UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Sesión 1.1: Preludio

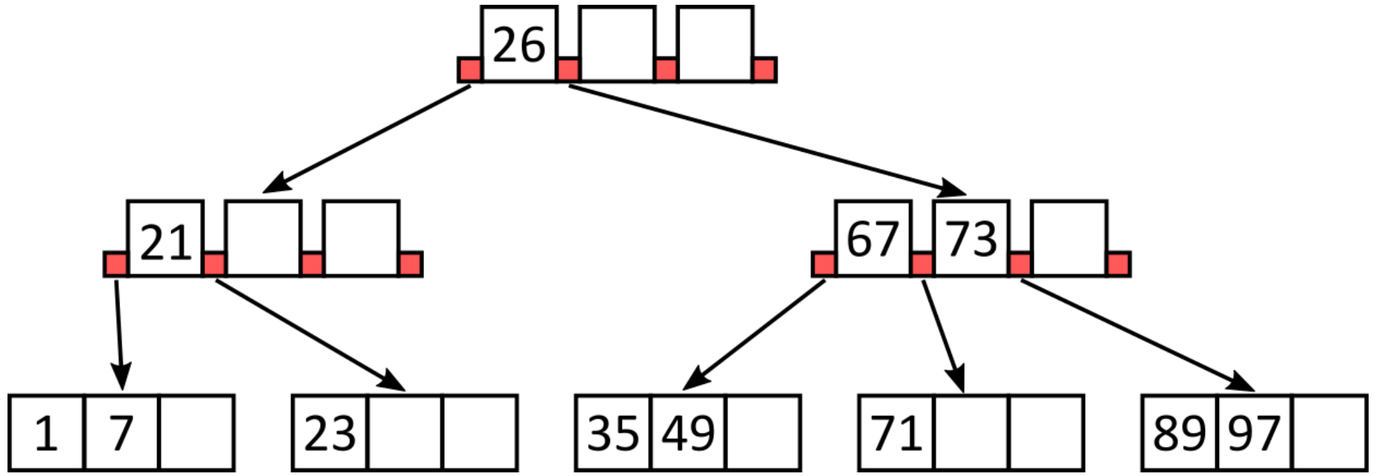
CS3102 EDA

Índice

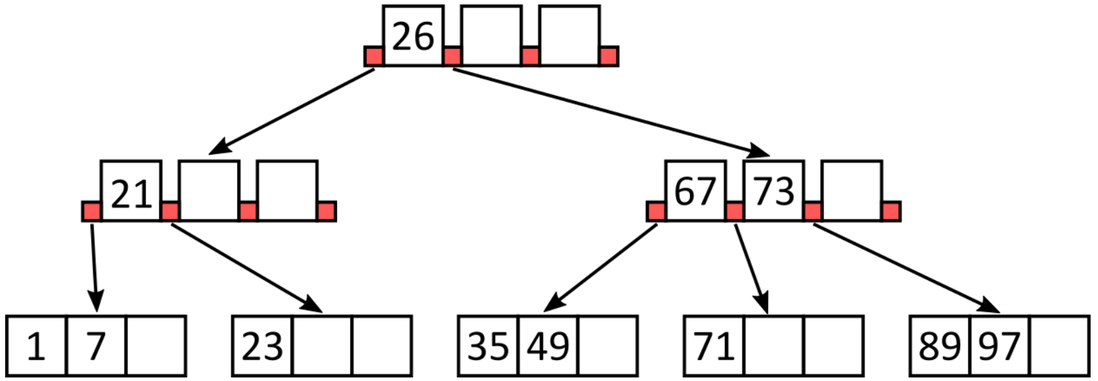
1. B-Tree
2. B+Tree
3. Range Tree

1. B-*Tree*

B-Tree



B-Tree



- Todos los nodos hoja deben estar al mismo nivel.
- Todos los nodos, excepto el raíz, deben tener al menos $\left\lceil \frac{m}{2} \right\rceil - 1$ claves y un máximo de $m - 1$ claves.
- Todos los nodos que no son hojas excepto la raíz (es decir, todos los nodos internos) deben tener al menos $\frac{m}{2}$ hijos.
- Si el nodo raíz no es un nodo hoja, entonces debe tener al menos 2 hijos.
- Un nodo no hoja con $n - 1$ claves debe tener n número de hijos.
- Todos los valores clave en un nodo deben estar en orden ascendente.

B-Tree: *Search*

Paso 1: lea el elemento de búsqueda del usuario.

Paso 2: compare el elemento de búsqueda con el primer valor clave del nodo raíz en el árbol.

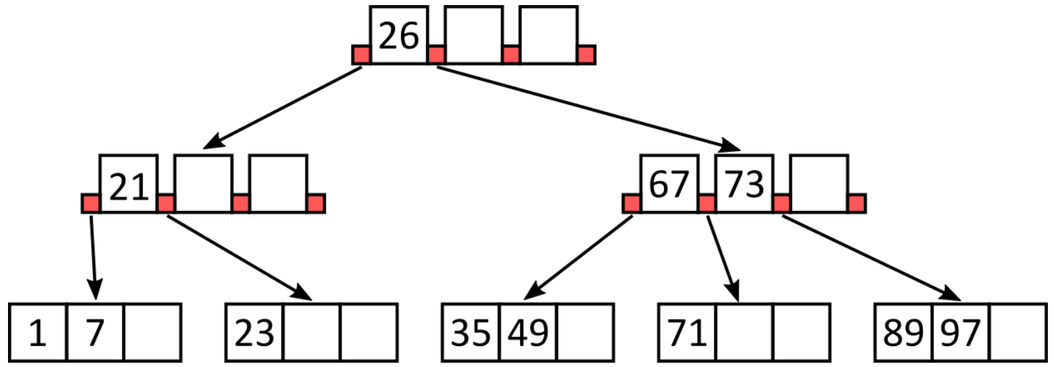
Paso 3: si ambos coinciden terminar la función

Paso 4: si no coinciden, compruebe si el elemento de búsqueda es más pequeño o más grande que el valor de la clave.

Paso 5: si el elemento de búsqueda es más pequeño, continúe el proceso de búsqueda en el subárbol izquierdo.

Paso 6: si el elemento de búsqueda es más grande, compare el elemento de búsqueda con el siguiente valor clave en el mismo nodo y repita los pasos 3, 4, 5 y 6 hasta que encontremos la coincidencia exacta o hasta que el elemento de búsqueda se compare con el último valor clave en el nodo hoja.

Paso 7: si el último valor clave en el nodo hoja tampoco coincide, el elemento no pertenece al árbol.



B-Tree: *Insert*

1, 21, 23, 67, 89, 97, 73, 7, 35, 49, 71, 26

Paso 1: compruebe si el árbol está vacío.

Paso 2: si el árbol está vacío, cree un nuevo nodo con un nuevo valor de clave e insértelo en el árbol como un nodo raíz.

Paso 3: si el árbol no está vacío, busque el nodo de hoja adecuado al que se agrega el nuevo valor clave mediante la lógica del árbol de búsqueda binaria.

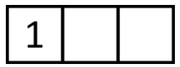
Paso 4: si ese nodo hoja tiene una posición vacía, agregue el nuevo valor clave a ese nodo hoja en orden ascendente de valor clave dentro del nodo.

Paso 5: si ese nodo de hoja ya está lleno, divida ese nodo de hoja enviando el valor medio a su nodo padre. Repita lo mismo hasta que el valor de envío se fije en un nodo.

Paso 6: si la división se realiza en el nodo raíz, el valor medio se convierte en un nuevo nodo raíz para el árbol y la altura del árbol aumenta en uno.

B-Tree: *Insert*

21, 23, 67, 89, 97, 73, 7, 35, 49, 71, 26



Paso 1: compruebe si el árbol está vacío.

Paso 2: si el árbol está vacío, cree un nuevo nodo con un nuevo valor de clave e insértelo en el árbol como un nodo raíz.

Paso 3: si el árbol no está vacío, busque el nodo de hoja adecuado al que se agrega el nuevo valor clave mediante la lógica del árbol de búsqueda binaria.

Paso 4: si ese nodo hoja tiene una posición vacía, agregue el nuevo valor clave a ese nodo hoja en orden ascendente de valor clave dentro del nodo.

Paso 5: si ese nodo de hoja ya está lleno, divida ese nodo de hoja enviando el valor medio a su nodo padre. Repita lo mismo hasta que el valor de envío se fije en un nodo.

Paso 6: si la división se realiza en el nodo raíz, el valor medio se convierte en un nuevo nodo raíz para el árbol y la altura del árbol aumenta en uno.

B-Tree: *Insert*

67, 89, 97, 73, 7, 35, 49, 71, 26

1	21	23

Paso 1: compruebe si el árbol está vacío.

Paso 2: si el árbol está vacío, cree un nuevo nodo con un nuevo valor de clave e insértelo en el árbol como un nodo raíz.

Paso 3: si el árbol no está vacío, busque el nodo de hoja adecuado al que se agrega el nuevo valor clave mediante la lógica del árbol de búsqueda binaria.

Paso 4: si ese nodo hoja tiene una posición vacía, agregue el nuevo valor clave a ese nodo hoja en orden ascendente de valor clave dentro del nodo.

Paso 5: si ese nodo de hoja ya está lleno, divida ese nodo de hoja enviando el valor medio a su nodo padre. Repita lo mismo hasta que el valor de envío se fije en un nodo.

Paso 6: si la división se realiza en el nodo raíz, el valor medio se convierte en un nuevo nodo raíz para el árbol y la altura del árbol aumenta en uno.

B-Tree: *Insert*

67, 89, 97, 73, 7, 35, 49, 71, 26

1	21	23	67

Paso 1: compruebe si el árbol está vacío.

Paso 2: si el árbol está vacío, cree un nuevo nodo con un nuevo valor de clave e insértelo en el árbol como un nodo raíz.

Paso 3: si el árbol no está vacío, busque el nodo de hoja adecuado al que se agrega el nuevo valor clave mediante la lógica del árbol de búsqueda binaria.

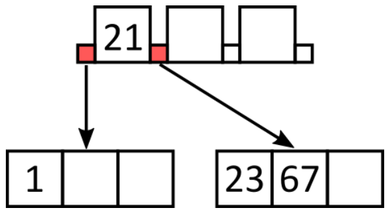
Paso 4: si ese nodo hoja tiene una posición vacía, agregue el nuevo valor clave a ese nodo hoja en orden ascendente de valor clave dentro del nodo.

Paso 5: si ese nodo de hoja ya está lleno, divida ese nodo de hoja enviando el valor medio a su nodo padre. Repita lo mismo hasta que el valor de envío se fije en un nodo.

Paso 6: si la división se realiza en el nodo raíz, el valor medio se convierte en un nuevo nodo raíz para el árbol y la altura del árbol aumenta en uno.

B-Tree: *Insert*

89, 97, 73, 7, 35, 49, 71, 26



Paso 1: compruebe si el árbol está vacío.

Paso 2: si el árbol está vacío, cree un nuevo nodo con un nuevo valor de clave e insértelo en el árbol como un nodo raíz.

Paso 3: si el árbol no está vacío, busque el nodo de hoja adecuado al que se agrega el nuevo valor clave mediante la lógica del árbol de búsqueda binaria.

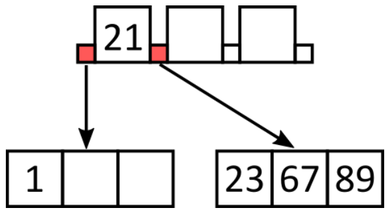
Paso 4: si ese nodo hoja tiene una posición vacía, agregue el nuevo valor clave a ese nodo hoja en orden ascendente de valor clave dentro del nodo.

Paso 5: si ese nodo de hoja ya está lleno, divida ese nodo de hoja enviando el valor medio a su nodo padre. Repita lo mismo hasta que el valor de envío se fije en un nodo.

Paso 6: si la división se realiza en el nodo raíz, el valor medio se convierte en un nuevo nodo raíz para el árbol y la altura del árbol aumenta en uno.

B-Tree: *Insert*

97, 73, 7, 35, 49, 71, 26



Paso 1: compruebe si el árbol está vacío.

Paso 2: si el árbol está vacío, cree un nuevo nodo con un nuevo valor de clave e insértelo en el árbol como un nodo raíz.

Paso 3: si el árbol no está vacío, busque el nodo de hoja adecuado al que se agrega el nuevo valor clave mediante la lógica del árbol de búsqueda binaria.

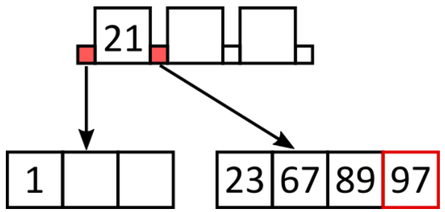
Paso 4: si ese nodo hoja tiene una posición vacía, agregue el nuevo valor clave a ese nodo hoja en orden ascendente de valor clave dentro del nodo.

Paso 5: si ese nodo de hoja ya está lleno, divida ese nodo de hoja enviando el valor medio a su nodo padre. Repita lo mismo hasta que el valor de envío se fije en un nodo.

Paso 6: si la división se realiza en el nodo raíz, el valor medio se convierte en un nuevo nodo raíz para el árbol y la altura del árbol aumenta en uno.

B-Tree: *Insert*

97, 73, 7, 35, 49, 71, 26



Paso 1: compruebe si el árbol está vacío.

Paso 2: si el árbol está vacío, cree un nuevo nodo con un nuevo valor de clave e insértelo en el árbol como un nodo raíz.

Paso 3: si el árbol no está vacío, busque el nodo de hoja adecuado al que se agrega el nuevo valor clave mediante la lógica del árbol de búsqueda binaria.

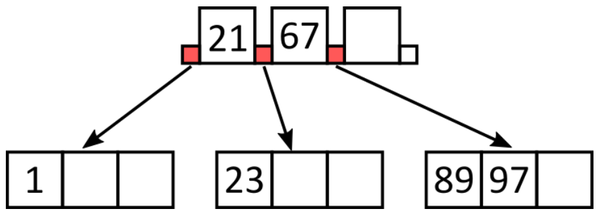
Paso 4: si ese nodo hoja tiene una posición vacía, agregue el nuevo valor clave a ese nodo hoja en orden ascendente de valor clave dentro del nodo.

Paso 5: si ese nodo de hoja ya está lleno, divida ese nodo de hoja enviando el valor medio a su nodo padre. Repita lo mismo hasta que el valor de envío se fije en un nodo.

Paso 6: si la división se realiza en el nodo raíz, el valor medio se convierte en un nuevo nodo raíz para el árbol y la altura del árbol aumenta en uno.

B-Tree: *Insert*

73, 7, 35, 49, 71, 26



Paso 1: compruebe si el árbol está vacío.

Paso 2: si el árbol está vacío, cree un nuevo nodo con un nuevo valor de clave e insértelo en el árbol como un nodo raíz.

Paso 3: si el árbol no está vacío, busque el nodo de hoja adecuado al que se agrega el nuevo valor clave mediante la lógica del árbol de búsqueda binaria.

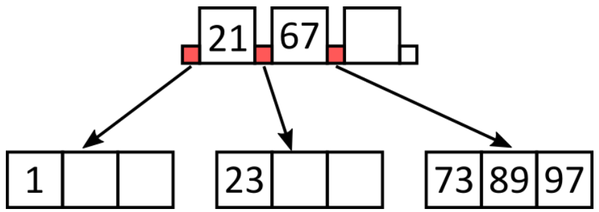
Paso 4: si ese nodo hoja tiene una posición vacía, agregue el nuevo valor clave a ese nodo hoja en orden ascendente de valor clave dentro del nodo.

Paso 5: si ese nodo de hoja ya está lleno, divida ese nodo de hoja enviando el valor medio a su nodo padre. Repita lo mismo hasta que el valor de envío se fije en un nodo.

Paso 6: si la división se realiza en el nodo raíz, el valor medio se convierte en un nuevo nodo raíz para el árbol y la altura del árbol aumenta en uno.

B-Tree: *Insert*

7, 35, 49, 71, 26



Paso 1: compruebe si el árbol está vacío.

Paso 2: si el árbol está vacío, cree un nuevo nodo con un nuevo valor de clave e insértelo en el árbol como un nodo raíz.

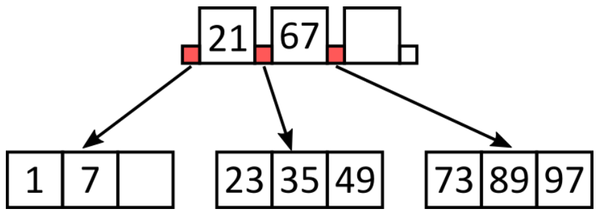
Paso 3: si el árbol no está vacío, busque el nodo de hoja adecuado al que se agrega el nuevo valor clave mediante la lógica del árbol de búsqueda binaria.

Paso 4: si ese nodo hoja tiene una posición vacía, agregue el nuevo valor clave a ese nodo hoja en orden ascendente de valor clave dentro del nodo.

Paso 5: si ese nodo de hoja ya está lleno, divida ese nodo de hoja enviando el valor medio a su nodo padre. Repita lo mismo hasta que el valor de envío se fije en un nodo.

Paso 6: si la división se realiza en el nodo raíz, el valor medio se convierte en un nuevo nodo raíz para el árbol y la altura del árbol aumenta en uno.

B-Tree: *Insert*



71, 26

Paso 1: compruebe si el árbol está vacío.

Paso 2: si el árbol está vacío, cree un nuevo nodo con un nuevo valor de clave e insértelo en el árbol como un nodo raíz.

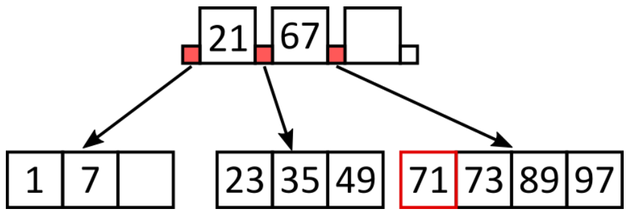
Paso 3: si el árbol no está vacío, busque el nodo de hoja adecuado al que se agrega el nuevo valor clave mediante la lógica del árbol de búsqueda binaria.

Paso 4: si ese nodo hoja tiene una posición vacía, agregue el nuevo valor clave a ese nodo hoja en orden ascendente de valor clave dentro del nodo.

Paso 5: si ese nodo de hoja ya está lleno, divida ese nodo de hoja enviando el valor medio a su nodo padre. Repita lo mismo hasta que el valor de envío se fije en un nodo.

Paso 6: si la división se realiza en el nodo raíz, el valor medio se convierte en un nuevo nodo raíz para el árbol y la altura del árbol aumenta en uno.

B-Tree: *Insert*



71, 26

Paso 1: compruebe si el árbol está vacío.

Paso 2: si el árbol está vacío, cree un nuevo nodo con un nuevo valor de clave e insértelo en el árbol como un nodo raíz.

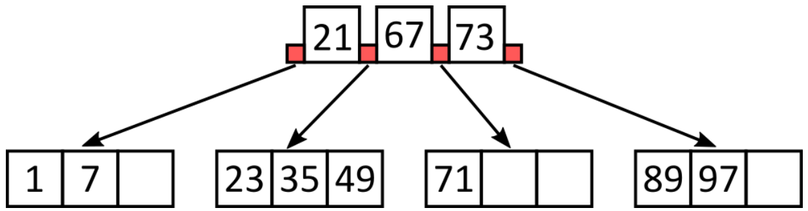
Paso 3: si el árbol no está vacío, busque el nodo de hoja adecuado al que se agrega el nuevo valor clave mediante la lógica del árbol de búsqueda binaria.

Paso 4: si ese nodo hoja tiene una posición vacía, agregue el nuevo valor clave a ese nodo hoja en orden ascendente de valor clave dentro del nodo.

Paso 5: si ese nodo de hoja ya está lleno, divida ese nodo de hoja enviando el valor medio a su nodo padre. Repita lo mismo hasta que el valor de envío se fije en un nodo.

Paso 6: si la división se realiza en el nodo raíz, el valor medio se convierte en un nuevo nodo raíz para el árbol y la altura del árbol aumenta en uno.

B-Tree: *Insert*



26

Paso 1: compruebe si el árbol está vacío.

Paso 2: si el árbol está vacío, cree un nuevo nodo con un nuevo valor de clave e insértelo en el árbol como un nodo raíz.

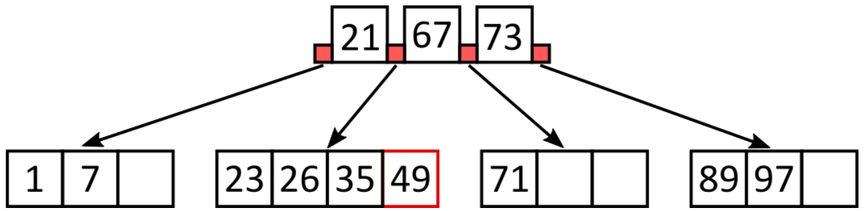
Paso 3: si el árbol no está vacío, busque el nodo de hoja adecuado al que se agrega el nuevo valor clave mediante la lógica del árbol de búsqueda binaria.

Paso 4: si ese nodo hoja tiene una posición vacía, agregue el nuevo valor clave a ese nodo hoja en orden ascendente de valor clave dentro del nodo.

Paso 5: si ese nodo de hoja ya está lleno, divida ese nodo de hoja enviando el valor medio a su nodo padre. Repita lo mismo hasta que el valor de envío se fije en un nodo.

Paso 6: si la división se realiza en el nodo raíz, el valor medio se convierte en un nuevo nodo raíz para el árbol y la altura del árbol aumenta en uno.

B-Tree: *Insert*



26

Paso 1: compruebe si el árbol está vacío.

Paso 2: si el árbol está vacío, cree un nuevo nodo con un nuevo valor de clave e insértelo en el árbol como un nodo raíz.

Paso 3: si el árbol no está vacío, busque el nodo de hoja adecuado al que se agrega el nuevo valor clave mediante la lógica del árbol de búsqueda binaria.

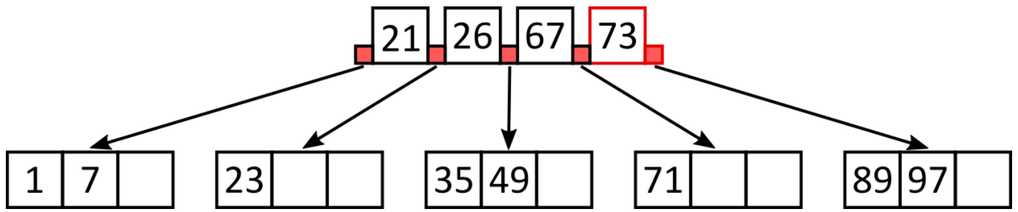
Paso 4: si ese nodo hoja tiene una posición vacía, agregue el nuevo valor clave a ese nodo hoja en orden ascendente de valor clave dentro del nodo.

Paso 5: si ese nodo de hoja ya está lleno, divida ese nodo de hoja enviando el valor medio a su nodo padre. Repita lo mismo hasta que el valor de envío se fije en un nodo.

Paso 6: si la división se realiza en el nodo raíz, el valor medio se convierte en un nuevo nodo raíz para el árbol y la altura del árbol aumenta en uno.

B-Tree: *Insert*

26



Paso 1: compruebe si el árbol está vacío.

Paso 2: si el árbol está vacío, cree un nuevo nodo con un nuevo valor de clave e insértelo en el árbol como un nodo raíz.

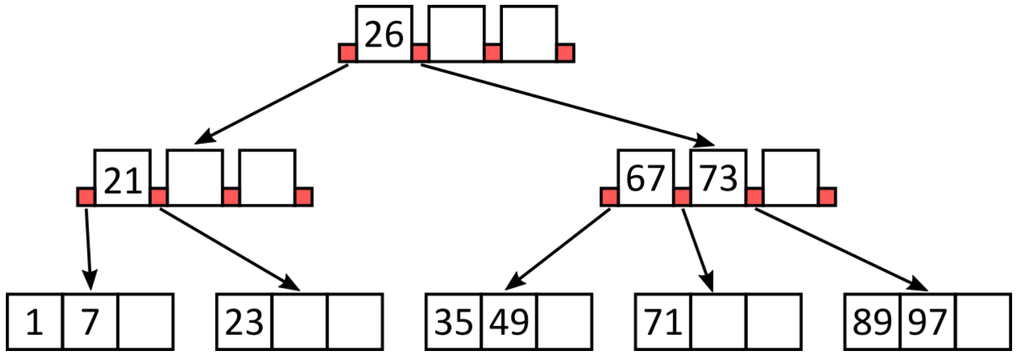
Paso 3: si el árbol no está vacío, busque el nodo de hoja adecuado al que se agrega el nuevo valor clave mediante la lógica del árbol de búsqueda binaria.

Paso 4: si ese nodo hoja tiene una posición vacía, agregue el nuevo valor clave a ese nodo hoja en orden ascendente de valor clave dentro del nodo.

Paso 5: si ese nodo de hoja ya está lleno, divida ese nodo de hoja enviando el valor medio a su nodo padre. Repita lo mismo hasta que el valor de envío se fije en un nodo.

Paso 6: si la división se realiza en el nodo raíz, el valor medio se convierte en un nuevo nodo raíz para el árbol y la altura del árbol aumenta en uno.

B-Tree: *Insert*



Paso 1: compruebe si el árbol está vacío.

Paso 2: si el árbol está vacío, cree un nuevo nodo con un nuevo valor de clave e insértelo en el árbol como un nodo raíz.

Paso 3: si el árbol no está vacío, busque el nodo de hoja adecuado al que se agrega el nuevo valor clave mediante la lógica del árbol de búsqueda binaria.

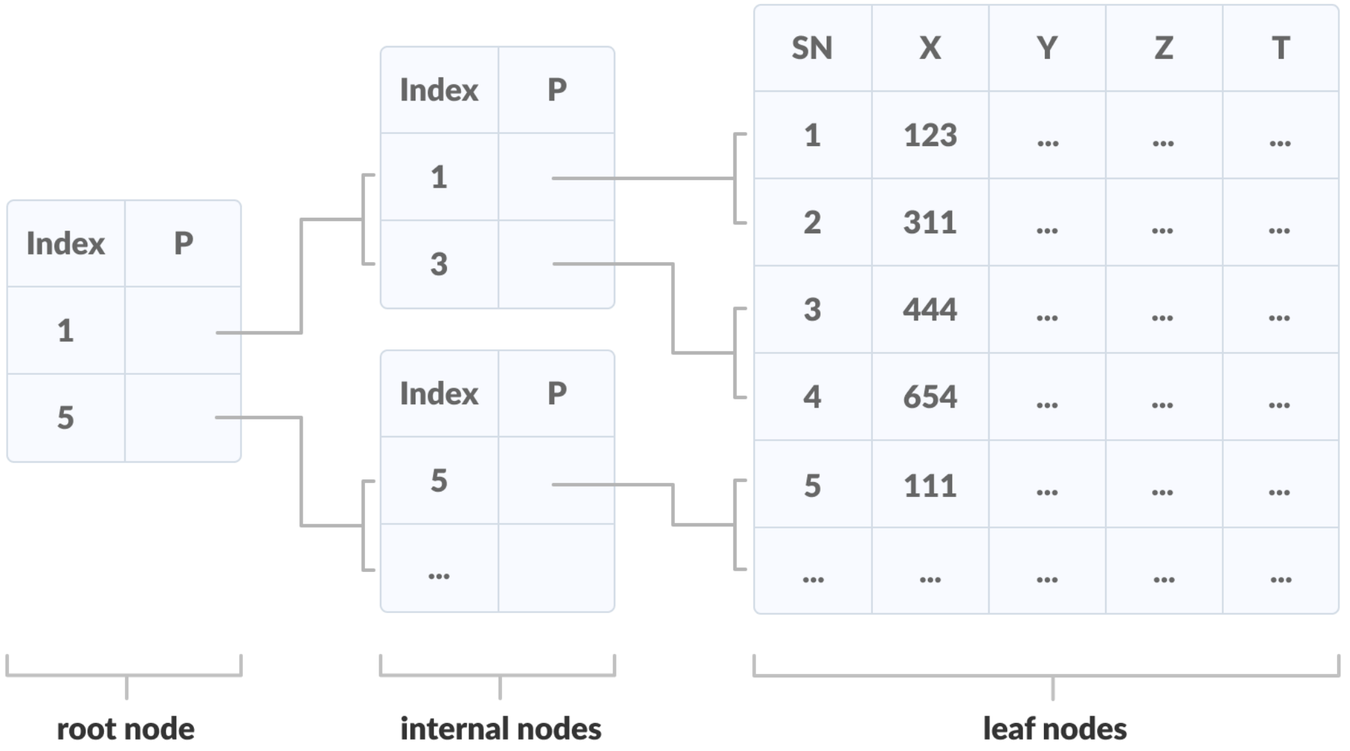
Paso 4: si ese nodo hoja tiene una posición vacía, agregue el nuevo valor clave a ese nodo hoja en orden ascendente de valor clave dentro del nodo.

Paso 5: si ese nodo de hoja ya está lleno, divida ese nodo de hoja enviando el valor medio a su nodo padre. Repita lo mismo hasta que el valor de envío se fije en un nodo.

Paso 6: si la división se realiza en el nodo raíz, el valor medio se convierte en un nuevo nodo raíz para el árbol y la altura del árbol aumenta en uno.

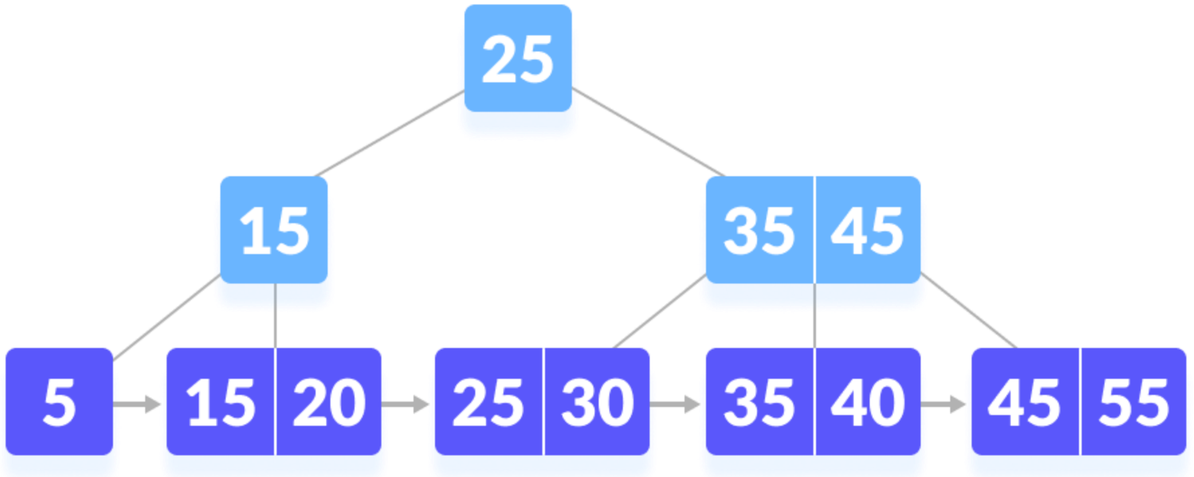
2. B+ *Tree*

B+Tree


Index
P
1
5
Index
P
1
3
Index
P
5
...
SN
X
Y
Z
T
1
123
...
...
...
2
311
...
...
...
3
444
...
...
...
4
654
...
...
...
5
111
...
...
...
...
...
...
...
...
root node
internal nodes
leaf nodes

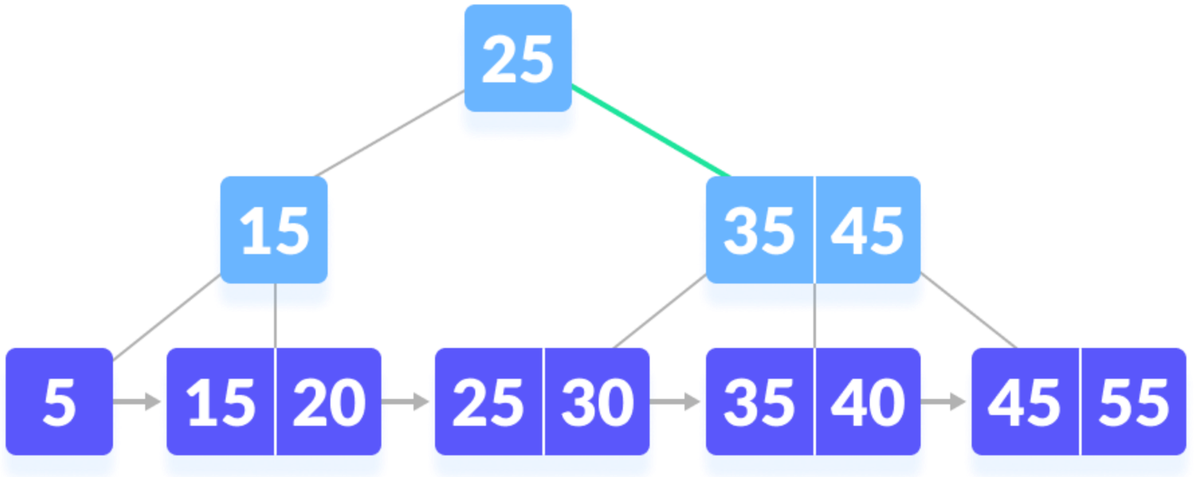
B+ Tree: *Search*

$k = 45$



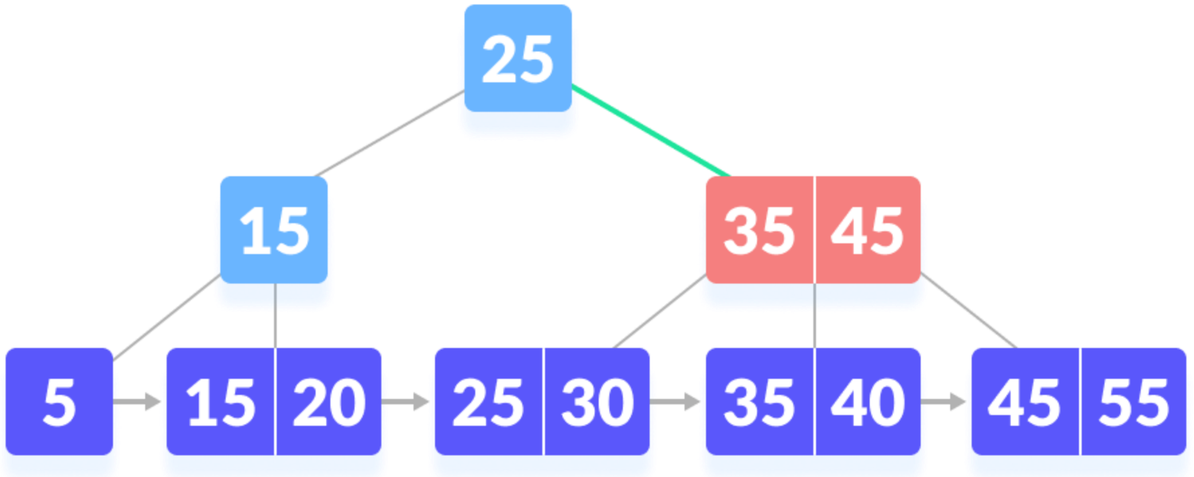
B+ Tree: *Search*

$k = 45$



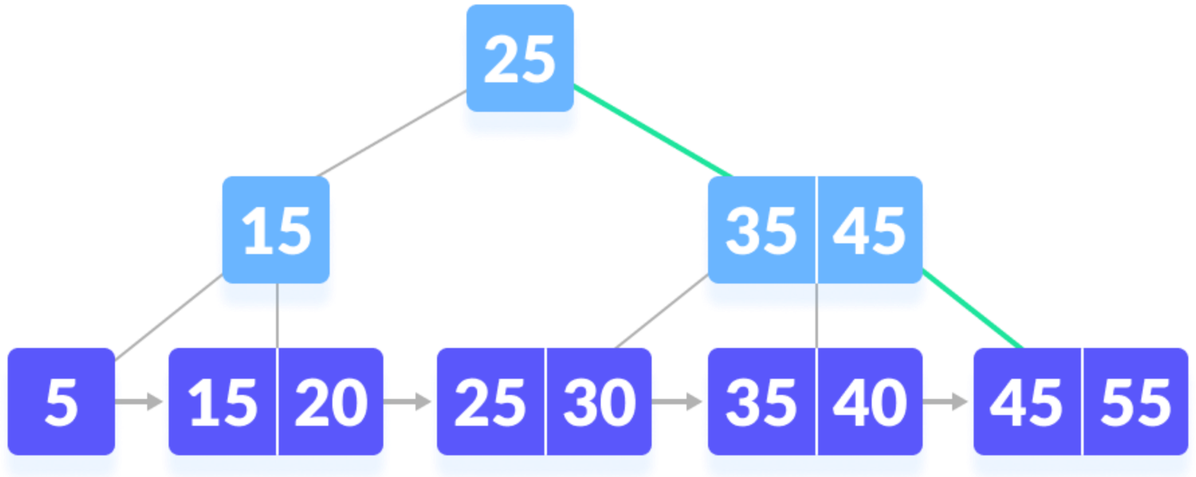
B+ Tree: *Search*

$k = 45$



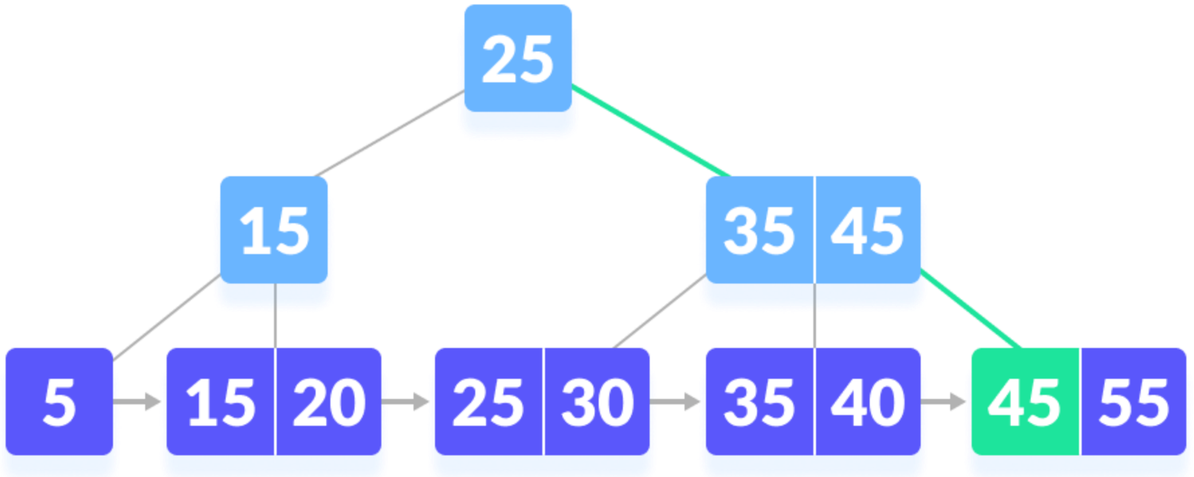
B+ Tree: *Search*

$k = 45$



B+ Tree: *Search*

$k = 45$



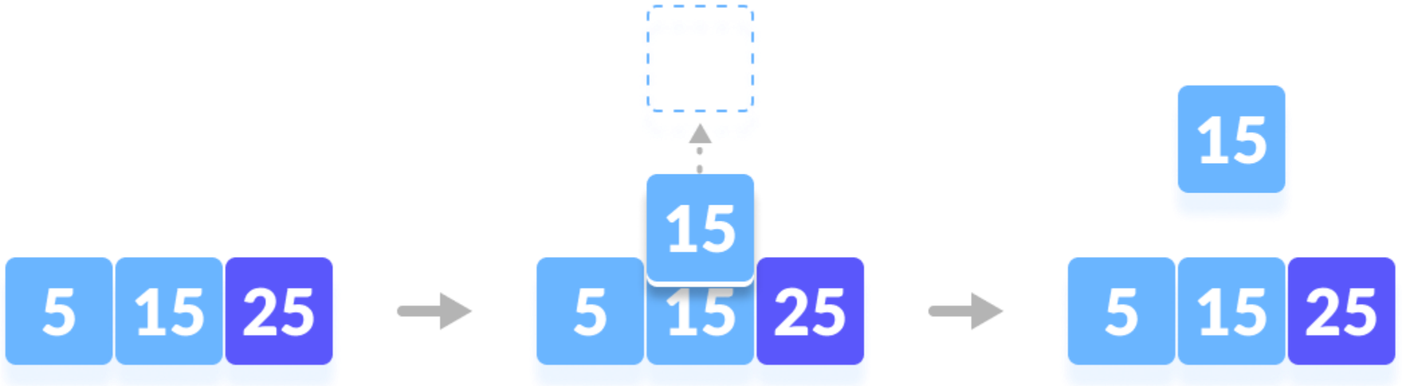
B+ Tree: *Insert* **5**

B+ Tree: *Insert* 15

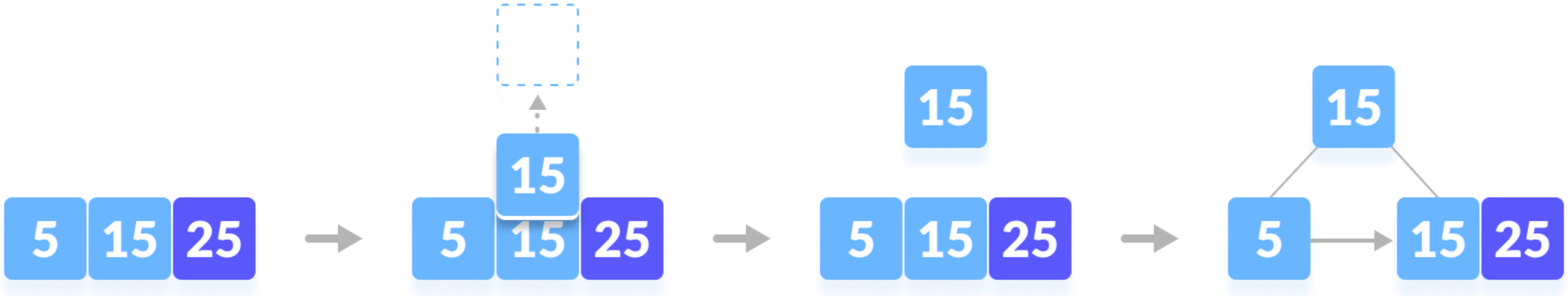
B+ Tree: *Insert* 25

B+ Tree: *Insert* 25

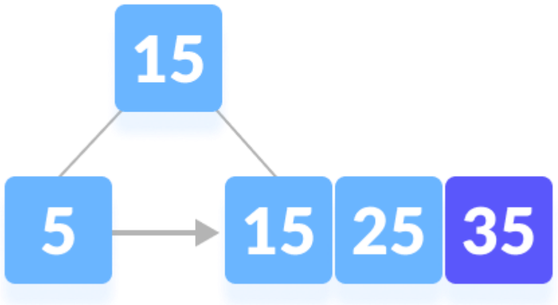
B+ Tree: *Insert* 25



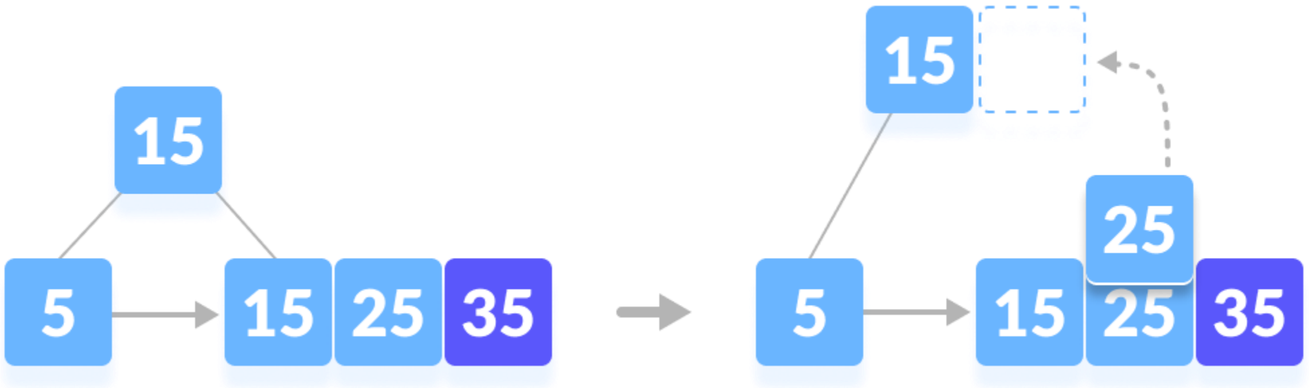
B+ Tree: *Insert* 25



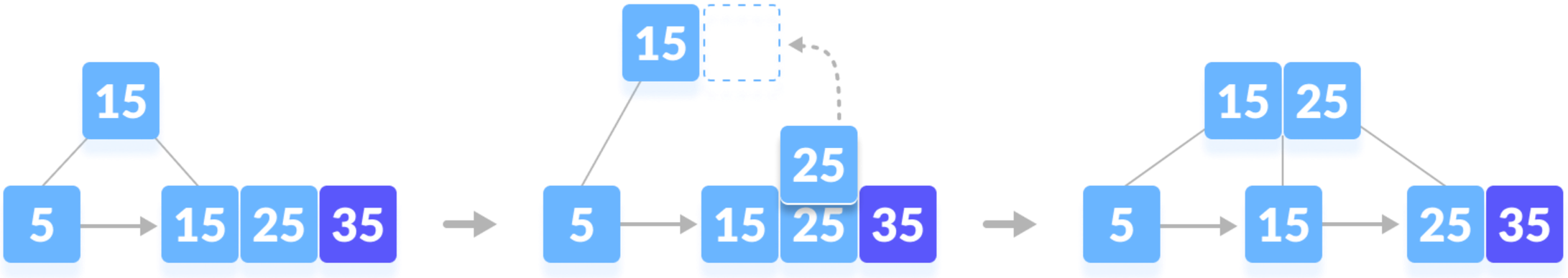
B+ Tree: *Insert* 35



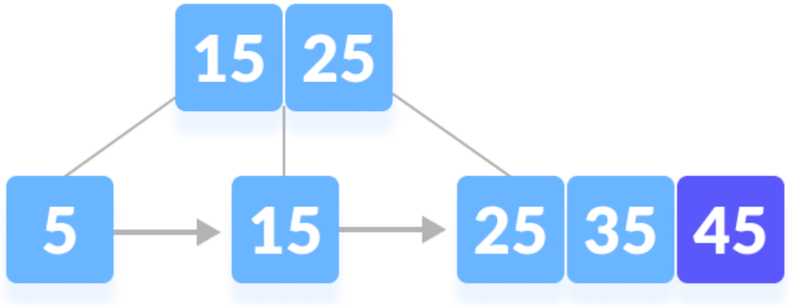
B+ Tree: *Insert* 35



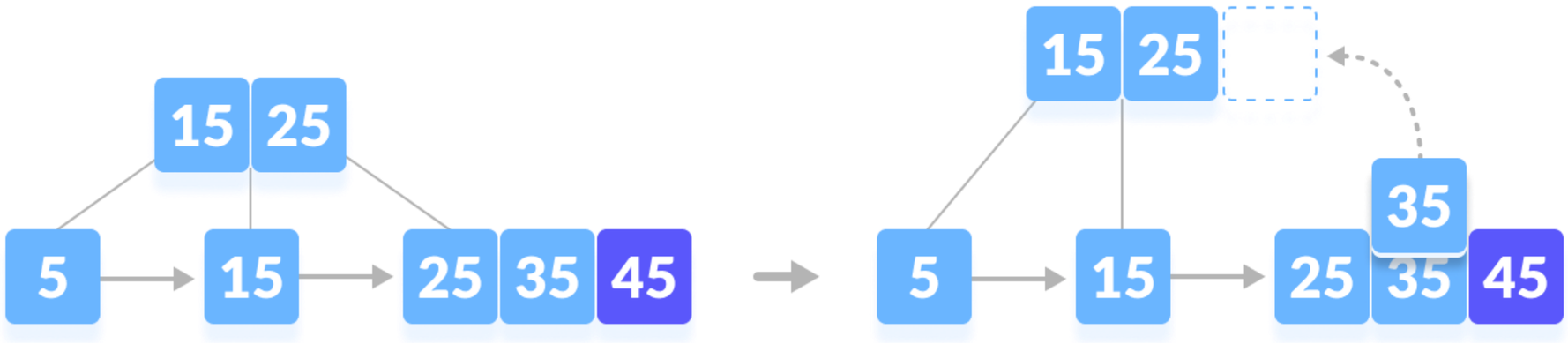
B+ Tree: *Insert* 35



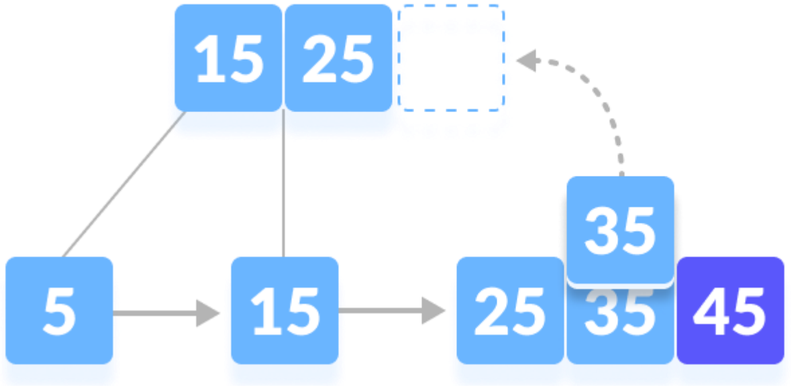
B+ Tree: *Insert* **45**



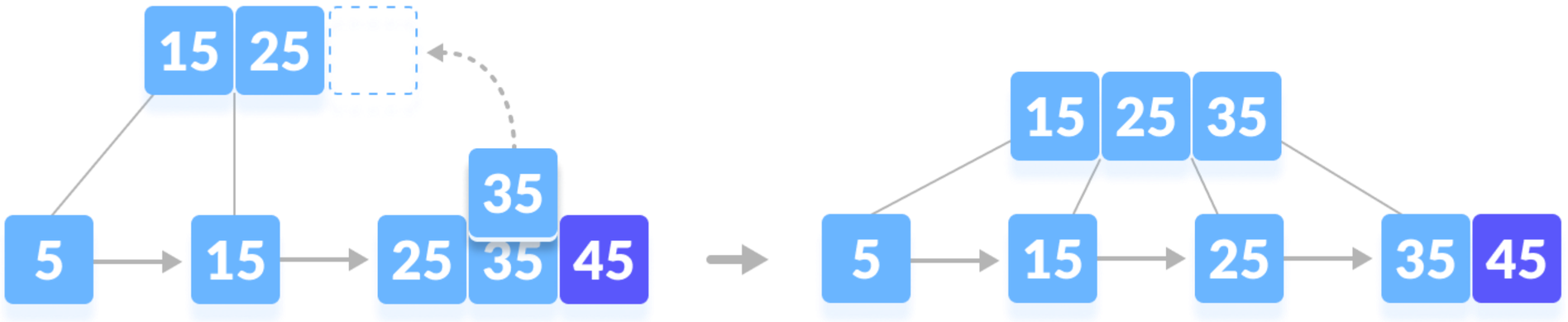
B+ Tree: *Insert* **45**



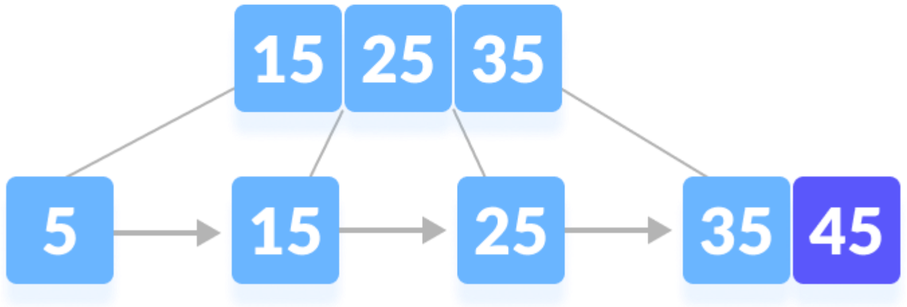
B+ Tree: *Insert* **45**



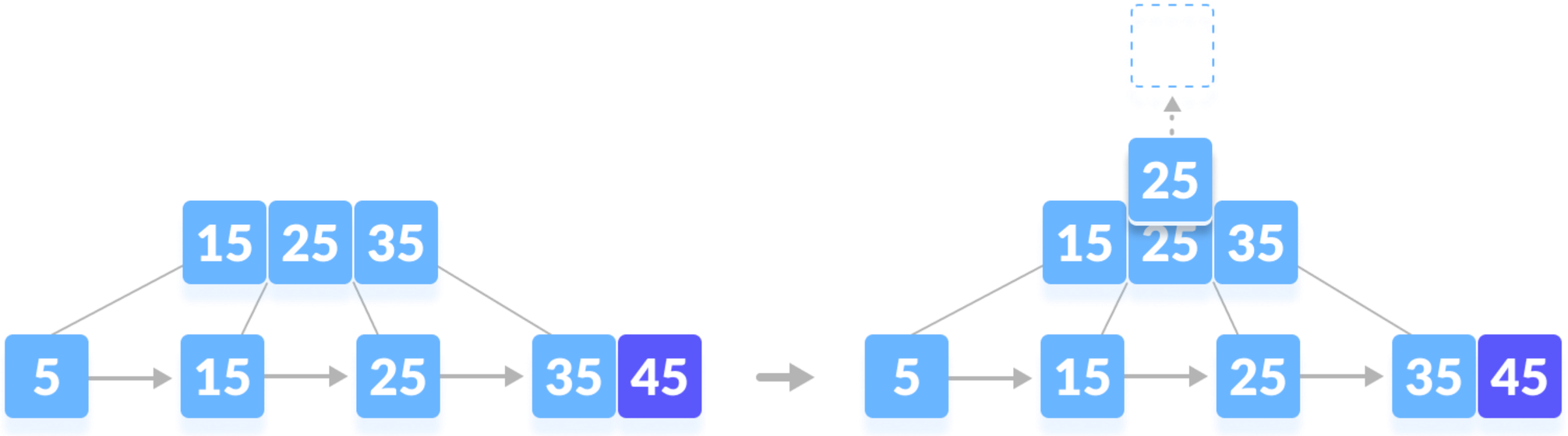
B+ Tree: *Insert* **45**



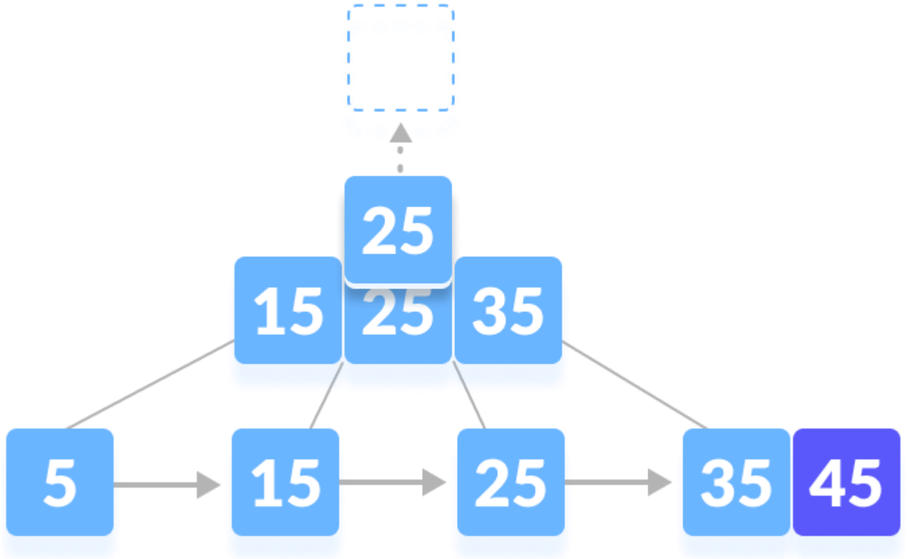
B+ Tree: *Insert* **45**



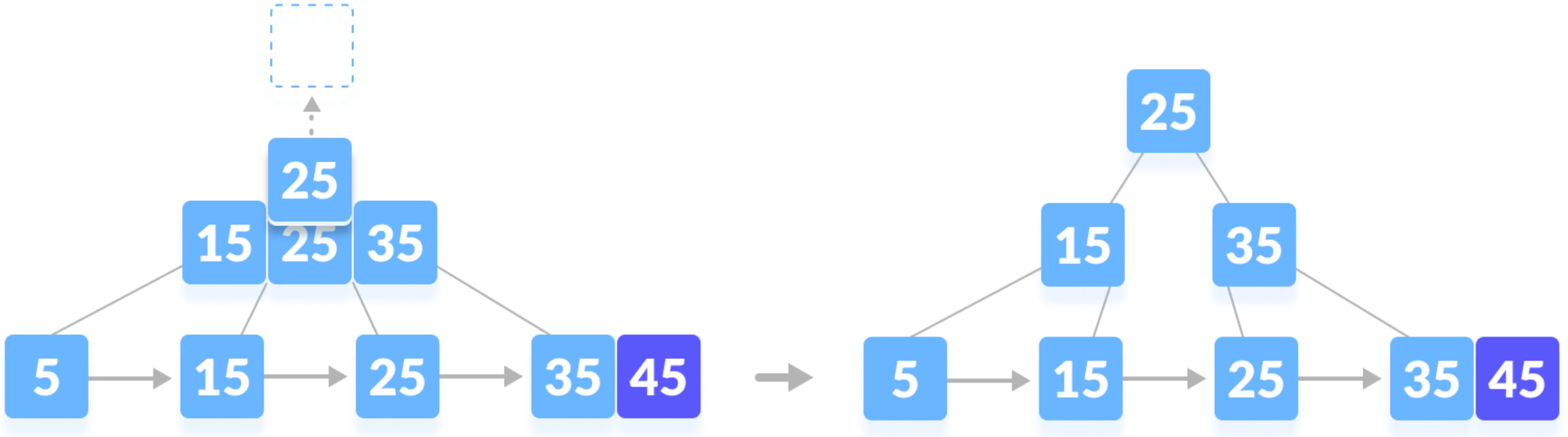
B+ Tree: *Insert* **45**



B+ Tree: *Insert* **45**

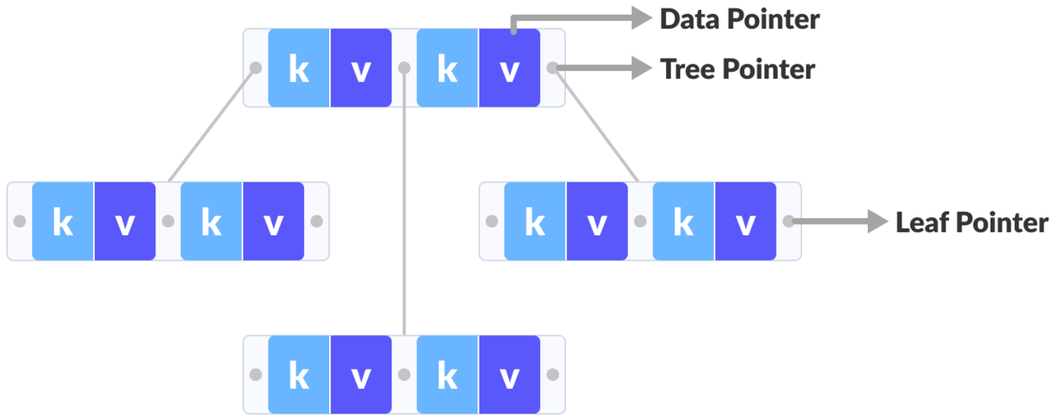


B+ Tree: *Insert* **45**

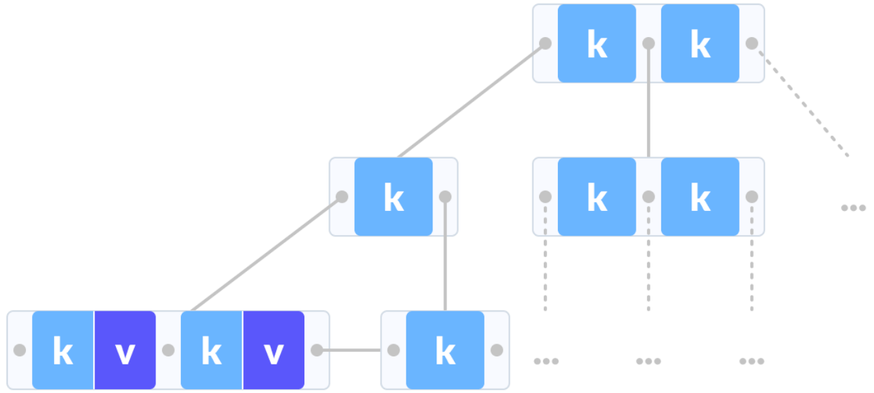


Compa*ración*

B Tree

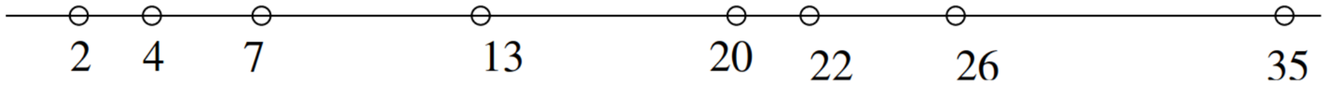


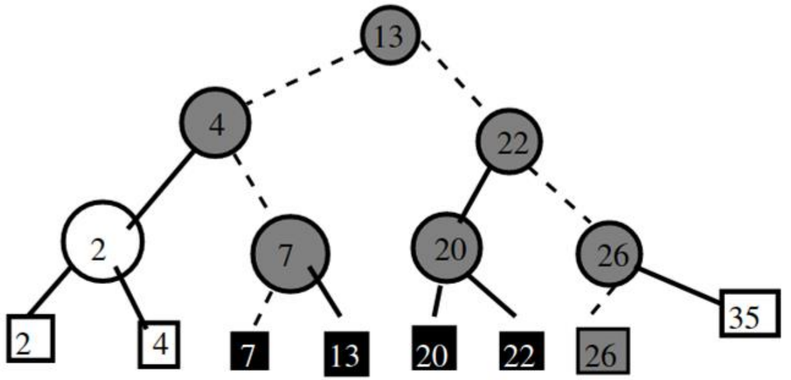
B+ Tree



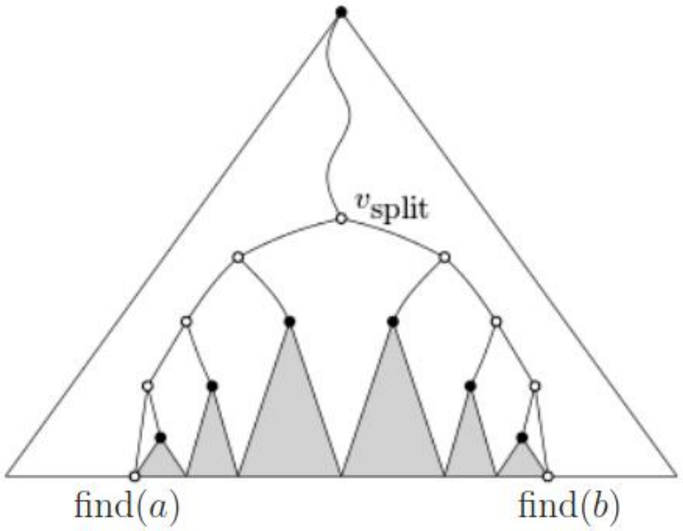
3. Range *Tree*

Range *Tree*





Range *Tree*



Buscar rango $[a, b]$

Paso 1: Buscar a.

Paso 2: Buscar b.

Paso 3: Encontrar el menor ancestro común de a y b.

Paso 4: Retornar todos los nodos y subárboles que estén dentro del rango:

- Retornar las hojas de los subárboles derechos de la ruta hacia a
- Retornar las hojas de los subárboles izquierdos de la ruta hacia b

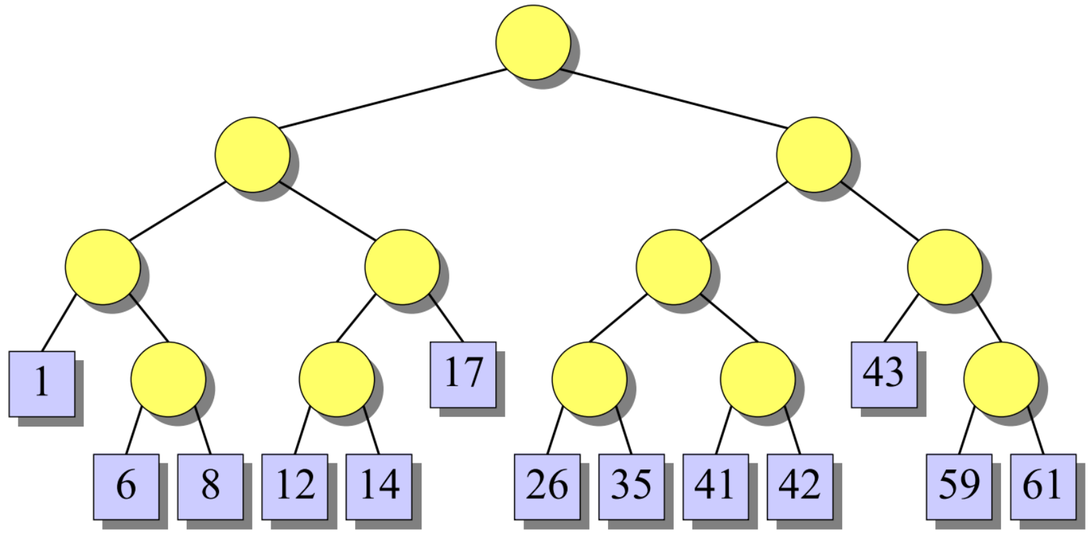
Range *Tree*

Estático

Entrada: 17, 8, 43, 12, 1, 61, 35, 41, 26, 6, 42, 14, 59

Índices ordenados: 1, 6, 8, 12, 14, 17, 26, 35, 41, 42, 43, 59, 61

Construir árbol:



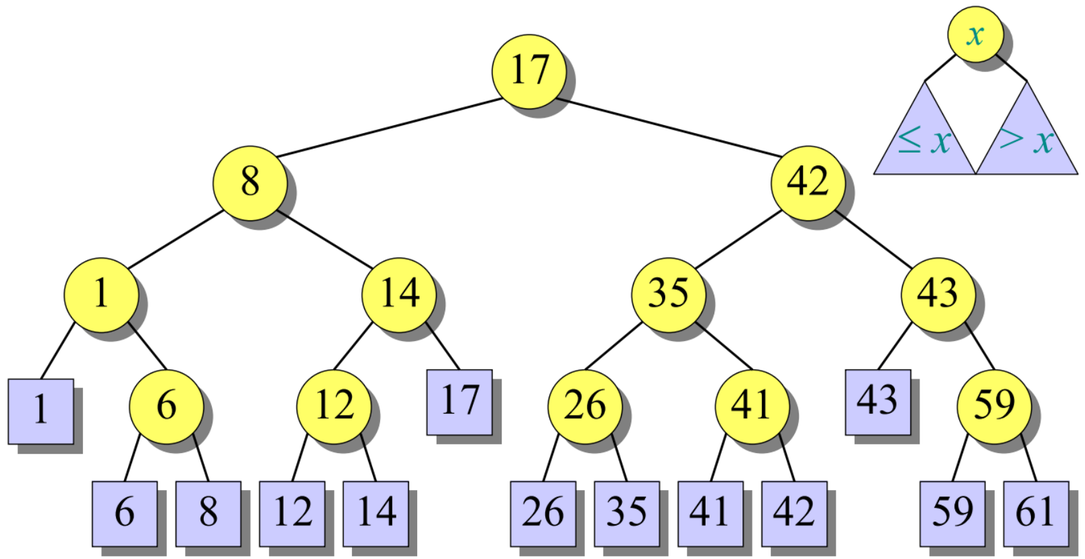
Range *Tree*

Estático

Entrada: 17, 8, 43, 12, 1, 61, 35, 41, 26, 6, 42, 14, 59

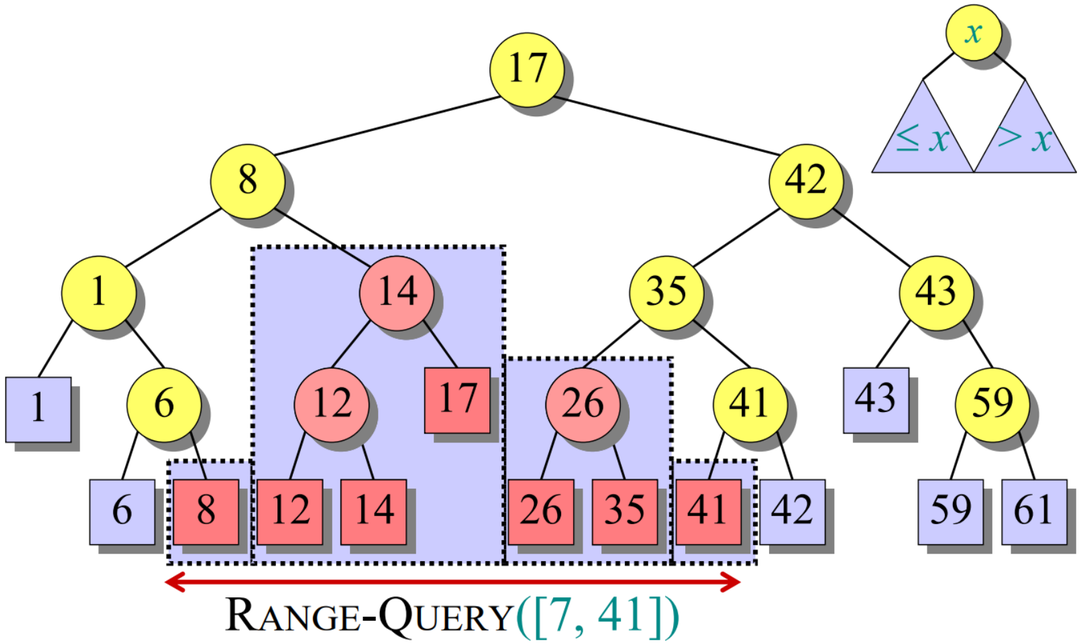
Índices ordenados: 1, 6, 8, 12, 14, 17, 26, 35, 41, 42, 43, 59, 61

Construir árbol:



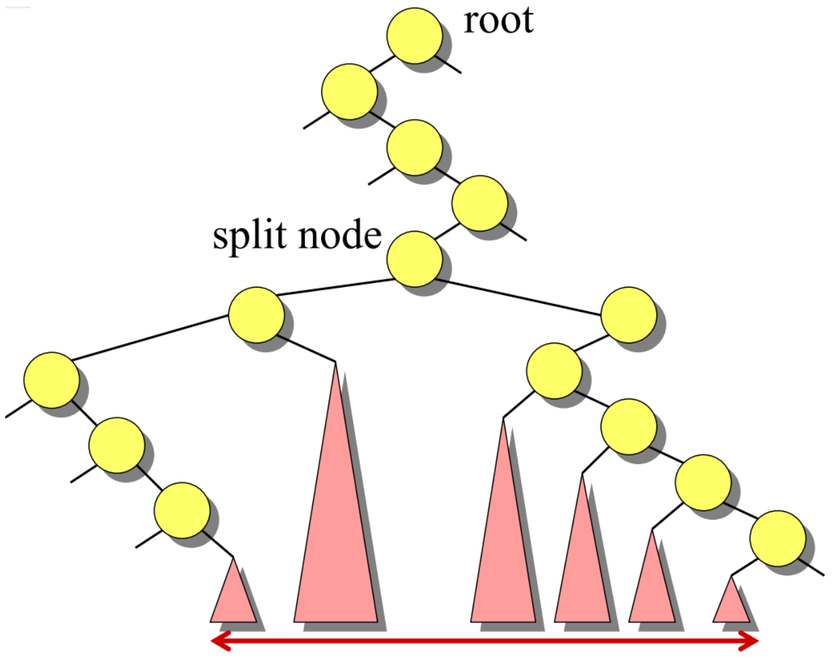
Range *Tree*

Estático



Range *Tree*

Estático

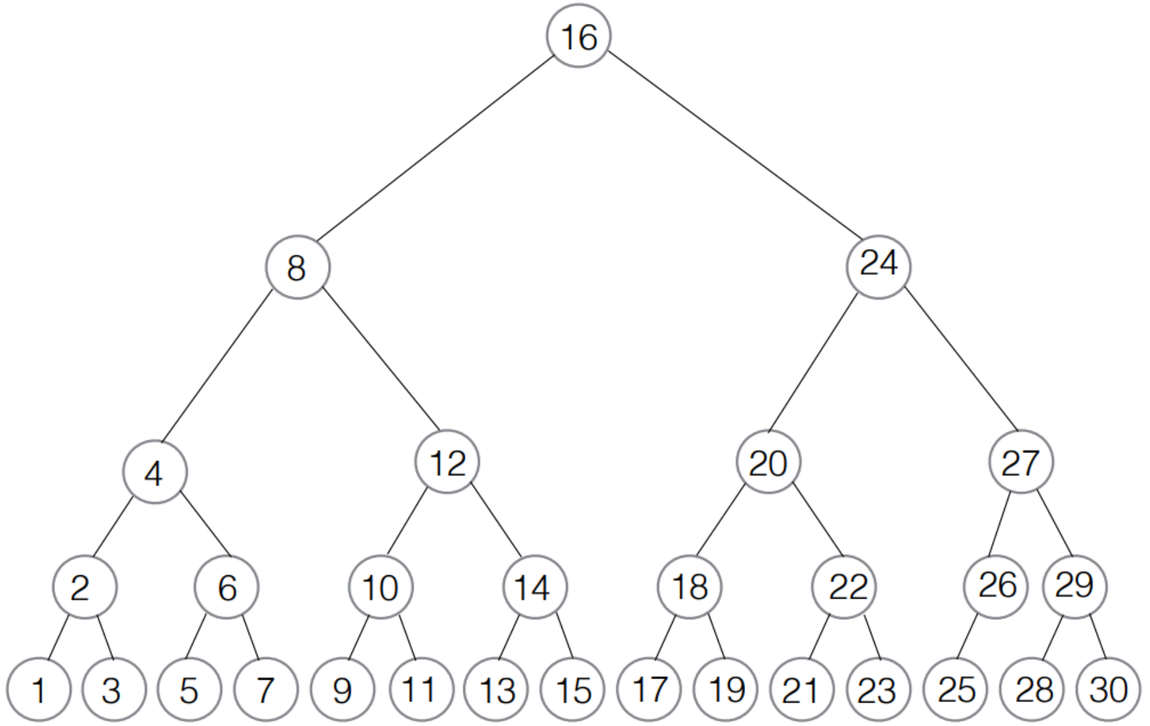


Range *Tree*

¿Cómo emplearía Range Tree cuando la cantidad de índices es diferente a una potencia de dos?

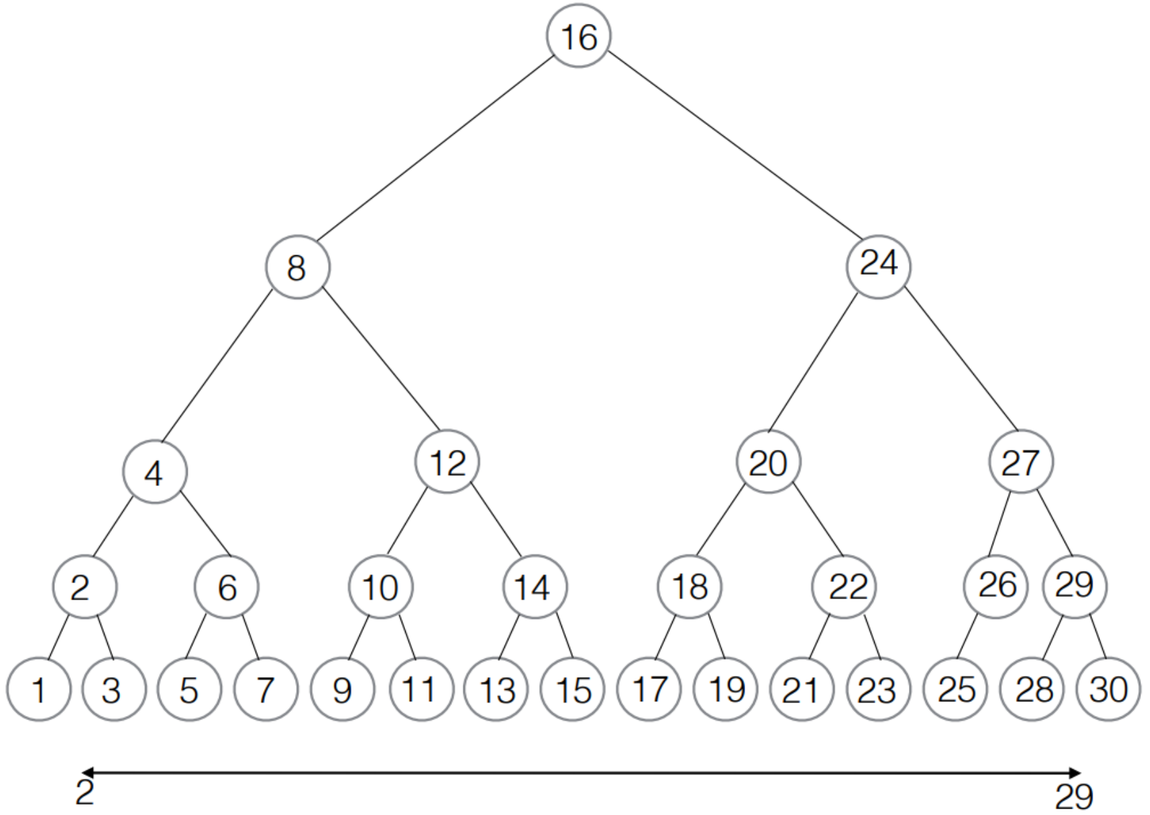
Range *Tree*

Dinámico Puedes utilizar AVL tree



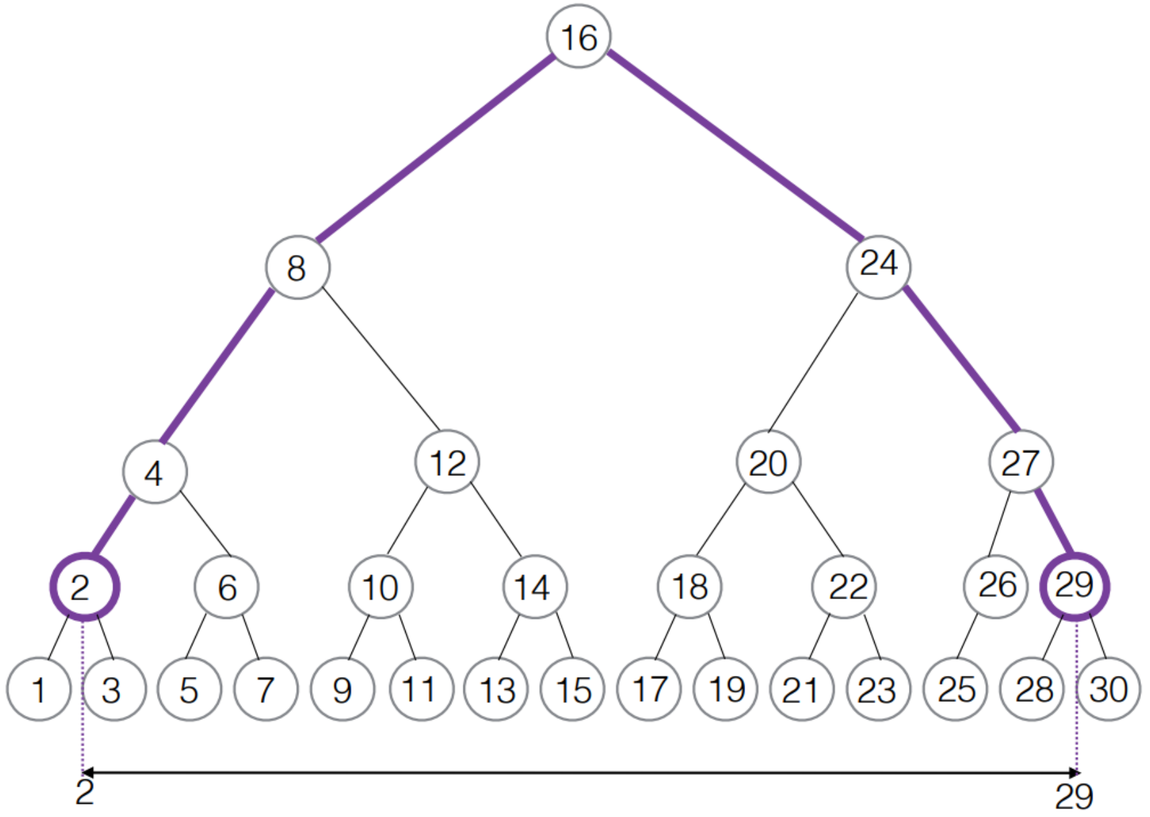
Range *Tree*

Dinámico Puedes utilizar AVL tree



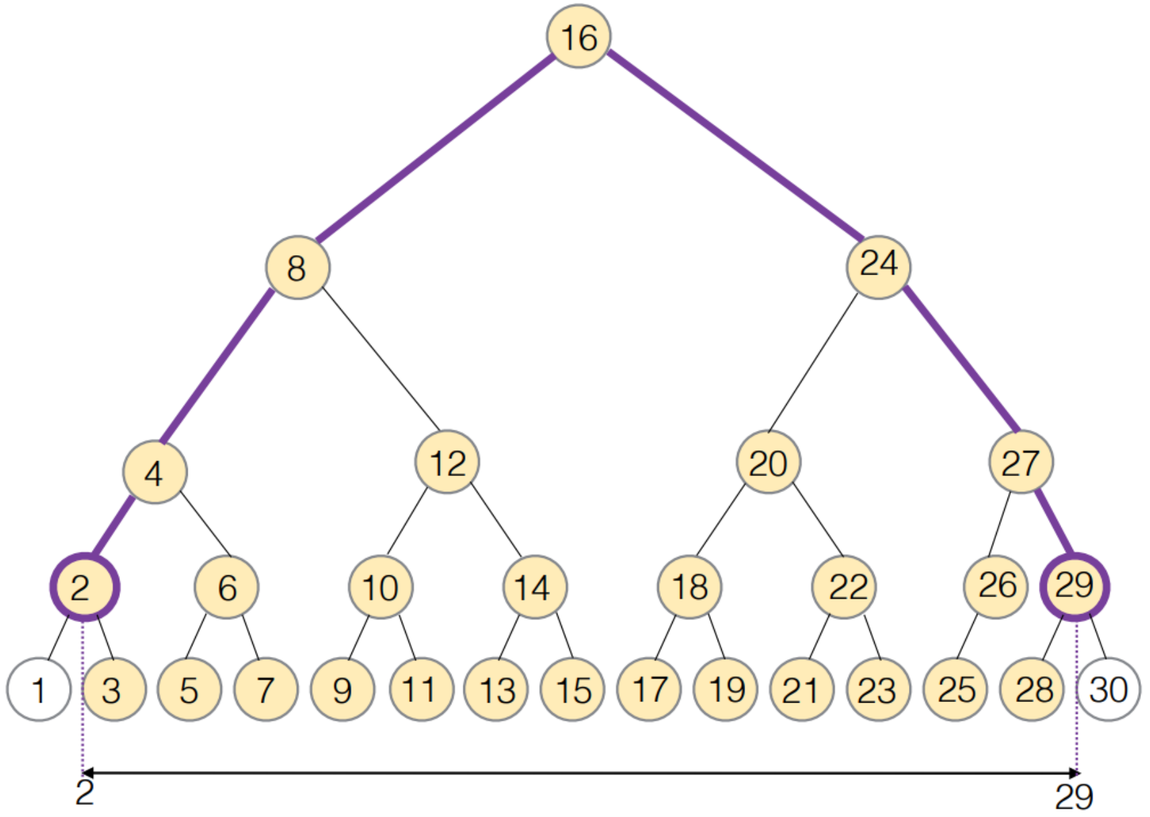
Range *Tree*

Dinámico Puedes u



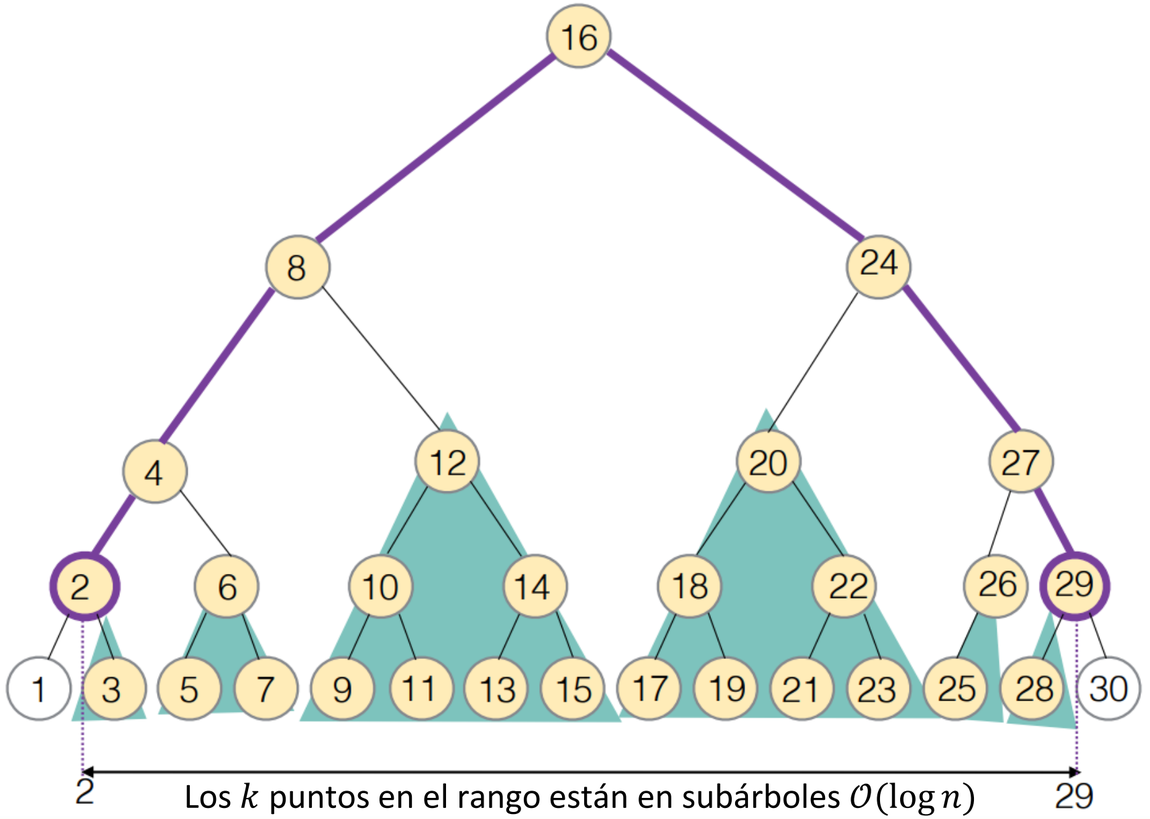
Range *Tree*

Dinámico Puedes utilizar AVL tree



Range *Tree*

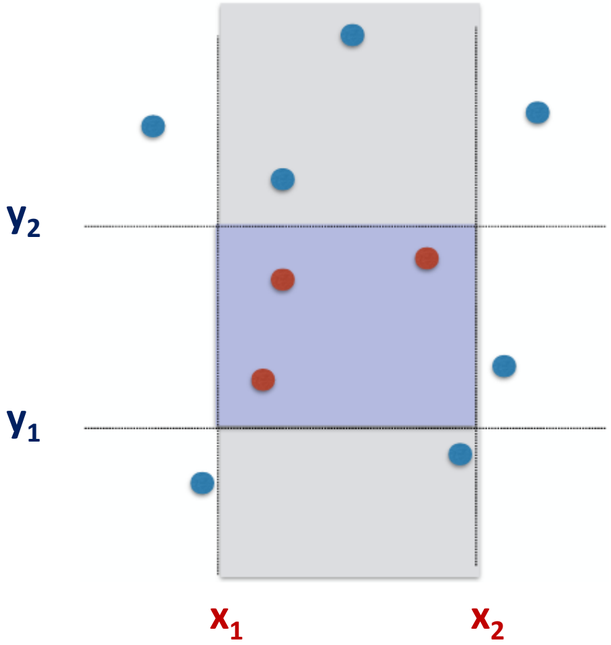
Dinámico Puedes utilizar AVL tree

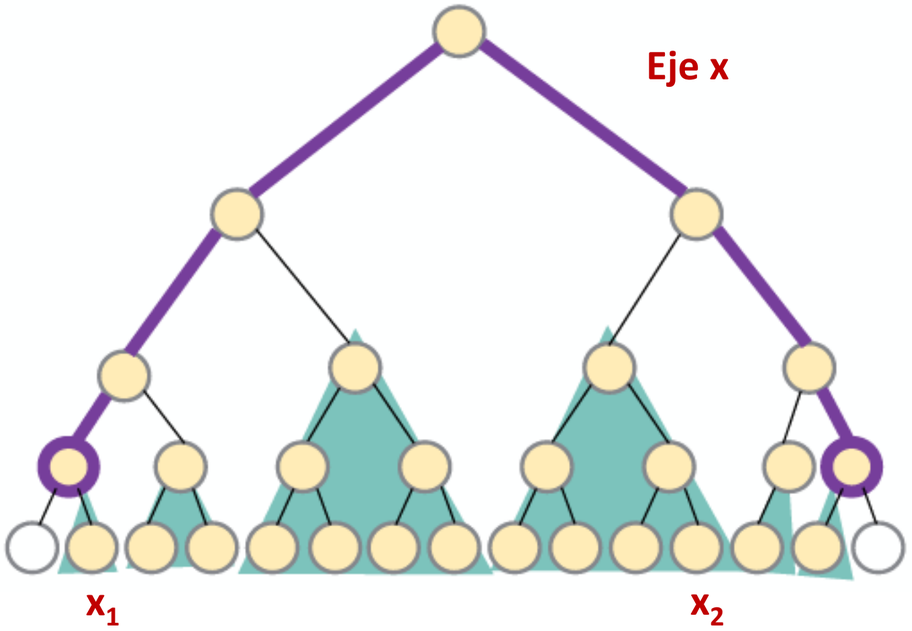


Range *Tree 2D*

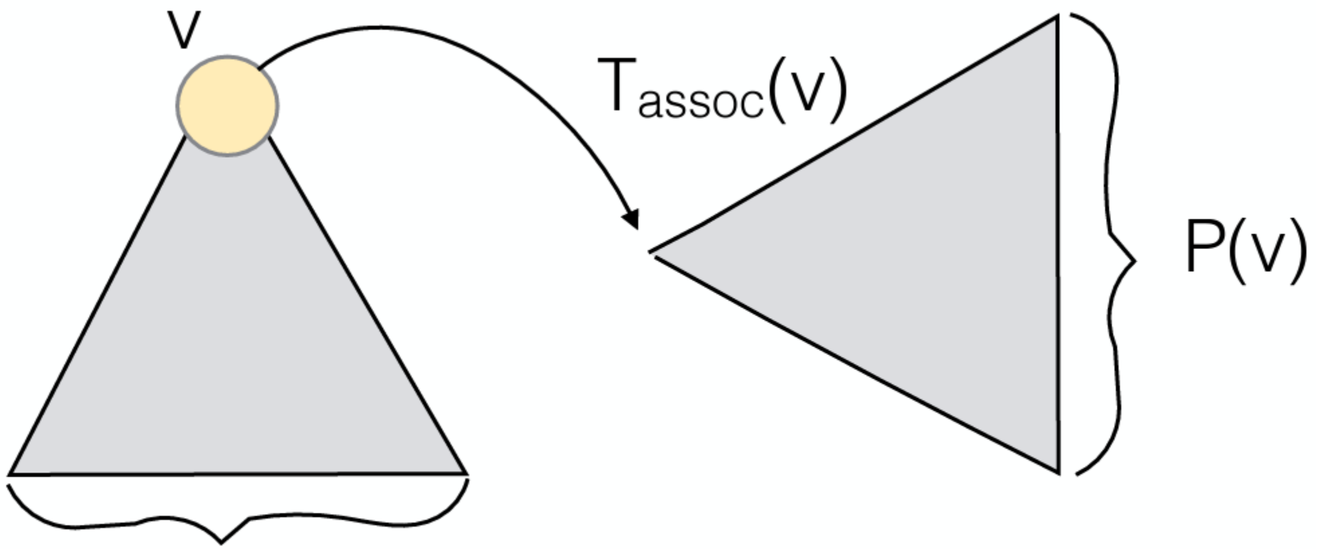
¿Cómo podríamos extender a dos dimensiones?

Range *Tree 2D*



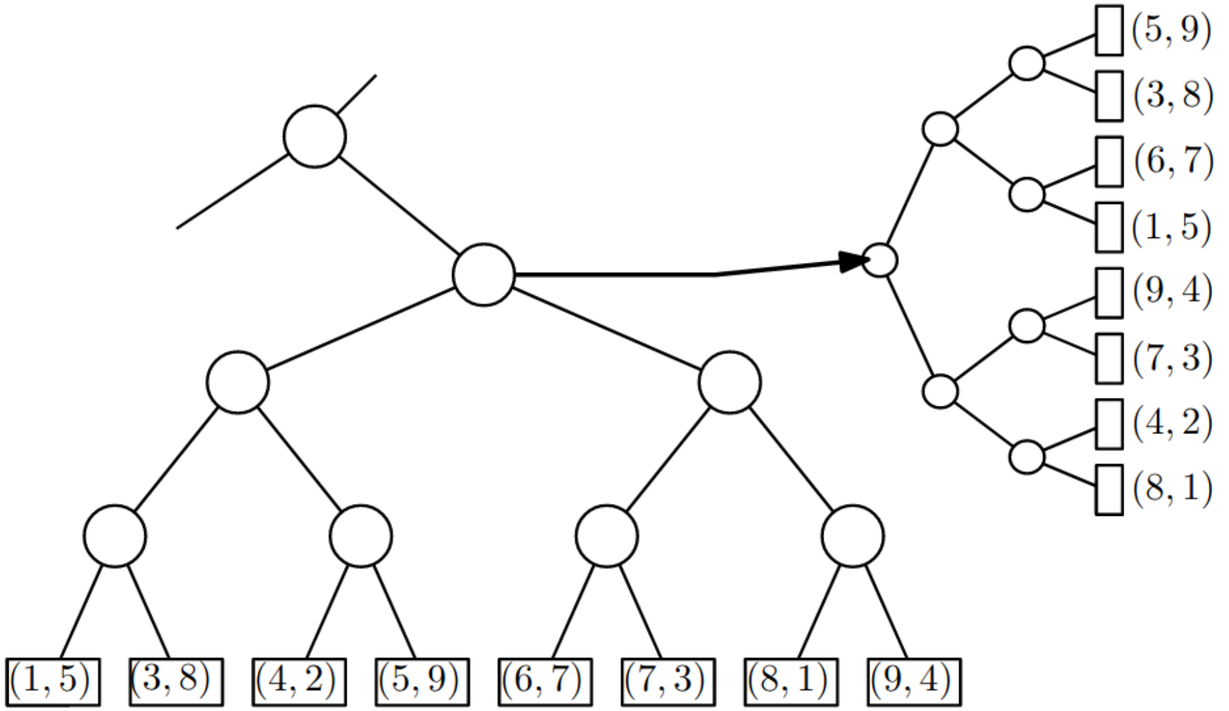


Range *Tree 2D*

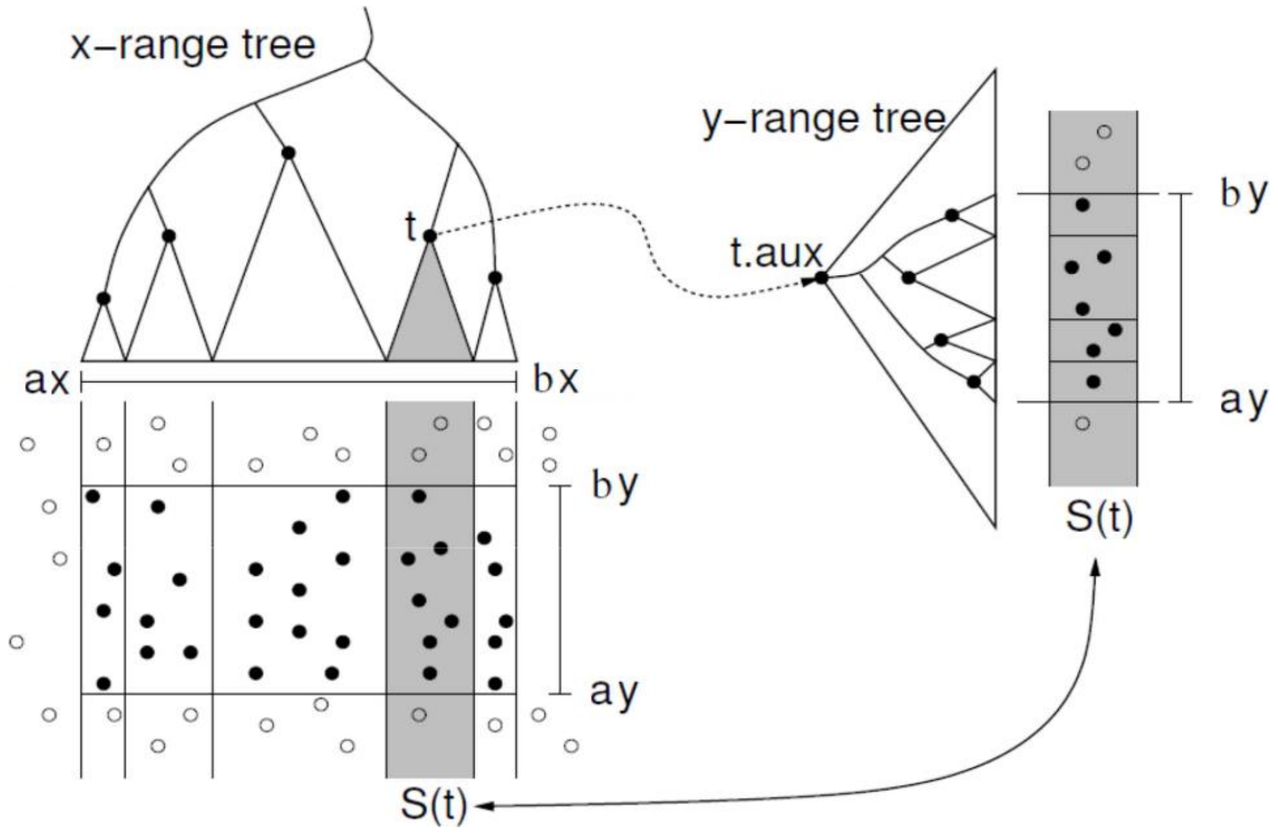


P(v): all points in subtree rooted at v

Range *Tree 2D*



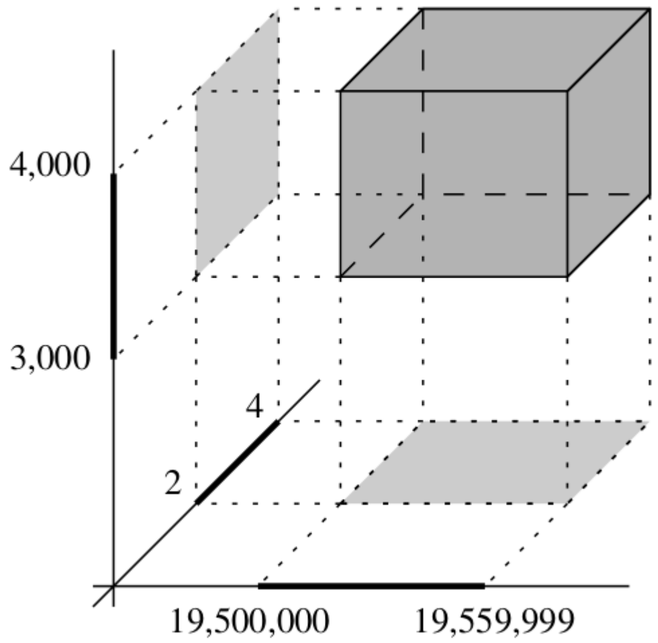
Range *Tree 2D*

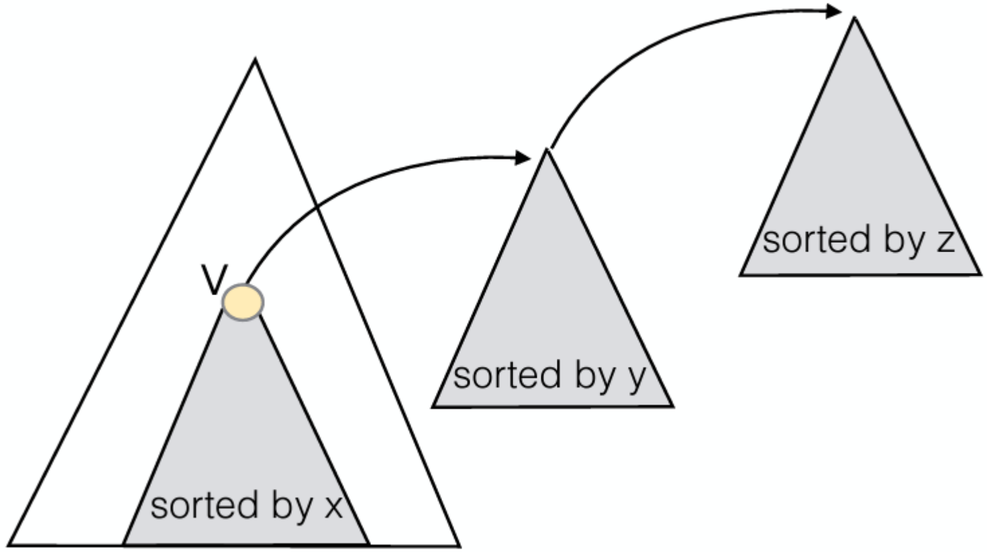


Range *Tree 3D*

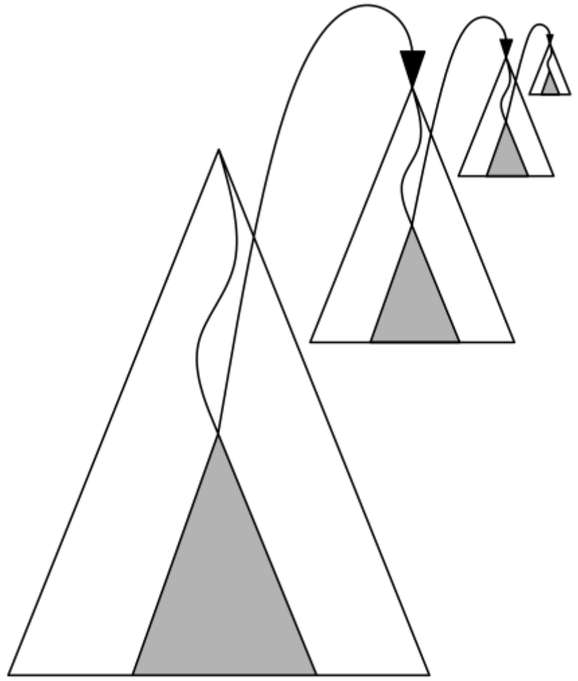
¿Cómo podríamos extender a tres dimensiones?

Range *Tree 3D*





Range *Tree nD*



Range *Tree nD*

Complejidad de consulta: $\mathcal{O}\left([\log n]^d\right)$

Complejidad de espacio: $\mathcal{O}\left(n[\log n]^{d-1}\right)$

Complejidad de construcción: $\mathcal{O}\left(n[\log n]^{d-1}\right)$

UTEC

UNIVERSIDAD DE INGENIERÍA
Y TECNOLOGÍA