Examen 2 Marzo 2015, preguntas y respuestas

Estructuras de datos y algoritmos (Universitat Politecnica de Valencia)

Resolución del Primer Parcial de EDA (2 de Marzo de 2015) - Puntuación 1.2

1.- La clase `LEGLPIDeComparables` hereda de `LEGListaConPI` e implementa una `ListaConPI` de elementos Comparables. Se pide:

(a) Escribir la cabecera de la clase. **(0.2 puntos)**

```
public class LEGLPIDeComparables<E extends Comparable<E>>
extends LEGListaConPI<E> implements ListaConPI<E> { … }
```

(b) Completar su método `insertar`, que aparece a continuación, tal como se indica en su especificación y utilizando única y exclusivamente los métodos de la interfaz `ListaConPI` (ver Anexo). **(0.3 puntos)**

```
/** Inserta e detrás cada uno de los elementos de una ListaConPI que sean menores que
 * él; si en la Lista no existe ningún elemento menor que e, inserta e en su final.
 * Así, por ejemplo: si la Lista está vacía, inserta e; si la Lista no está vacía
 * PERO no contiene elementos menores que e, inserta e en su final; si la Lista
 * contiene 2 elementos menores que e, inserta e tras cada uno de ellos; etc. */
 public void insertar(E e) {
     inicio(); int menoresQueE = 0;
     while (!esFin()) {
          if (recuperar().compareTo(e) < 0) {
             siguiente(); super.insertar(e); menoresQueE++;
          }
          else siguiente();
     }
     if (menoresQueE == 0)  super.insertar(e);
 }
```

2.- Una fábrica produce monedas de un determinado peso; pero, por un defecto de fabricación, entre ellas existe una moneda de peso inferior al de las demás. Suponiendo que un array `v` contiene las referencias de todas las monedas fabricadas (`String`) y que se dispone de un método `balanza(v, i, j, k, l)` tal que, en tiempo constante, dados dos (sub)array de `v` de igual talla `v[i, j]` y `v[k, l]` ...

- devuelve 0 si las monedas de `v[i, j]` pesan lo mismo que las de `v[k, l]`;
- devuelve un valor negativo si las monedas de `v[i, j]` pesan menos que las de `v[k, l]`;
- devuelve un valor positivo si las monedas de `v[i, j]` pesan más que las de `v[k, l]`;

Se pide escribir un método Divide y Vencerás que, con coste logarítmico en el peor caso, devuelva el índice del array `v` donde se encuentra la referencia de la moneda defectuosa. **(0.4 puntos)**

NOTA: comprueba que tu método funciona bien tanto si el número de monedas de `v` es par como si es impar.

```
public static int farsaMonea(String[] v) { return farsaMonea(v, 0, v.length – 1); }
private static int farsaMonea(String[] v, int i, int j) {
     if (i == j) return i;
     int  m = (i + j) / 2;
     if ((j - i + 1) % 2 == 0) { //si la talla es par
        if (balanza(v, i, m, m + 1, j) < 0) return farsaMonea(v, i, m);
        else return farsaMonea(v, m + 1, j);
     }
     else { // si la talla es impar
         int resC = balanza(v, i, m - 1, m + 1, j);
         if (resC == 0) return m;
         if (resC < 0) return farsaMonea(v, i, m - 1);
         return farsaMonea(v, m + 1, j);
     }
 }
```

3.- Se desea analizar el coste Temporal del siguiente método recursivo: **(0.3 puntos)**

```
private static int metodoR(int[] v, int i, int f) {
   if (i < f) {
     int mitad = (i + f) / 2;
     if (v[mitad] <= 0) {
       if (v[mitad + 1] > 0) return mitad;
       else return metodoR(v, mitad + 1, f);
     }
     else return metodoR(v, i, mitad);
   }
   else return -1;
}
```

Para ello **se pide**:

a) Expresar la talla x del problema en función de los parámetros del método. **(0.05 puntos)**

x = f – i + 1

b) Para una talla x dada, marcar con una cruz la casilla que se considere correcta y rellenar el recuadro en blanco que tenga asociado. **(0.1 puntos)**

☒ Sí existen instancias significativas

Mejor Caso: `v[mitad]` negativo o cero y `v[mitad + 1]` positivo

Peor Caso: los datos del array v son, o bien todos positivos, o bien todos negativos

☐ No existen instancias significativas

porque

c) Escribir las Relaciones de Recurrencia que requiera la respuesta dada en el apartado **b)**; resolverlas y acotar su solución usando los teoremas de coste (ver Anexo). **(0.1 puntos)**

Mejor Caso: $T_{metodoR}^{M}(x > 1) = k$. Por tanto, $T_{metodoR}^{M}(x) \in \Theta(1)$

Peor Caso: $T_{metodoR}^{P}(x > 1) = 1 * T_{metodoR}^{P}(x / 2) + k'$. Por tanto, por Teorema 3 con a=1, c=2 y sobrecarga constante, $T_{metodoR}^{M}(x) \in \Theta(\log x)$

d) A partir de las cotas obtenidas en el apartado c), escribir el coste Temporal Asintótico del método. **(0.05 puntos)**

$T(x) \in \Omega(1)$ y $T(x) \in O(\log x)$

ANEXO

La interfaz `ListaConPI` del paquete `modelos`.

```
public interface ListaConPI<E> {
   void insertar(E e);
   /** SII !esFin() */ void eliminar();
   void inicio();
   /** SII !esFin() */ void siguiente();
   void fin();
   /** SII !esFin() */ E recuperar();
   boolean esFin();
   boolean esVacia();
   int talla();
}
```

Teoremas de coste

Teorema 1: $f(x) = a\cdot f(x - c) + b$, con $b \geq 1$

- si $a=1$, $f(x) \in \Theta(x)$;
- si $a>1$, $f(x) \in \Theta(a^{x/c})$;

Teorema 2: $f(x) = a\cdot f(x - c) + b\cdot x + d$, con b y $d\geq 1$

- si $a=1$, $f(x) \in \Theta(x^2)$;
- si $a>1$, $f(x) \in \Theta(a^{x/c})$;

Teorema 3: $f(x) = a\cdot f(x/c) + b$, con $b \geq 1$

- si $a=1$, $f(x) \in \Theta(\log_c x)$;
- si $a>1$, $f(x) \in \Theta(x^{\log_c a})$;

Teorema 4: $f(x) = a\cdot f(x/c) + b\cdot x + d$, con b y $d\geq 1$

- si $a<c$, $f(x) \in \Theta(x)$;
- si $a=c$, $f(x) \in \Theta(x\cdot \log_c x)$;
- si $a>c$, $f(x) \in \Theta(x^{\log_c a})$;

Teoremas maestros

Teorema para recurrencia divisora: la solución a la ecuación $T(n) = a\cdot T(n/b) + \Theta(n^k)$, con $a\geq 1$ y $b>1$ es:

- $T(n) = O(n^{\log_b a})$ si $a>b^k$;
- $T(n) = O(n^k\cdot \log n)$ si $a=b^k$;
- $T(n) = O(n^k)$ si $a<b^k$;

Teorema para recurrencia sustractora: la solución a la ecuación $T(n) = a\cdot T(n-c) + \Theta(n^k)$ es:

- $T(n) = \Theta(n^k)$ si $a<1$;
- $T(n) = \Theta(n^{k+1})$ si $a=1$;
- $T(n) = \Theta(a^{n/c})$ si $a>1$;