

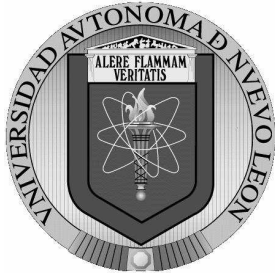
# ***Estructuras de datos***

## ***Árboles B***

Dra. Elisa Schaeffer

`elisa.schaeffer@gmail.com`

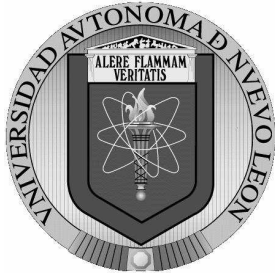
PISIS / FIME / UANL



# Árboles B

Árboles B son árboles balanceados que **no son binarios**.

Todos los vértices contienen datos y el número por datos por vértice puede ser **mayor a uno**.

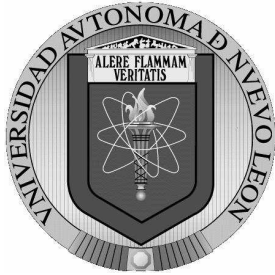


# Claves versus hijos

Si un vértice internal contiene  $k$  claves  $a_1, a_2, \dots, a_k$ , tiene necesariamente  $k + 1$  hijos que contienen las claves en los intervalos  $[a_1, a_2], [a_2, a_3], \dots, [a_{k-1}, a_k]$ .

Cada vértice contiene la información siguiente:

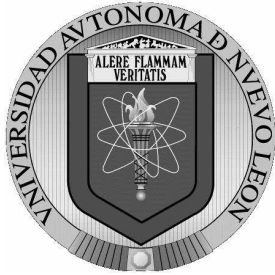
1. su número de claves  $k$
2. las  $k$  claves en orden no decreciente
3. un valor binario que indica si o no el vértice es una hoja
4. si no es una hoja,  $k + 1$  punteros a sus hijos  
 $c_1, c_2, \dots, c_{k+1}$



# Propiedades

Aplica para las  $d$  claves  $b_1, \dots, b_d$  en el ramo del hijo  $c_i$  que  $a_i \leq b_j \leq a_{i+1}$  para cada  $j \in [1, d]$ .

Todas las hojas del árbol tienen la misma profundidad y la profundidad es exactamente la altura del árbol.



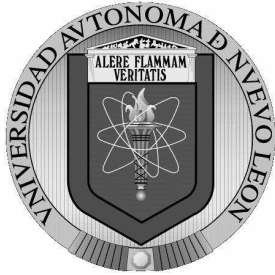
# Grado máximo y mínimo

- (I) Cada vértice salvo que la raíz debe contener por lo menos  $t - 1$  claves<sup>a</sup>.
- (II) Cada vértice puede contener al máximo  $2t - 1$  claves.

En consecuencia, cada vértice que no es hoja tiene por lo menos  $t$  hijos y al máximo  $2t$  hijos. Un vértice es **lleno** si contiene el número máximo permitido de claves.

---

<sup>a</sup>En los árboles  $B^*$ , se exige que esten por lo menos  $\frac{2}{3}$  llenos.

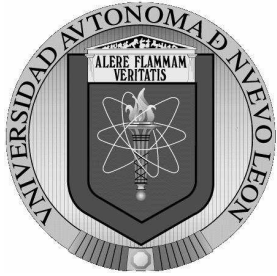


# *Altura y búsqueda*

En los árboles B aplica para la altura  $a$  del árbol que (omitimos la demostración) para  $t \geq 2$ ,

$$a \leq \log_t \frac{n+1}{2}.$$

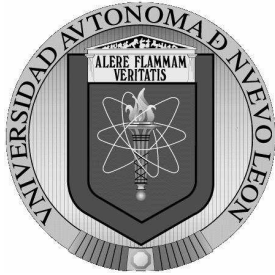
Búsqueda de una clave en un árbol B no diferencia mucho de la operación de búsqueda en árboles binarios, el único cambio siendo que habrá que elegir entre varias alternativas en cada vértice intermedio.



# ***Inserciones***



- 6 Buscamos la posición en dónde insertar la clave.

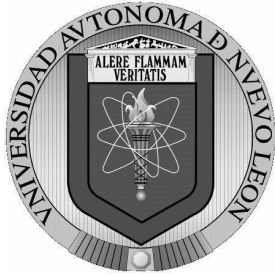


# ***Inserciones***



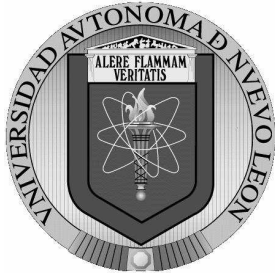
- ⑥ Buscamos la posición en dónde insertar la clave.
- ⑥ Si el vértice donde deberíamos realizar la inserción todavía no está lleno, insertamos la clave.





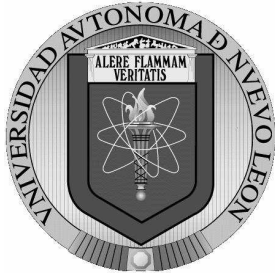
# ***Inserciones***

- ⑥ Buscamos la posición en dónde insertar la clave.
- ⑥ Si el vértice donde deberíamos realizar la inserción todavía no está lleno, insertamos la clave.
- ⑥ Si el vértice es lleno, habrá que identificar su clave mediana y dividir el vértice en dos partes.



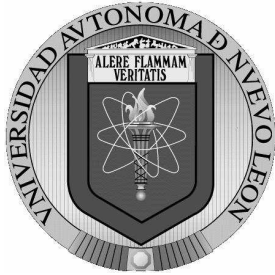
# ***Inserciones***

- ⑥ Buscamos la posición en dónde insertar la clave.
- ⑥ Si el vértice donde deberíamos realizar la inserción todavía no está lleno, insertamos la clave.
- ⑥ Si el vértice es lleno, habrá que identificar su clave mediana y dividir el vértice en dos partes.
- ⑥ La mediana moverá al vértice padre para marcar la división.



# ***Inserciones***

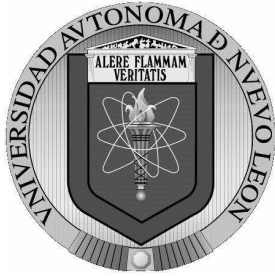
- ⑥ Buscamos la posición en dónde insertar la clave.
- ⑥ Si el vértice donde deberíamos realizar la inserción todavía no está lleno, insertamos la clave.
- ⑥ Si el vértice es lleno, habrá que identificar su clave mediana y dividir el vértice en dos partes.
- ⑥ La mediana moverá al vértice padre para marcar la división.
- ⑥ Esto puede causar que el padre también tendrá que dividirse.



# ***Inserciones***

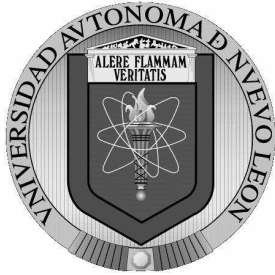
- ⑥ Buscamos la posición en dónde insertar la clave.
- ⑥ Si el vértice donde deberíamos realizar la inserción todavía no está lleno, insertamos la clave.
- ⑥ Si el vértice es lleno, habrá que identificar su clave mediana y dividir el vértice en dos partes.
- ⑥ La mediana moverá al vértice padre para marcar la división.
- ⑥ Esto puede causar que el padre también tendrá que dividirse.

Las divisiones pueden continuar recursivamente hasta la raíz.



## ***Eliminaciones***

Como también impusimos una cota inferior al número de claves, al eliminar una clave podemos causar que un vértice sea “demasiado vacío”.

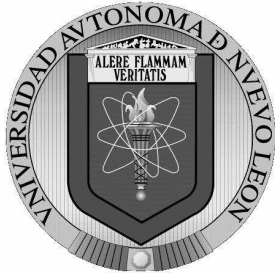


# ***Eliminaciones***

Como también impusimos una cota inferior al número de claves, al eliminar una clave podemos causar que un vértice sea “demasiado vacío”.

Al eliminar claves, los vértices “chupan” claves de reemplazo de sus hojas o de su padre.

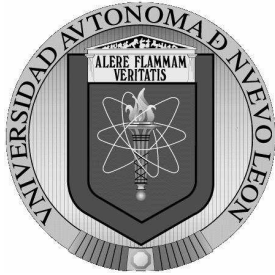
La operación que resulta se divide en varios casos posibles.



# Árboles multcaminos B+

Los árboles **multcaminos** (inglés: B+ trees) tienen además punteros extras entre vértices que son “hermanos” para ofrecer más posibilidades simples para mover claves al buscar, insertar y eliminar claves.

La otra diferencia entre los árboles B y los B+ es que los B+ son árboles **externos**: únicamente se guarda claves en las hojas y los vértices internos son puramente para el ruteo.



## ***Tarea para entregar el martes***

**Demuestra** que la altura de un árbol rojo-negro es al máximo  $2 \log(n + 1)$  utilizando la notación auxiliar siguiente y caracterizando el número de vértices de ruteo através las cinco propiedades de los árboles rojo-negro, no olvidando que son árboles binarios llenos:

El número de vértices negros en el camino desde  $v$  a una hoja (sin incluir a  $v$  mismo) es la **altura negra** de  $v$ ,  **$an(v)$** .