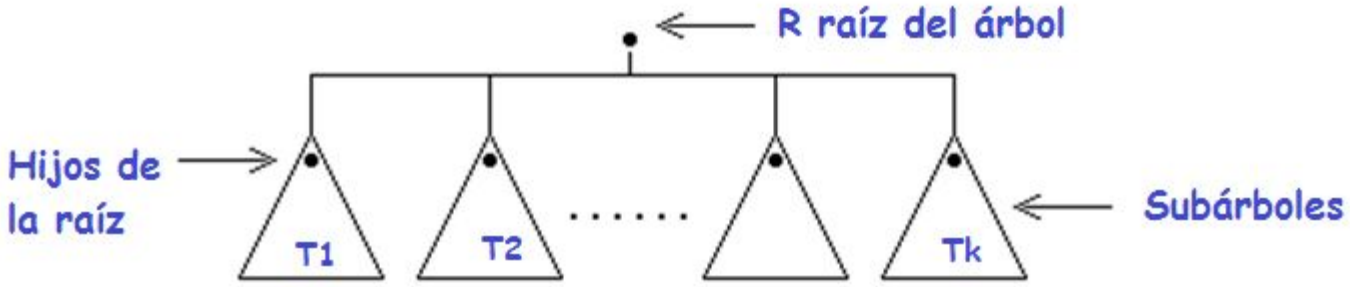
# Árboles Generales

# Agenda

- Definición
- Descripción y terminología
- Ejemplos
- Representaciones
- Recorridos

# Definición

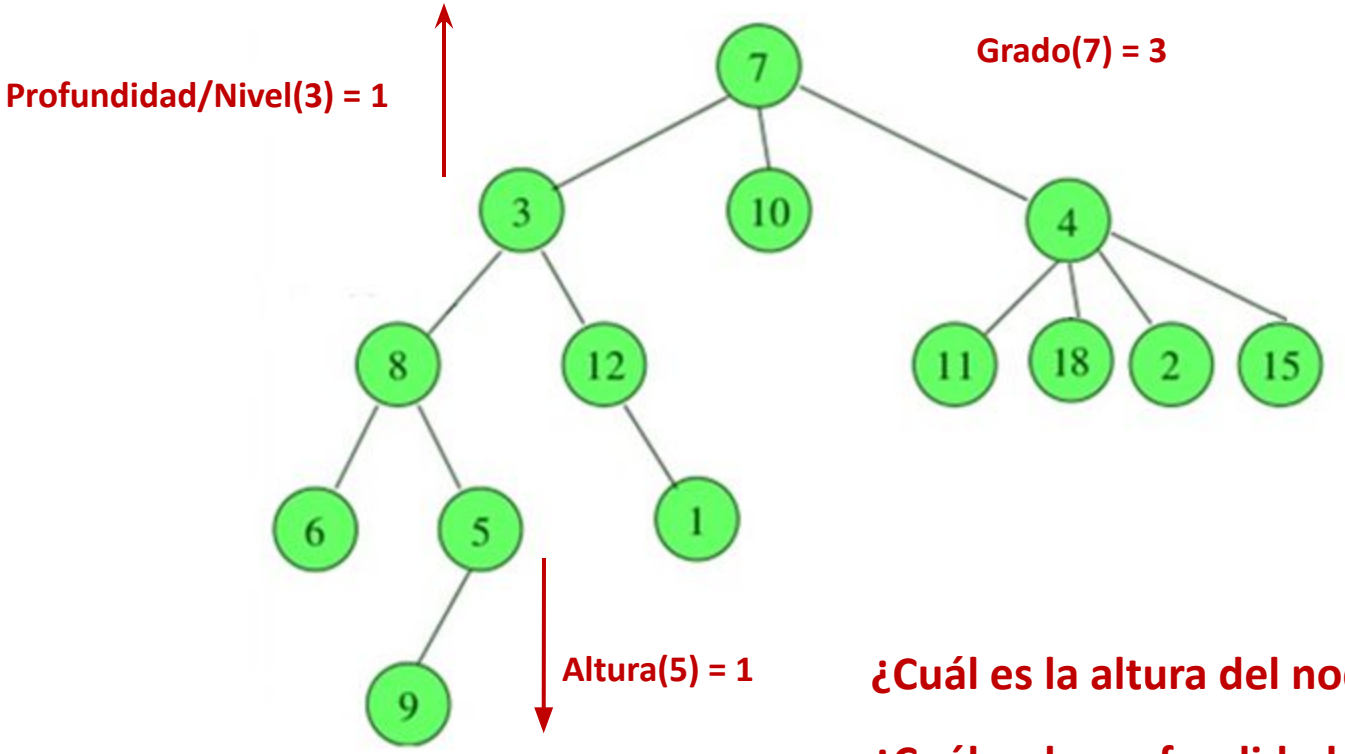
- *Un árbol es una colección de nodos, tal que:*
  - *puede estar vacía. (Árbol vacío)*
  - *puede estar formada por un nodo distinguido R, llamado **raíz** y un conjunto de árboles $T_1$, $T_2$, ....$T_k$, $k \geq 0$ (subárboles), donde la raíz de cada subárbol $T_i$ está conectado a R por medio de una arista*



# Descripción y terminología

- ***Grado*** de $n_i$ es el número de hijos del nodo $n_i$ .
  - Grado del árbol es el grado del nodo con mayor grado.
- ***Altura*** de $n_i$ es la longitud del camino más largo desde $n_i$ hasta una hoja.
  - Las hojas tienen altura **cero**.
  - La altura de un árbol es la altura del nodo raíz.
- ***Profundidad / Nivel***: de $n_i$ es la longitud del único camino desde la raíz hasta $n_i$.
  - La raíz tiene profundidad o nivel **cero**.

# Descripción y terminología



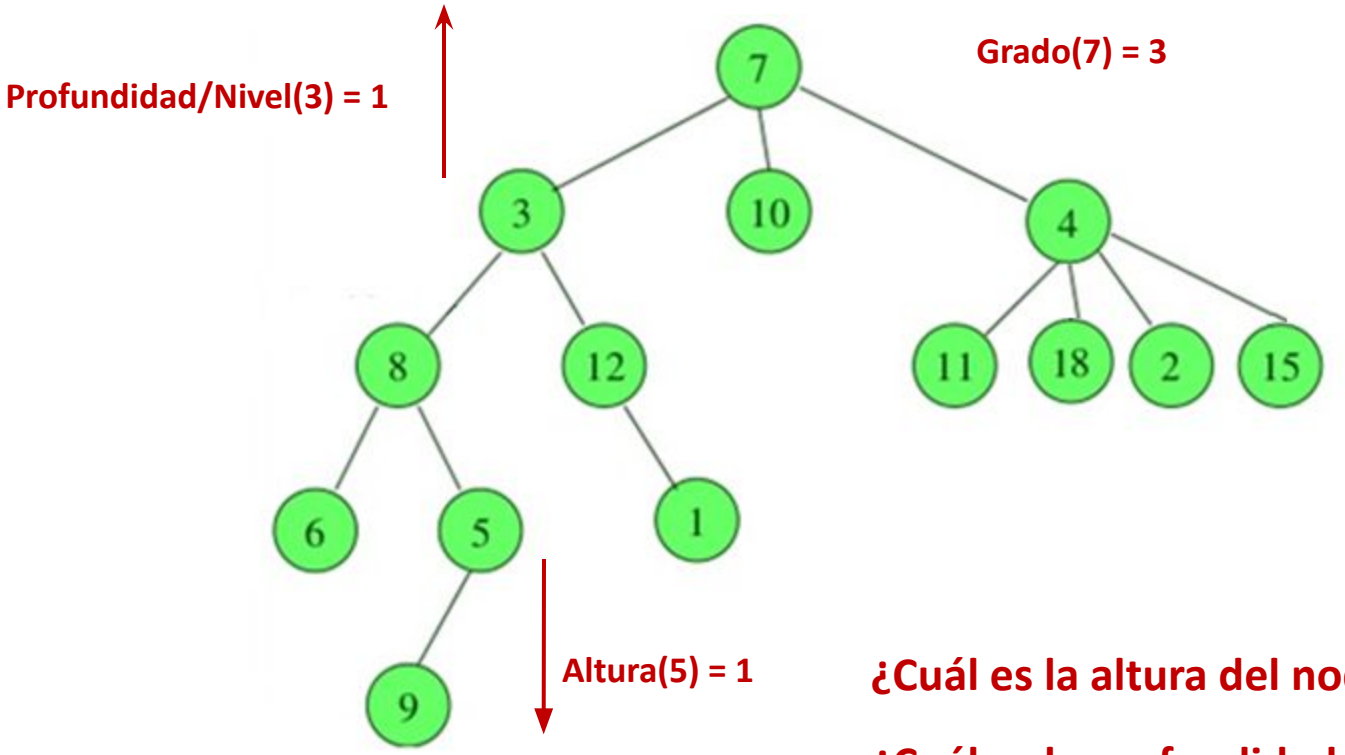

**¿Cuál es la altura del nodo 3?**

**¿Cuál es la profundidad del nodo 12?**

**¿Cuál es el grado del árbol?**

**¿Cuál es la altura del árbol?**

# Descripción y terminología



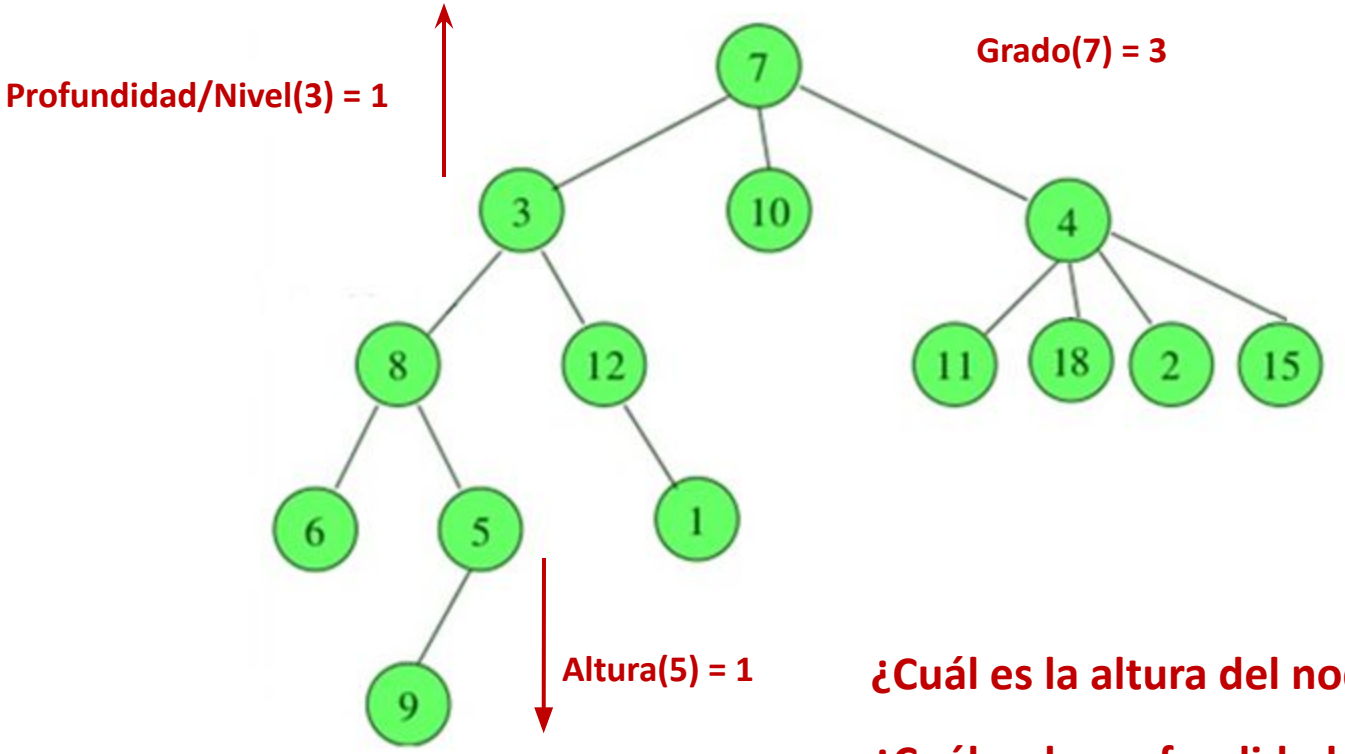

**¿Cuál es la altura del nodo 3?** **3**

**¿Cuál es la profundidad del nodo 12?** **2**

**¿Cuál es el grado del árbol?**

**¿Cuál es la altura del árbol?**

# Descripción y terminología




**¿Cuál es la altura del nodo 3?** 3

**¿Cuál es la profundidad del nodo 12?** 2

**¿Cuál es el grado del árbol?** 4

**¿Cuál es la altura del árbol?** 4

# Descripción y terminología

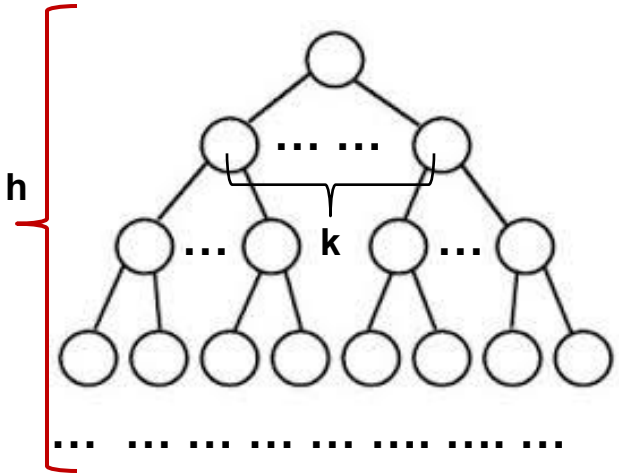
- ***Árbol lleno***: Dado un árbol T de grado ***k*** y altura ***h***, diremos que T es *lleno* si cada nodo interno tiene grado k y todas las hojas están en el mismo nivel (h).

  Es decir, recursivamente, T es *lleno* si :
  1.- T es un nodo simple ( árbol lleno de altura 0), o
  2.- T es de altura h y todos sus sub-árboles son llenos de altura h-1.

# Descripción y terminología

- ***Árbol completo***: Dado un árbol T de grado ***k*** y altura ***h***, diremos que T es *completo* si es lleno de altura h-1 y el nivel h se completa de izquierda a derecha.
- ***Cantidad de nodos en un árbol <u>lleno</u>***:

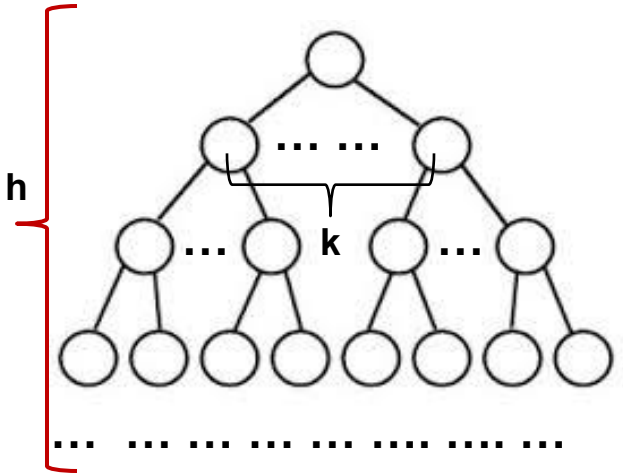
  Sea T un árbol lleno de grado ***k*** y altura ***h***, la cantidad de nodos N es $\mathbf{(k^{h+1} - 1) / (k-1)}$ ya que:



# Descripción y terminología

- ***Árbol completo***: Dado un árbol T de grado ***k*** y altura ***h***, diremos que T es *completo* si es lleno de altura h-1 y el nivel h se completa de izquierda a derecha.
- ***Cantidad de nodos en un árbol <u>lleno</u>***:

  Sea T un árbol lleno de grado ***k*** y altura ***h***, la cantidad de nodos N es **$(k^{h+1} - 1) / (k-1)$** ya que:




Nivel 0 -> $k^0$ nodos

Nivel 1 -> $k^1$ nodos

Nivel 2 -> $k^2$ nodos

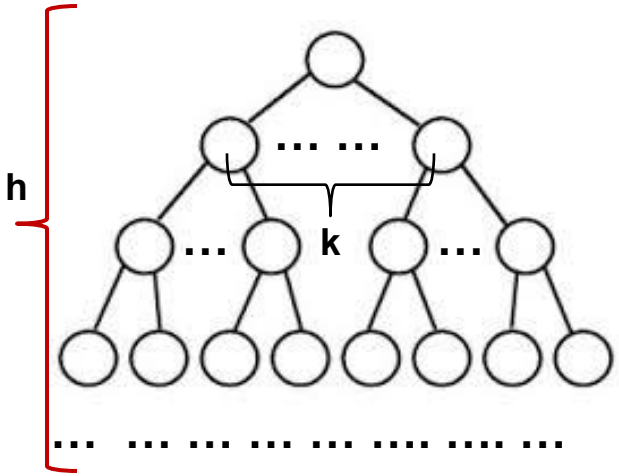
Nivel 3 -> $k^3$ nodos

.....

# Descripción y terminología

- ***Árbol completo***: Dado un árbol T de grado ***k*** y altura ***h***, diremos que T es *completo* si es lleno de altura h-1 y el nivel h se completa de izquierda a derecha.
- ***Cantidad de nodos en un árbol <u>lleno</u>***:

  Sea T un árbol lleno de grado ***k*** y altura ***h***, la cantidad de nodos N es **$(k^{h+1} - 1) / (k-1)$** ya que:




Nivel 0 -> $k^0$ nodos

Nivel 1 -> $k^1$ nodos

Nivel 2 -> $k^2$ nodos

Nivel 3 -> $k^3$ nodos

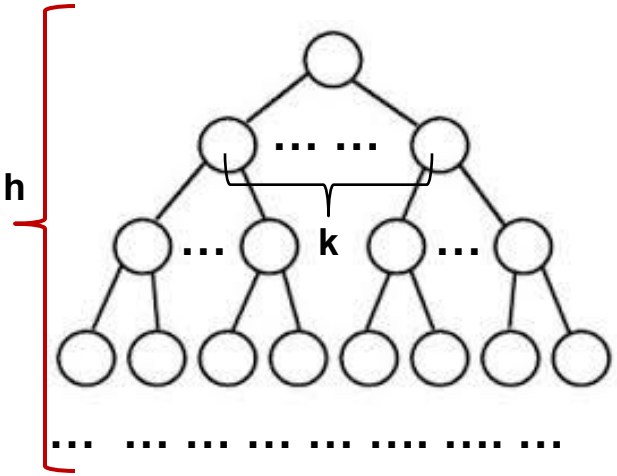
.....

$$N = k^0 + k^1 + k^2 + k^3 + \ldots + k^h$$

# Descripción y terminología

- ***Árbol completo***: Dado un árbol T de grado ***k*** y altura ***h***, diremos que T es *completo* si es lleno de altura h-1 y el nivel h se completa de izquierda a derecha.
- ***Cantidad de nodos en un árbol <u>lleno</u>***:

  Sea T un árbol lleno de grado ***k*** y altura ***h***, la cantidad de nodos N es **$(k^{h+1} - 1) / (k-1)$** ya que:




Nivel 0 -> $k^0$ nodos

Nivel 1 -> $k^1$ nodos

Nivel 2 -> $k^2$ nodos

Nivel 3 -> $k^3$ nodos

.....

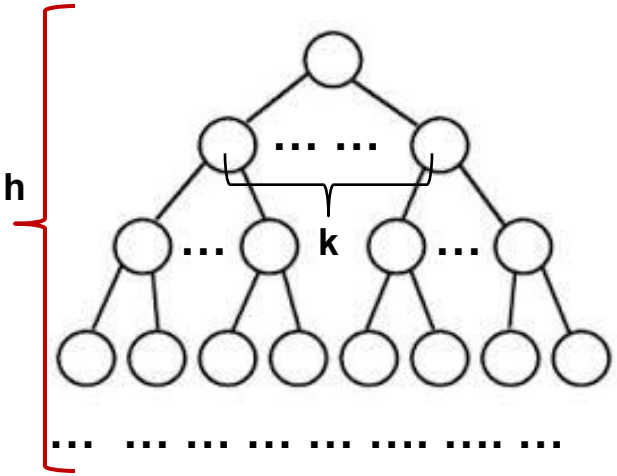
**$N = k^0 + k^1 + k^2 + k^3 + \ldots + k^h$**

La suma de los términos de una serie geométrica de razón k es:

# Descripción y terminología

- ***Árbol completo***: Dado un árbol T de grado ***k*** y altura ***h***, diremos que T es *completo* si es lleno de altura h-1 y el nivel h se completa de izquierda a derecha.
- ***Cantidad de nodos en un árbol lleno***:

  Sea T un árbol lleno de grado ***k*** y altura ***h***, la cantidad de nodos N es **$(k^{h+1} - 1) / (k-1)$** ya que:




Nivel 0 -> $k^0$ nodos

Nivel 1 -> $k^1$ nodos

Nivel 2 -> $k^2$ nodos

Nivel 3 -> $k^3$ nodos

.....

**$N = k^0 + k^1 + k^2 + k^3 + \ldots + k^h$**

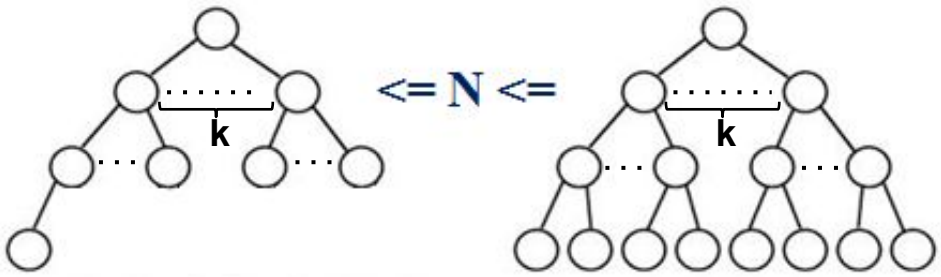
La suma de los términos de una serie geométrica de razón k es:

**$(k^{h+1} - 1) / (k-1)$**

# Descripción y terminología

- ***Cantidad de nodos en un árbol <u>completo</u>***:

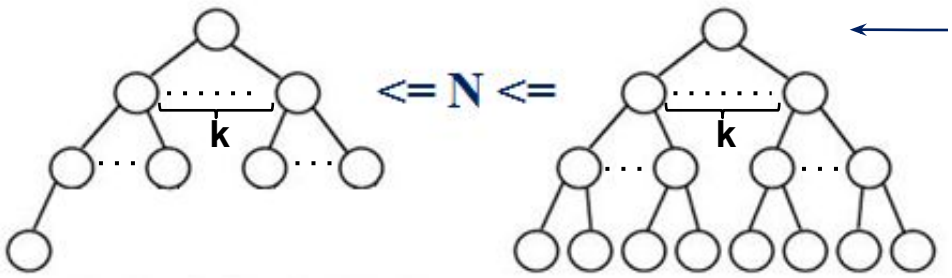
Sea T un árbol completo de grado ***k*** y altura ***h***, la cantidad de nodos N varía entre $(k^{h}+k-2) / (k-1)$ y $(k^{h+1} - 1) / (k-1)$ ya que …



# Descripción y terminología

- ***Cantidad de nodos en un árbol <u>completo</u>***:

Sea T un árbol completo de grado ***k*** y altura ***h***, la cantidad de nodos N varía entre $(k^{h}+k-2) / (k-1)$ y $(k^{h+1} - 1) / (k-1)$ ya que …



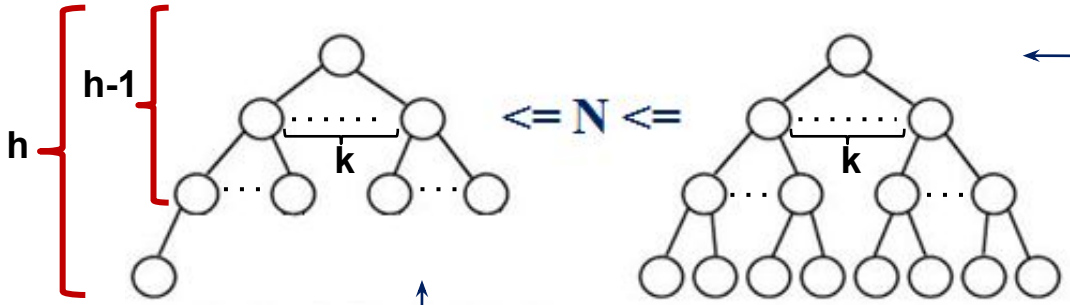

- Si el árbol es lleno

  $\mathbf{N} = (k^{h+1}-1)/(k-1)$

# Descripción y terminología

- ***Cantidad de nodos en un árbol <u>completo</u>***:

Sea T un árbol completo de grado ***k*** y altura ***h***, la cantidad de nodos N varía entre $(k^{h}+k-2)/(k-1)$ y $(k^{h+1}-1)/(k-1)$ ya que …

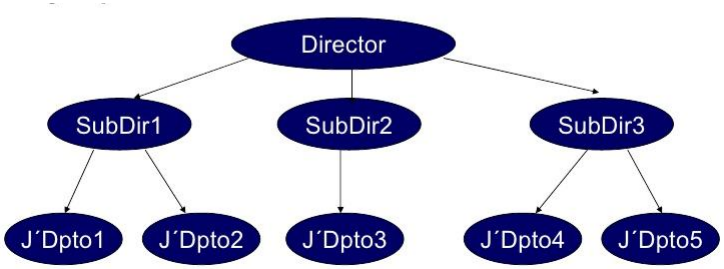


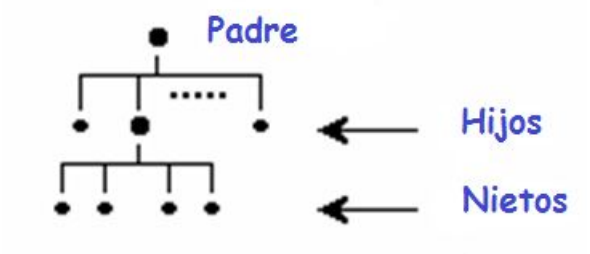

- Si el árbol es lleno
  $\mathbf{N} = (k^{h+1}-1)/(k-1)$

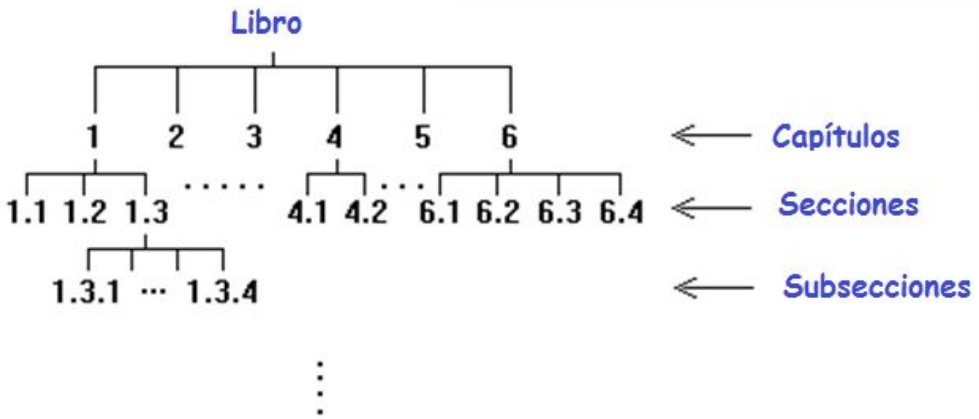
- Si no, el árbol es lleno en la altura *h-1* y tiene por lo menos un nodo en el nivel *h:*
  $\mathbf{N} = (k^{h-1+1}-1)/(k-1)+1=(k^{h}+k-2)/(k-1)$

# Ejemplos

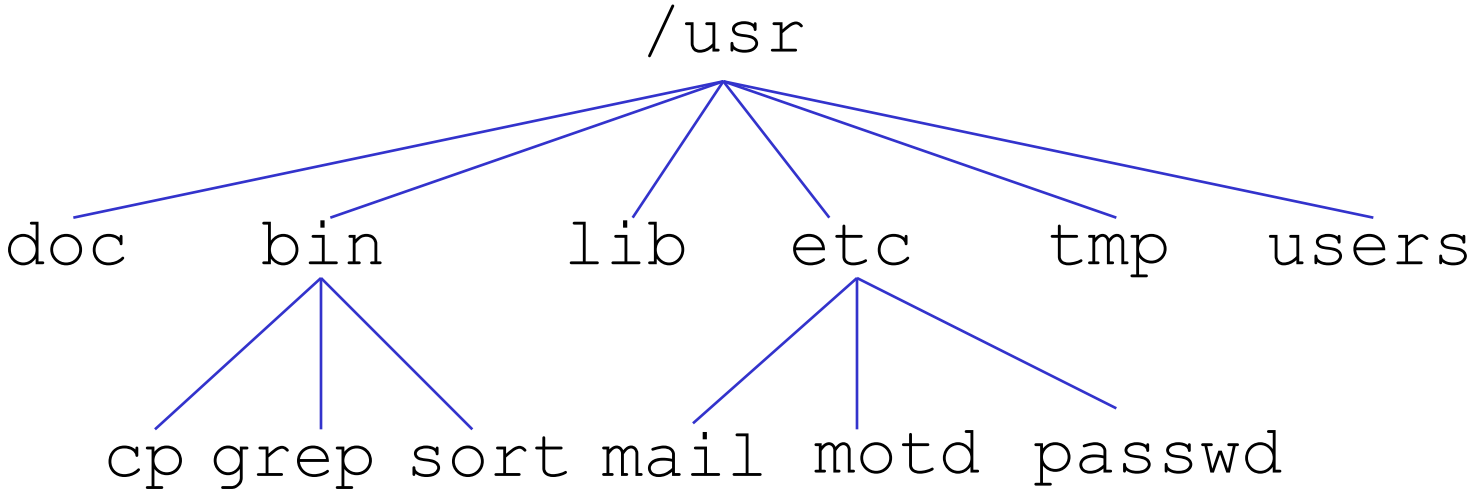
- ✔ Organigrama de una empresa
- ✔ Árboles genealógicos
- ✔ Taxonomía que clasifica organismos
- ✔ Sistemas de archivos
- ✔ Organización de un libro en capítulos y secciones



# Ejemplo: Sistema de archivos



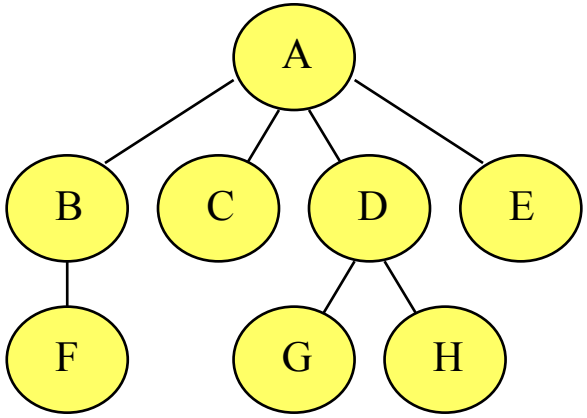
# Representaciones

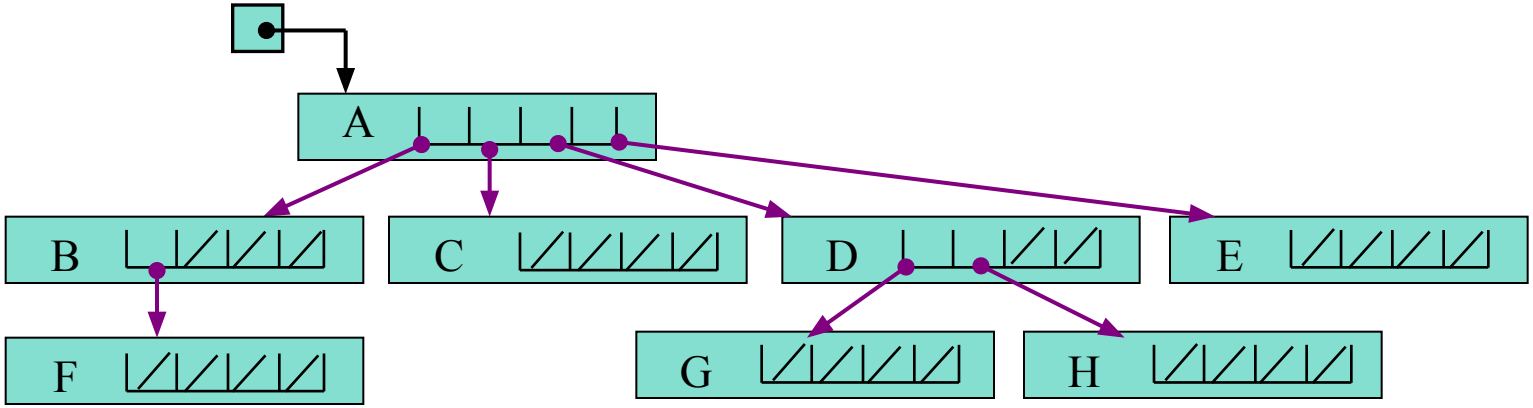
- ✔ Lista de hijos
  - Cada nodo tiene:
    - Información propia del nodo
    - Una lista de todos sus hijos
- ✔ Hijo más izquierdo y hermano derecho
  - Cada nodo tiene:
    - Información propia del nodo
    - Referencia al hijo más izquierdo
    - Referencia al hermano derecho

# Representación: Lista de hijos

- ✔ La lista de hijos, puede estar implementada a través de:
  - Arreglos
    - Desventaja: espacio ocupado
  - Listas dinámicas
    - Mayor flexibilidad en el uso

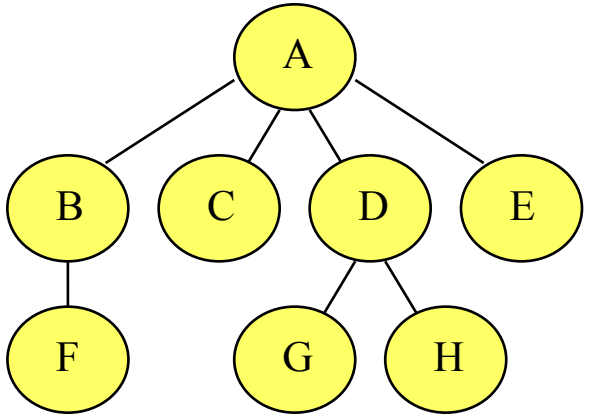
# Representación: Lista de hijos
## Implementada con Arreglos

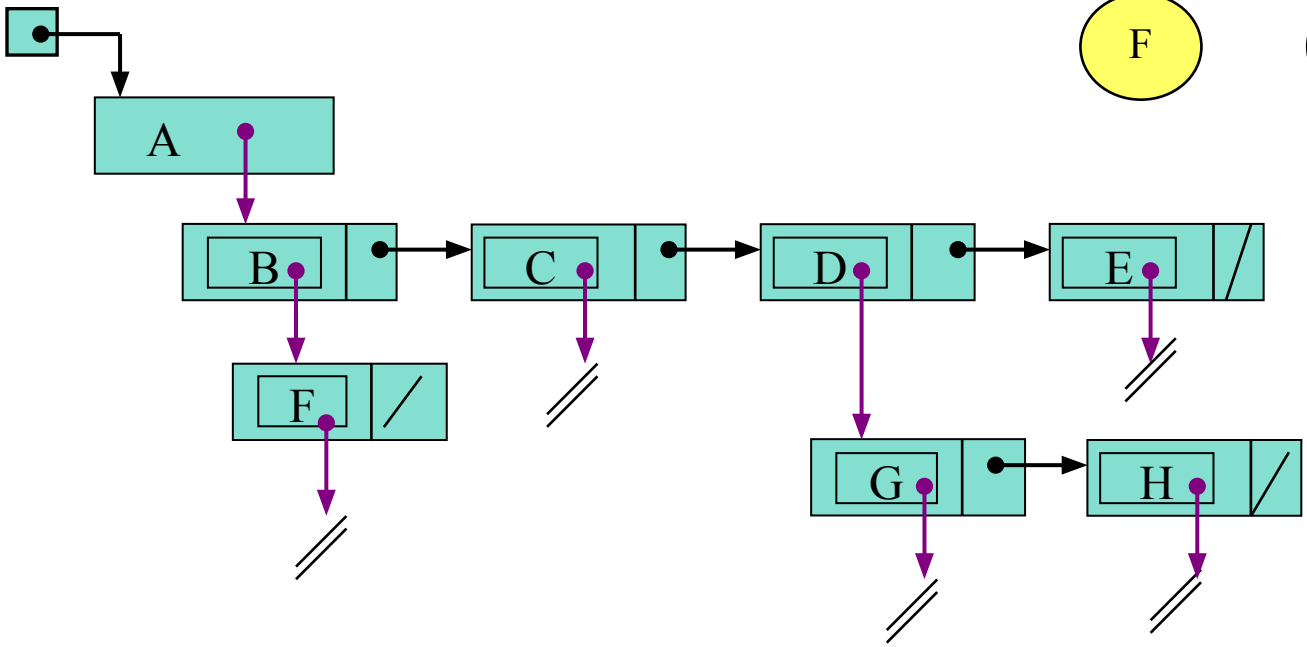


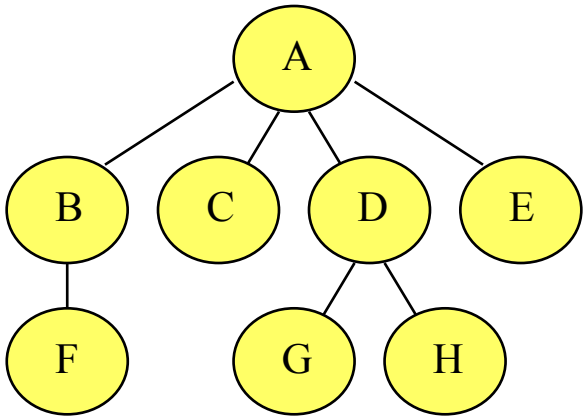
# Representación: Lista de hijos

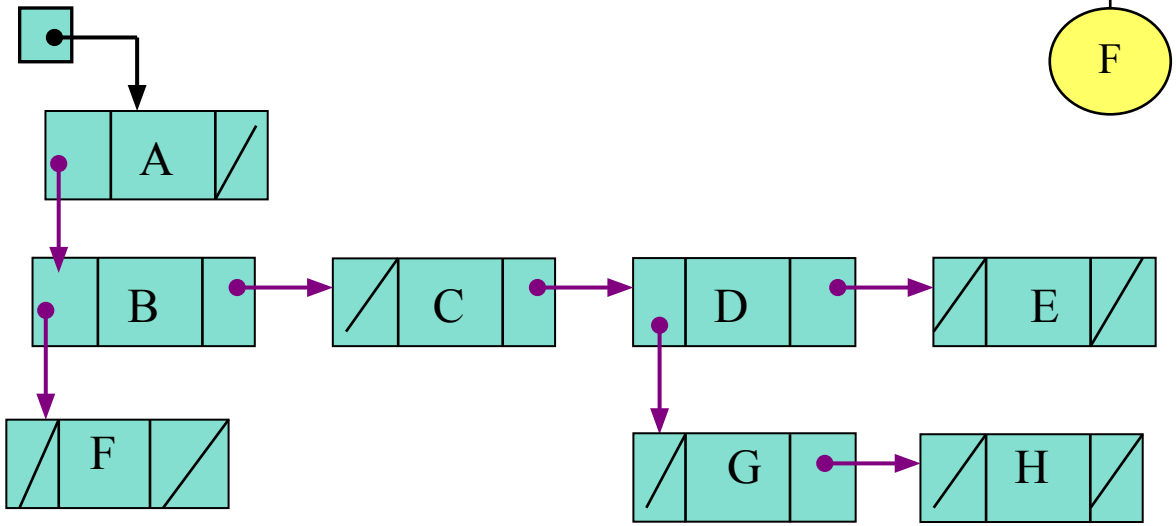
## Implementada con Listas enlazadas



# Representación: Hijo más izquierdo y hermano derecho



# Recorridos

- **Preorden**

  Se procesa primero la raíz y luego los hijos

- **Inorden**

  Se procesa el primer hijo, luego la raíz y por último los restantes hijos

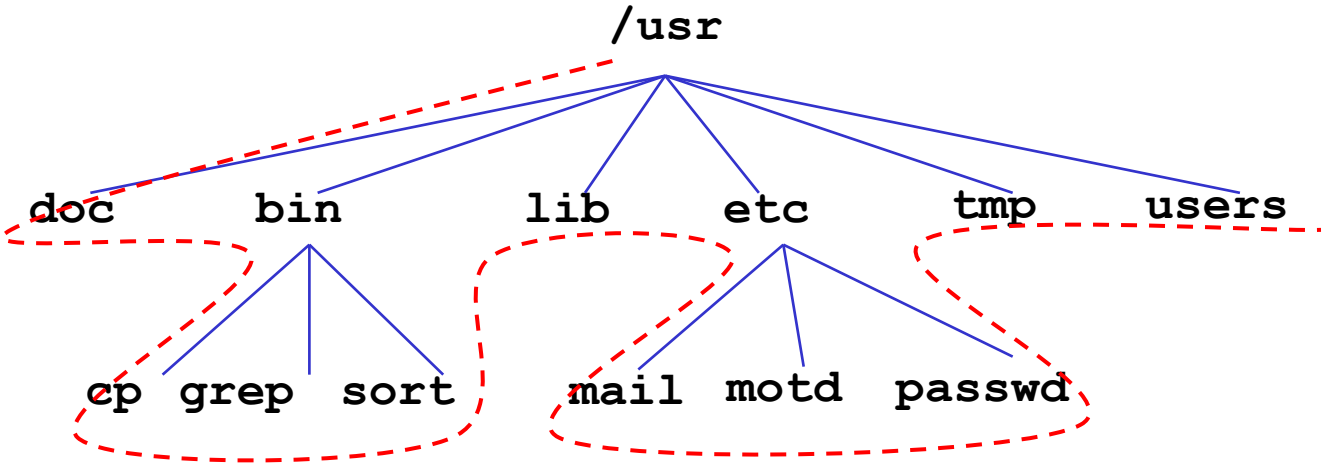
- **Postorden**

  Se procesan primero los hijos y luego la raíz

- **Por niveles**

  Se procesan los nodos teniendo en cuenta sus niveles, primero la raíz, luego los hijos, los hijos de éstos, etc.

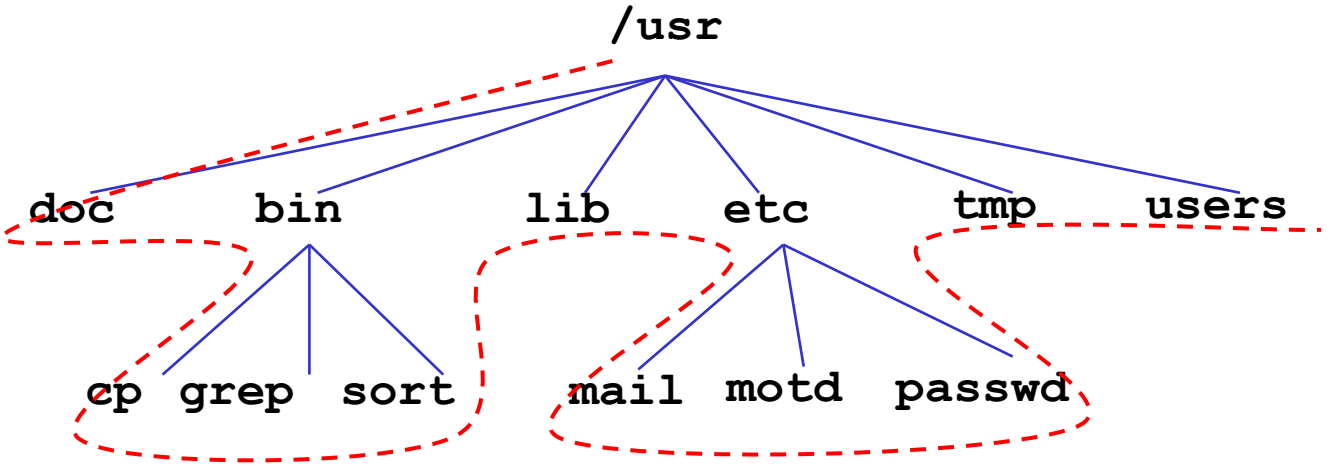
# Árbol General
## Recorrido en preorden




```
public void preOrden() {
   imprimir (dato);
   obtener lista de hijos;
   mientras (lista tenga datos) {
    hijo ← obtenerHijo;
    hijo.preOrden();
   }
}
```

**Ejemplo: Listado del contenido de un directorio**

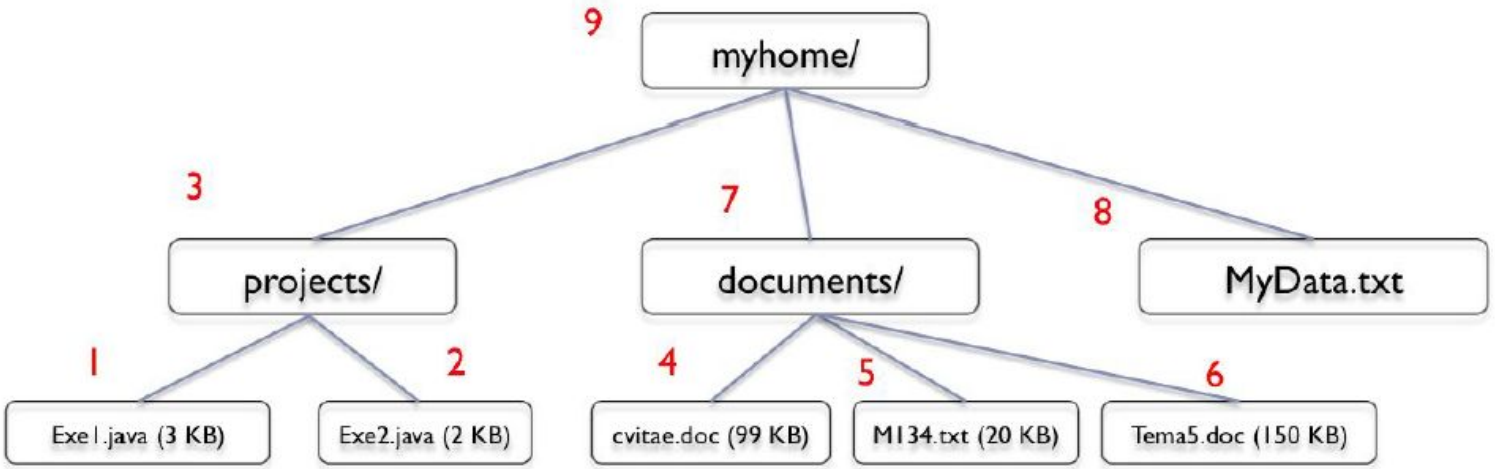
# Árbol General
## Recorrido en postorden



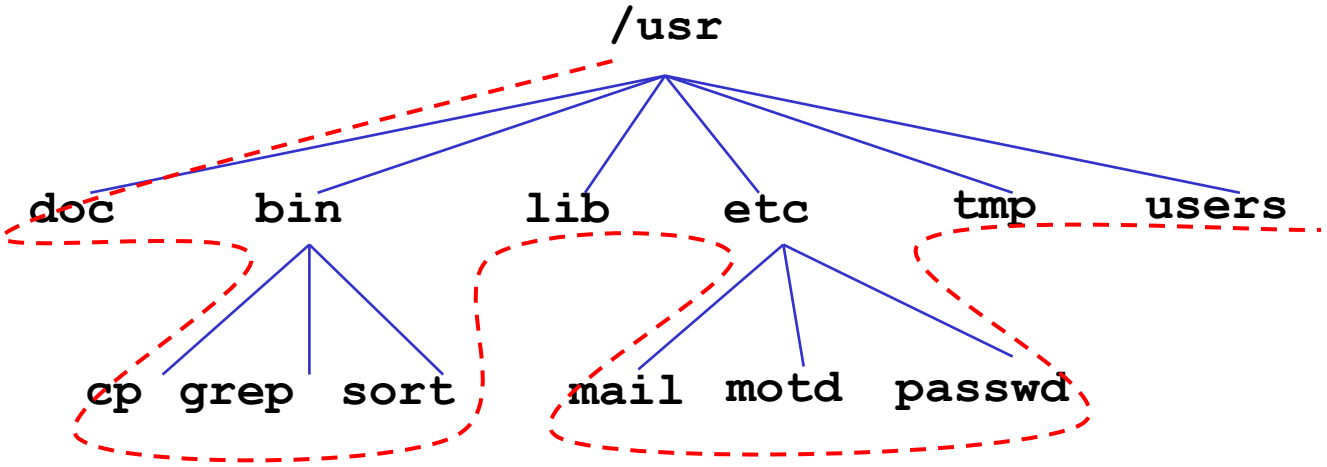

```
public void postOrden() {
   obtener lista de hijos;
   mientras (lista tenga datos) {
    hijo ← obtenerHijo;
    hijo.postOrden();
   }
  imprimir (dato);
}
```

# Recorrido: Postorden

## Ejemplo: Calcular el tamaño ocupado por un directorio



# Árbol General
## Recorrido por niveles

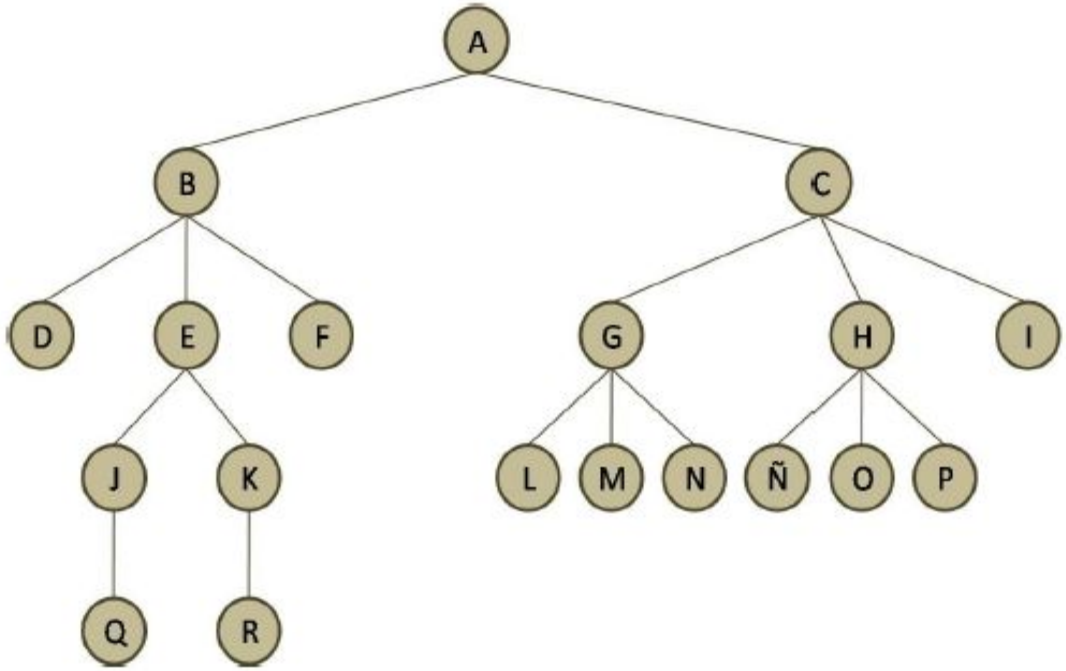



```
public void porNiveles() {

   encolar(raíz);
   mientras cola no se vacíe {
      v ← desencolar();
      imprimir (dato de v);
      para cada hijo de v
      encolar(hijo);
   }
}
```

# Ejercicio

**Dado el siguiente árbol, escriba los recorridos preorden, inorden y postorden**



# Ejercicio

## Abeto navideño

Problem - B - Codeforces

Considere un árbol general. Recordemos que el vértice **u** se llama hijo del vértice **v** y el vértice **v** se llama padre del vértice **u** si existe una arista dirigida de v a u. El árbol tiene un vértice distinguido llamado raíz, que es el único vértice que no tiene padre. Un vértice se llama hoja si no tiene hijos y tiene padre.
Llamaremos **abeto** a un árbol si cada vértice no hoja tiene al menos 3 hijos hojas.
Dado un árbol general, compruebe si es un abeto.

**Input**
La primera línea contiene un entero n: el número de vértices en el árbol ($3 \leq n \leq 1000$). Cada una de las siguientes n - 1 líneas contiene un entero pi ($1 \leq i \leq n - 1$) — el índice del padre del i + 1-ésimo vértice ($1 \leq pi \leq i$).
El vértice 1 es la raíz. Está garantizado que la raíz tiene al menos 2 hijos.

**Output**
Imprima "Yes" si el árbol es un abeto y "No" de lo contrario.

# Ejercicio

## Abeto navideño

Problem - B - Codeforces

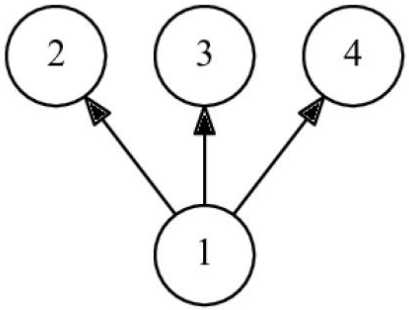
**Ejemplo 1**

**Input**

4
1
1
1

**Output**

Yes



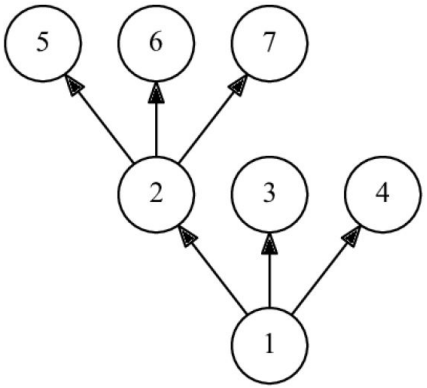

**Ejemplo 2**

**Input**

7
1
1
1
2
2
2

**Output**

No




**Ejemplo 3**

**Input**

8
1
1
1
1
3
3
3

**Output**

Yes