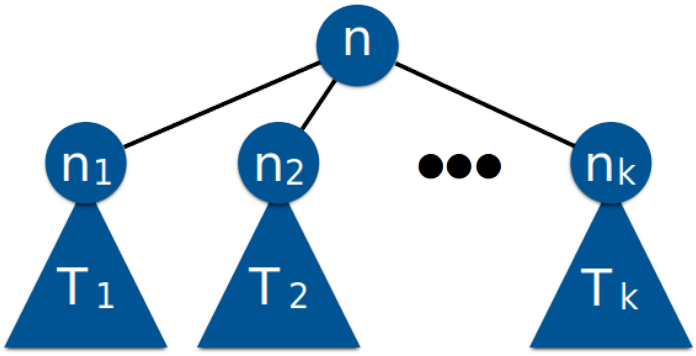
ESTRUCTURA DE DATOS

Árboles

Árbol n-ario



Conceptos

Base: Un nodo es un árbol n-ario (si el árbol tiene un sólo nodo, éste es el nodo raíz)

Recurrencia: Si n es un nodo y T1, ..., Tk son árboles narios con raíces n1,..., nk, respectivamente, podemos construir un árbol que tenga como raíz el nodo n y subárboles T1, ..., Tk

A los nodos que son hijos de un mismo padre se les denomina **hermanos.**

Se llama **grado de un nodo** al número de subárboles (de hijos) que tiene dicho nodo. Los nodos de grado 0 se denominan **hojas** o **nodos terminales**. El resto se llaman nodos **no terminales** o **interiores**.

El **grado de un árbol** es el máximo de los grados de sus nodos.

El **camino entre dos nodos**, ni y nj se define como la secuencia de nodos del árbol necesaria para alcanzar el nodo nj desde el nodo ni

La **longitud del camino** entre dos nodos es igual al número de nodos que forman el camino menos largo.

Nivel de un nodo

- Base: El nivel del nodo raíz es 0
- Recurrencia: si un nodo está en el nivel i, todos sus hijos están en el nivel i+1

La **profundidad de un árbol** es el máximo de los niveles de los nodos del árbol.

Recorridos

Recorridos en profundidad:

- **Preorden**: raíz, Pre(T1), Pre(T2),..., Pre(Tk)
- **Inorden**: In(T1), raíz, In(T2),..., In(Tk)
- **Postorden**: Pos(T1), Pos(T2),..., Pos(Tk), raíz

Recorrido en anchura:

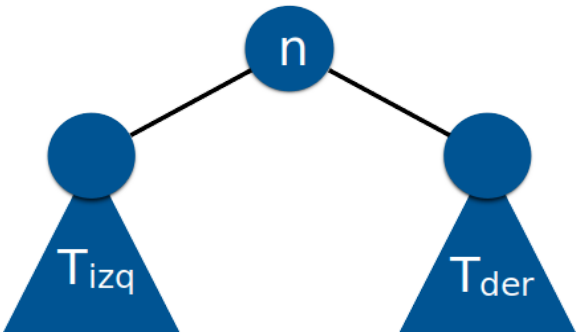
- **Por niveles:** de arriba a abajo y de izquierda a derecha, empezando por la raíz

Árbol binario

Conceptos

Base: Un árbol vacío es un árbol binario

Recurrencia: Si n es un nodo y T_{izq} y T_{der} son árboles binarios, podemos construir un nuevo árbol binario que tenga como raíz el nodo n y como subárboles T_{izq} y T_{der} (subárbol izquierdo y derecho, respectivamente)

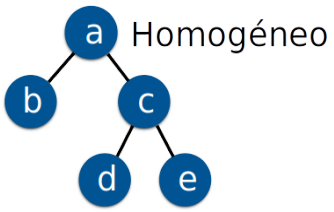


Un árbol binario NO es un árbol n-ario de grado 2.

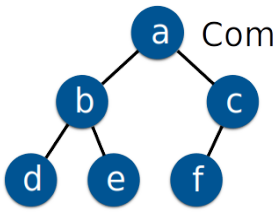
En un árbol binario, el **número máximo de nodos** que puede haber en el nivel i es 2i

Tipos

- **Árbol binario homogéneo**: aquél cuyos nodos tienen grado 0 ó 2 (no hay ninguno de grado 1)
- **Árbol binario completo**: aquél que tiene todos los niveles llenos excepto, quizá, el último, en cuyo caso los huecos deben quedar a la derecha



Homogéneo



Completo

Recorridos en árboles binarios

Recorridos en profundidad:

- **Preorden**: raíz, Pre(Tizq), Pre(Tder)
- **Inorden**: In(Tizq), raíz, In(Tder)
- **Postorden**: Pos(Tizq), Pos(Tder), raíz

Se pueden realizar de forma recursiva, siguiendo el esquema de construcción recursivo de árboles binarios •

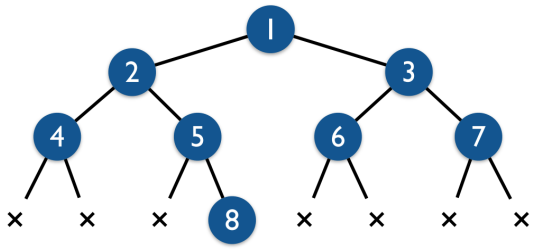
Recorrido en anchura:

- **Por niveles**: de izquierda a derecha Se realiza de forma iterativa.

Pre(n) < Pre(m) && Post(m) < Post(n)

⇕

n ancestro de m



	Pre(n)<Pre(m)	In(n)<In(m)	Pos(n)<Pos(m)
n a la izquierda de m	✅	✅	✅
n a la derecha de m	❌	❌	❌
n descendiente de m	❌	✅❌	✅
n ancestro de m	✅	✅❌	❌

Lectura/escritura de un árbol

- **Preorden** 1 2 4 x x 5 x 8 x x 3 6 x x 7 x x 1 2 4 5 8 3 6 7
- **Por niveles** 1 2 3 -1 4 5 6 7 -1 -1 -1 -1 8 -1 -1 -1 -1 -1

Función f(t) sobre un árbol binario, t

- **Base**: Valor de la función si t es el árbol vacío
- **Recurrencia**: se supone conocida la función para cada uno de los subárboles Tizq y Tder de t.

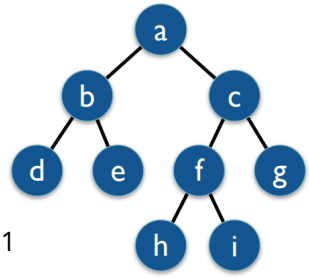
Se calcula el valor final de la función suponiendo conocidos los valores anteriores.

Representación mediante vectores

Las **etiquetas** de los nodos se almacenan en un vector.

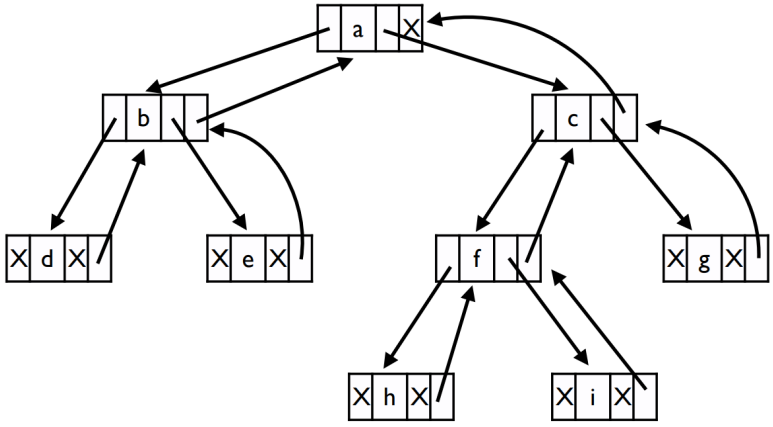
Los **nodos** se enumeran de la siguiente forma:

- A la raíz le corresponde el índice 0
- Si a un nodo le corresponde el índice k:
  - Su hijo izquierdo, si tiene, está en la posición 2*(k+1)-1 = 2*k+1
  - Su hijo derecho, si tiene, está en la posición 2*(k+1) = 2*k+2
  - Su padre, si tiene, está en la posición (k-1)/2



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	...
a	b	c	d	e	f	g					h	i			...

Representación mediante celdas enlazadas

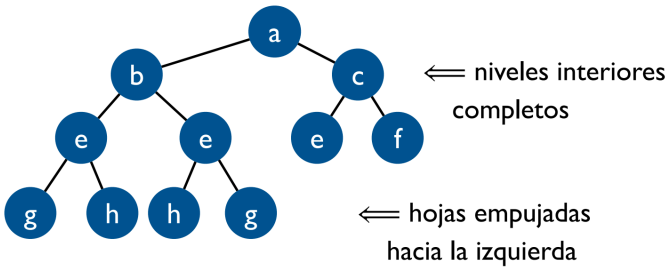


Árboles Binarios Parcialmente Ordenados (APO)

Definición

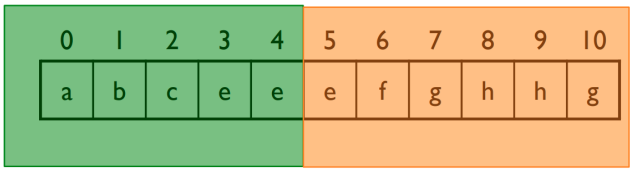
Un árbol binario es un APO si cumpe:

la etiqueta de cada nodo es menor o igual que las etiquetas de los hijos, manteniéndose tan equilibrado (balanceado) como sea posible (hojas empujadas a la izquierda)



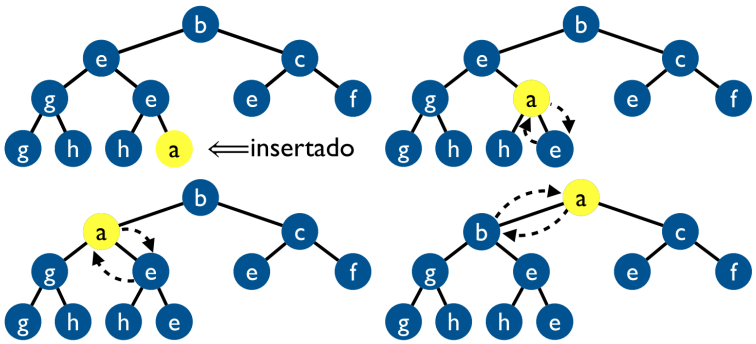
Representación

La **representación** para los APO es la **del montón** (**Heap**). En este contexto, el montón M será un vector en el que se guardará el APO por niveles, de la siguiente forma:



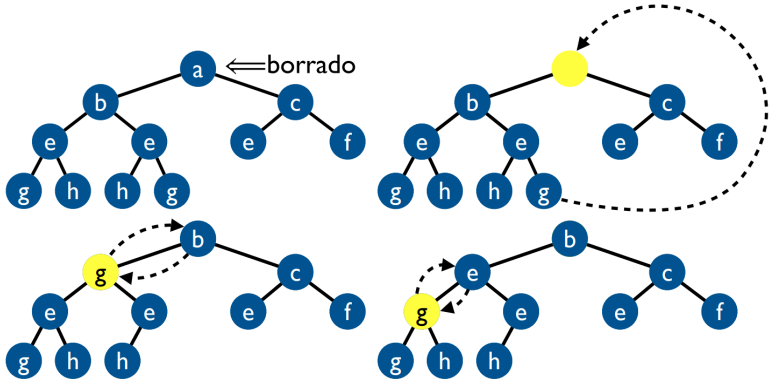
- M[0] aloja la raíz.
- Los hijos izquierdo y derecho (si existen) del nodo M[k] estarán en las posiciones M[2k+1] y M[2k+2], lo que equivale a decir que el padre de M[k] es M[(k-1) ∀k>0

Inserción



$O(\log_2 n)$

Borrado



$O(\log_2 n)$

Árboles Binarios de Búsqueda (ABB)

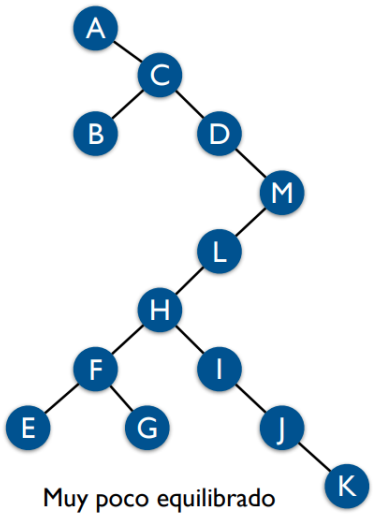
Definición

Un ABB es un árbol binario con la propiedad de que todos los elementos almacenados en el **subárbol izquierdo** de cualquier nodo x (incluyendo la raíz) **son menores** (o iguales***) **que** el elemento almacenado en **x** y todos los elementos almacenados en el **subárbol derecho** de x son **mayores** que el elemento almacenado en **x**.

Propiedades

- La búsqueda de un elemento en el árbol reproduce la **búsqueda binaria** ➡ $O(\log_2(n))$
- El recorrido en el **InOrden** de un ABB produce un **listado ordenado** de las etiquetas

Árboles Binarios Equilibrados - AVL



Muy poco equilibrado

En ocasiones, la construcción de los ABB conduce a árboles con características muy pobres para la búsqueda ➡

Idea: Construir ABB equilibrados, impidiendo que en ningún nodo las alturas de los subárboles izquierdo y derecho difieran en más de una unidad

Concepto

Diremos que un **ABB** es un AVL (o que está equilibrado en el sentido de Addelson-Velski-Landis) si para cada uno de sus nodos se cumple que las **alturas de sus dos subárboles difieren como máximo en 1**

Características

- En el peor de los casos, la búsqueda se puede realizar en O(log2 n)

Equilibrio en inserciones y borrados

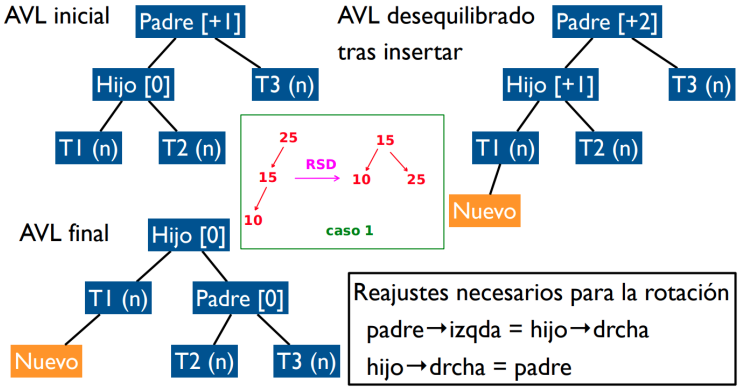
Idea: Usar un campo altura en el registro que represente cada uno de los nodos del AVL para determinar el factor de equilibrio (diferencia de altura entre los subárboles izquierdo y derecho), de forma que cuando esa diferencia sea >1 se hagan los reajustes necesarios en los punteros para que tenga una diferencia de alturas ≤1

- Notaremos los subárboles como T_k, anotando entre paréntesis su altura (la altura de su raíz)
- Notaremos el factor de equilibrio como un valor con signo ubicado entre corchetes junto a cada padre o hijo

Las dos situaciones posibles que pueden presentarse son:

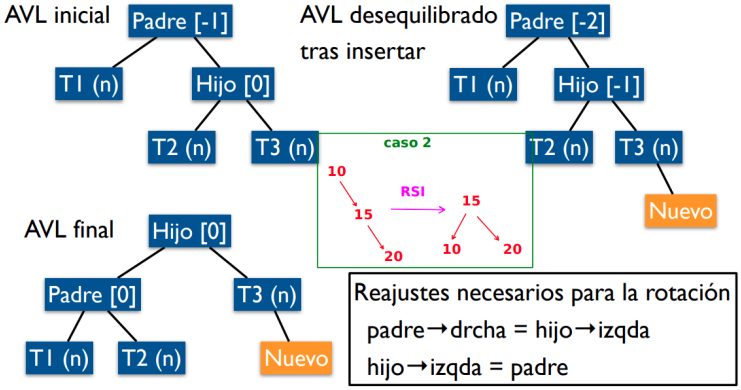
- Rotaciones simples
  - Se preserva el inorden
  - Altura del árbol final = altura árbol inicial
- Rotaciones dobles
  - Compuesto por dos rotaciones

Rotación simple a la derecha



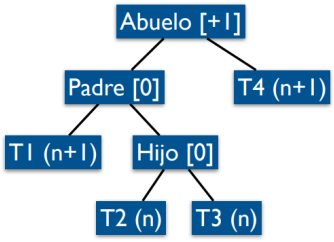
Reajustes necesarios para la rotación
padre→izqda = hijo→drcha
hijo→drcha = padre

Rotación simple a la izquierda

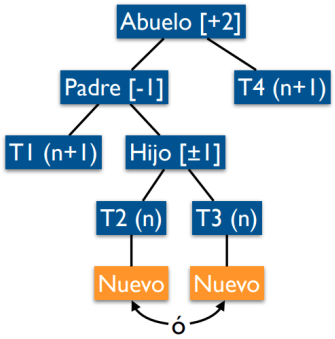


Reajustes necesarios para la rotación
padre→drcha = hijo→izqda
hijo→izqda = padre

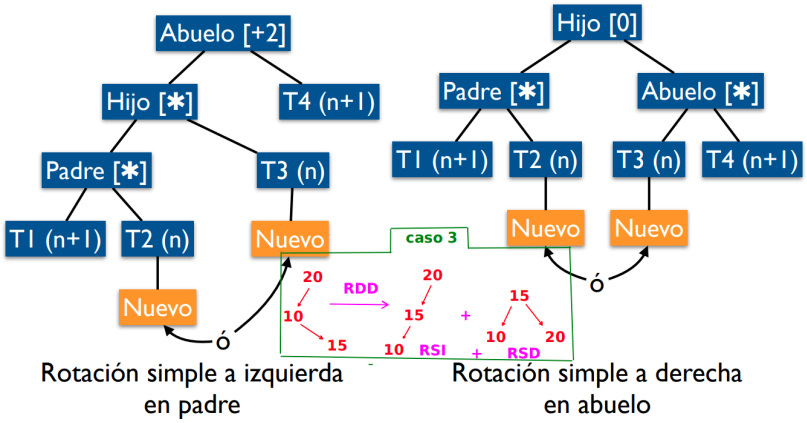
Rotación doble a la derecha



AVL inicial



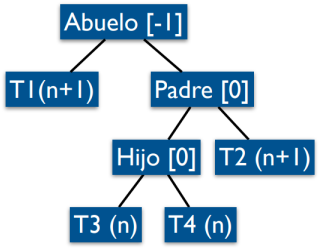
AVL desequilibrado
tras insertar



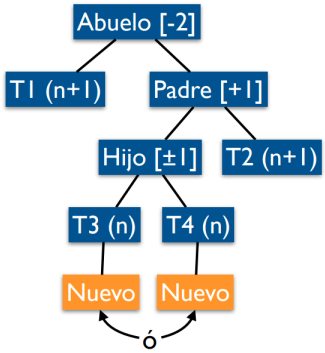
Rotación simple a izquierda
en padre

Rotación simple a derecha
en abuelo

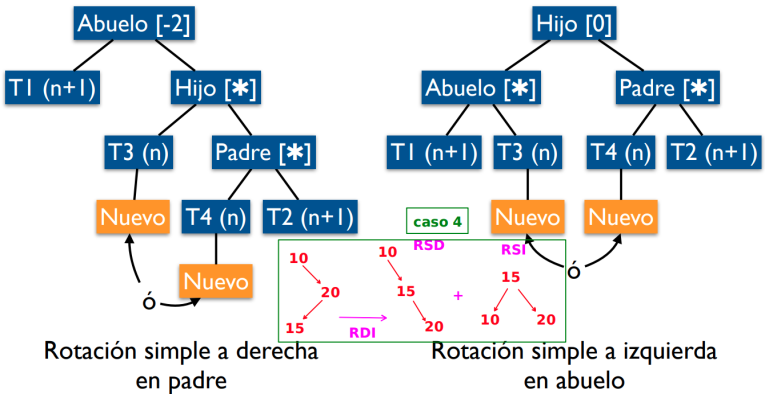
Rotación doble a la izquierda



AVL inicial



AVL desequilibrado
tras insertar



Rotación simple a derecha
en padre

Rotación simple a izquierda
en abuelo

¿Qué rotación utilizar?

Si la inserción se realiza en:

- El **hijo izquierdo del hijo izquierdo** del nodo desequilibrado ➡ **RSD**
- El **hijo derecho del hijo derecho** del nodo desequilibrado ➡ **RSI**
- El **hijo derecho del hijo izquierdo** del nodo desequilibrado ➡ **RDD**
- El **hijo izquierdo del hijo derecho** del nodo desequilibrado ➡ **RDI**

Ejemplo