



Algoritmos y Estructuras de Datos

**Cursada 2017
Redictado**

Prof. Alejandra Schiavoni (ales@info.unlp.edu.ar)

Prof. Catalina Mostaccio (catty@lifa.info.unlp.edu.ar)

Prof. Pablo Iuliano (piuliano@info.unlp.edu.ar)

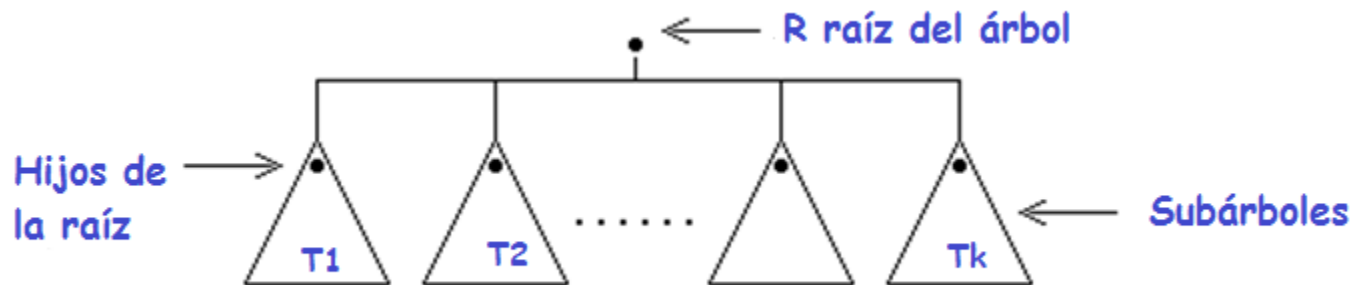
Agenda

- 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, \dots, 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* del árbol es el grado del nodo con mayor grado.
- *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 :

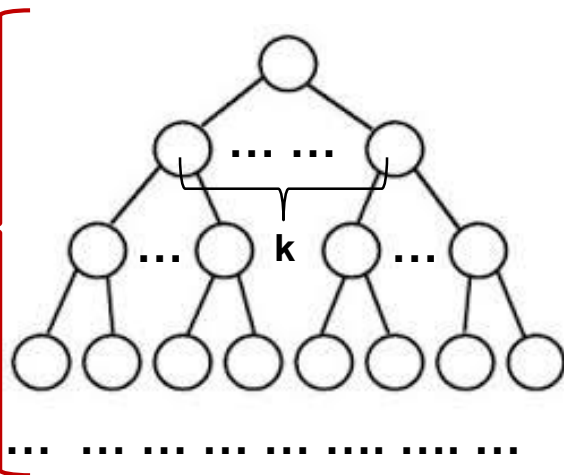
- 1.- T es un nodo simple (árbol lleno de altura 0), o
- 2.- T es de altura h y todos sus sub-árboles son llenos de altura $h-1$.

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 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 $\rightarrow k^0$ nodos

Nivel 1 $\rightarrow k^1$ nodos

Nivel 2 $\rightarrow k^2$ nodos

Nivel 3 $\rightarrow k^3$ nodos

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

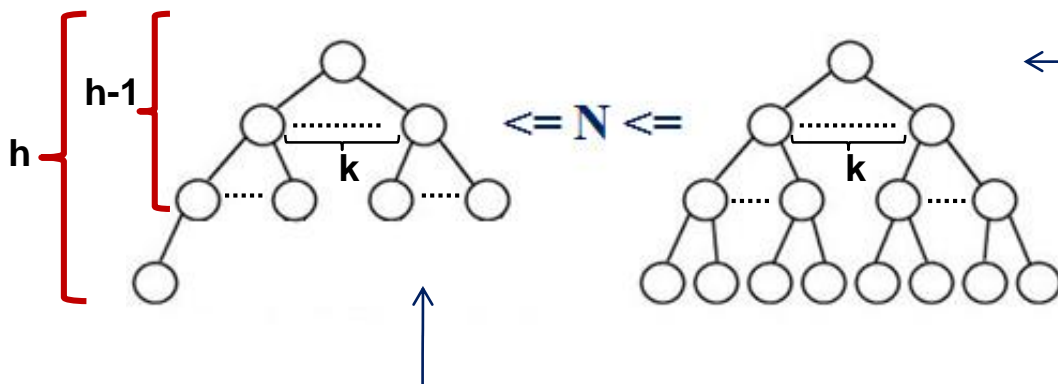
La suma de los términos de una serie geométrica de razón k es:

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

Descripción y terminología

- *Cantidad de nodos en un árbol completo:*

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 ...

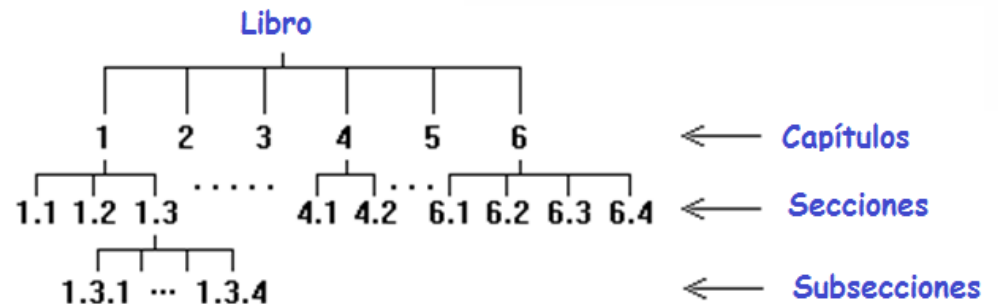
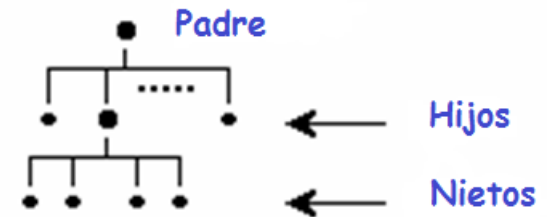
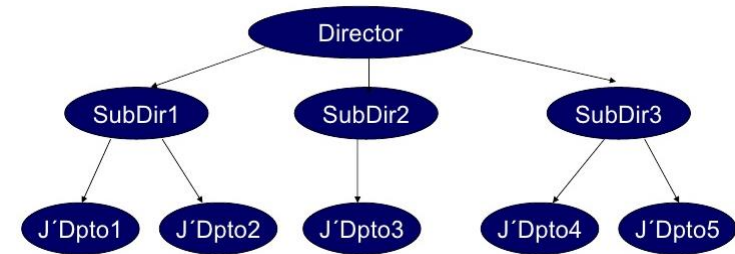


• Si el árbol es lleno
 $N = (k^{h+1} - 1) / (k - 1)$

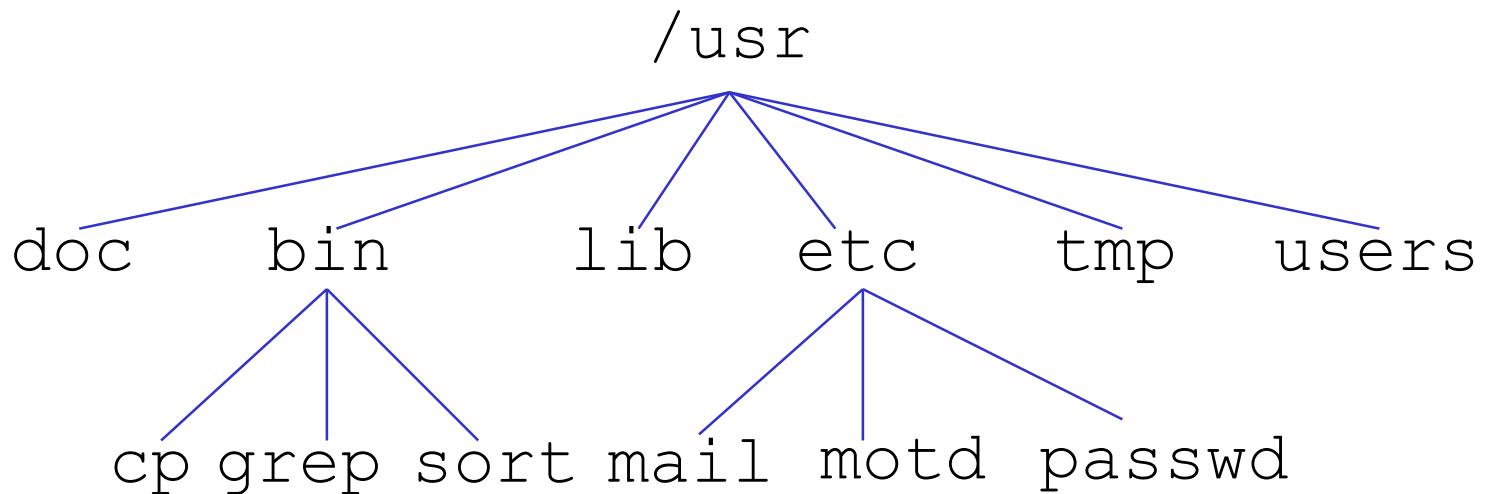
• Si no, el árbol es lleno en la altura $h-1$ y tiene por lo menos un nodo en el nivel h :
 $N = (k^{h-1+1} - 1) / (k - 1) + 1 = (k^h + k - 2) / (k - 1)$

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

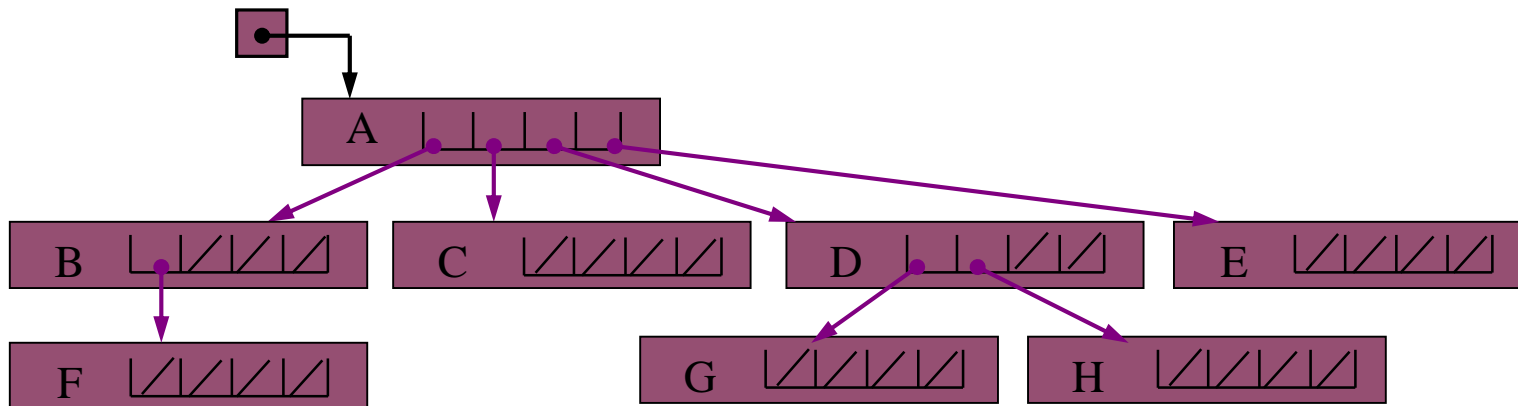
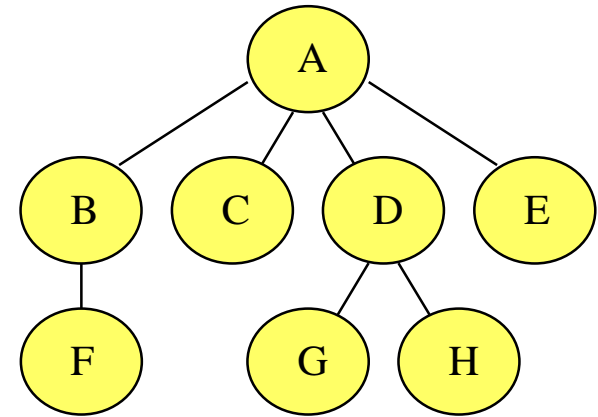
- ✓ Lista de hijos
 - Cada nodo tiene:
 - Información propia del nodo
 - Una lista de todos sus hijos
- ✓ Hijo más izquierdo y hermano derecho
 - Cada nodo tiene:
 - Información propia del nodo
 - Referencia al hijo más izquierdo
 - Referencia al hermano derecho

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

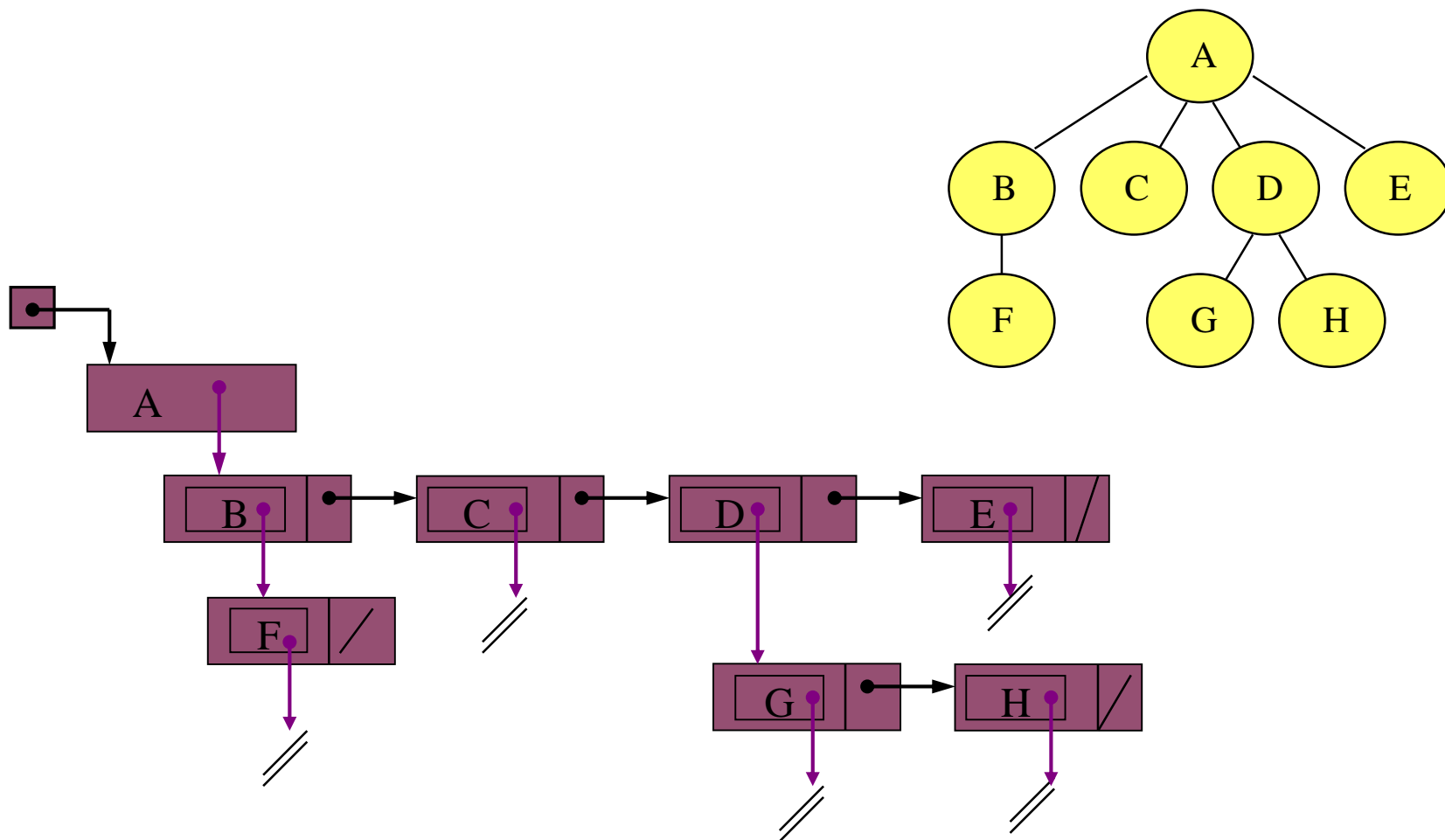
Representación: Lista de hijos

Implementada con Arreglos

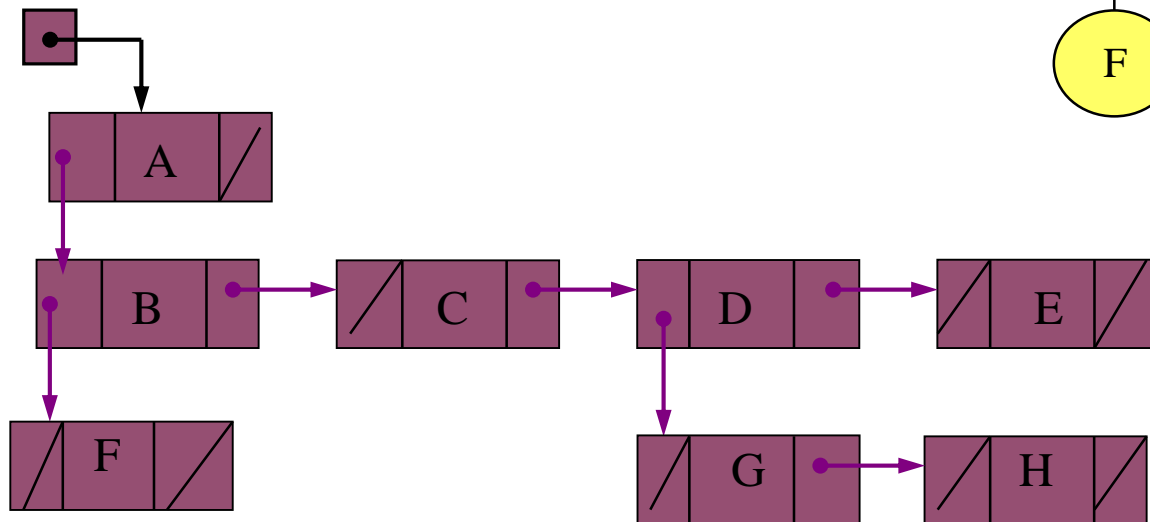
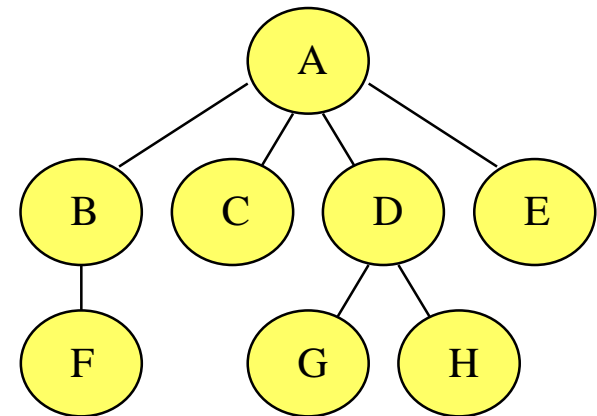


Representación: Lista de hijos

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

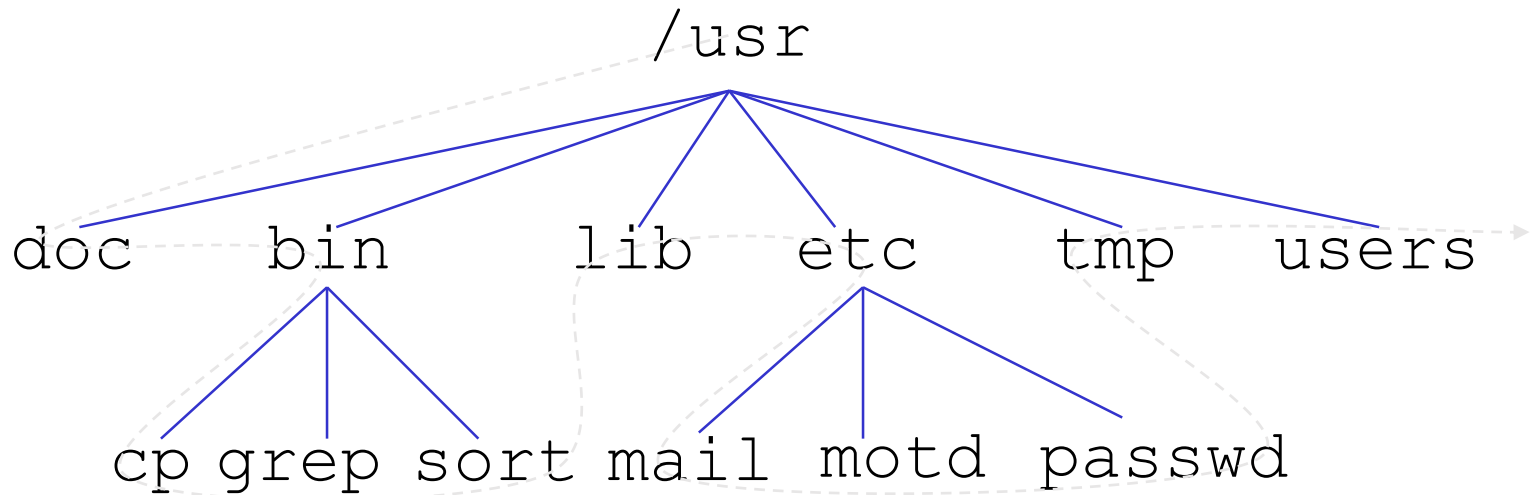
Se procesan primero los hijos y luego la raíz



Por niveles

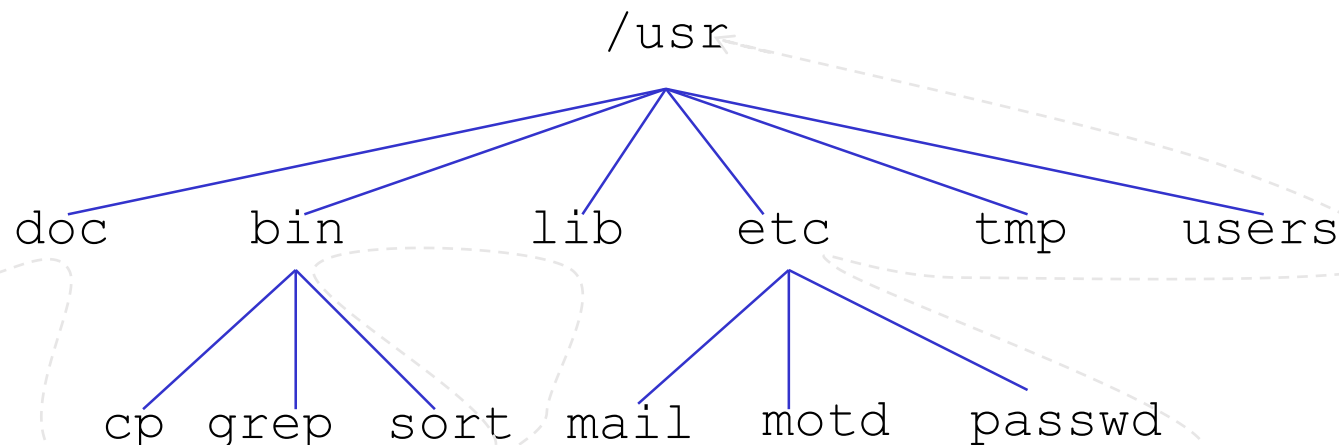
Se procesan los nodos teniendo en cuenta sus niveles, primero la raíz, luego los hijos, los hijos de éstos, etc.

Recorrido: Preorden

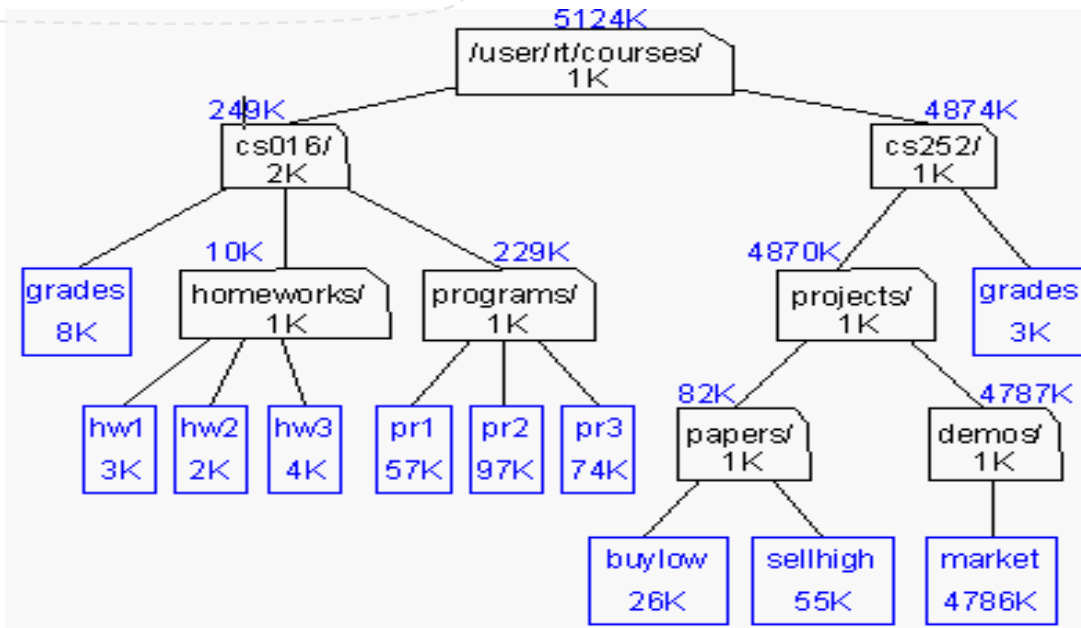


Ejemplo: Listado del contenido de un directorio

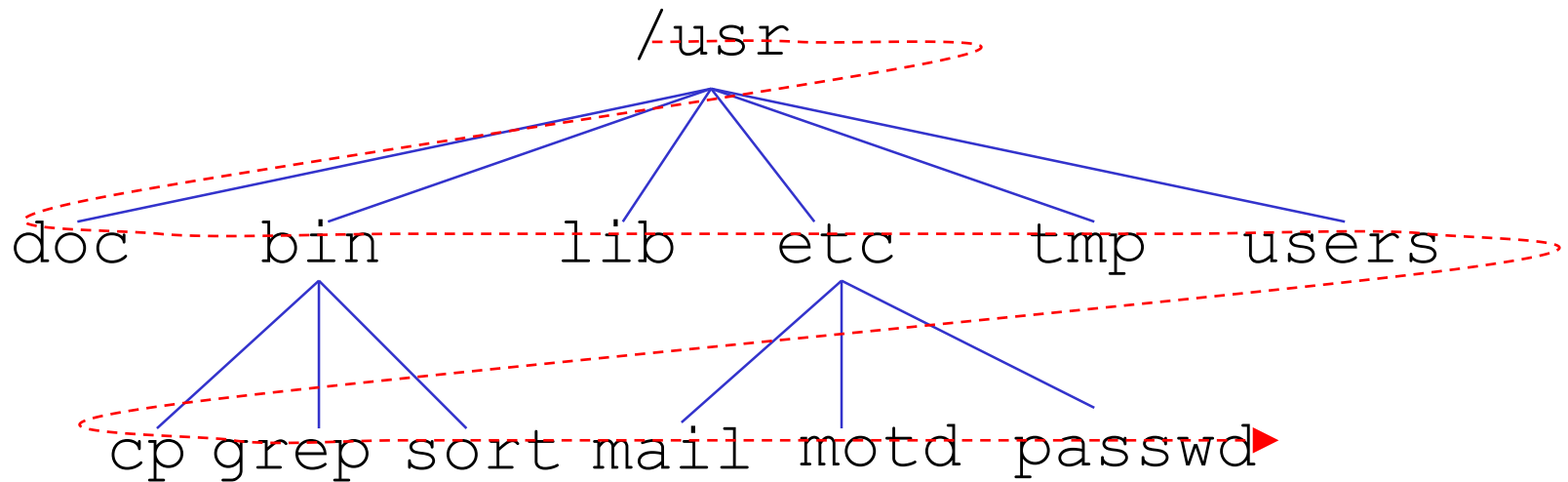
Recorrido: Postorden



Ejemplo: Calcular el tamaño ocupado por un directorio



Recorrido: Por niveles



Recorrido: Preorden

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

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);  
    }  
}
```