



Algoritmos y Estructuras de Datos

Cursada 2021

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

Agenda

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

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

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

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

Recorrido: Por niveles

```
public void porNiveles() {  
    encolar(raíz);  
    mientras (cola no se vacíe) {  
        desencolar(v);  
        imprimir (dato de v);  
        si (tiene hijo_izquierdo)  
            encolar(hijo_izquierdo);  
        si (tiene hijo_derecho)  
            encolar(hijo_derecho);  
    }  
}
```

Recorrido por niveles: Tiempo de Ejecución

$$\begin{aligned} T(n) &= cte_1 + \sum_{i=1}^n(cte_2) \\ &= cte_1 + n * cte_2 \end{aligned}$$

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

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

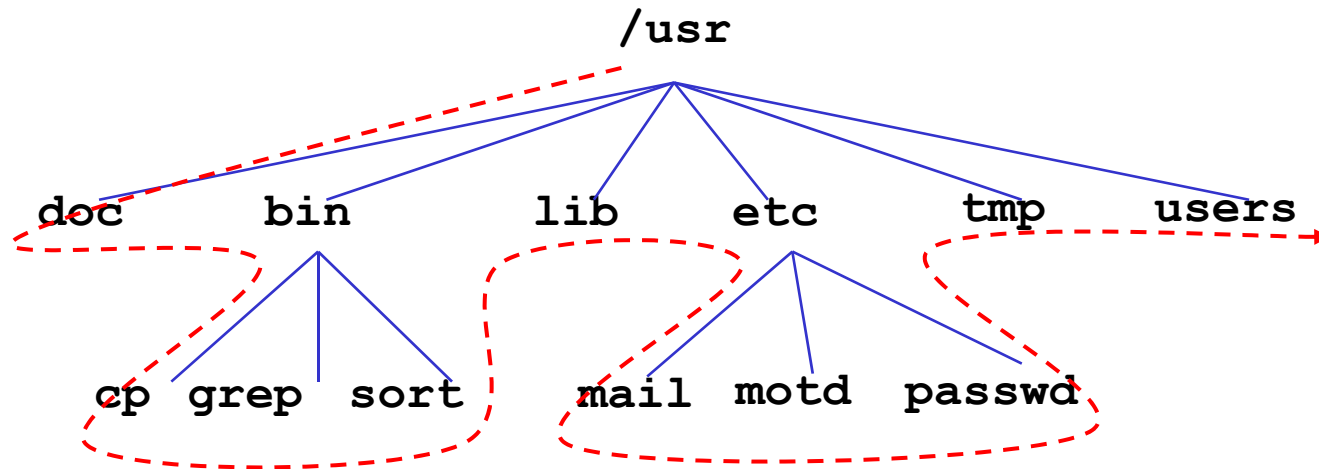
- Se procesan primero los hijos y luego la raíz



Por niveles

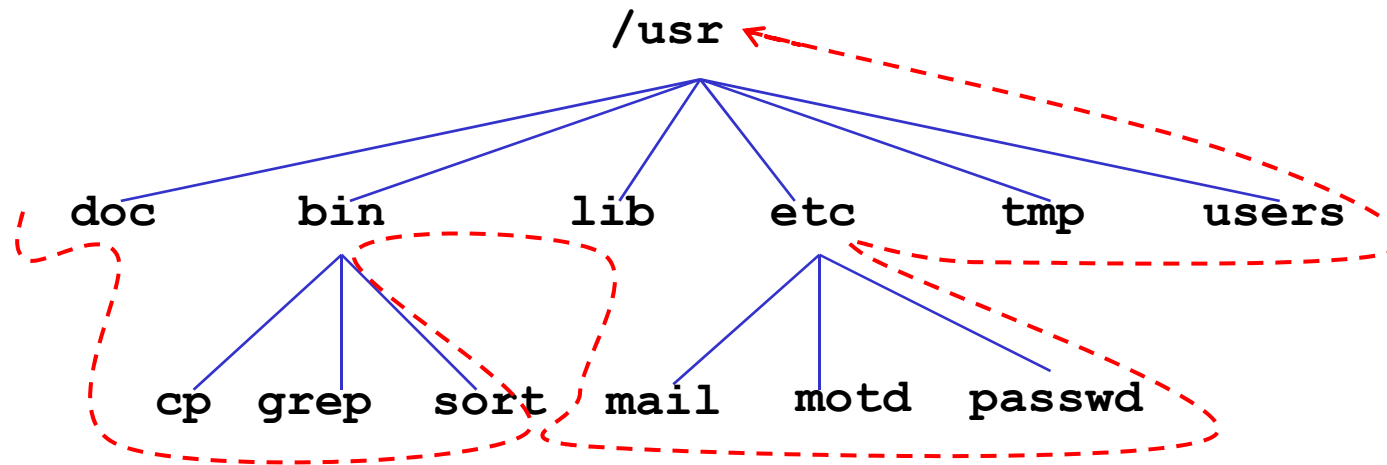
- Se procesan los nodos teniendo en cuenta sus niveles, primero la raíz, luego los hijos, los hijos de éstos, etc.

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 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 $O(n)$

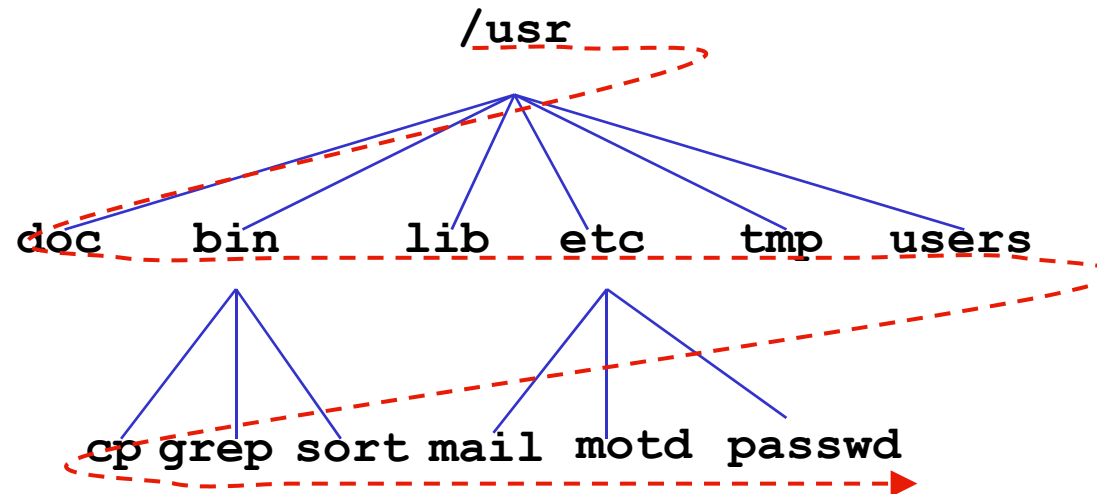
Recorrido Preorden: Tiempo de Ejecución

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 $O(n)$

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 por niveles: Tiempo de Ejecución

```
public void porNiveles() {  
1.   encolar(raíz);  
2.   mientras cola no se vacíe {  
3.       v ← desencolar();  
4.       imprimir (dato de v);  
5.       para cada hijo de v  
6.           encolar(hijo);  } }
```

$$T(n) = cte_1 + \sum_{i=1}^n (cte_2 + h_{v_i} * cte_1)$$

$$= cte_1 + n * cte_2$$

$$+ (h_{v_1} * cte_1 + h_{v_2} * cte_1 + \dots + h_{v_n} * cte_1)$$


$$= cte_1 + n * cte_2 + (h_{v_1} + h_{v_2} + \dots + h_{v_n}) * cte_1$$

$$= cte_1 + n * cte_2 + (n - 1) * cte_1$$

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

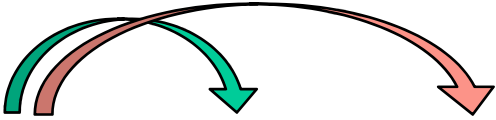
Recorrido por niveles: Tiempo de Ejecución

```
public void porNiveles() {  
1.   encolar(raíz);  
2.   mientras cola no se vacíe {  
3.       v ← desencolar();  
4.       imprimir (dato de v);  
5.       para cada hijo de v  
6.           encolar(hijo);  } }
```


$$\begin{aligned} T(n) &= \underset{1.}{cte_1} + \sum_{i=1}^n \underset{2.}{(} \underset{3.y\ 4.}{cte_2 + h_{v_i} * } \underset{5.}{cte_1} \underset{6.}{)} \\ &= cte_1 + n * cte_2 \\ &\quad + (h_{v_1} * cte_1 + h_{v_2} * cte_1 + \dots + h_{v_n} * cte_1) \\ &= cte_1 + n * cte_2 + (h_{v_1} + h_{v_2} + \dots + h_{v_n}) * cte_1 \\ &= cte_1 + n * cte_2 + (n - 1) * cte_1 \\ T(n) &\text{ es } O(n) \end{aligned}$$

Recorrido por niveles: Tiempo de Ejecución

```
public void porNiveles() {  
1.   encolar(raíz);  
2.   mientras cola no se vacíe {  
3.       v ← desencolar();  
4.       imprimir (dato de v);  
5.       para cada hijo de v  
6.           encolar(hijo);  } }
```


$$T(n) = cte_1 + \sum_{i=1}^n (cte_2 + h_{v_i} * cte_1)$$

1. 2. 3.y 4. 5. 6.

$$= cte_1 + n * cte_2$$

$$+ (h_{v_1} * cte_1 + h_{v_2} * cte_1 + \dots + h_{v_n} * cte_1)$$

$$= cte_1 + n * cte_2 + (h_{v_1} + h_{v_2} + \dots + h_{v_n}) * cte_1$$

$$= cte_1 + n * cte_2 + (n - 1) * cte_1$$

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

Recorrido por niveles: Tiempo de Ejecución

```
public void porNiveles() {  
1.   encolar(raíz);  
2.   mientras cola no se vacíe {  
3.       v ← desencolar();  
4.       imprimir (dato de v);  
5.       para cada hijo de v  
6.           encolar(hijo);  } }
```

$$\begin{aligned} T(n) &= \underset{1.}{cte_1} + \sum_{i=1}^n (\underset{2.}{cte_2} + \underset{3.y\ 4.}{h_{v_i}} * \underset{5.}{cte_1}) \\ &= cte_1 + n * cte_2 \\ &\quad + (h_{v_1} * cte_1 + h_{v_2} * cte_1 + \dots + h_{v_n} * cte_1) \\ &= cte_1 + n * cte_2 + (h_{v_1} + h_{v_2} + \dots + h_{v_n}) * cte_1 \\ &= cte_1 + n * cte_2 + (n - 1) * cte_1 \end{aligned}$$

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