PRG - ETSInf. TEORÍA. Curso 2013-14. Recuperación Parcial 1. 23 de junio de 2014. Duración: 1h 50m.

1. $\boxed{\text{3 puntos}}$ Se desea diseñar un método recursivo que determine si cierto array de enteros `v` contiene los elementos iniciales de una sucesión de Fibonacci. Para ello el array se proporciona al menos con 3 elementos. Recuérdese que la sucesión de Fibonacci es la siguiente: 0,1,1,2,3,5,8... Esto es, el término n-ésimo es `n` si n=0 o n=1; en otro caso, se calcula como la suma de los dos términos anteriores ((n-1)-ésimo y (n-2)-ésimo).

   (a) (2.5 puntos). Escribir el método de acuerdo a las consideraciones anteriores.

   Solución: Solución mediante recorrido ascendente:

```
/** v.length >= 3, 2 <= i <= v.length */
public static boolean es_fibo (int[] v, int i) {
  if (i==v.length) return true;
  if (v[0]!=0 || v[1]!=1) return false;
  else return (v[i]==v[i-1]+v[i-2]) && es_fibo(v,i+1);
}
```

   Solución mediante recorrido descendente:

```
/** v.length >= 3, 1 <= i <= v.length-1 */
public static boolean es_fibo (int[] v, int i) {
  if (i<=1) return (v[0]==0 && v[1]==1);
  else return (v[i]==v[i-1]+v[i-2]) && es_fibo(v,i-1);
}
```

   (b) (0.5 puntos). De acuerdo a la implementación del método anterior, indicar la llamada inicial para que se verifique la propiedad sobre todo el array.

   Solución: Asumiendo `v` un array con al menos 3 elementos, la llamada inicial sería, para la solución recursiva ascendente:

   `es_fibo(v,2)`

   Y para la solución recursiva descendente:

   `es_fibo(v,v.length-1)`

2. $\boxed{\text{3 puntos}}$ El siguiente método recursivo, `inversion(String)`, obtiene un `String` con la inversión de los caracteres de la que recibe como argumento. Por ejemplo, la inversión de la cadena `hola` es `aloh`.

```
public static String inversion (String s) {
    if (s.length() <= 1) return s;
    else return inversion(s.substring(1)) + s.charAt(0);
}
```

   Se quiere estudiar su coste temporal en las dos situaciones siguientes:

   1. Suponer que tanto la operación `substring(int)` como la operación de concatenación (operador `+`) tienen un **coste constante** con la longitud del String `s`.
   2. Asumir que la operación `substring(int)` tiene un **coste lineal** con la longitud del String `s`, mientras que la operación de concatenación (operador `+`) tiene un **coste constante** con el número total de caracteres que se concatenan.

   Para cada una de las dos situaciones **se pide**:

   (a) (0.25 puntos) Indicar cuál es el tamaño o talla del problema, así como la expresión que lo representa.

   Solución: Para ambas situaciones, la talla del problema es el número de caracteres del `String s`, que cambiará en cada llamada recursiva. La expresión que la define es `s.length`, que llamaremos `n`.

   (b) (0.5 puntos) Indicar si existen diferentes instancias significativas para el coste temporal del algoritmo, e identificarlas si es el caso.

Solución: Para ambas situaciones, no existen instancias significativas porque se trata de un recorrido sobre el `String s`.

(c) (1.5 puntos) Escribir las ecuaciones de recurrencia del coste temporal en función de la talla, resolviéndolas por sustitución.

Solución:
Para el caso de `substring` y operador de concatenación con costes constantes:

$$T(n) = \begin{cases} k' & si\ n \leq 1 \\ T(n-1) + k & si\ n > 1 \end{cases}$$

siendo k y k' constantes positivas, en alguna unidad de tiempo. Resolviendo por sustitución:

$$\begin{aligned} T(n) &= T(n-1) + k = T(n-2) + 2k = T(n-3) + 3k = \cdots = \\ &= T(n-i) + i \cdot k = \cdots = \\ &(caso\ base:\ n-i=1,\ i=n-1) \\ &= T(1) + (n-1)k = k' + nk - k \end{aligned}$$

Para el caso de `substring` con coste lineal y operador de concatenación con coste constante:

$$T(n) = \begin{cases} k' & si\ n \leq 1 \\ T(n-1) + kn & si\ n > 1 \end{cases}$$

siendo k y k' constantes positivas, en alguna unidad de tiempo. Resolviendo por sustitución y menospreciando términos de orden inferior:

$$\begin{aligned} T(n) &= T(n-1) + kn = T(n-2) + k(n-1) + kn = T(n-3) + k(n-2) + k(n-1) + kn = \cdots = \\ &= T(n-i) + \textstyle\sum_{j=n-(i-1)}^{n} kj = \cdots = \\ &(caso\ base:\ n-i=1,\ i=n-1,\ n-(i-1)=2) \\ &= T(1) + \textstyle\sum_{j=2}^{n} kj = k' + k\Big(\frac{n(n+1)}{2} - 1\Big) \end{aligned}$$

(d) (0.5 puntos) Expresar el resultado anterior en notación asintótica.

Solución:
Para el caso de `substring` y operador de concatenación con costes constantes: $T(n) \in \theta(n)$.
Para el caso de `substring` con coste lineal: $T(n) \in \theta(n^2)$.

(e) (0.25 puntos) ¿Cuáles de las dos situaciones crees que es la más favorable desde un punto de vista de coste temporal? Justifica tu respuesta.

Solución: A la vista del coste temporal asintótico es más favorable la situación donde `substring` presenta coste constante puesto que el algoritmo tiene un coste lineal con la talla del problema.

3. $\boxed{\text{4 puntos}}$ El siguiente algoritmo iterativo devuelve un entero correspondiente a la suma máxima de los valores almacenados en posiciones consecutivas de un array dado `a`.

```
/** Precondición: a.length >= 1 */
public static int metodo (int[] a) {
    int n = a.length, max = a[0];
    for (int i=0; i<=n-1; i++) {
        int suma = 0;
        for (int j=i; j<=n-1; j++) {
            suma = suma + a[j];
            if ( suma > max ) max = suma;
        }
    }
    return max;
}
```

Se pide:

a) (0.5 puntos) Indicar cuál es el tamaño o talla del problema, así como la expresión que lo representa.

> **Solución:** La talla es la longitud del array, `a.length`, que llamaremos `n`.

b) (0.5 puntos) Indicar si existen diferentes instancias significativas para el coste temporal del algoritmo, e identificarlas si es el caso.

> **Solución:** No hay instancias significativas, porque es un problema de recorrido de un array.

c) (2 puntos) Elegir una unidad de medida para el coste (pasos de programa, instrucción crítica) y acorde con ella, obtener una expresión lo más precisa posible del coste temporal del programa (para el caso mejor y el caso peor si es el caso).

> **Solución:**
> En pasos de programa:
>
> $$T(n) = 1 + \sum_{i=0}^{n-1}(1 + \sum_{j=i}^{n-1} 1) = 1 + \sum_{i=0}^{n-1}(1 + n - i) = 1 + n + n^2 - \frac{n(n-1)}{2} = 1 + \frac{3}{2}n + \frac{1}{2}n^2 \ p.p.$$
>
> Si tomamos como instrucción crítica: `suma = suma + a[j];` y considerándola de coste unitario:
>
> $$T(n) = \sum_{i=0}^{n-1}\sum_{j=i}^{n-1} 1 = \sum_{i=0}^{n-1}(n-i) = n\sum_{i=0}^{n-1} 1 - \sum_{i=0}^{n-1} i = n^2 - \frac{n(n-1)}{2} = n^2 - \frac{1}{2}(n^2 - n) = \frac{1}{2}n^2 + \frac{1}{2}n \ \ \textit{instr. críticas}$$

d) (1 punto) Expresar el resultado anterior en notación asintótica.

> **Solución:** El coste es cuadrático con la talla del problema, $T(n) \in \theta(n^2)$.