

Complejidad en AVL y árboles 2-3

Clase 08

IIC 2133 - Sección 1

Prof. Sebastián Buggedo

Sumario

Obertura

Complejidad en AVL

Árboles 2-3

Inserciones

Epílogo

¿Cómo están?



Miau

aus Frankreich

1.
Mi - au, mi - au! Hörst du mich schrei-en? Mi - au, mi - au, ich will dich frei-en.

2.
Folgst du mir aus den Ge-mä-chern, sin-gen wir hoch auf den Dä-chern.

3.
Mi - au, komm, ge-lieb-te Kat-ze, mi - au, reich mir dei-ne Tat-ze!

Miau, miau, hörst du mich schreien?
Miau, miau, ich will dich freien.

Folgst du mir aus den Gemächern,
singen wir hoch auf den Dächern.

Miau, komm, geliebte Katze,
miau, reich mir deine Tatze!

Segundo Acto: Diccionarios

Árboles y tablas de hash



Playlist 2



Playlist: DatiWawos Segundo Acto

Además sigan en instagram:
[@orquesta_tamen](#)

Árboles AVL: práctica

Ejercicio

Construya un árbol AVL cuyas llaves se insertan en el siguiente orden:

M, D, H, B, A, C, S, K, I

Luego de cada inserción, rebalancee los subárboles en caso de ser necesario.



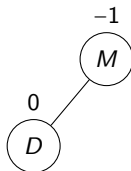
Ejercicio: inserción y rotaciones

Insertamos la llave M



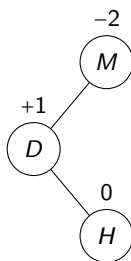
Ejercicio: inserción y rotaciones

Insertamos la llave D



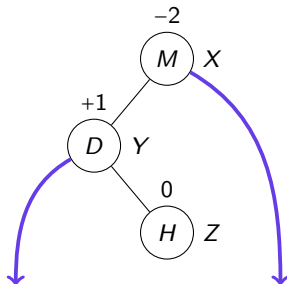
Ejercicio: inserción y rotaciones

Insertamos la llave M y se produce desbalance AVL



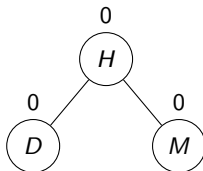
Ejercicio: inserción y rotaciones

Identificamos los nodos de la rotación (doble) y las direcciones para rotar



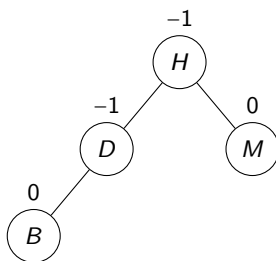
Ejercicio: inserción y rotaciones

Resultado de la rotación



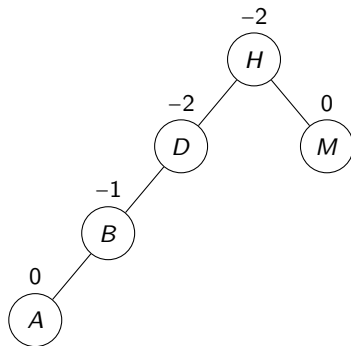
Ejercicio: inserción y rotaciones

Insertamos B y no se produce desbalance



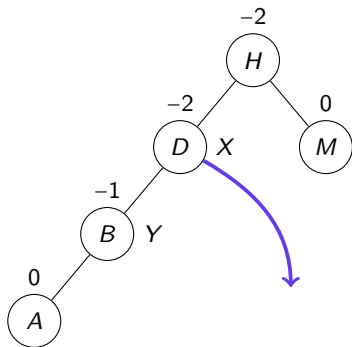
Ejercicio: inserción y rotaciones

Insertamos *A* y se produce desbalance



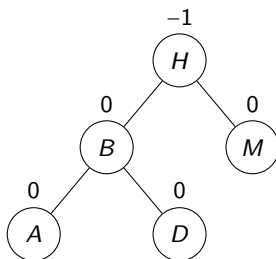
Ejercicio: inserción y rotaciones

Identificamos nodos para la rotación (simple) y la dirección



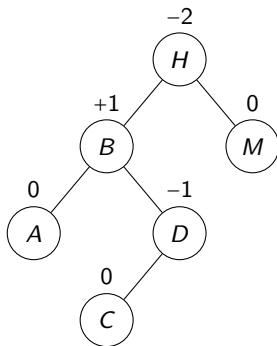
Ejercicio: inserción y rotaciones

Resultado de la rotación



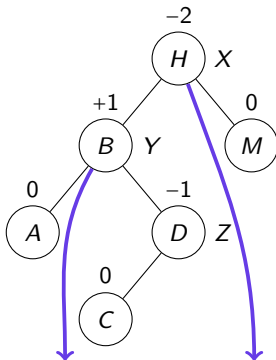
Ejercicio: inserción y rotaciones

Insertamos C y producimos desbalance



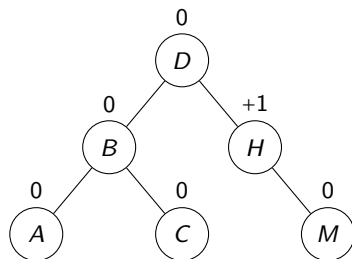
Ejercicio: inserción y rotaciones

Identificamos nodos de la rotación (doble) y las direcciones



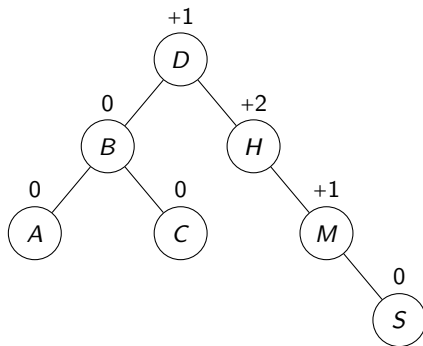
Ejercicio: inserción y rotaciones

Resultado de la rotación



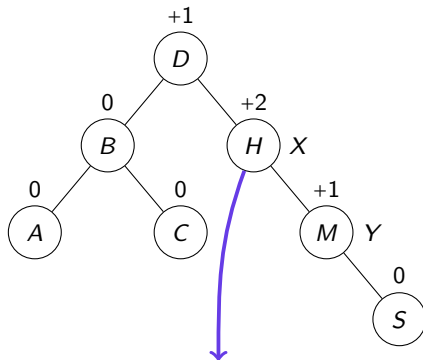
Ejercicio: inserción y rotaciones

Insertamos *S* y se produce desbalance



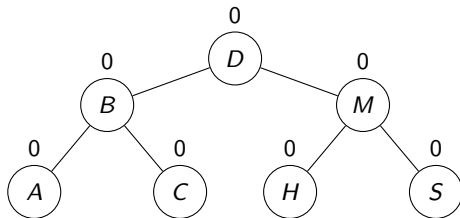
Ejercicio: inserción y rotaciones

Identificamos nodos para rotación (simple) y la dirección



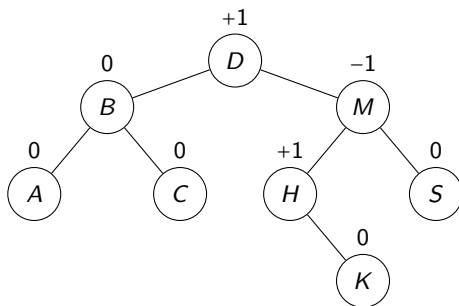
Ejercicio: inserción y rotaciones

Resultado de la rotación



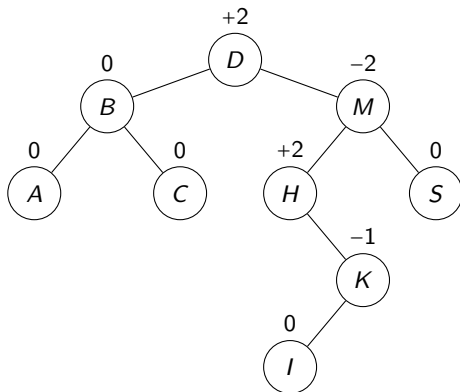
Ejercicio: inserción y rotaciones

Insertamos K y no hay desbalance



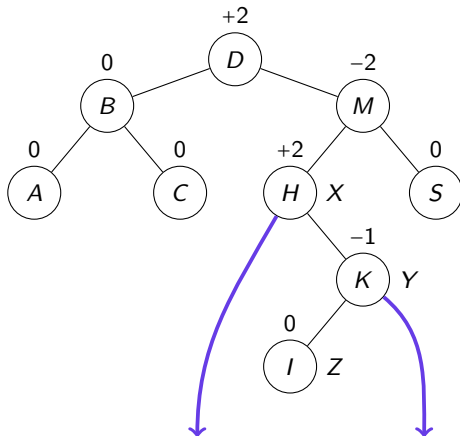
Ejercicio: inserción y rotaciones

Insertamos *I* y se produce desbalance



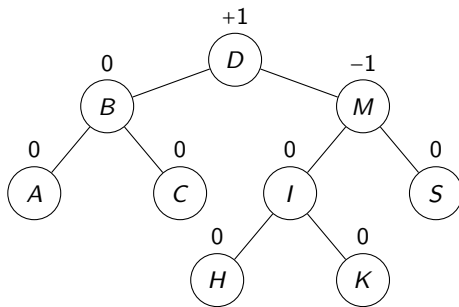
Ejercicio: inserción y rotaciones

Identificamos nodos de la rotación (doble) y las direcciones



Ejercicio: inserción y rotaciones

Resultado de la rotación



Objetivos de la clase

- ☐ Determinar la complejidad de operaciones en árboles AVL
- ☐ Comprender el modelo de árboles 2-3
- ☐ Distinguir el impacto de este modelo en la altura del árbol

Sumario

Obertura

Complejidad en AVL

Árboles 2-3

Inserciones

Epílogo

Complejidad de las rotaciones

Una rotación tiene **costo constante**

- En una rotación simple se cambian 3 punteros: $\mathcal{O}(1)$
- En una rotación doble se cambian 5 punteros: $\mathcal{O}(1)$

Además, al rotar para restaurar balance de X, este se resuelve sin crear un nuevo balance

- No se aumentan las diferencias de altura respecto al resto del árbol
- En el peor caso, se realiza a lo más **una rotación** (simple o doble) **por inserción**

Complejidad de las rotaciones

Gracias a esto, para un árbol de altura h , una inserción contempla

- Inserción propiamente tal $\mathcal{O}(h)$
- Detectamos los nodos que participan de la rotación (si aplica) $\mathcal{O}(h)$
- En el peor caso, aplica rotar y se requiere una rotación $\mathcal{O}(1)$
- Estos pasos se realizan consecutivos Total $\mathcal{O}(h)$

¿Cuál es la altura de un árbol AVL en función de n ?

Altura de un árbol AVL

Antes de atacar el problema, pensemos en la relación inversa: cómo depende n de h .

Para un árbol AVL T

- ¿Cuál es el máximo número de nodos de T si tiene altura h ?
- ¿Y el mínimo?

Si probamos un crecimiento exponencial para n en función de h ,
probaremos que la altura está acotada por $\log(n)$

Altura de un árbol AVL

En primer lugar, consideremos el número máximo en un árbol AVL

Esto ocurre cuando el árbol está **lleno**, es decir, para cada nivel del árbol existe la máxima cantidad de nodos posible. Este caso claramente es un árbol AVL

Si k es el nivel de los nodos (recordando que la raíz está en el nivel 0 y que la altura es $\#$ niveles), la cantidad de nodos total es

$$n = \sum_{k=0}^{h-1} 2^k = 2^h - 1$$

con lo cual

$$n \in \Theta(2^h) \quad \Leftrightarrow \quad 2^h \in \Theta(n)$$

Como se trata de funciones crecientes deducimos que $h \in \Theta(\log(n))$

Altura de un árbol AVL

En segundo lugar, consideramos el número mínimo $m(h)$ de nodos que puede tener un árbol AVL de altura h

Plantearemos una recurrencia que defina m

- Observamos que $m(1) = 1$ y $m(2) = 2$
- Para el caso general, nos interesa minimizar el número de nodos
- Nos concentramos en el caso en que los hijos difieren su altura en 1
- Además, dado que el árbol principal tiene altura h , uno de los hijos debe tener $h - 1$ y el otro $h - 2$
- Finalmente, exigimos que cada uno de estos hijos también tenga el mínimo posible. Con esto, planteamos una ecuación de recurrencia

$$m(h) = m(h - 1) + m(h - 2) + 1$$

Altura de un árbol AVL

Esta recurrencia es similar a la recurrencia de Fibonacci

$$F(0) = 0, F(1) = 1, \quad F(h) = F(h-1) + F(h-2), h \geq 2$$

cuyo término general satisface la aproximación $F(h) \approx \frac{\varphi^h}{\sqrt{5}}$. Con esto, se puede probar por inducción que

$$m(h) = F(h+3) - 1$$

y utilizando la aproximación se obtiene

$$m(h) = F(h+3) - 1 \approx \frac{\varphi^{h+3}}{\sqrt{5}} - 1$$

y deducimos que

$$h+3 \approx \log_{\varphi}(\sqrt{5}(m(h)+1))$$

Concluimos que $h \in \mathcal{O}(\log(n))$

Altura de un árbol AVL

Teorema

Todo árbol AVL con n nodos tiene altura h tal que

$$h \in \mathcal{O}(\log(n))$$

El problema del balance

Tenemos una primera definición de *balance* para **árboles binarios**

- Propiedad AVL definida recursivamente
- Esto es: diferencia de alturas entre hermanos a lo más 1

Pero podemos dar otra definición: todas las hojas están a la misma profundidad y que sea $\mathcal{O}(\log(n))$

¿Es posible conseguir esto con árboles binarios?

Sumario

Obertura

Complejidad en AVL

Árboles 2-3

Inserciones

Epílogo

Un nuevo acercamiento al problema

En lugar de utilizar árboles binarios o ternarios, usaremos una mezcla de ellos

La idea de esta **nueva estructura** incluye

- Tener dos tipos de nodos
- **2-Nodos** que tendrán una llave, y si no son hojas tendrán **2 hijos**
- **3-Nodos** que tendrán **dos llaves** distintas y ordenadas, y si no son hojas tendrán **3 hijos**

Esta estrategia permitirá tener las hojas a la misma profundidad

Además garantizará profundidad $\mathcal{O}(\log(n))$ al almacenar n llaves

Árboles de búsqueda 2-3

Definición

Un **árbol de búsqueda 2-3** es una EDD que almacena **(llave, valor)** según

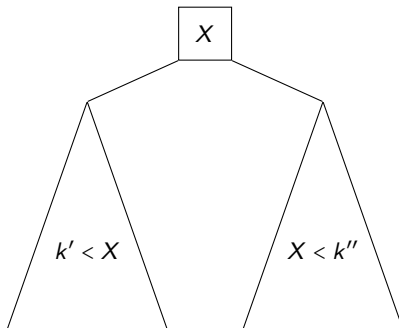
1. Un árbol 2-3 tiene una nodo que puede ser un **2-nodo** (con una llave) o un **3-nodo** (con 2 llaves distintas y ordenadas)
2. El nodo puede no tener hijos o tener exactamente
 - 2 hijos árboles 2-3 si es un 2-nodo
 - 3 hijos árboles 2-3 si es un 3-nodo

y que además, satisface la **propiedad de árboles 2-3**

- Si es 2-nodo con llave k
 - las llaves k' del hijo izquierdo son $k' < k$
 - las llaves k'' del hijo derecho son $k < k''$
- Si es 3-nodo con llaves $k_1 < k_2$
 - las llaves k' del hijo izquierdo son $k' < k_1$
 - las llaves k'' del hijo central son $k_1 < k'' < k_2$
 - las llaves k''' del hijo derecho son $k_2 < k'''$

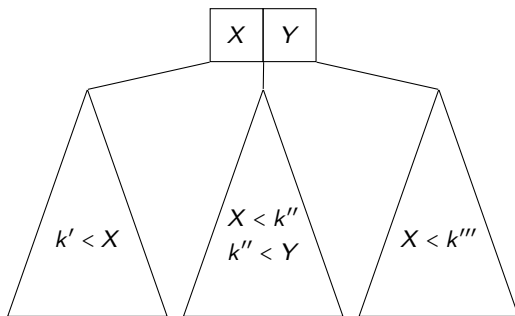
Árboles de búsqueda 2-3

Un 2-nodo con llave X que no es hoja tiene la siguiente estructura



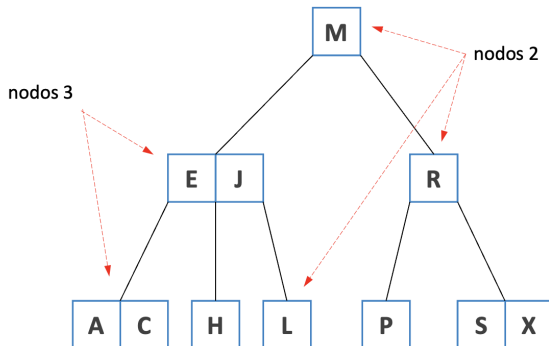
Árboles de búsqueda 2-3

Un 3-nodo con llaves $X < Y$ que no es hoja tiene la siguiente estructura



Árboles de búsqueda 2-3

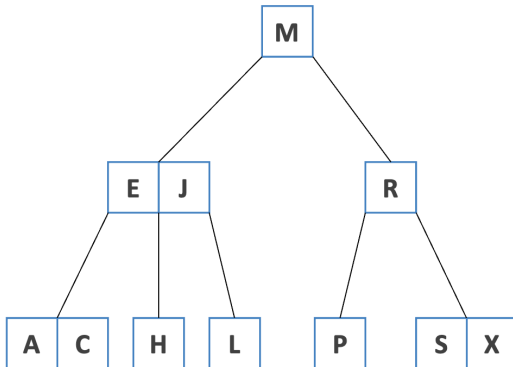
Un ejemplo de árbol 2-3 con llaves alfabéticas



Observemos que tal como en los ABB,
el orden de las llaves está implícito

Árboles de búsqueda 2-3

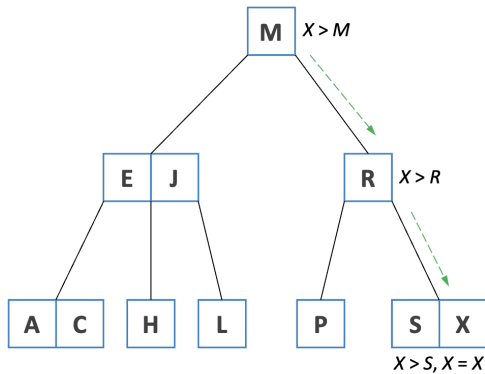
No olvidemos que los árboles 2-3 son **árboles de búsqueda**



Podemos buscar elementos comparando llaves recursivamente

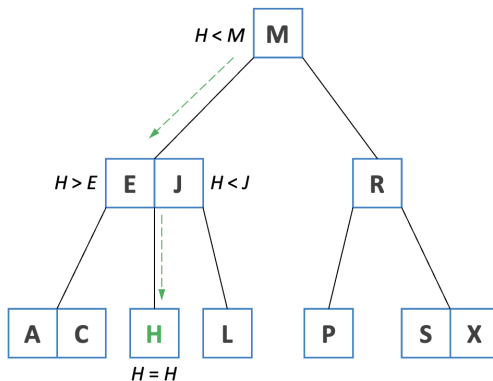
Árboles de búsqueda 2-3

Buscamos la llave X



Árboles de búsqueda 2-3

Buscamos la llave H



Observemos que en el peor caso, tenemos que comparar dos llaves por nodo

Sumario

Obertura

Complejidad en AVL

Árboles 2-3

Inserciones

Epílogo

Operaciones en árboles 2-3

Nos planteamos el mismo desafío que en los ABB: implementar operaciones

- Queremos que sean eficientes
- Recordemos que al insertar podría cambiar la altura
- El objetivo de los árboles 2-3 es que las hojas estén en el mismo nivel
- **Ojo:** esto no significa que la profundidad no cambie

¿Cómo insertar llaves manteniendo la profundidad pareja?

Inserciones en árboles 2-3

Al insertar llaves seguiremos la siguiente estrategia

- Se inserta como llave *múltiple* en una hoja existente
- Si la hoja era 2-nodo, queda como 3-nodo y terminamos
- Si la hoja era 3-nodo, queda como 4-nodo (con 3 llaves) por ahora
- La llave central del 4-nodo sube como llave múltiple al padre (**split**)
- Se repite la modificación de forma recursiva hacia la raíz

El árbol solo crece en altura cuando la raíz se llena
al recibir una llave desde un hijo

Inserciones en árboles 2-3

Insertamos la llave D que será la raíz inicial



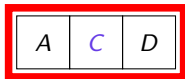
Inserciones en árboles 2-3

Insertamos la llave A

A	D
-----	-----

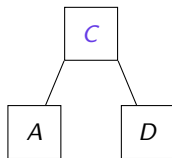
Inserciones en árboles 2-3

Insertamos la llave *C*, produciendo un nodo no válido



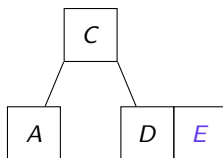
Inserciones en árboles 2-3

Efectuamos un **split** para subir la llave central como nueva raíz



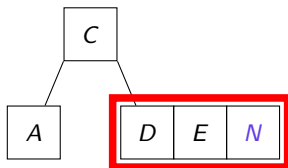
Inserciones en árboles 2-3

Insertamos la llave *E*



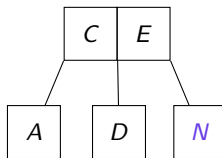
Inserciones en árboles 2-3

Insertamos la llave N , produciendo un nodo no válido



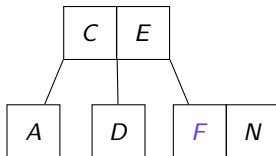
Inserciones en árboles 2-3

Efectuamos un **split** subiendo la llave *E* e insertándola ordenadamente



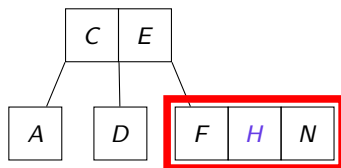
Inserciones en árboles 2-3

Insertamos la llave *F*



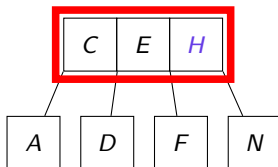
Inserciones en árboles 2-3

Insertamos la llave *H* y se produce un nodo no válido



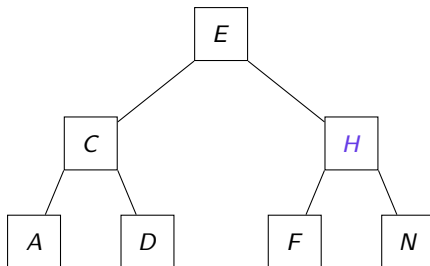
Inserciones en árboles 2-3

Subimos la llave *H* y se produce un nuevo nodo no válido



Inserciones en árboles 2-3

Hacemos **split** de la raíz actual, subiendo la llave *E* como nueva raíz



Sumario

Obertura

Complejidad en AVL

Árboles 2-3

Inserciones

Epílogo

Objetivos de la clase

- ☐ Determinar la complejidad de operaciones en árboles AVL
- ☐ Comprender el modelo de árboles 2-3
- ☐ Distinguir el impacto de este modelo en la altura del árbol