

ESTRUCTURA DE DATOS Y ALGORITMOS

Grado de Ingeniero de Informática de Gestión y Sistemas de Información



Vitoria-Gasteiz, 21 de diciembre de 2011

2. Ejercicio (2 puntos)

1. Especificar / Parametrizar

Entrada: A, un arbol binario.

Salida: un entero, que representa la longitud del camino de mayor coste

```
public int longitudCaminoCosteMaximo();
```

A la hora de tratar un nodo del árbol y poder calcular la longitud del camino de mayor coste, debemos de conocer tres cosas: 1) si el nodo actual es izquierdo o derecho, para poder calcular el coste total que supone visitar dicho nodo; 2) el coste máximo de recorrer uno de los caminos de los subárboles izquierdo y/o derecho; y 3) la longitud máxima de los caminos de los subárboles izquierdo y/o derecho cuyo recorrido suponga el coste máximo \rightarrow INMERSION

Entrada: A, un arbol binario y esDerecho, un booleano que representa si la raíz de A es derecho (true) o izquierdo (false).

Salida: dos enteros. El primero representa el coste máximo de recorrer uno de los caminos del árbol A y el segundo representa la longitud máxima de los caminos cuyo recorrido suponga el coste máximo.

Se necesita una clase Contexto para poder devolver dos valores:

```
public class Contexto {
   int coste;
   int longitud;
}
```



ESTRUCTURA DE DATOS Y ALGORITMOS

Grado de Ingeniero de Informática de Gestión y Sistemas de Información



Vitoria-Gasteiz, 21 de diciembre de 2011

2. Diseño

Casos triviales

```
Si esVacio (A) \rightarrow devolver new Contexto(0, 0)
```

Si esHoja (A) y no(esDerecho) \rightarrow devolver new Contexto(raiz(A), 0)

Si esHoja (A) y esDerecho \rightarrow devolver new Contexto(2*raiz(A), 0)

Caso General

```
Si No esHoja(A) →

costeA = (esDerecho? 2:1) * raiz(A);

contextoIzq = longitudCaminoCosteMaximo1 (subarbolIzq(A), true);

contextoDch = longitudCaminoCosteMaximo1 (subarbolDch(A), false);

Si (contextoIzq. coste > contextoDch.coste) entonces

coste = contextoIzq.coste + costeA;

longitud = contextoIzq.longitud + 1;

si no (contextoIzq. coste < contextoDch.coste) entonces

coste = contextoDch. coste + costeA;

longitud = contextoDch.longitud + 1;

si no

coste = contextoDch. coste + costeA;

longitud = maximo(contextoIzq.longitud, contextoDch.longitud) + 1;

fin si

contexto = new Contexto(coste, longitud);

devolver contexto;
```



ESTRUCTURA DE DATOS Y ALGORITMOS

Grado de Ingeniero de Informática de Gestión y Sistemas de Información



Vitoria-Gasteiz, 21 de diciembre de 2011

3-Implementación

```
class BinTreeItr<T> {
  public int longitudCaminoCosteMaximo() {
     Contexto contexto = longitudCaminoCosteMaximo1 (
                                       (BTNode<Integer> bTree.root, false);
     return contexto.longitud;
  }
  private Contexto longitudCaminoCosteMaximo1(BTNode<? extends Integer> A,
                                              boolean esDerecho) {
     int coste, longitud;
     if (A==null)
       return new Contexto(0, 0);
     else if ((A.left==null) && (A.right==null) ) { // A es hoja
        coste = (esDerecho? 2 : 1) * A.content.intValue();
        return new Contexto(coste, 0);
     }
     else {
       int costeA = (esDerecho? 2 : 1) * A.content.intValue();
       Contexto contextoIzq = longitudCaminoCosteMaximo1(A.left, false);
       Contexto contextoDch = longitudCaminoCosteMaximo1(A.right, true);
       if (contextoIzg.coste > contextoDch.coste){
         coste = contextoIzq.coste + costeA;
         longitud = contextoIzq.longitud + 1;
       else if (contextoIzq.coste < contextoDch.coste){</pre>
         coste = contextoDch.coste + costeA;
         longitud = contextoDch.longitud + 1;
       else { //el coste máximo es el mismo
         coste = contextoDch.coste + costeA;
         longitud = Math.max(contextoIzq.longitud, contextoDch.longitud)+1;
       Contexto contexto = new Contexto(coste, longitud);
       return contexto;
     }
public class Contexto {
     public int coste, longitud;
     public Contexto(int coste, int longitud) {
       this.coste = coste;
       this.longitud = longitud;
}
```