

# Estructuras de Datos y Algoritmos II UPE 2016

## Trabajo Práctico N°2

Fecha parcialito:03/05/2015

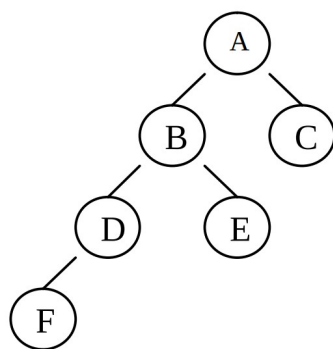


1. Dadas las fuentes de Arboles Binarios proporcionadas por la cátedra, construya un árbol binario de números enteros de por lo menos 5 niveles, y realice la impresión en pantalla de todo sus elementos, realizándolo con los recorridos: preorden, inorden, posorden y por niveles.

2. Agregue a la clase **ArbolBinario** los siguientes métodos:

**a) frontera(): list.** Se define frontera de un árbol binario, a las hojas de un árbol binario recorridos de izquierda a derecha.

Ejemplo:



La lista deberá devolver:  
F,E,C

**b) lleno(): boolean.** Devuelve true si el árbol es lleno. Un árbol binario es lleno si tiene todas las hojas en el mismo nivel y además tiene todas las hojas posibles (es decir todos los nodos intermedios tienen dos hijos).

3. Una red binaria es una red que posee una topología de árbol binario completo (vea la figura 1 como ejemplo).

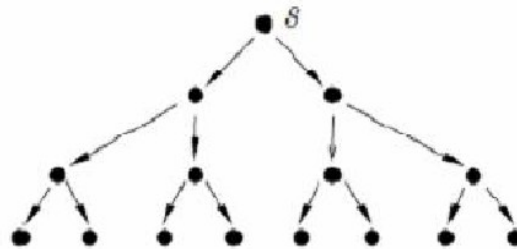


Figura 1: Red binaria completa

Los nodos que conforman una red binaria completa tiene la particularidad de que todos ellos conocen cual es su retardo de reenvío. El retardo de reenvío se define como el período comprendido entre que un nodo recibe un mensaje y lo reenvía a sus dos hijos.

Implementar un algoritmo que calcule el mayor retardo posible, en el camino que realiza un mensaje desde la raíz hasta llegar a las hojas en una red binaria completa.

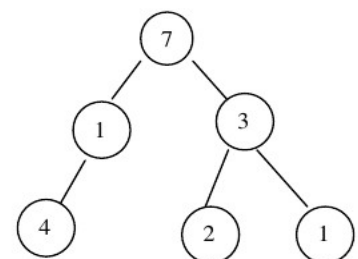
4. Implemente la operación **trayectoriaPesada(ab: ArbolBinario) : list** // Retorna el valor de la trayectoria pesada de cada una de las hojas del árbol binario ab.

Se define el valor de la trayectoria pesada de una hoja de un árbol binario como la suma del contenido de todos los nodos desde la raíz a la hoja multiplicada por el nivel en el que se encuentra. Ejemplo:

Trayectoria Pesada hoja 4 es  $4*2 + 1*1 + 7*0 = 9$

Trayectoria Pesada hoja 2 es  $2*2 + 3*1 + 7*0 = 7$

Trayectoria Pesada hoja 1 es  $1*2 + 3*1 + 7*0 = 5$



5. Willy es una abeja macho y está interesado en saber como está formado su árbol genealógico y cuantas hembras y machos hay en él.

Willy sabe que las féminas tienen 2 padres (un padre y una madre), pero los machos tienen únicamente madre (no padre).

Usted debe construir el árbol genealógico de Willy hasta el nivel "x" (el cual es pasado como parámetro).

Tenga en cuenta que la estructura del árbol genealógico puede deducirse a partir de cualquier abeja (por ejemplo: Willy es macho, por lo tanto, tiene sólo madre. La

madre de Willy es hembra, por lo tanto, tiene padre y madre. El abuelo de Willy, es macho, por lo tanto tiene sólo madre... etc.)

Teniendo las siguientes clases, implemente en la clase Apicola el método ***arbolGenealogico(int nivel, Abeja abeja):ArbolBinario<Abeja>*** que recibe como parámetros el nivel, del árbol genealógico a crear según lo enunciado anteriormente, y la abeja, de la cual hay que crear el árbol. Asuma que el resto de los métodos se disponen.

Abeja
-tipo:String # "m" o "f"
+ getTipo():String + setTipo(tipo:String)

Apicola
+arbolGeneralógico(nivel:int, abeja:Abeja):ArbolBinario<Abeja>

### Tiempo de Ejecución

6. Calcule el T(n).

i)

```
c = 1;
while ( c < n ):
    algo_de _ O(1)
    c = c * 2
```

ii)

```
c = i;
while ( c > 1 ):
    algo_de _ O(1)
    c = c / 2
```

7. Para cada uno de los algoritmos presentados:

a) Determinar cuál es el peor caso. Fundamentar su respuesta.

b) Expresar en función de n el tiempo de ejecución del peor caso

i)

```
def rec1(n):
    if n <= 1 :
        return 1
    else:
        return rec1(n-1) + rec1(n-1)
```

ii)

```
def rec2(n):
    if n <= 1:
        return 1
    else:
```

```
return 2 * rec2(n-1)
```

iii)

```
def rec3(n):  
    if n == 0:  
        return 0  
    elif n == 1:  
        return 1  
    else:  
        return rec3(n-2) * rec3(n-2)
```

8. Dado la siguiente función, plantear y resolver la función de recurrencia:

```
def funcion(n):  
    x = 0  
    if n <= 1:  
        return 1  
    else:  
        for i in range(1,n):  
            x = 1  
            while x < n :  
                x = x*2  
        return funcion(n/2) + funcion(n/2)
```