Resolucion del Primer Parcial marzo 2014.pdf

Estructuras de datos y algoritmos (Universitat Politecnica de Valencia)

Resolución del Primer Parcial de EDA (24 de Marzo de 2014)

1.- En el paquete `lineales`, se desea implementar la siguiente subinterfaz de `ListaConPI` mediante una clase que extienda `LEGListaConPI`.

```
public interface ListaConPIPlus<E> extends ListaConPI<E> {
   /** Si el Elemento e está en una Lista Con PI elimina su última aparición y devuelve
    *  true como resultado; sino, si e no está en la lista, devuelve false para advertirlo */
   public boolean eliminarUltimo(E e);
}
```

Se pide: **(3.25 puntos)**

(a) Escribir la cabecera de dicha clase. **(0.75 puntos)**

```
public class LEGListaConPIPlus<E> extends LEGListaConPI<E> implements ListaConPIPlus<E> {…}
```

(b) Implementar el método `eliminarUltimo`. Sólo se podrán utilizar los métodos de `ListaConPI` (ver Anexo). **(2 puntos)**

```
public boolean eliminarUltimo(E e) {
    this.inicio(); int pos = 0, posUltima = -1;
    while (!this.esFin()) {
       if (this.recuperar().equals(e)) posUltima = pos;
       pos++;
       this.siguiente();
    }
    if (posUltima == -1) return false;
    this.inicio(); pos = 0;
    while (pos < posUltima) {
       pos++;
       this.siguiente();
    }
    this.eliminar();
    return true;
}
```

(c) Utilizando la notación asintótica (O y Ω o bien Θ) y tomando como talla del problema el número de elementos de la `ListaConPI`, escribe el coste de `eliminarUltimo`. **(0.5 puntos)**

No hay Mejor ni Peor de los Casos porque en el primer bucle siempre se recorre toda la Lista con PI; por tanto, si x es su talla, la complejidad temporal del método es Θ(x).

2.- Se tiene un conjunto de bolas blancas y negras dispuestas sobre un array `v` de modo que al principio están todas las blancas y a continuación las negras. Considerando que una bola se representa mediante un `String` cuyo valor es "`blanca`" o "`negra`" y que al menos hay una de cada color, s**e pide:** **(3.5 puntos)**

(a) Diseñar un método Divide y Vencerás `contarBlancas` que, con coste logarítmico, devuelva el número de bolas blancas que hay en `v`. **(3 puntos)**

```
public static int contarBlancas(String[] v) {
    return contarBlancas(v, 0, v.length - 1)
}
private static int contarBlancas(String[] v, int i, int f) {
   // Búsqueda con garantía de éxito de la última bola blanca de v:
   // usando una estrategia DyV, se divide el problema en 2 subproblemas
   int mitad = (i + f) / 2;
   // Para que en mitad esté la última bola blanca, al menos v[mitad] es blanca
   if (v[mitad].equals("blanca") {
       //si v[mitad+1] es negra entonces mitad+1 es la solución
       if (v[mitad + 1].equals("negra") return mitad + 1;
       // sino, la última bola blanca solo puede estar a partir de mitad+1,
       // i.e. en el subArray v[mitad+1...f]
       else return contarBlancas(v, mitad + 1, f);
    }
    // Si v[mitad] es una bola negra, la última blanca solo puede estar antes,
    // i.e. en el subArray v[i...mitad-1]
    else return contarBlancas(v, i, mitad - 1);
}
```

(b) Justificar su coste logarítmico utilizando los teoremas de los costes. **(0.5 puntos)**

En el Peor de los Casos la talla `x` decrece geométricamente en la única llamada que se produce y el coste del resto de operaciones (la sobrecarga) es constante. Por tanto, aplicando el Teorema 3 con a = 1 y c = 2, se tiene que el coste es logarítmico, i.e. $T(x) \in O(\log x)$.

3.- Se dispone de los siguientes métodos `examen`: **(3.25 puntos)**

```
public int examen(int[] v) { return examen(v, 0, v.length-1); }
protected int examen(int[] v, int p, int f) {
  if (p > f) return 0;
  else if (p == f) return v[p];
  else {   int c = (p + f) / 2;
           int sol1 = examen(v, p, c);  int sol2 = examen(v, c + 1, f);
           if (sol1 == sol2) return sol1;
           if (sol1 > 0) if (test(v, p, f, sol1)) return sol1;
           if (sol2 > 0) if (test(v, p, f, sol2)) return sol2;
           return 0;
  }
}
```

Sabiendo que todos los elementos de `v` son mayores o iguales que cero y que el método `test` tiene siempre un coste lineal con la talla del subarray sobre el que se aplica, **se pide**:

(a) Si `v.length=8`, completar los siguientes enunciados sobre el método `protected examen`: **(0.5 puntos)**

- La llamada `examen(v, 6, 6)` es la penúltima que se genera y `examen(v, 7, 7)` la última.
- La llamada `examen(v, 0, 0)` es la primera que se resuelve y `examen(v, 0, 7)` la última.

(b) Marcar con una cruz la casilla correspondiente al tamaño o talla, `x`, del problema que resuelve el método `protected examen`. **(0.25 puntos)**

☐ `x = v.length` ☐ `x = (p + f) / 2` ☒ `x = f - p + 1` ☐ `x = p - f + 1`

☐ Según sea el Peor o Mejor de los Casos, `x=v.length` o $x \leq 1$ respectivamente.

(c) Para una talla `x` dada, marcar aquellas que entre las siguientes (una o más) sean las instancias significativas que presenta el método `protected examen`. **(0.5 puntos)**

- ☐ No existen instancias significativas, pues siempre se genera el mismo número de llamadas recursivas.
- ☐ El Mejor de los Casos se da cuando el subarray v[p...f] que se le pasa como argumento en la llamada principal tiene cero o un elemento, i.e. cuando `x=0` o `x=1` respectivamente.
- ☒ El Mejor de los Casos se da cuando el subarray v[p...f] que se le pasa como argumento en la llamada principal tiene todos sus elementos iguales.
- ☐ El Peor de los Casos se da cuando el subarray v[p...f] que se le pasa como argumento en su llamada principal tiene todos sus elementos iguales.
- ☒ El Peor de los Casos se da cuando en todas las llamadas sucede que sol1 es distinto de sol2.

(d) En función de la respuesta dada en el apartado anterior, completar y resolver las Ecuaciones de Recurrencia que expresan la Complejidad Temporal del método, indicando su solución con notación asintótica. Si no hubieran instancias significativas, usa el espacio reservado al Peor de los Casos en el siguiente recuadro. **(1.5 puntos)**

Caso base: si $x \leq 1$ entonces $T(x) = k$

Mejor de los Casos, caso general: Si $x > 1$ entonces $T^M(x) = 2 \cdot T^M(x/2) + k'$. Por tanto, por Teorema 3 con $a = c = 2$ y sobrecarga constante, $T^M(x) \in \Theta(x)$

Peor de los Casos, caso general: Si $x > 1$ entonces $T^P(x) = 2 \cdot T^P(x/2) + k'' \cdot x$. Por tanto, por Teorema 4 con $a = c = 2$ y sobrecarga lineal, $T^P(x) \in \Theta(x \cdot \log x)$

(e) En función del resultado del apartado (d), indicar el coste Temporal Asintótico de dicho método. **(0. 5 puntos)**

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

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 $a=1$;