



Phantone

www.wuolah.com/student/Phantone

695

ED-T4-1.pdf

Apuntes Teoría



2º Estructuras de Datos



Grado en Ingeniería Informática



Escuela Técnica Superior de Ingenierías Informática y de
Telecomunicación
Universidad de Granada



**El más PRO del lugar
puedes ser Tú.**

¿Quieres eliminar toda la publi
de tus apuntes?

Hazte PRO

4,95€/mes

W

WUOLAH



El más PRO del lugar puedes ser Tú.



¿Quieres eliminar toda la publi de tus apuntes?



¡Fuera Publi!
Concéntrate al máximo



Apuntes a full.
Sin publi y sin gastar coins

Para los amantes de la inmediatez, para los que no desperdician ni un solo segundo de su tiempo o para los que dejan todo para el último día.



Quiero ser PRO

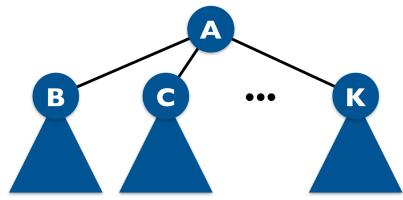
4,95 / mes

TEMA 4.1: ÁRBOLES

Árbol n-ario

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

Recurrencia: si n es un nodo y T_1, \dots, T_k son árboles n-arios con raíces n_1, \dots, n_k , respectivamente, podemos construir un árbol que tenga como raíz el nodo n y subárboles T_1, \dots, T_k .



Conceptos sobre árboles

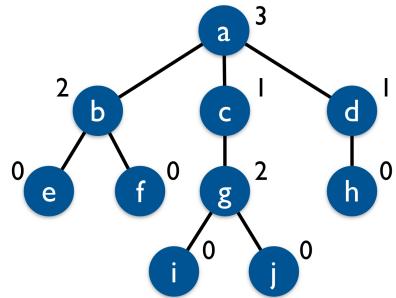
Se dice que un árbol está etiquetado si todos sus nodos contienen una etiqueta.

A los nodos que son hijos de un mismo parente se les llama hermanos.

Se llama grado de un nodo al número de subárboles (hijos) que tiene dicho nodo.

Los nodos de grado 0 se denominan hojas o nodos terminales. El resto se llaman no terminales o interiores.

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



El camino entre dos nodos n_i y n_j se define como la secuencia de nodos del árbol necesaria para alcanzar el nodo n_j desde el nodo n_i .

La longitud del camino entre dos nodos es igual al número de nodos que forman el camino menos 1. (nº ejes del camino).

Ej: camino $(a, f) = \{a, b, f\}$, longitud 2.

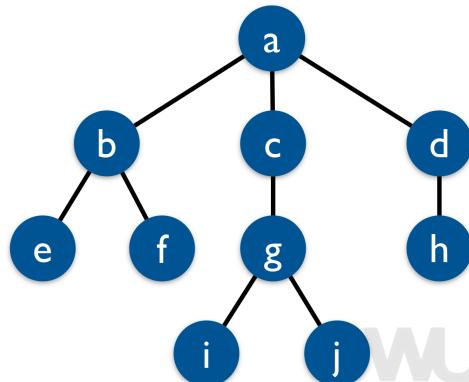
Nivel de un nodo:

Base: el nivel del nodo raíz es 0.

Recurrencia: si un nodo está en el nivel i , todos sus hijos están en el nivel $i+1$.

Altura y profundidad de un árbol

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



ÁRBOLES BINARIOS

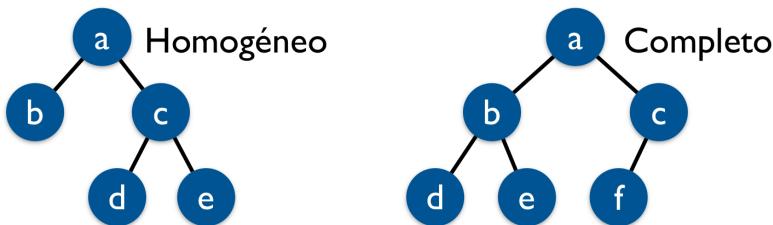
Base: un árbol vacío es un árbol binario.

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).

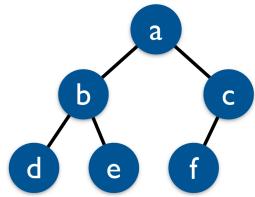
Un árbol binario NO es un árbol n -ario de grado 2.

Árbol binario completo: es aquel que tiene todos los niveles llenos, excepto, quizá, el último (en cuyo caso los huecos deben quedar a la derecha).

Árbol binario homogéneo: aquel cuyos nodos tienen grado 0 ó 2 (no hay ninguno de grado 1).



En un árbol binario completo con n nodos el camino más largo de la raíz a las hojas no atravesará más de $\log_2 n$ nodos.



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

Ej:	Nivel	Nodos
	0	$2^0 = 1$
	1	$2^1 = 2$
	2	$2^2 = 4$

En un árbol binario completo de altura K , el número máximo de nodos es $2^{K+1} - 1$.

RECORRIDO DE UN ÁRBOL BINARIO

Recorridos en profundidad:

- **Preorden:** raíz, Pre(T_1), Pre(T_2), ..., Pre(T_k).
- **Inorden:** In(T_1), raíz, In(T_2), ..., In(T_k).
- **Posorden:** Pos(T_1), Pos(T_2), ..., Pos(T_k), raíz.

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

Recorrido en anchura: por niveles, de arriba a abajo y de izquierda a derecha, empezando por la raíz.

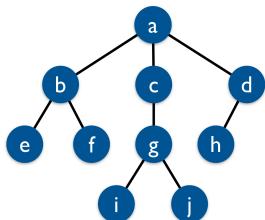
Se realiza de forma iterativa.

Una página más, y a por un café

Ánimo, tu puedes

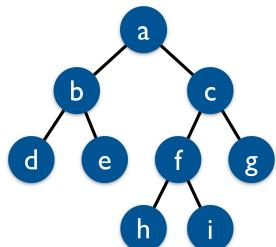
Estudialo bien, que
tiene pinta de importante

Ejemplo:



Preorden: a, b, e, f, c, g, i, j, d, h. }
 Inorden: e, b, f, a, i, g, j, c, h, d. } Recursivos
 Postorden: e, f, b, i, j, g, c, h, d, a }
 Por niveles: a, b, c, d, e, f, g, h, i, j } Iterativo

Esto lo pregunta
seguuuuuuuuuro!

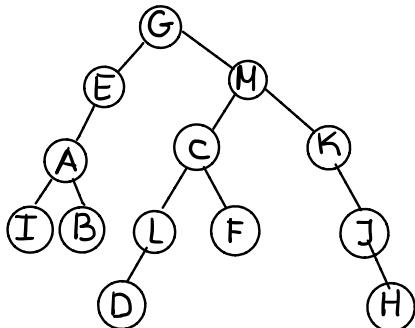


Preorden: a, b, d, e, c, f, h, i, g
 Inorden: d, b, e, a, h, f, i, c, g.
 Postorden: d, e, b, h, i, f, g, c, a
 Por niveles: a, b, c, d, e, f, g, h, i.

Echa un vistazo
verás que guay



	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	✓	✓✗	✗



Preorden: G E A I B M C L D F K J H
Inorden: I A B E G L D C F M K H T



Clases en **DIRECTO**



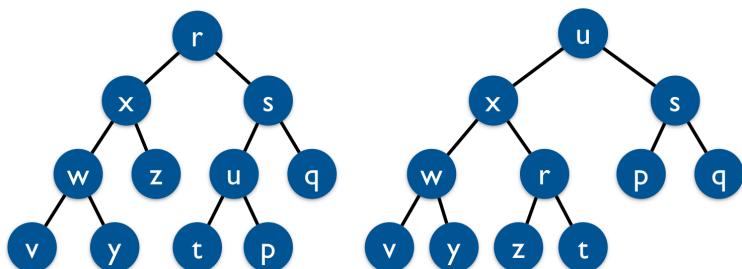
Audio y vídeo **PROFESIONAL**



Pizarra digital **COMPARTIDA**



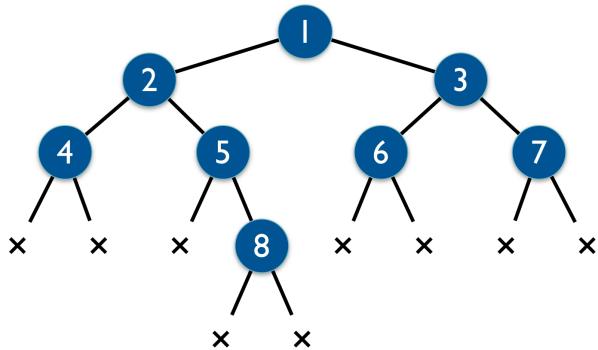
Máximo
10 PERSONAS



Inorden (en ambos): $v, w, y, x, z, r, t, u, p, s, q$

En general un árbol no puede recuperarse con sólo uno de sus recorridos.

LECTURA / ESCRITURA DE UN ÁRBOL



Preorden:

n 1 n 2 n 4 x x n 5 x n 8 x x
n 3 n 6 x x n 7 x x

Por niveles:

1	2	3	-1	4	5	6	7	-1	-1	-1
-1	8	-1	-1	-1	-1	-1	-1	-1	-1	

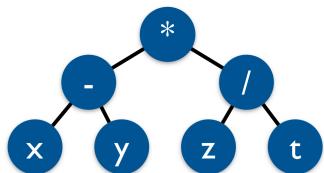
APLICACIÓN : ÁRBOLES DE EXPRESIÓN

Árboles sintácticos: árboles que contienen las derivaciones de una gramática necesarias para obtener una frase del lenguaje.

Árboles de expresión: etiquetamos:

- hojas con un operando.
- nodos interiores con un operador

Ejemplo: $(x - y) * (z / t) \rightarrow$ Inorden

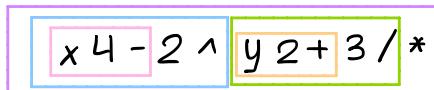


Preorden: * - x y / z t \rightarrow Representación prefija
Postorden: x y - z t / * \rightarrow Representación postfija } No necesitan parentesis

Resolución de ambigüedades: recorridos en preorden y postorden más:

- nivel de cada nodo.
- n.º de hijos de cada nodo.

Ejemplo: $x 4 - 2^y 2 + 3 / *$ (postfijo)



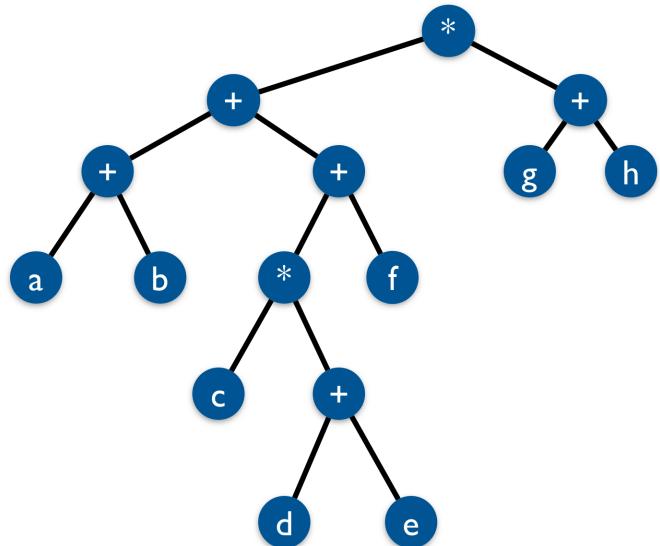
- $x + 4 \rightarrow (x + 4) 2 \wedge y 2 + 3 / *$
- $(x + 4) \wedge 2 \rightarrow ((x + 4) \wedge 2) y 2 + 3 / *$
- $y + 2 \rightarrow ((x + 4) \wedge 2) (y + 2) 3 / *$
- $(y + 2) / 3 \rightarrow ((x + 4) \wedge 2) ((y + 2) / 3) *$
- $((x + 4) \wedge 2) * ((y + 2) / 3)$

Las notaciones prefija y postfija facilitan la evaluación automática de expresiones aritméticas.

Ejemplo:

$((15 / (7 - (1 + 1))) * 3) - (2 + (1 + 1))$

$$\begin{aligned}
 & [(a+b) + (c * (d+e) + f)] * (g+h) \Rightarrow * E_1 E_2 \\
 & \quad \boxed{E_1} \quad \boxed{E_2} \\
 & [(a+b) + (c * (d+e) + f)] \Rightarrow * + E_{11} E_{12} E_2 \\
 & \quad \boxed{E_{11}} \quad \boxed{E_{12}} \quad \boxed{E_2} \\
 & E_{11} \equiv +ab \Rightarrow * + + a b E_{12} E_2 \\
 & E_{12} \equiv [(c * (d+e) + f] \Rightarrow + E_{121} E_{122} \\
 & \quad \boxed{E_{121}} \quad \boxed{E_{122}} \\
 & E_{121} \equiv c * (d+e) \Rightarrow * c E_{1212} \equiv * c + de \\
 & \quad \boxed{E_{1211}} \quad \boxed{E_{1212}} \\
 & * + + a b + * c + d e f E_2 \\
 & \quad \boxed{* + + a b} \quad \boxed{* c + d e f} \quad \boxed{E_2} \\
 & E_2 \equiv (g+h) \equiv + g h \\
 & \quad \boxed{E_2} \\
 & \left. \begin{aligned}
 & * + + a b + * c + d e f + g h \\
 & \quad \boxed{* + + a b} \quad \boxed{* c + d e f} \quad \boxed{g h}
 \end{aligned} \right\}
 \end{aligned}$$



Definición de funciones en árboles: generalmente la forma más fácil de definir una función sobre un árbol es trasladar la definición recurrente (re recursiva) del dominio a la definición de ésta.

Esto no quiere decir que toda función definida sobre un árbol deba ser recursiva. Podemos encontrar problemas cuya solución exija diseñar funciones iterativas, ya que no es posible encontrar una función recursiva (por extensión de la definición recurrente del dominio) que lo resuelva.

Ejemplo: el recorrido por niveles del arbol.

Función $f(t)$ sobre un árbol binario, t : definición por extensión de la definición del conjunto de árboles binarios.

Base: valor de la función si t es el árbol vacío.

Recurrencia: se supone conocida la función para cada uno de los subárboles T_{izq} y T_{der} de t . Se calcula el valor final de la función suponiendo conocidos los valores anteriores.

Ejemplo: igualdad de árboles binarios.

Base: Si t_1 y t_2 son árboles binarios vacíos, son iguales.

Recurrencia: hipótesis.

- igual (t_{izq1}, t_{izq2}) } t_{izq1} y t_{der1} son los subárboles izquierdo
- igual (t_{der1}, t_{der2}) } y derecho de t_i .

Tesis: los árboles binarios t_1 y t_2 serán iguales si se cumplen:

- $t_1.\text{etiqueta}() == t_2.\text{etiqueta}()$.
 - $\text{igual}(t_{izq1}, t_{izq2}) \text{ e}$
 - $\text{igual}(t_{der1}, t_{der2})$

Estudialo bien, que
tiene pinta de importante

Ejemplo: árboles binarios isomorfos.

Base: si t_1 y t_2 son arboles binarios vacíos, son isomorfos.

Recurrencia: Hipótesis

- $\text{iso}(t_{izq1}, t_{izq2})$
 - $\text{iso}(t_{der1}, t_{der2})$
 - $\text{iso}(t_{der1}, t_{izq2})$

Este lo pregunta
seguuuuuuuuucro!

Tesis: los árboles binarios t_1 y t_2 serán isomorfos si se cumplen las condiciones:

- $t_1.\text{etiqueta}() == t_2.\text{etiqueta}()$.
 - $\text{iso}(t_{izq1}, t_{izq2})$ y $\text{iso}(t_{der1}, t_{der2})$, ó
 - $\text{iso}(t_{izq1}, t_{der2})$ y $\text{iso}(t_{der1}, t_{izq2})$.

Representación mediante vectores :

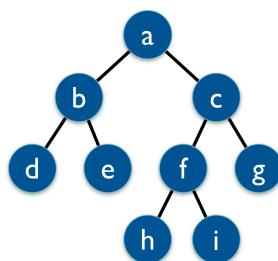
Las etiquetas de los nodos se almacenan en un vector.

Los nodos se enumeran de la siguiente manera:

Echa un vistazo
verás que guay



- Su hijo izquierdo, si tiene, está en la posición $2 * K + 1$.
 - Su hijo derecho, si tiene, se encuentra en la posición $2 * K + 2$.
 - 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:



Clases en **DIRECTO**



■ **Audio y vídeo
PROFESIONAL**



Pizarra digital
COMPARTIDA



**Máximo
10 PERSONAS**

