

# ÁRBOLES

## Conceptos sobre árboles

Joaquín Fernández-Valdivia

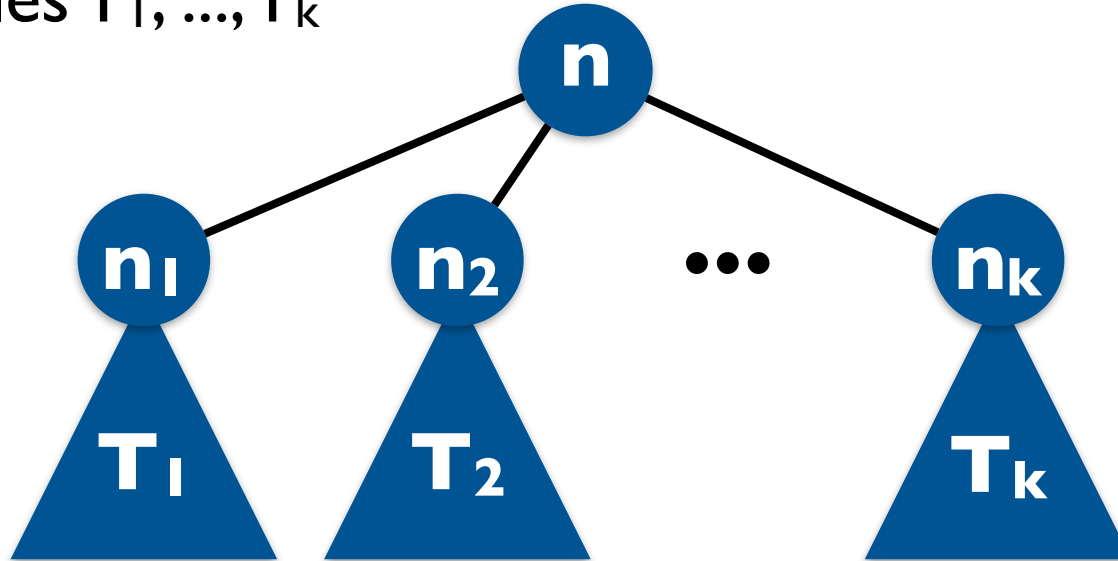
Javier Abad

Dpto. de Ciencias de la Computación e Inteligencia Artificial  
Universidad de Granada



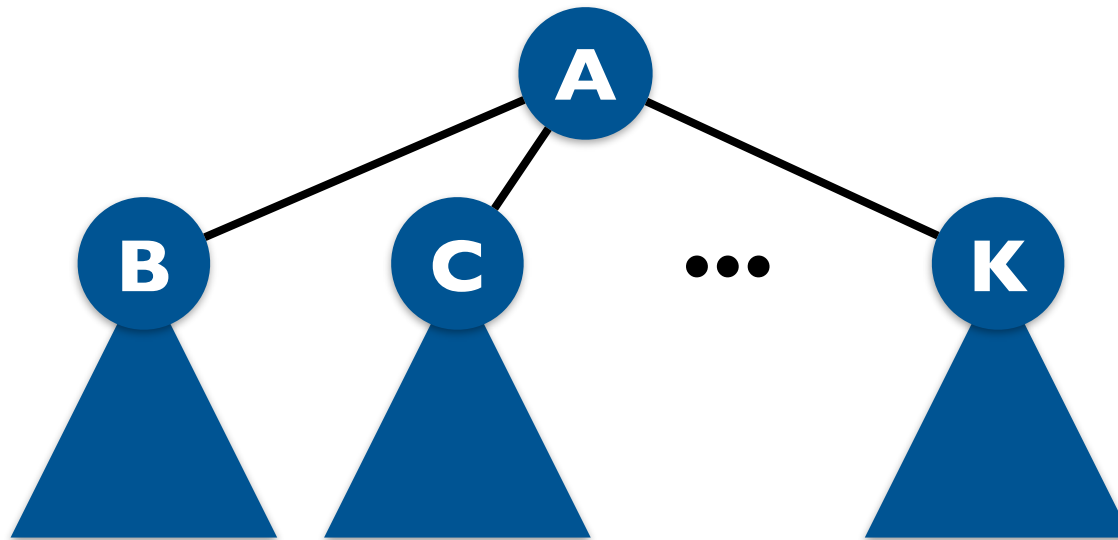
# Árbol n-ario

- 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  $T_1, \dots, T_k$  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

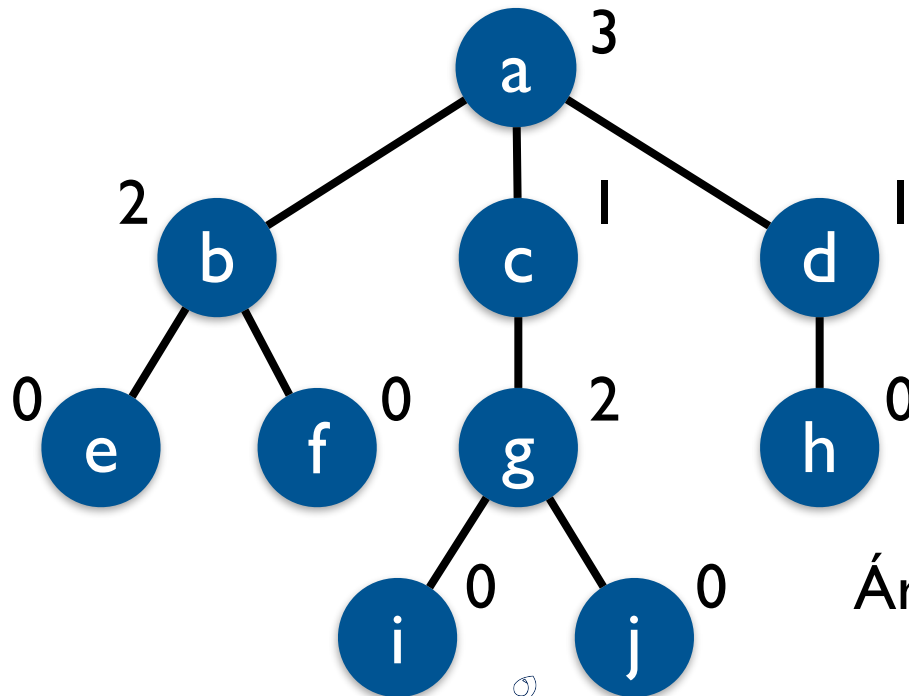
- Se dice que **un árbol está etiquetado** si todos sus nodos contienen una etiqueta



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

# Conceptos sobre árboles

- 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

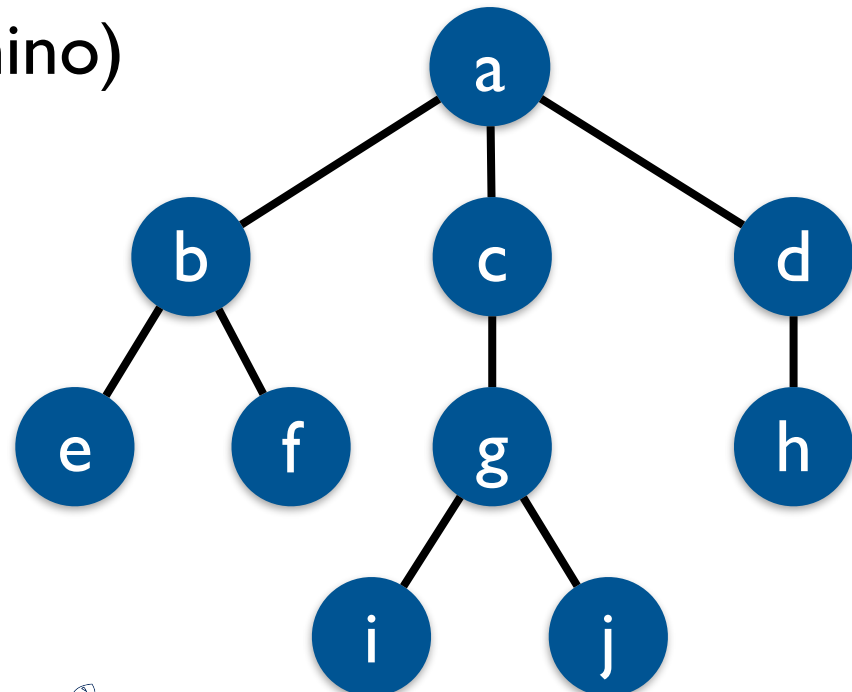


Árbol de grado 3

# Conceptos sobre árboles

- 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 (número de ejes del camino)

$\text{camino}(a,f) = \{a,b,f\}$   
Longitud 2



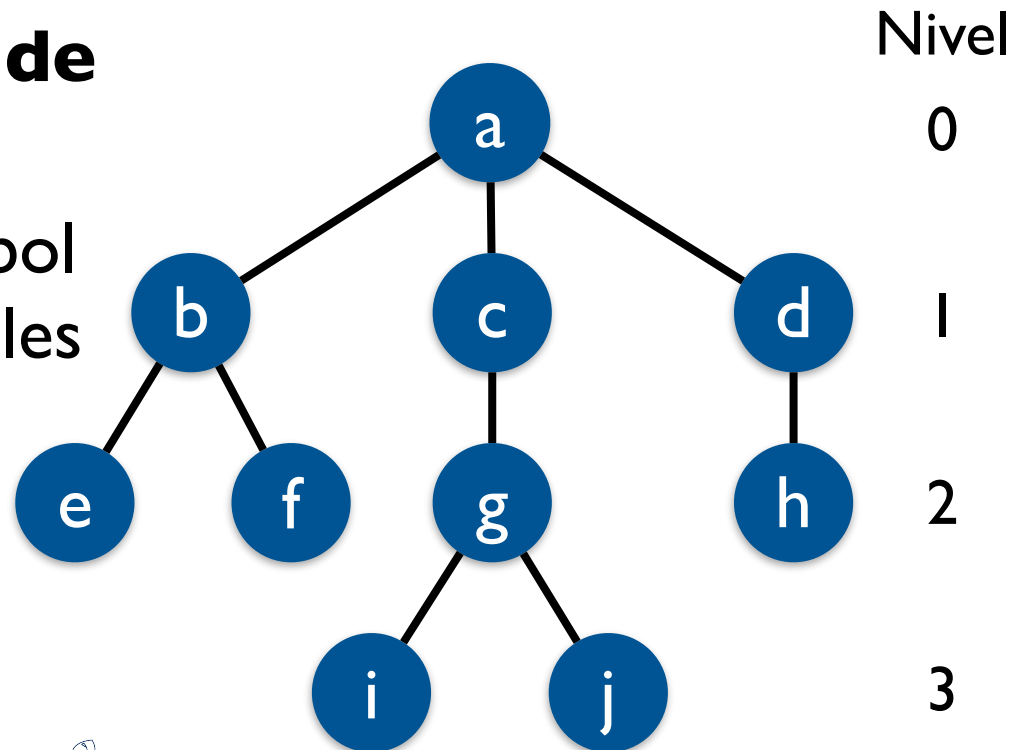
# Conceptos sobre árboles

## 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  $i+1$

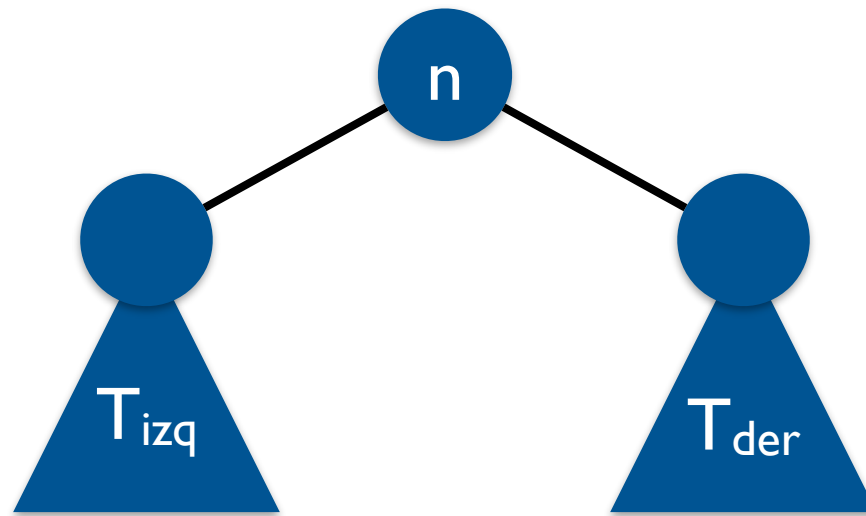
## Altura y profundidad de un árbol

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



# Árboles binarios

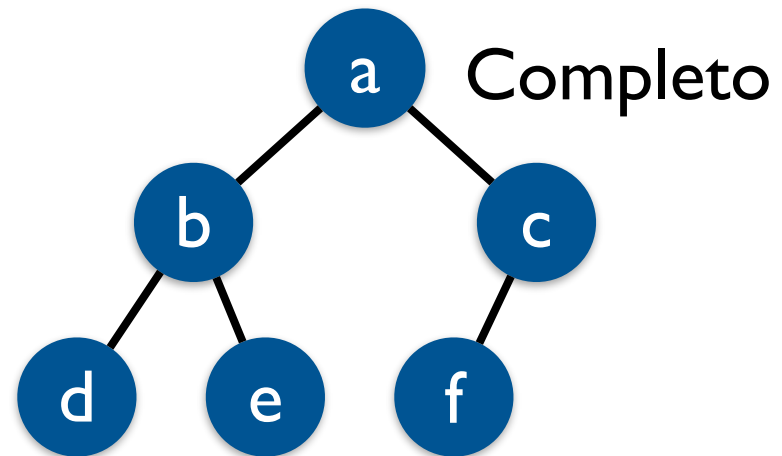
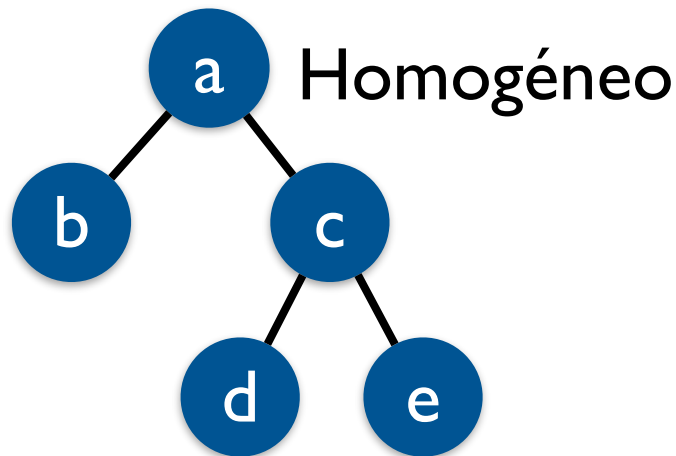
- Base: Un árbol vacío es un árbol binario
- Recurrencia: Si  $n$  es un nodo y  $T_{izq}$  y  $T_{der}$  son árboles binarios, podemos construir un nuevo árbol binario que tenga como raíz el nodo  $n$  y como subárboles  $T_{izq}$  y  $T_{der}$  (subárbol izquierdo y derecho, respectivamente)



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

# Árboles binarios

- **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

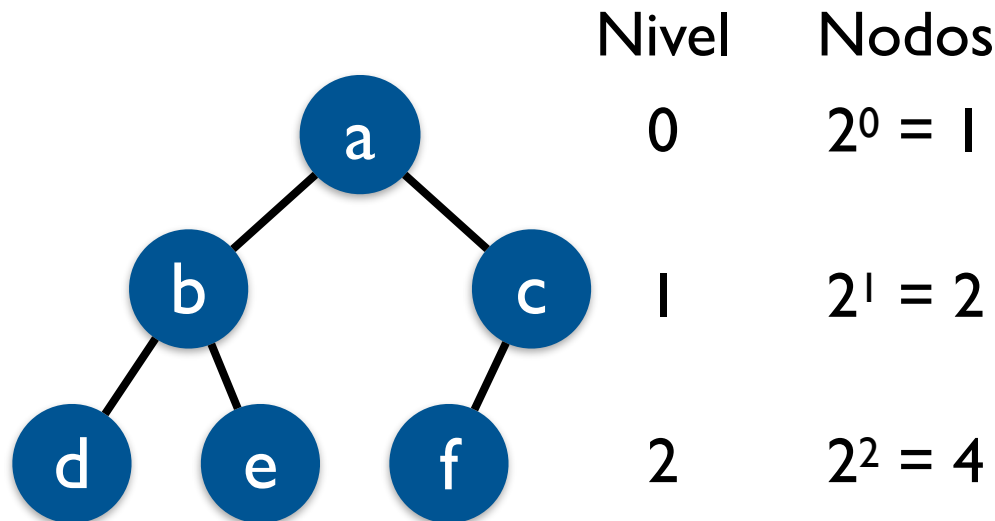


En un árbol binario completo con  $n$  nodos el camino más largo de la raíz a las hojas no atraviesa más de  $\log_2 n$  nodos



# Árboles binarios

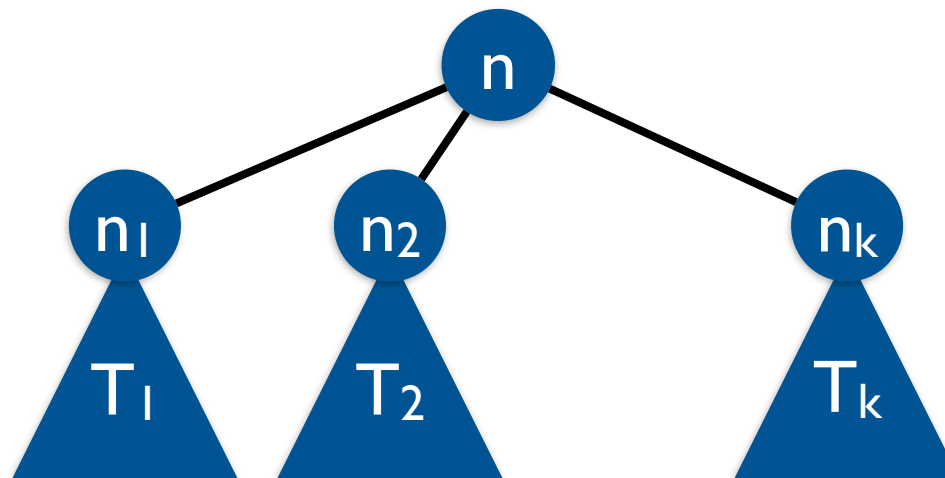
- En un árbol binario, el número máximo de nodos que puede haber en el nivel  $i$  es  $2^i$



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

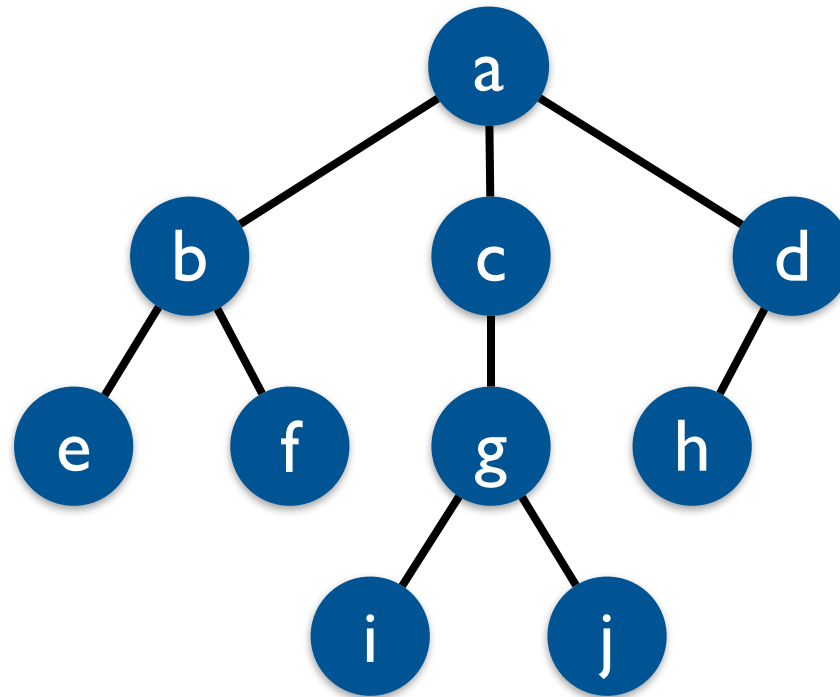
# Recorridos en árboles n-arios

- Recorridos en profundidad:
  - **Preorden:** raíz,  $\text{Pre}(T_1)$ ,  $\text{Pre}(T_2)$ , ...,  $\text{Pre}(T_k)$
  - **Inorden:**  $\text{In}(T_1)$ , raíz,  $\text{In}(T_2)$ , ...,  $\text{In}(T_k)$
  - **Postorden:**  $\text{Pos}(T_1)$ ,  $\text{Pos}(T_2)$ , ...,  $\text{Pos}(T_k)$ , raíz



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

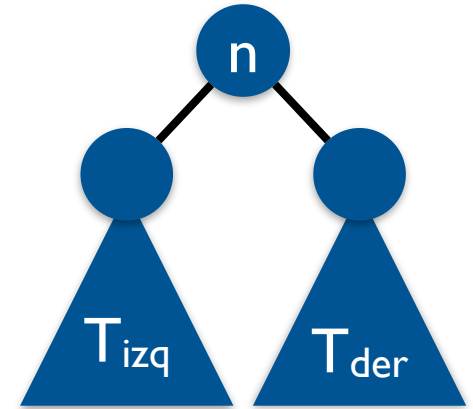
# Recorridos en árboles n-arios



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

# Recorridos en árboles binarios

- Recorridos en profundidad:
  - **Preorden:** raíz,  $\text{Pre}(T_{\text{izq}})$ ,  $\text{Pre}(T_{\text{der}})$
  - **Inorden:**  $\text{In}(T_{\text{izq}})$ , raíz,  $\text{In}(T_{\text{der}})$
  - **Postorden:**  $\text{Pos}(T_{\text{izq}})$ ,  $\text{Pos}(T_{\text{der}})$ , raíz



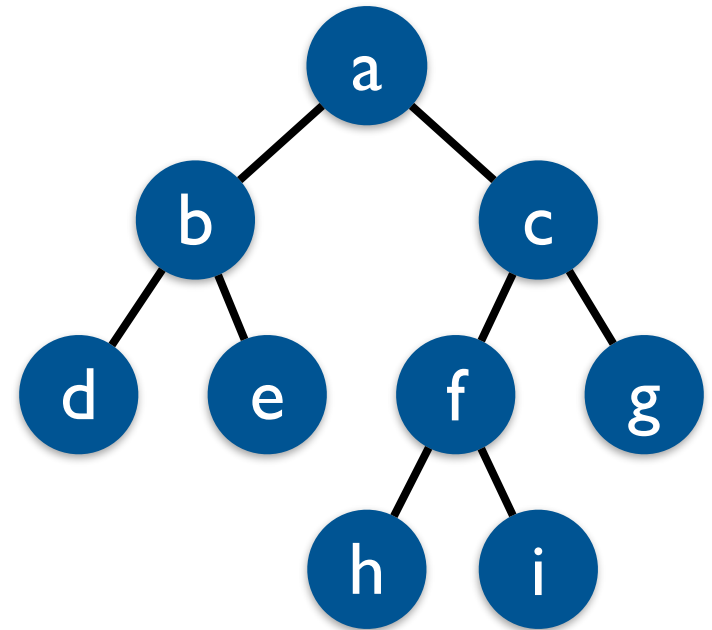
Se pueden realizar de forma recursiva, siguiendo el esquema de construcción recursivo de árboles binarios

- Recorrido en anchura:
  - **Por niveles**, de izquierda a derecha

Se realiza de forma iterativa

# Recorridos en árboles binarios

- 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

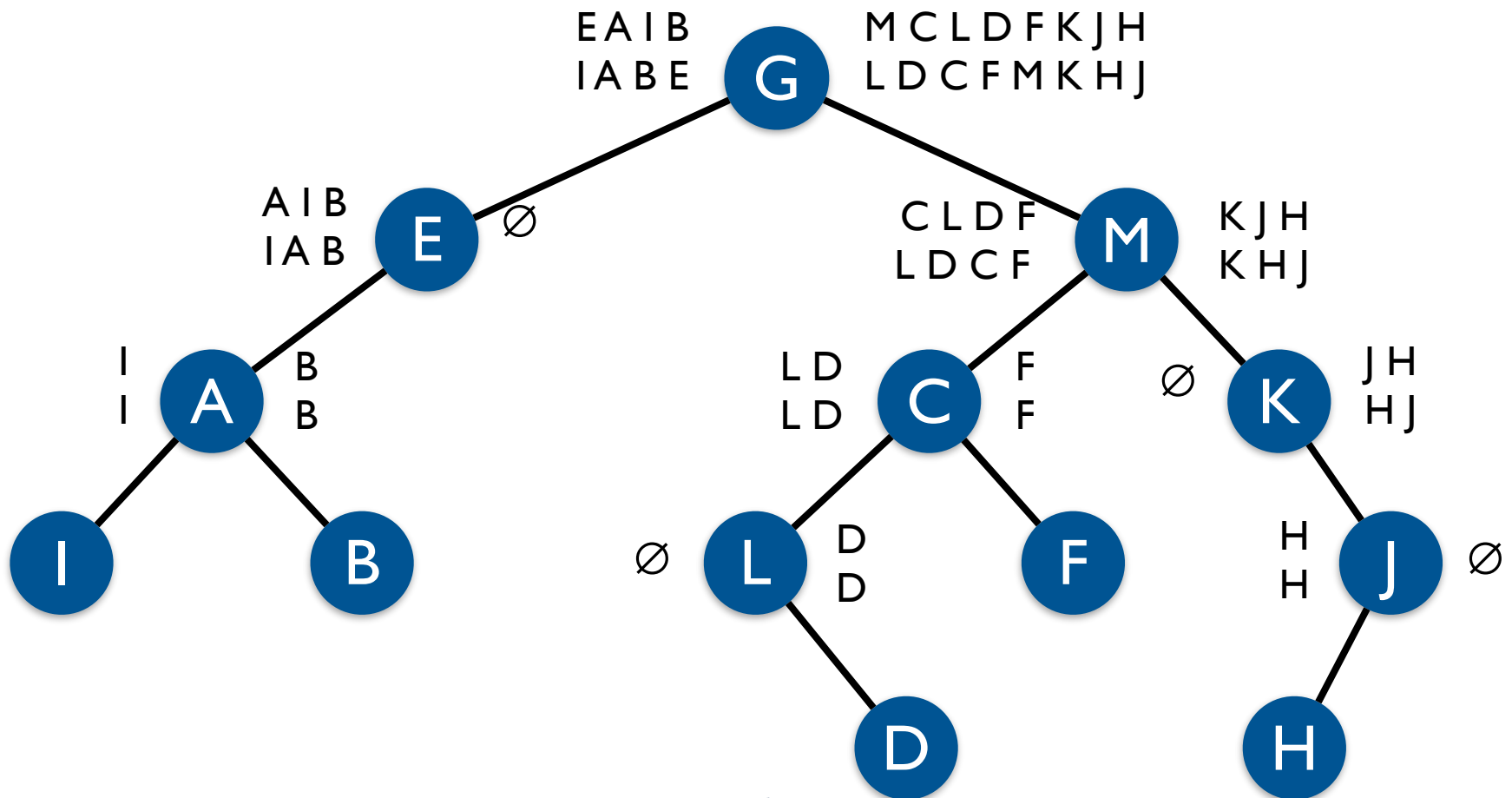


# Recorridos en árboles binarios

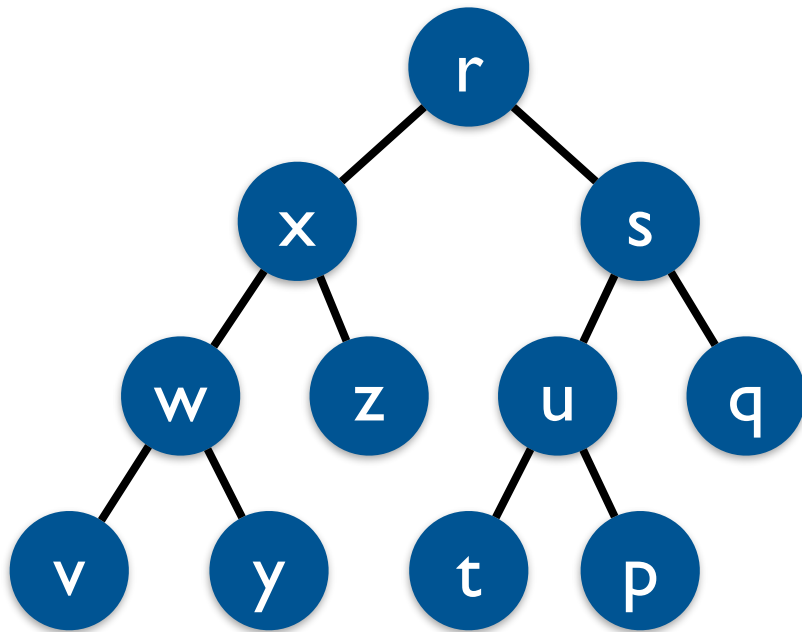
	$\text{Pre}(n) < \text{Pre}(m)$	$\text{In}(n) < \text{In}(m)$	$\text{Pos}(n) < \text{Pos}(m)$
n a la izquierda de m	✓	✓	✓
n a la derecha de m	✗	✗	✗
n descendiente de m	✗	✓✗	✓
n ancestro de m	✓	✓✗	✗

# Recorridos en árboles binarios

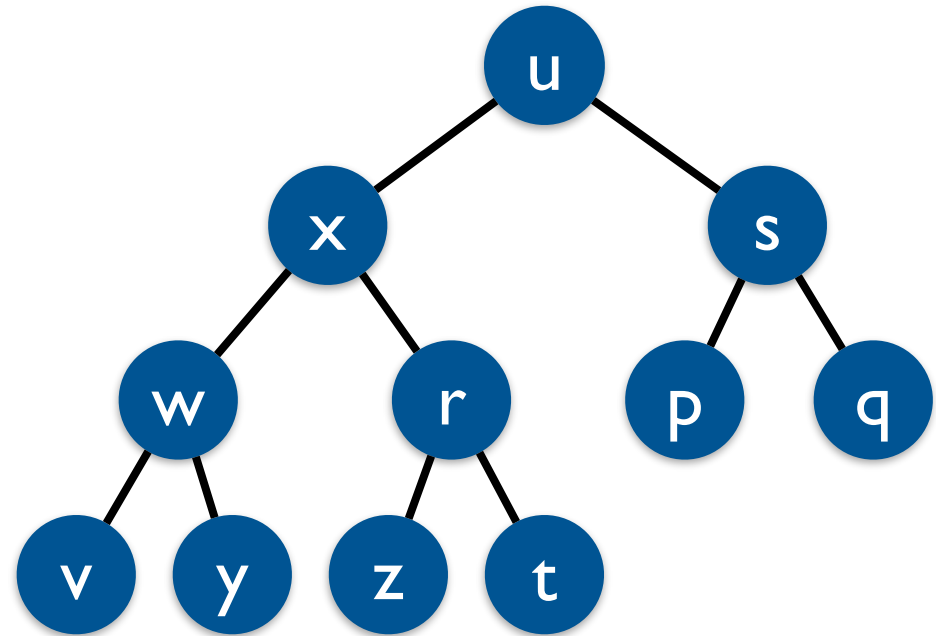
- Preorden: G E A I B M C L D F K J H
- Inorden: I A B E G L D C F M K H J



# Recorridos en árboles binarios



Inorden: v w y x z r t u p s q

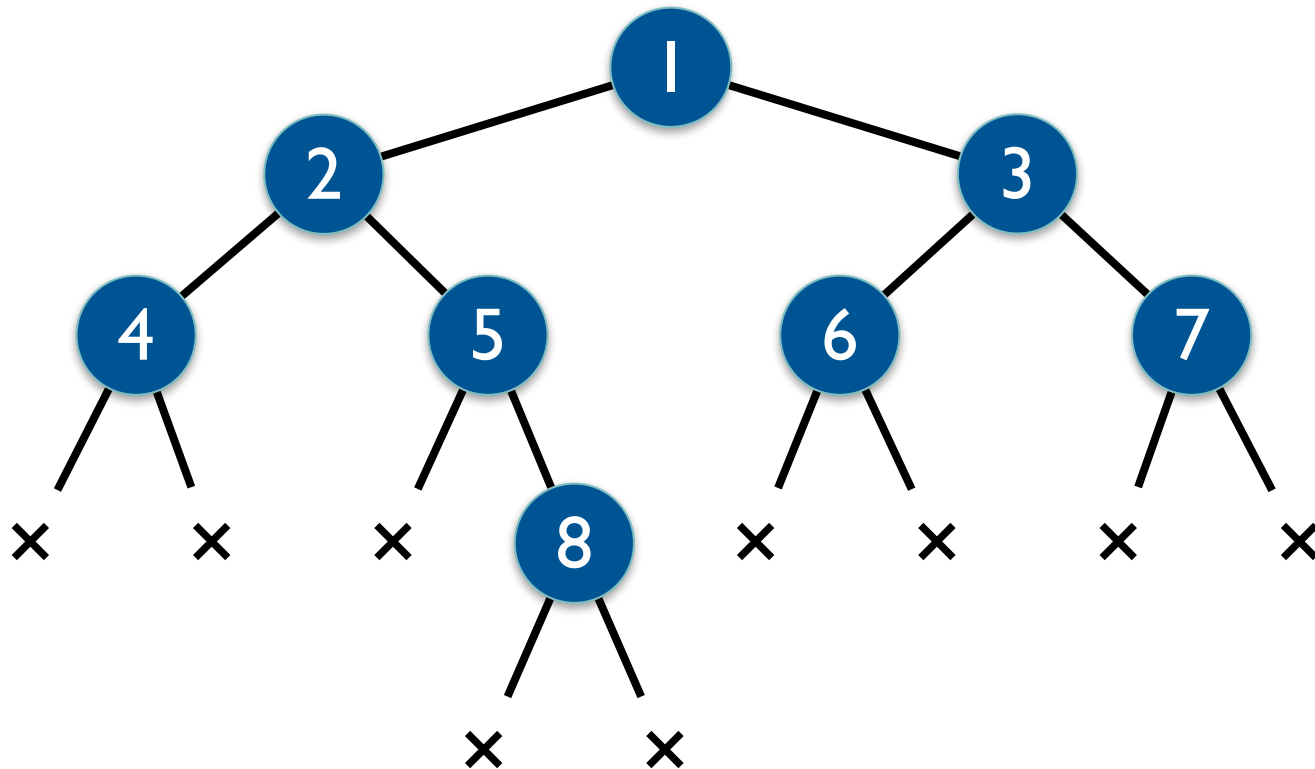


Inorden: v w y x z r t u p s q

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



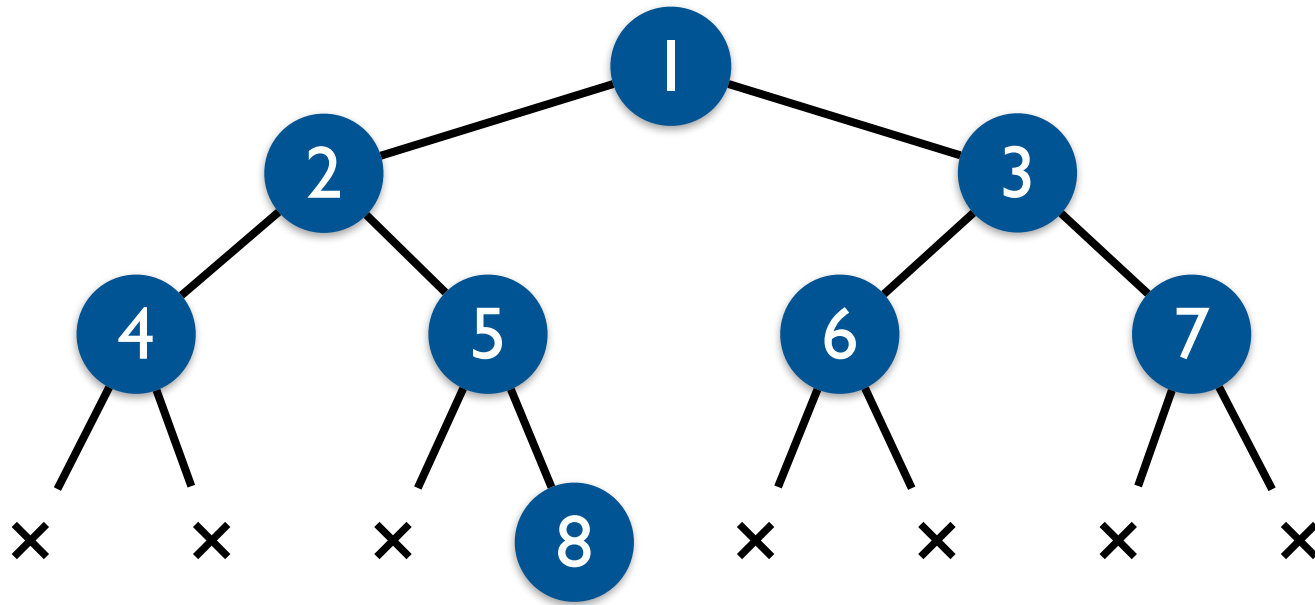
# Lectura/escritura de un árbol



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  
1 2 4 5 8 3 6 7

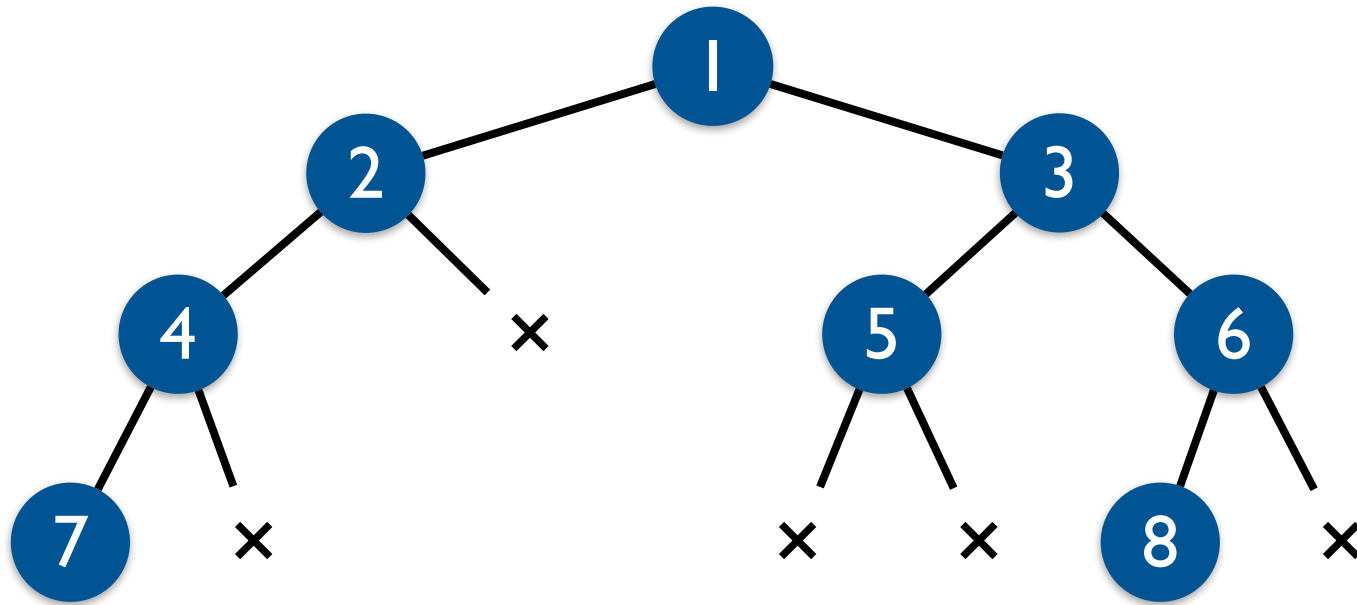
# Lectura/escritura de un árbol



Por niveles

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

# Lectura/escritura de un árbol

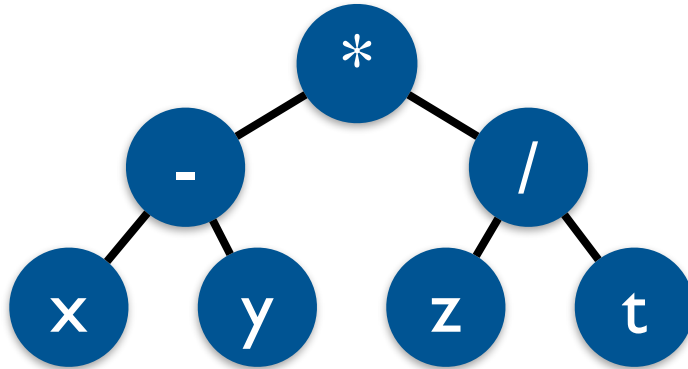
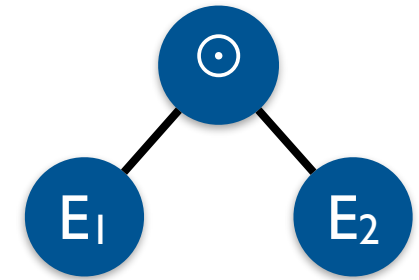


Por niveles

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



$$((x)-(y))^*((z)/(t))$$

III

$$(x-y) * (z/t) \text{ Inorden}$$

Preorden:  $*-xy/zt$  ➤ Representación prefija } No necesita  
Postorden:  $xy-zt/*$  ➤ Representación postfija } paréntesis

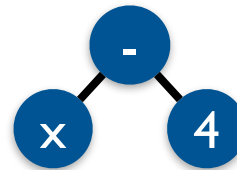
# Aplicación: árboles de expresión

- **Resolución de ambigüedades:** recorridos en preorden o postorden más
  - Nivel de cada nodo, ó
  - Número de hijos de cada nodo

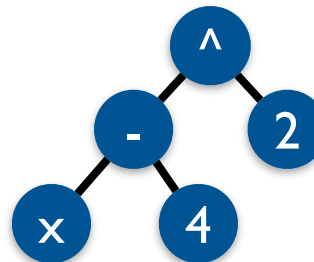
Ejemplo:  $x4-2^y2+3/*$  (postfijo)

Los operadores  $-$ ,  $^$ ,  $+$ ,  $/$  y  $*$  son binarios

$x4-2^y2+3/*$   
 $(x-4)$



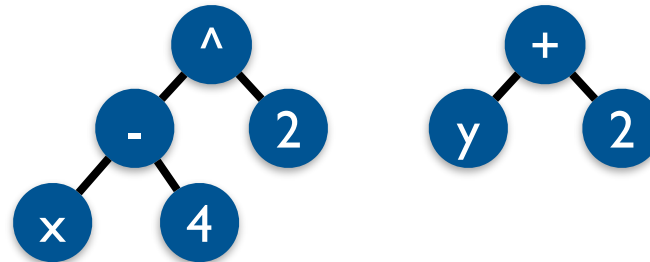
$(x-4)2^y2+3/*$   
 $((x-4)^2)$



# Aplicación: árboles de expresión

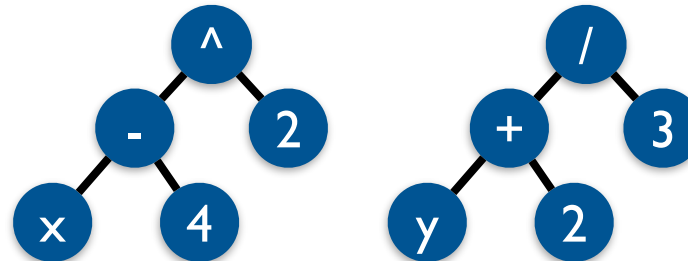
$$((x-4)^2) \text{ y } 2 + 3 / *$$

$$((x-4)^2) (y+2)$$



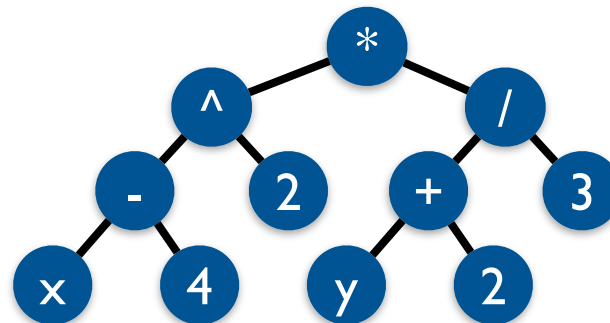
$$((x-4)^2) (y+2) 3 / *$$

$$((x-4)^2) ((y+2)/3)$$



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

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



# Aplicación: árboles de expresión

- Las notaciones prefija y posfija facilitan la evaluación automática de expresiones aritméticas
- Ejemplo:  $((15 / (7 - (1 + 1))) * 3) - (2 + (1 + 1))$

-	*	/	15	-	7	+	1	1	3	+	2	+	1	1	=
-	*	/	15	-	7		2		3	+	2	+	1	1	=
-	*	/	15			5			3	+	2	+	1	1	=
-	*			3					3	+	2	+	1	1	=
-					9					+	2	+	1	1	=
-					9					+	2		2		=
-					9						4				=
									5						

# Aplicación: árboles de expresión

$$\underbrace{[(a+b) + (c * (d+e) + f)]}_{E_1} * \underbrace{(g+h)}_{E_2} \Rightarrow * E_1 E_2$$

$$\underbrace{[(a+b)]}_{E_{11}} + \underbrace{(c * (d+e) + f)}_{E_{12}} \Rightarrow * + E_{11} E_{12} E_2$$

$$\left. \begin{array}{l} E_{11} \equiv +ab \Rightarrow * + + a b E_{12} E_2 \\ E_{12} \equiv [(c * (d+e) + f)] \Rightarrow + E_{121} E_{122} \end{array} \right\} * + + a b + E_{121} E_{122} E_2$$

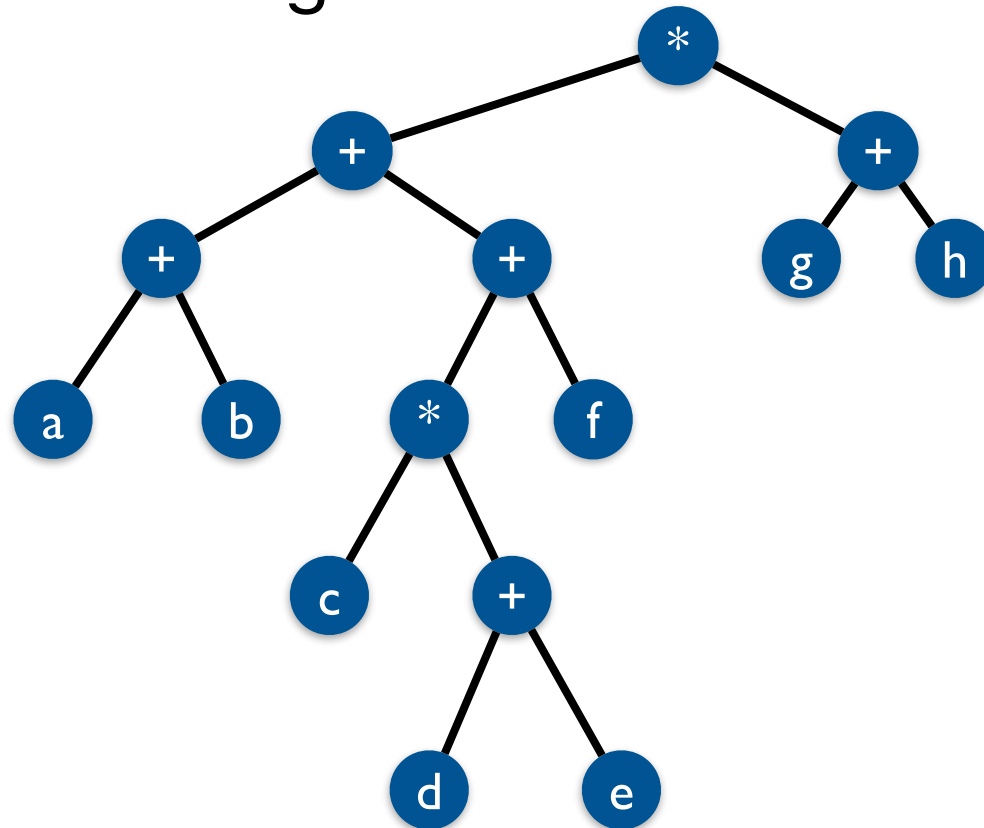
$$E_{121} \equiv \underbrace{c}_{E_{1211}} * \underbrace{(d+e)}_{E_{1212}} \Rightarrow * c E_{1212} \equiv * c + de$$

$$\left. \begin{array}{l} * + + a b + * c + d e f E_2 \\ E_2 \equiv (g+h) \equiv + g h \end{array} \right\} * + + a b + * c + d e f + g h$$



# Aplicación: árboles de expresión

\*++ab+\*c+def+gh

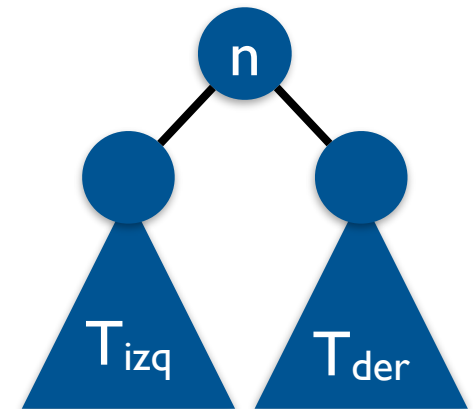


# Árboles binarios

- **Definición de funciones en árboles:**  
generalmente la forma más simple de definir una función sobre un árbol es trasladar la definición recurrente (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. Ej: el recorrido por niveles del árbol

# Árboles binarios

- **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

# Árboles binarios

Ejemplo: igualdad de árboles binarios

- Base: Si  $t_1$  y  $t_2$  son árboles binarios vacíos, son iguales
  - Recurrencia: Hipótesis
    - $\text{igual}(t_{\text{izq1}}, t_{\text{izq2}})$
    - $\text{igual}(t_{\text{der1}}, t_{\text{der2}})$
- $t_{\text{izqi}}$  y  $t_{\text{deri}}$  son los subárboles izquierdo y derecho de  $t_i$
- Tesis: Los árboles binarios  $t_1$  y  $t_2$  serán iguales si se cumplen las condiciones:
    - $t_1.\text{etiqueta()} == t_2.\text{etiqueta()}$
    - $\text{igual}(t_{\text{izq1}}, t_{\text{izq2}})$ , e
    - $\text{igual}(t_{\text{der1}}, t_{\text{der2}})$

# Árboles binarios

Ejemplo: altura de un árbol binario

- Base: Si  $t$  es un árbol binario vacío, su altura es 0
  - Recurrencia: Hipótesis
    - $\text{altura}(t_{\text{izq}}) = a_{\text{izq}}$
    - $\text{altura}(t_{\text{der}}) = a_{\text{der}}$
- $t_{\text{izq}}$  y  $t_{\text{der}}$  son los subárboles izquierdo y derecho de  $t$
- Tesis: la altura se calcula como:
$$1 + \text{máximo}(a_{\text{izq}}, a_{\text{der}})$$
  - Ejercicios:
    - Contar el número de nodos de un árbol
    - Calcular el grado de un árbol

# Árboles binarios

Ejemplo: árboles binarios isomorfos

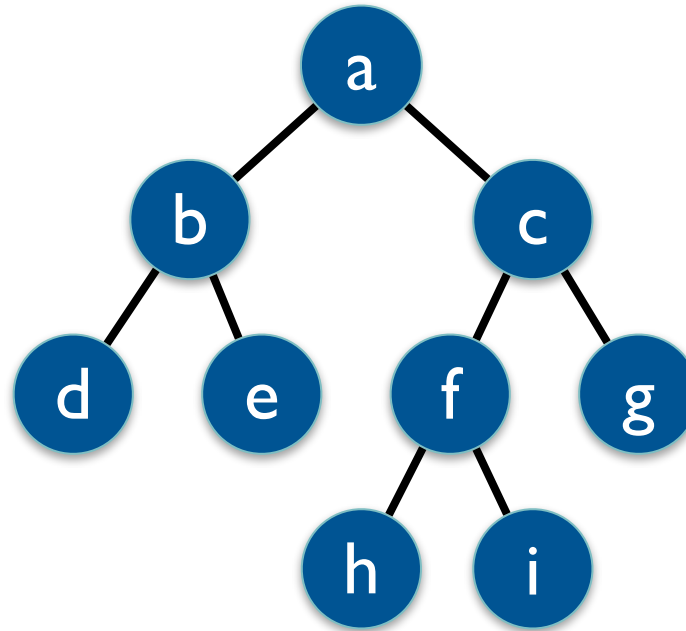
- Base: Si  $t_1$  y  $t_2$  son árboles binarios vacíos, son isomorfos
- Recurrencia: Hipótesis
  - $iso(t_{izq1}, t_{izq2})$  •  $iso(t_{izq1}, t_{der2})$
  - $iso(t_{der1}, t_{der2})$  •  $iso(t_{der1}, t_{izq2})$
- Tesis: Los árboles binarios  $t_1$  y  $t_2$  serán isomorfos si se cumplen las condiciones:
  - $t_1.etiqueta() == t_2.etiqueta()$
  - $iso(t_{izq1}, t_{izq2})$  e  $iso(t_{der1}, t_{der2})$ , ó
  - $iso(t_{izq1}, t_{der2})$  e  $iso(t_{der1}, t_{izq2})$

$t_{izqi}$  y  $t_{deri}$  son los subárboles izquierdo y derecho de  $t_i$

# 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  $(k-1)/2$

# Representación mediante vectores



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

