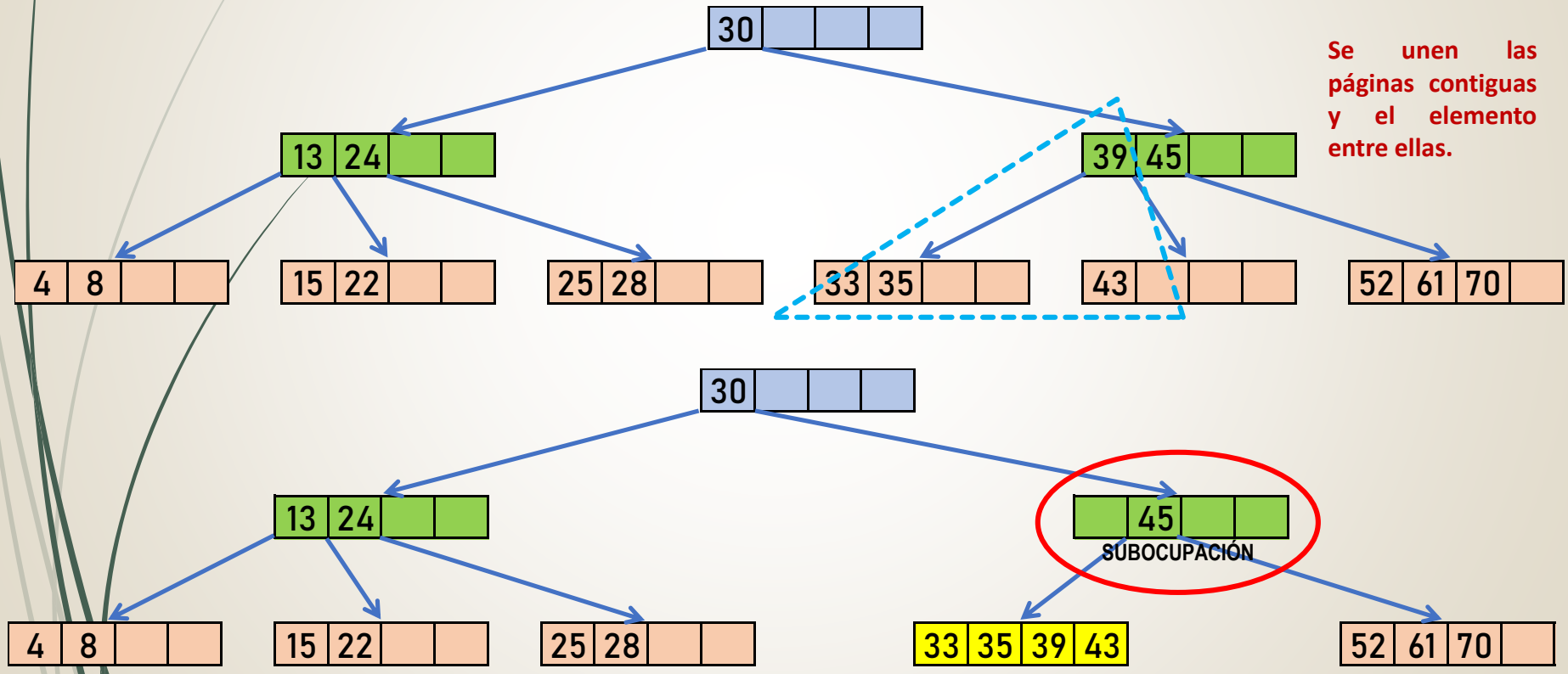
- ¿Cómo se resuelve la subocupación?



Eliminación. Caso 1B (2)

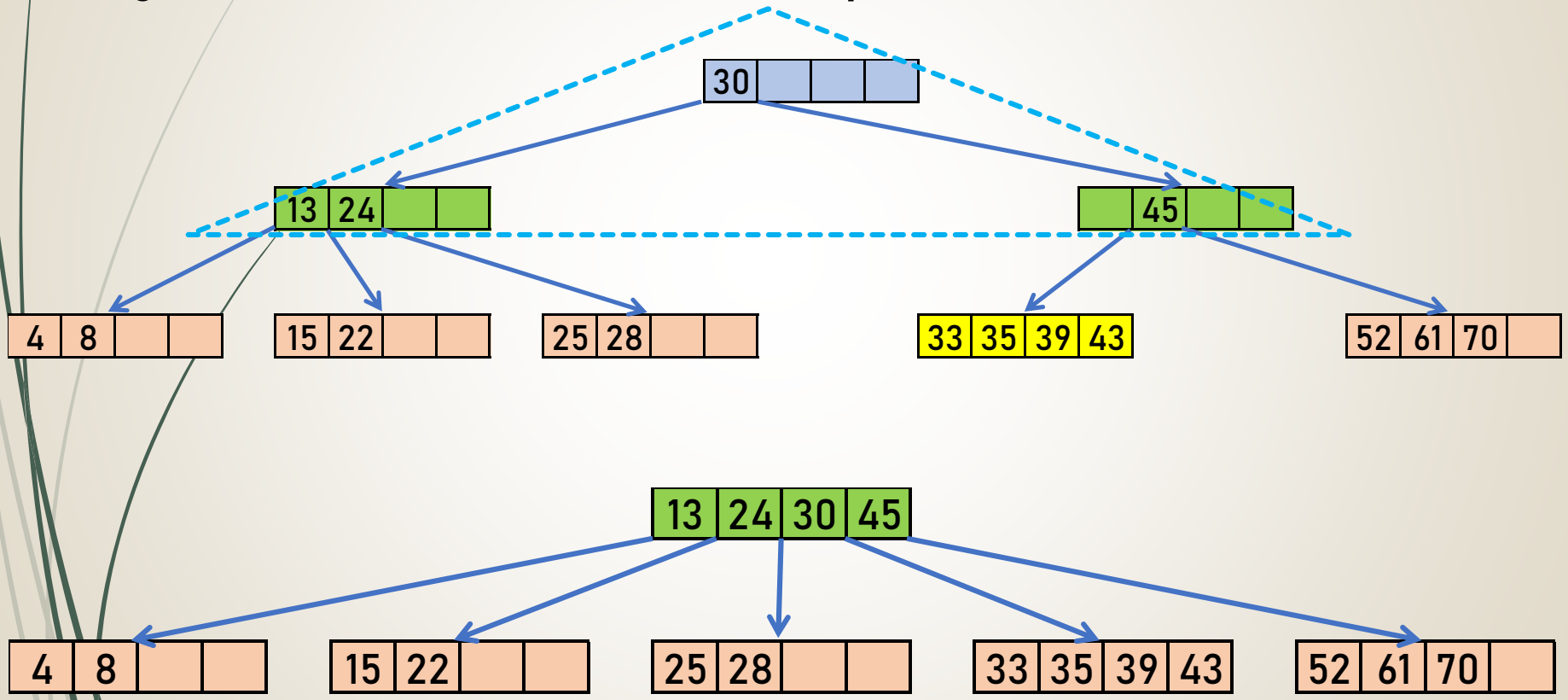
ALTERNATIVA 2

- ¿Cómo se resuelve la subocupación?



Eliminación. Caso 1B (3)

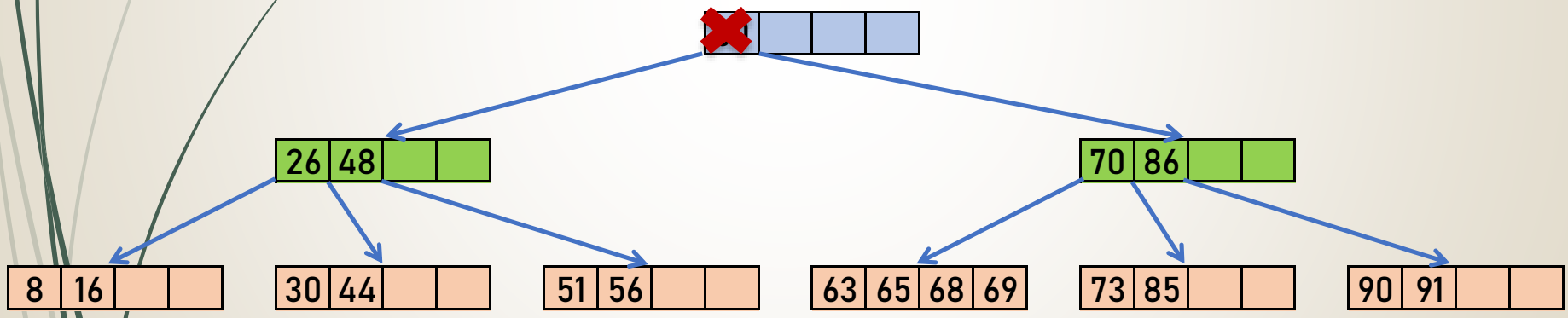
- ¿Cómo se resuelve la subocupación?



Eliminación. Caso 2 (1)

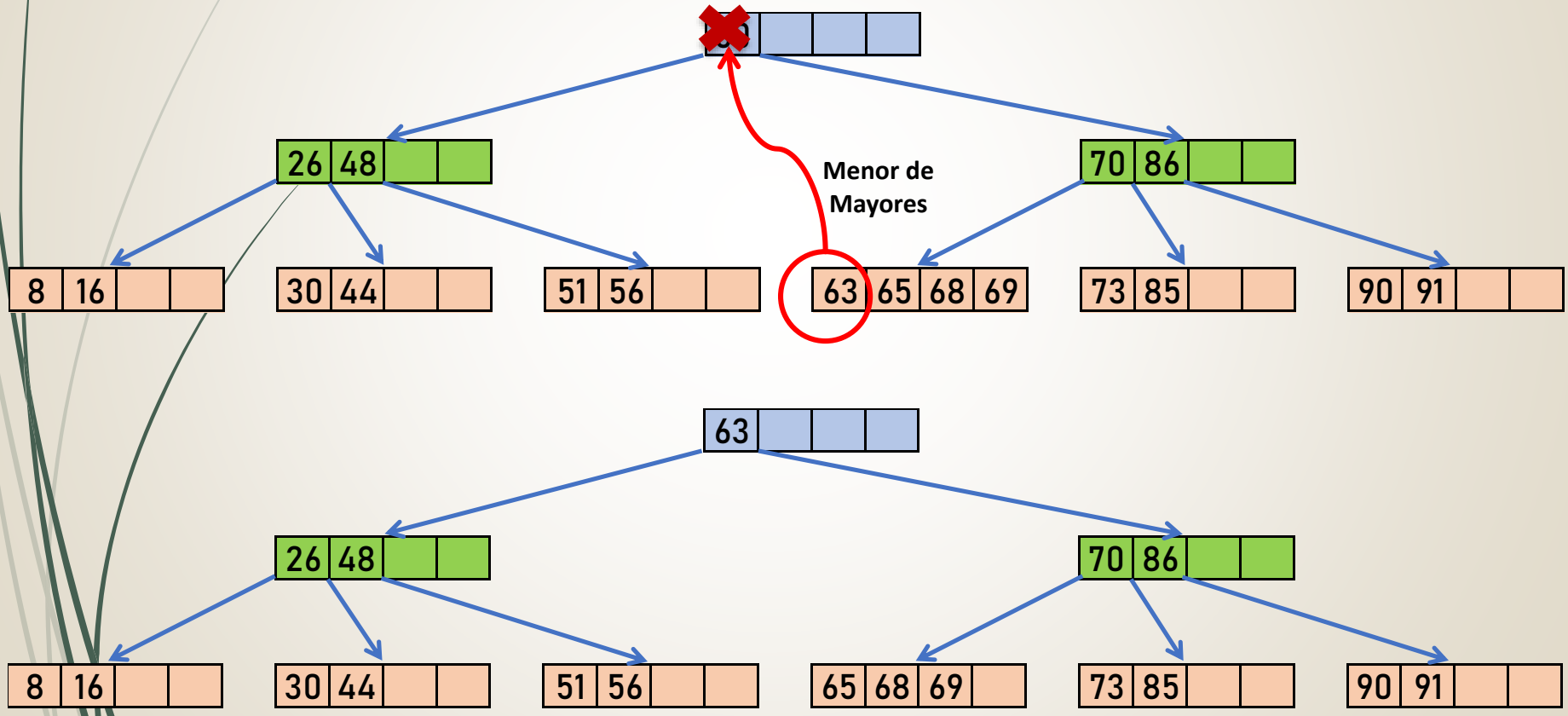
- Caso 2: Eliminar un elemento que pertenece a una página interior.

  En este caso el elemento a eliminar se sustituye por otro elemento del árbol.



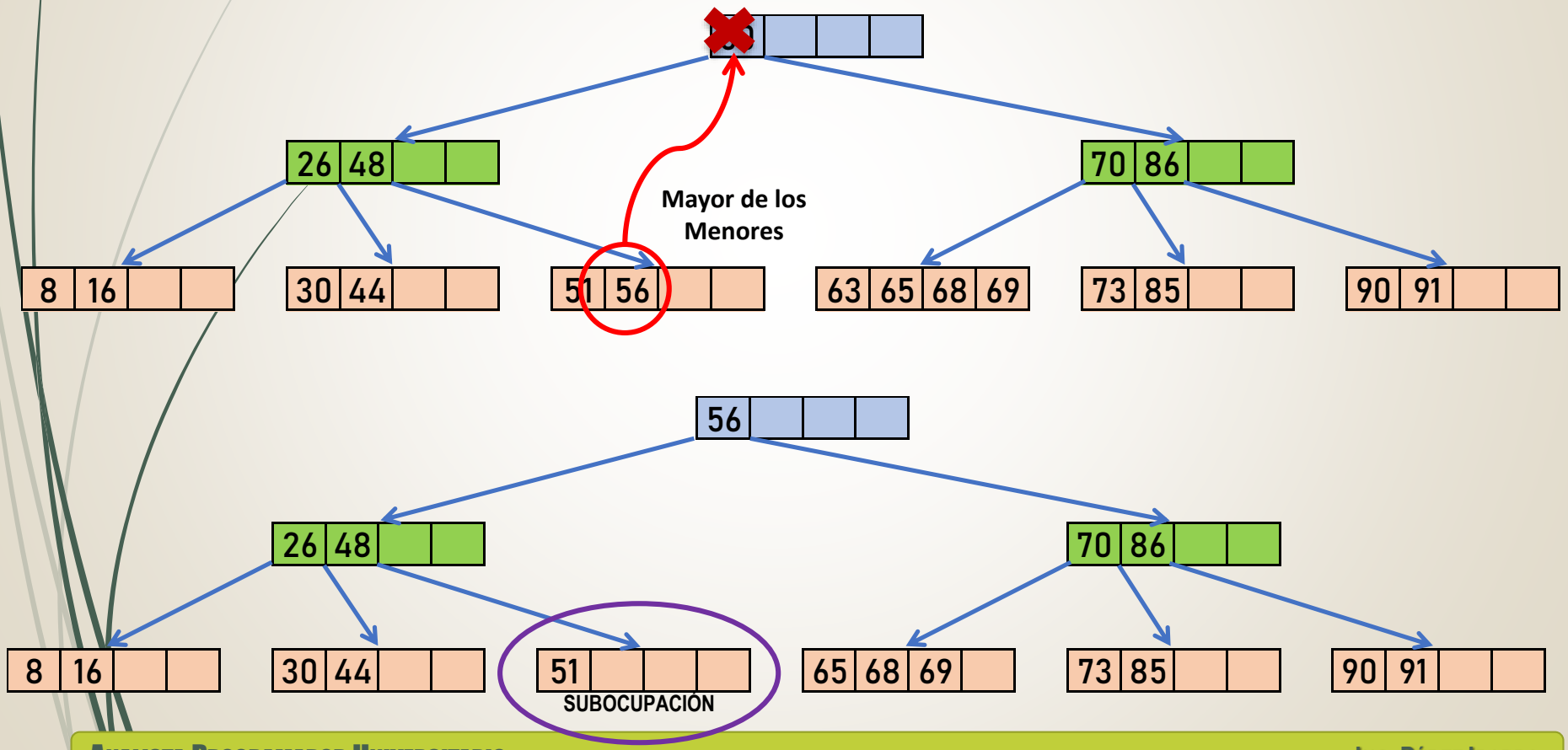
Eliminación. Caso 2 (2)

- Sustituir por el Menor de los Mayores



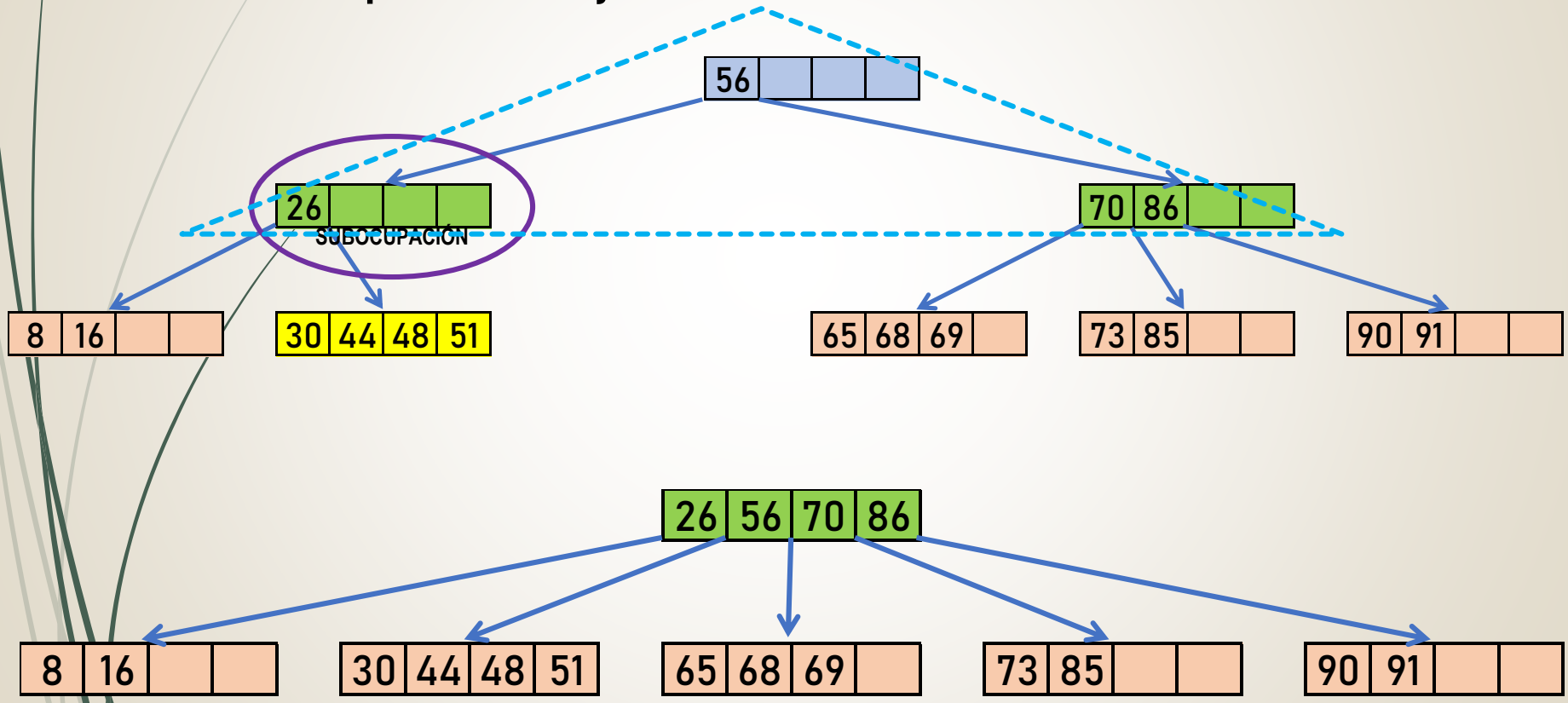
Eliminación. Caso 2 (3)

- Sustituir por el Mayor de los Menores



Eliminación. Caso 2 (3)

- Sustituir por el Mayor de los Menores



Bibliografía

- Hernández, Roberto *et al*. Estructuras de Datos y Algoritmos. Prentice Hall. 2001.
- Joyanes Aguilar *et al*. Estructuras de Datos en C++. Mc Graw Hill. 2007.
- De Giusti, Armando *et al.* Algoritmos, datos y programas, conceptos básicos. Editorial Exacta. 1998.
- Joyanes Aguilar, Luis. Fundamentos de Programación. Mc Graw Hill. 1996.