Exámenes

Tema 4 - S2 - Parte 1: Cuestiones sobre los métodos de un ABB y sus costes

Volver a la Lista de Exámenes

Parte 1 de 1 - 10.0/ 10.0 Puntos

Preguntas 1 de 7

1.0/ 1.0 Puntos

El atributo `talla` definido en la clase `NodoABB` permite calcular el tamaño de cualquier nodo de un ABB en tiempo constante.

El siguiente método calcula el tamaño del nodo `actual` contando explícitamente sus descendientes.

```
  protected int talla(NodoABB<E> actual) {
       if (actual == null)  { return 0; }
       else { return 1 + talla(actual.izq) +
talla(actual.der); }
   }
```

¿Cuál es el coste temporal asintótico de este método?

- ◯ **Lineal con la altura del nodo `actual`**
- ◯ **Logarítmico con la altura del nodo `actual`**
- ✔ ◯ **Lineal con la talla del nodo `actual`**
- ◯ **Logarítmico con la talla del nodo `actual`**

Respuesta correcta:C

Comentarios:

El método `talla` implementa un Recorrido que procesa todos los Nodos de `actual`. Por lo tanto, su coste es lineal con su tamaño.

Preguntas 2 de 7

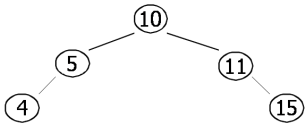
1.0/ 1.0 Puntos

El siguiente método permite contar el **número de hojas** de un ABB:

```
    public int numHojas() {
        if  (this.raiz == null) { return 0; }
        return numHojas(this.raiz);
```

```
    }
    //SII actual != null: obtiene el nº de hojas del
nodo actual
    protected int numHojas(NodoABB<E> actual) {
        int  res = 0;
        if  (actual.izq == null  &&  actual.der ==
null) { res = 1; }
        else {
            if  (actual.izq != null)  { res +=
numHojas(actual.izq); }
            if  (actual.der != null)  { res +=
numHojas(actual.der); }
        }
        return res;
    }
```

Dado el siguiente ABB, indica el número total de llamadas recursivas que se generan para la ejecución del método -recursivo- sobre su nodo raíz.



- ◯ **11**
- ◯ **33**
- ✔ ◯ **5**
- ◯ **7**

Respuesta correcta:C

Comentarios:

Se generan las siguientes llamadas recursivas en este orden:

```
numHojas(10)
    → numHojas(5)
        → numHojas(4)   // se alcanza una hoja: caso base, en el que se inicializa el resultado
    → numHojas(11)
        → numHojas(15)  // se alcanza una hoja: caso base, en el que se inicializa el resultado
```

Fíjate que el caso base elegido, un nodo hoja tiene una hoja: hace que NO se generen llamadas con parámetro `null`.

Preguntas 3 de 7

2.0/ 2.0 Puntos

***Sin añadir blanco alguno,* completa los huecos del siguiente método para que elimine todas las Hojas de un ABB:**

```
public void eliminarHojas() {
    if (this.raiz != null) { this.raiz = eliminarHojas(this.raiz); }
}
//SII actual != null: elimina las hojas del nodo actual
protected NodoABB<E> eliminarHojas(NodoABB<E> actual) {
    NodoAB<E> res = ✔actual ;
    if (actual.izq == null && actual.der == null) { res = ✔null ; }
    else {
        if (actual.izq != null) { res. ✔izq = eliminarHojas(actual.izq);
}
        if (actual.der != null) { res. ✔der = eliminarHojas(actual.der);
}
        res.talla = 1 + talla(res.izq) + talla(res.der);
    }
    return res;
}
```

Respuesta correcta:actual, null, izq, der

Comentarios:

El resultado de `eliminarHojas` es el mismo nodo `actual` **PERO** sin hojas. Por tanto,

- Si `actual` es una hoja, caso base, eliminarla es substituirla por un `null`.
- Si `actual` NO es una hoja, caso general, eliminar sus hojas es eliminar las de sus hijos izquierdo y derecho, si es que los tiene.

Preguntas 4 de 7

2.0/ 2.0 Puntos

El siguiente método obtiene de forma ineficiente el nodo con la 1ª aparición en Pre-Orden de `e` en el Nodo `actual` (`null` si `e` no está en `actual`). Completa los huecos para realizar el análisis de su coste Temporal Asintótico:

```
protected NodoABB<E> recuperarMalo(E e, NodoABB<E> actual) {
    NodoABB<E> res = null;  // Caso base equivalente: res = actual
    if  (actual != null) {
         if (actual.dato.equals(e))  {  res = actual;  }
         else {
             res = recuperarMalo(e, actual.izq);
             if (res == null)  {  res = recuperarMalo(e, actual.der);  }
         }
    }
    return res;
}
```

Paso 1. La talla x del problema es ✔el_tamaño del Nodo `actual`
NOTA. *Sin añadir blanco alguno*, utiliza una de las siguientes opciones para rellenar: la_altura, el_tamaño, el_tamaño/2, la_altura/2.

Paso 2. Sí existen instancias significativas, pues se trata de una búsqueda. Indica cuáles de las siguientes lo son poniendo V o F (*sin añadir blanco alguno*) en los huecos que figuran delante de ellas:

- **✔F El mejor de los casos se produce cuando `actual` es `null`, pues ni siquiera se ejecuta el `equals`.**
- **✔V El mejor de los casos se produce cuando el primer dato en Pre-Orden del nodo `actual` es `e`.**
- **✔F El mejor de los casos se produce cuando el dato `e` está en el hijo izquierdo de `actual`.**
- **✔F El peor de los casos se produce bien cuando el dato `e` no está en el hijo izquierdo de `actual`, sino en el derecho.**
- **✔V El peor de los casos se produce cuando el dato `e` no está en el nodo `actual`.**

Paso 3. Escribe las relaciones de recurrencia del caso general para las instancias significativas detectadas, completando cada hueco con alguna de las siguientes opciones y *sin añadir blanco alguno*: >1, >0, k, k*x, k'*x, 1, 2, x-1, x/2, x-2, logx:

- **En el mejor de los casos, tanto si el nodo `actual` está Equilibrado como si está Completamente Degenerado, si x ✔>0**

  $T^M_{recuperarMalo}(x) =$ **✔k**
- **En el peor de los casos, si el nodo `actual` está...**
  - **Equilibrado:** $T^P_{recuperarMalo}(x) =$ **✔2** $* T^P_{recuperarMalo}($ **✔x/2** $) +$ **✔k**
  - **Completamente Degenerado:** $T^P_{recuperarMalo}(x) =$ **✔1** $* T^P_{recuperarMalo}($ **✔x-1** $) +$ **✔k**

Paso 4. Utilizando los teoremas de coste, obtén el coste temporal asintótico completando cada hueco con una de las siguientes opciones y *sin añadir blanco alguno*: Theta, Omega, O, 1, x, x^2, logx:

$T^{M}_{recuperarMalo}(x) \in$ ✔Theta (✔1) $\rightarrow T_{recuperarMalo}(x) \in$ ✔Omega (✔1)

$T^{P}_{recuperarMalo}(x) \in$ ✔Theta (✔x) $\rightarrow T_{recuperarMalo}(x) \in$ ✔O (✔x)

Respuesta correcta:el_tamaño, F, V, F, F, V, >0, k, 2, x/2, k, 1, x-1, k, Theta, 1, Omega, 1, Theta, x, O, x

Preguntas 5 de 7

1.0/ 1.0 Puntos

Señala de entre los siguientes, los métodos que aprovechando la propiedad de orden del ABB pueden ser diseñados con coste logarítmico (puedes consultar el código disponible en la clase ABB utilizada en el tema):

- ✔ ☐ `recuperarMalo(e)`
- ☐ `eliminarHojas()`
- ☐ `altura()`
- ☐ `toStringPreOrden()`

Respuesta correcta:A

Preguntas 6 de 7

2.0/ 2.0 Puntos

***Sin añadir blanco alguno*, completa el siguiente método para que inserte el dato `e` en un ABB; si `e` ya fuera un elemento del ABB, entonces lo debe insertar en su hijo izquierdo:**

```
/** inserta e en un ABB; si ya está lo inserta en su Hijo Izquierdo **/
public  void  insertarConDuplicados(E e)  {
    this.raiz = insertarConDuplicados(e, this.raiz);
}
protected NodoABB<E> insertarConDuplicados(E e, NodoABB<E> actual) {

    NodoAB<E> res = ✔actual ;
    if  (actual != null) {
        int resC = actual. ✔dato.compareTo(e) ;
        if (resC < 0)  {  res. ✔der = insertarConDuplicados(e, actual.der);
}
        else   {  res. ✔izq = insertarConDuplicados(e, actual.izq);  }
        res.talla++;
    }
    else  {   ✔res = new NodoABB<E>(e);  }
    return res;
}
```

Respuesta correcta:actual, dato.compareTo(e)|dato.compareTo(e), der, izq, res

Preguntas 7 de 7

1.0/ 1.0 Puntos

¿Cuál es el coste del método `insertarConDuplicados` diseñado en este mismo examen si se realiza sobre un ABB Equilibrado?

- [ ] Constante
- [ ] Lineal con el tamaño del ABB
- ✔ [ ] Logarítmico con el tamaño del ABB
- ✔ [ ] Lineal con la altura del ABB
- [ ] Logarítmico con la altura del ABB

Respuesta correcta:C, D

Comentarios:

El coste temporal asintótico de `insertarConDuplicados` es lineal con la altura del longitud del camino de inserción que nunca puede ser superior a la altura del ABB. Por otro lado, en un ABB equilibrado su altura es del mismo orden que el logaritmo de su tamaño.

- PoliformaT
- UPV

- Powered by Sakai
-