Recuperación Primer Parcial de PRG - ETSInf
Fecha: 19 de junio de 2012. Duración: 2 horas

1. (2 puntos) Sea `a` un array de `int`, y `x` un `int`. Se pide un método recursivo que indique si los elementos de `a` forman una progresión geométrica de razón `x`, es decir, si cada componente `a[i+1]` del array vale `a[i]*x`. Por ejemplo, para `a = {3,6,12,24,48}` y `x=2` el método debe devolver `true`, para `a = {3,6,12,33,48}` y `x=2` el método debe devolver `false`. Si sólo hay un elemento, se entiende que es una progresión geométrica sea cual sea el valor de `x`.

   Indicar cuál deberá ser la primera llamada.

   Solución:

```
/** Comprueba si las componentes de a[i..a.length-1], 0<=i<a.length,
 *  forman una progresión geométrica de razón x
 */
public static boolean geometrica(int[] a,int i,int x){
   if (i==a.length-1) return true;
   else { if (a[i+1]==a[i]*x) return geometrica(a,i+1,x);
          else return false;
        }
}
```

   La primera llamada debería ser `geometrica(a,0,x)`.

2. (2 puntos) Escribir un método recursivo que dado un entero $n \geq 0$, escriba en la salida estándar, y en la misma línea los valores $-n - (n-1) \ldots$ `-2 -1 0 1 2` $\ldots (n-1)\ n$. Por ejemplo, para $n = 3$, en la salida se debe escribir:

```
-3 -2 -1 0 1 2 3
```

   Solución:

```
/** n>=0 */
public static void escribe(int n) {
  if (n==0) System.out.print(0);
  else { System.out.print(-n + " ");
         escribe(n-1);
         System.out.print(" " + n);
       }
}
```

3. (2.5 puntos) Dado el siguiente método:

```
/** n>=0, 1<=x<=9 */
public static boolean buscarX(int n,int x){
   if (n>0)
       if (n%10==x) return true;
       else return buscarX(n/10,x);
   else return false;
}
```

Se pide:

a) Indicar cuál es la talla del problema y qué expresión la define.

b) Determinar si existen instancias significativas. Si hay, identificar las que representan los casos mejor y peor del algoritmo.

c) Escribir la ecuación de recurrencia del coste temporal en función de la talla para cada uno de los casos si los hubiera, o una única ecuación si sólo hubiese un caso. Hay que resolverla por sustitución.

d) Expresar el resultado anterior usando notación asintótica.

Solución:

a) La talla del problema es el valor del argumento del método, es decir, `n`.

b) Sí que hay instancias significativas, es una búsqueda. En el mejor caso, `x` es la cifra de las unidades de `n`, y en el caso peor ninguna cifra de `n` es `x`.

c) En el caso mejor $T^m(n) \in \Theta(1)$. En el caso peor, la ecuación de recurrencia para el coste, en pasos de programa, es:

$$T^p(n) = \begin{cases} T^p(n/10) + 1 & \text{si } n > 0 \\ 1 & \text{si } n = 0 \end{cases}$$

Resolviendo por sustitución:

$T^p(n) = T^p(n/10) + 1 = T^p(n/10^2) + 2 = \ldots = T^p(n/10^i) + i$. Cuando $i = 1 + \lfloor log_{10} n \rfloor$ se llega al caso base en el que $T^p(0) = 1$.

Con lo que $T^p(n) = 2 + \lfloor log_{10} n \rfloor$.

d) En notación asintótica: $T^m(n) \in \Theta(1)$ y $T^p(n) \in \Theta(\log n)$. Por tanto, $T(n) \in \Omega(1)$ y $T(n) \in O(\log n)$.

4. (3.5 puntos) Sea `a` un array $\{\mathtt{a}_0, \mathtt{a}_1, \mathtt{a}_2, \ldots, \mathtt{a}_{n-1}\}$ de `double`, que representa los coeficientes de un polinomio $a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$.

Para calcular el valor del polinomio en un x dado, se proponen los siguientes métodos:

Método 1:

```
/** a.length>=1 */
public static double polinomio1(double[] a, double x) {
    double result = a[0];
    for(int i=1; i<a.length; i++){
        double pot = 1;
        for(int k=1; k<=i; k++) pot = pot*x;
        result += a[i]*pot;
    }
    return result;
}
```

Método 2:

```
/** a.length>=1 */
public static double polinomio2(double[] a, double x) {
    double result = a[a.length-1];
    for(int i=a.length-2; i>=0; i--)
        result = result*x + a[i];
    return result;
}
```

Se pide:

a) **Para cada método:**

1. Indicar cuál es la tamaño o talla del problema, así como la expresión que la representa.
2. Identificar, en caso de que hubiese, las instancias del problema que representan el caso mejor y peor del algoritmo.
3. Elegir una unidad de medida para la estimación del coste (pasos de programa, instrucción crítica) y de acuerdo con ella, obtener una expresión matemática del coste temporal del programa lo más precisa posible, ya sea a nivel global o en las instancias más significativas si las hay.
4. Expresar el resultado anterior utilizando notación asintótica.

b) Indicar cuál de los dos métodos es más eficiente y porqué.

Solución:

a) **Método 1.**

1 La talla $n =$`a.length`, es decir, la cantidad de elementos de `a`.

2 Sólo hay una instancia. El algoritmo sólo tiene una función de coste $T(n)$, válida para cualquier entrada de talla n.

3 En pasos de programa,

$$T(n) = 1 + \sum_{i=1}^{n-1}(1+i) = 1 + (2 + 3 + \ldots + n) = \frac{(n+1)\cdot n}{2} \in \Theta(n^2)$$

4 En resumen, $T(n) \in \Theta(n^2)$.

Método 2.

1) Ídem que para el método 1.

2) Ídem que para el método 1.

3) En pasos de programa,

$$T(n) = 1 + \sum_{i=1}^{n-1} 1 = n \in \Theta(n)$$

4) En resumen, $T(n) \in \Theta(n)$.

b) El primer método es cuadrático respecto a la talla del problema y el segundo es lineal, por lo que se puede concluir que el segundo es más eficiente.

Del análisis realizado se observa que el primer método recalcula completamente la potencia de `x` elevado a `i` en cada pasada del bucle, operación que va resultando más costosa a medida que aumenta `i`. El segundo método lo evita haciendo que las potencias de `x` acumuladas en `result` vayan aumentado su grado en 1 en cada pasada del bucle.