



Algoritmos y Estructuras de Datos

Cursada 2017

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

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

Prof. Laura Fava (lfava@info.unlp.edu.ar)

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

Análisis de los algoritmos de recorridos en Árboles binarios y generales

Agenda

- Tiempo de ejecución de los recorridos
 - Árboles Binarios
 - Árboles Generales
 - Árboles AVL

Árboles binarios

Recorridos



Preorden

- Se procesa primero la raíz y luego sus hijos, izquierdo y derecho.



Inorden

- Se procesa el hijo izquierdo, luego la raíz y último el hijo derecho



Postorden

- Se procesan primero los hijos, izquierdo y derecho, 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

```
public void preOrden() {  
    imprimir (dato);  
    si (tiene hijo_izquierdo)  
        hijoIzquierdo.preOrden();  
    si (tiene hijo_derecho)  
        hijoDerecho.preOrden();  
}
```

Recorrido Preorden: Tiempo de Ejecución

Considerando un árbol binario **lleno** y altura ***h***

$$T(n) = \begin{cases} c & n = 1 \\ c + 2 T((n - 1) / 2) & n > 1 \end{cases}$$

$T(n)$ es $O(n)$

Recorrido Preorden: Tiempo de Ejecución

Otra forma: expresando en función de la altura

$$T(h) = \begin{cases} c & h = 0 \\ c + 2 T(h - 1) & h > 0 \end{cases}$$

$$T(n) \text{ es } O(n)$$

Árboles Generales

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

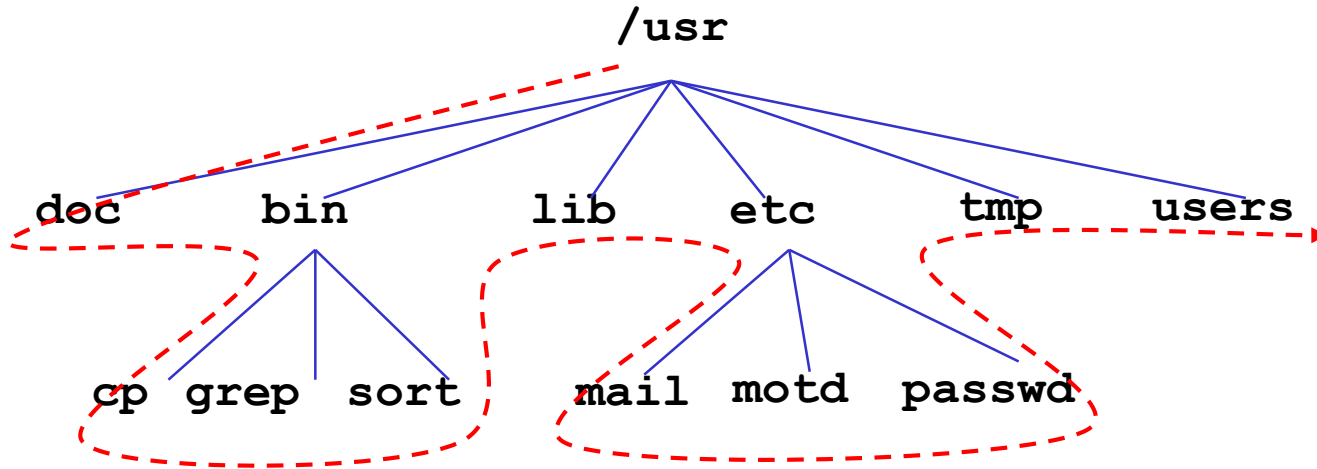
- Se procesan primero los hijos y luego la raíz



Por niveles

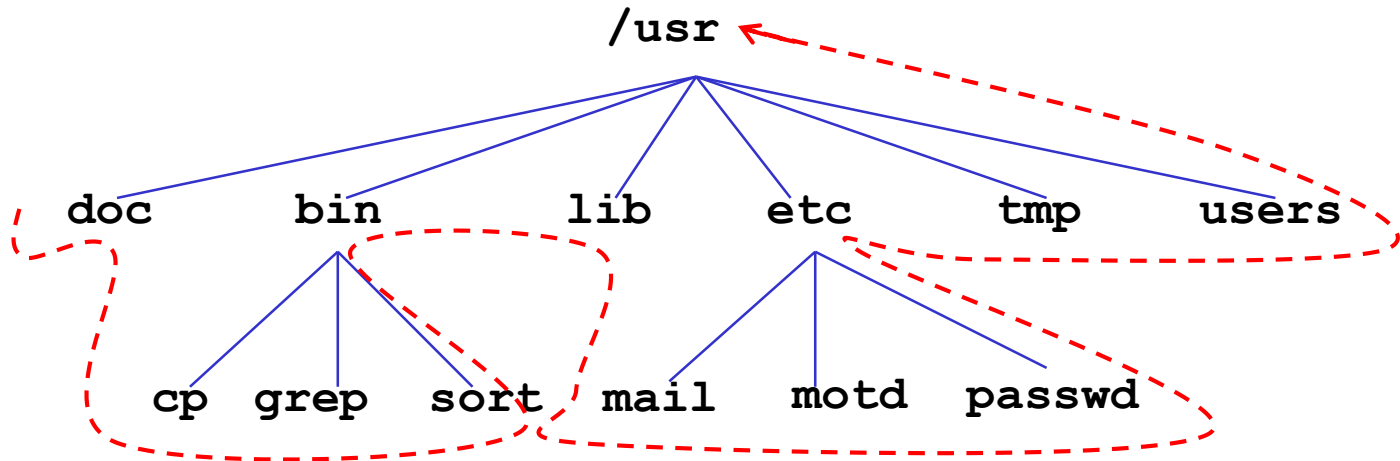
- Se procesan los nodos teniendo en cuenta sus niveles, primero la raíz, luego los hijos, los hijos de éstos, etc.

Recorrido Preorden



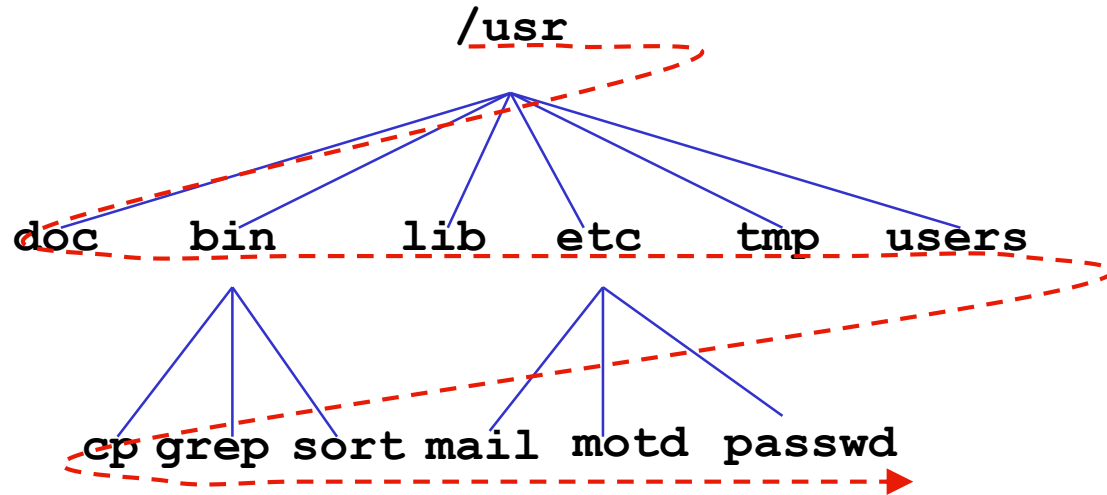
```
public void preOrden() {  
    imprimir (dato);  
    obtener lista de hijos;  
    mientras (lista tenga datos) {  
        hijo ← obtenerHijo;  
        hijo.preOrden();  
    }  
}
```

Recorrido Postorden



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

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);  
    }  
}
```

Recorrido Preorden: Tiempo de Ejecución

Considerando un árbol lleno de grado ***k*** y altura ***h***

$$T(n) = \begin{cases} c & n = 1 \\ c + k T((n-1)/k) & n > 1 \end{cases}$$

$T(n)$ es $O(n)$

Recorrido Preorden: Tiempo de Ejecución

Otra forma: expresando en función de la altura

$$T(h) = \begin{cases} c & h = 0 \\ c + k T(h - 1) & h > 0 \end{cases}$$

$T(n)$ es $O(n)$

Árboles AVL

Árboles AVL - Tiempo de Ejecución

Dado que el árbol AVL es un árbol binario de búsqueda balanceado, tomamos N_h como el número de nodos en un árbol AVL de altura h

$$\begin{aligned} N_h &\geq N_{h-1} + N_{h-2} + 1 \\ &\geq 2 N_{h-2} + 1 \\ &\geq 1 + 2(1 + 2 N_{h-4}) = 1 + 2 + 2^2 N_{h-4} \\ &\geq 1 + 2 + 2^2 + 2^3 N_{h-6} \\ &\dots \\ &\geq 1 + 2 + 2^2 + 2^3 + \dots + 2^{h/2} = 2^{h/2+1} - 1 \end{aligned}$$

continúa ...

Árboles AVL - Tiempo de Ejecución

Entonces,

$$2^{h/2+1} - 1 \leq n$$

$$h/2 \leq \log_2(n + 1) - 1$$

$$h \leq 2 \log_2(n + 1) - 2$$

Un análisis cuidadoso basado en la teoría de los números de Fibonacci, nos da un valor más ajustado de $1.44 \log_2(n + 2)$.