FAQ – v.2
Ejercicio 2 – Preguntas frecuentes

josé a. mañas

29.2.2016

1 ¿Qué hay que hacer?

1. Implementar un diccionario usando un array ordenado
2. Implementar un diccionario utilizando un árbol binario de búsqueda
3. Medir tiempos de la operación get() en ambos diccionarios

2 ¿Cómo se implementa DiccionarioBinario?

Se puede partir de DiccionarioLineal del ejercicio 1, aprovechando los chequeos de casos excepcionales.

Hay que implementar la función busca(clave) que usa el mecanismo de búsqueda binaria para encontrar en qué posición está la clave.

Se recomienda pasar pruebas de busca().

2.1 Método get()

Use el método busca(). Si encuentra la clave, devuelva el valor asociado. Si no, devuelva null.

2.2 Método put()

Puede usar el método busca() para saber dónde está la clave.

- Si la encuentra, cambie el valor.
- Si no la encuentra, desplace los datos desde la posición donde debe estar la clave hasta el final, haciendo un hueco para la nueva clave

2.3 Método remove()

Puede usar el método busca() para saber dónde está la clave.

- Si la encuentra, desplace los datos desde el final hasta donde está la clave, cerrando el hueco.
- Si no la encuentra, devuelva null.

2.4 Pruebas

Pase las pruebas del ejecicio 1.

Puede ser interesante poner trazas para ver que el array está en todo momento ordenado, en particular al acabar put() y remove():

```
private void dump() {
    for (int i = 0; i < size(); i++)
        System.out.print(datos[i].getClave() + " ");
```

```
    System.out.println();
}
```

Puede ser interesante chequear que el array es consistente, en particular al acabar put() y remove():

```
    private void check() {
        int pos = 0;
        while (pos < datos.length && datos[pos] != null) {
            if (pos + 1 < datos.length && datos[pos + 1] != null) {
                String k1 = datos[pos].getClave();
                String k2 = datos[pos + 1].getClave();
                if (k1.compareTo(k2) > 0)
                    System.err.println("ERROR: datos desordenados");
                if (k1.equals(k2))
                    System.err.println("ERROR: datos duplicados");
            }
            pos++;
        }
        if (pos != nDatos)
            System.err.println("ERROR: datos mal contados");
        while (pos < datos.length) {
            if (datos[pos] != null)
                System.err.println("ERROR: datos en la zona vacia");
            pos++;
        }
    }
```

3 Medidas de DiccionarioBinario

3.1 Banco de pruebas

Necesita un banco de pruebas como en recogido en

http://www.dit.upm.es/~pepe/doc/adsw/tema2/Meter1Ops.java

debiendo adaptarlo a su código y a los datos de medición que se recomiendan en el enunciado del ejercicio.

3.2 Código preparado para medir

Debe tunear su código para que las llamadas a compareTo() en el método get() llamen a OpMeter.

O sea, donde pone

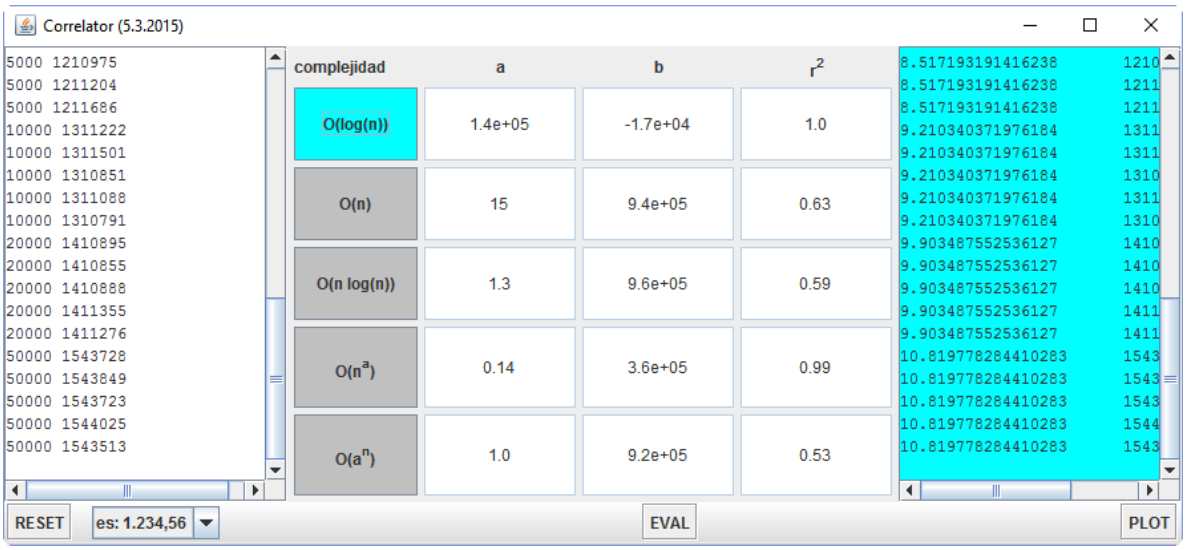
```
int cmp = clave.compareTo(datos[m].getClave());
```

pasar a

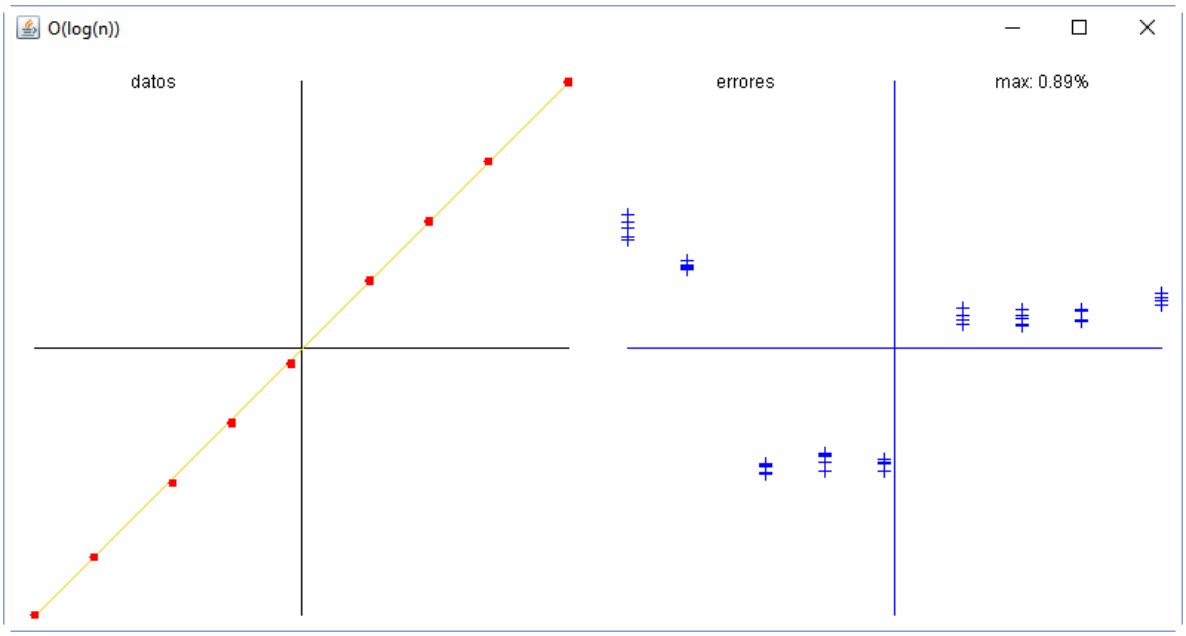
```
int cmp = OpMeter.compareTo(clave, datos[m].getClave());
```

3.3 Entrega

Ejecute, pese los datos a Correlator y presente las 2 pantallas en un documento en PDF:



y



4 ¿Cómo de implementa BST?

Es más complicado de lo que parece y cuesta mucho esfuerzo poner a punto el método remove().

Por eso se recomienda partir de alguna implementación correcta de las muchas que hay en Internet y adaptarla a la interfaz Diccionario:

- casos excepcionales
- métodos públicos, parámetros y resultado

4.1 Método get()

Es muy sencillo como método recursivo y algo más entretenido como bucle iterativo.

4.2 Método put()

Es muy sencillo como método recursivo y algo más entretenido como bucle iterativo.

Hay que ajustar para que cuando detecta una clave existente, modifique su valor sin crear nuevos nodos.

4.3 Método remove()

Es bastante complicado. Hay varios casos. Si al reordenar el árbol se equivoca, el error deja el árbol inconsistente y esa inconsistencia puede aflorar más tarde.

Una particularidad de la adaptación a Diccionario es que tiene que devolver el valor que había antes. Pero la mayor parte de los códigos en Internet sólo explican cómo modificar el árbol.

Se suele combinar ambas funcionalidades usando esta estructura:

```
String found = get(clave);
if (found == null)
    return null;
root = remove(root, clave);
nDatos--;
return found;
```

4.4 Pruebas

Pase las pruebas del ejercicio 1. Quizás le interesa meter más pruebas para árboles variados.

Puede ser interesante chequear que el árbol es consistente en todo momento. Esta función puede ayudar:

class Nodo

```
public void check() {
    if (izq != null) {
        if (izq.clave.compareTo(clave) > 0)
            System.out.printf("ERROR: %s > %s$n", izq.clave, clave);
        izq.check();
    }
    if (der != null) {
        if (clave.compareTo(der.clave) > 0)
            System.out.printf("ERROR: %s > %s$n", clave, der.clave);
        der.check();
    }
}
```

5 Medidas de BST

Ver anteriormente para DiccionarioBinario.

Se puede reutilizar el banco de pruebas simplemente cambiando

new DiccionarioBinario → new BST()