PROGRAMACIÓN 2. Curso 2017-18. 2ª prueba voluntaria de evaluación. Grupo 411

Esta es la segunda prueba de **evaluación voluntaria** que se plantea en la asignatura ***Programación 2*** a los alumnos matriculados en el grupo de mañanas. Debe ser resuelta individualmente y deberá ser entregada **antes de las 24 horas del viernes 4 de mayo de 2018**.

Para cada uno de los dos ejercicios se deben resolver las siguientes cuestiones, tanto en **tiempo** como en **espacio**:

- Definir el tamaño del problema
- Analizar y justificar si existen casos mejor y peor, dependiendo de los valores de los datos involucrados
- Establecer las funciones de coste asintótico (dos distintas si existieran mejor y peor caso)
- Caracterizar el orden de las funciones anteriores.

Es importante que las soluciones estén claramente explicadas y justificadas.

Ejercicio 1 [6.0 puntos]

```
//-----------------------------------------------------------
//Pre:  0 ≤ f ∧ f < N ∧ 0 ≤ c ∧ c < N
//Post:  Irrelevante
template<typename T, int N>
void processElement(T M[N][N], int f, int c) {
    for(int i=0; i<c; i++) {                                  // t_a
        M[f][c] = M[f][c]+M[f][i];                            // t_b
    }
}
//-----------------------------------------------------------
//Pre:   Irrelevante
//Post:  Irrelevante
template<typename T, int N>
void process(T M[N][N]) {
    for (int i=0; i<N; i++) {                                 // t_c
        int j = N-1;                                          // t_d
        while (j>=0) {                                        // t_e
            processElement(M, i, j);                          // t_f
            j--;                                              // t_g
        }
    }
}
```

Una posible solución

Dado que $process$ invoca a $processElement$, habrá que llevar a cabo el cálculo para ambas.

Coste en tiempo

- El coste en tiempo de $processElement$ depende únicamente del parámetro c, que marcará el número de iteraciones del bucle. Por lo tanto, este será el tamaño de problema elegido para este procedimiento. Por otro lado, el tiempo que cada iteración tarde no depende de los valores de los datos, con lo que no cabe distinguir casos mejor y peor. $t_{pE}(c) = t_{inv} + t_a + \sum_{i=0}^{c-1}(t_a + t_b) = (t_a + t_b) \cdot c + t_a \in O(c)$
- Tampoco aquí cabe la distinción entre casos mejor y peor ya que siempre se ejecuta una única instrucción. El tiempo de ejecución de $process$ depende exclusivamente de N, ya que este valor determina el número de iteraciones de ambos bucles. Así

$$
\begin{aligned}
t_p(N) &= t_{inv} + t_c + \sum_{i=0}^{N-1}(t_c + t_d + t_e + \sum_{j=0}^{N-1}(t_e + t_{pE}(j) + t_g)) && (1)\\
&= t_{inv} + t_c + \sum_{i=0}^{N-1}(t_c + t_d + t_e + \sum_{j=0}^{N-1}(t_e + K \cdot j + t_g)) && (2)\\
&= t_{inv} + t_c + \sum_{i=0}^{N-1}(t_c + t_d + t_e + (t_e + t_g) \cdot N + K \cdot \sum_{j=0}^{N-1} j) && (3)\\
&= t_{inv} + t_c + \sum_{i=0}^{N-1}(t_c + t_d + t_e + (t_e + t_g - 0{,}5 \cdot K) \cdot N + 0{,}5 \cdot K \cdot N^2)\\
&= t_{inv} + t_c + (t_c + t_d + t_e) \cdot N + (t_e + t_g - 0{,}5 \cdot K) \cdot N^2 + 0{,}5 \cdot K \cdot N^3 \in O(N^3)
\end{aligned}
$$

  Se trata de un coste polinomial (cúbico). Alternativamente, podríamos haber razonado en la ecuación (3), diciendo que, asintóticamente, $K \cdot \sum_{j=0}^{N-1} j$ es del orden $O(N^2)$ (suma de los $N-1$ primeros números naturales) y, por lo tanto, sustituir esa suma por $K' \cdot N$, siguendo a partir de ahí.

Coste en memoria

- El coste en memoria de $processElement$ va a ser constante, por lo que se podría elegir como tamaño del problema cualquiera. Por homogeneidad con el caso de coste en tiempo, tomaremos c. La función de coste es $mem_{pE}(c) = mem_{inv} + m_{ref} + 3 \cdot m_{int} \in O(1)$
- El coste en memoria de $process$ también va a ser constante, y depende del coste de $processElement(M, i, j)$ que, al ser este $O(1)$, lo aproximaremos por un valor constante K $mem_p(c) = mem_{inv} + m_{ref} + 2 \cdot m_{int} + K \in O(1)$. Por lo tanto, coste constante

Ejercicio 2 [4.0 puntos]

```
//------------------------------------------------------------
//Pre:   Irrelevante
//Post:  Irrelevante
//Com:  Asumimos que el coste en tiempo de ejecutar ‘‘=’’ y de evaluar ‘‘+’’
//       para datos de tipo T es O(1)
template<typename T, int N>
void trickyI (T M[N][N], int i) {
    if (i>0) {                                          // t_a
        if (i == 1) {                                   // t_b
            M[1][1] = M[0][0]+M[1][1];                  // t_c
        } else {
            trickyI (M, i-2);                           // t_d
            trickyI (M, i-1);                           // t_e
            M[i][i] = M[i-1][i-1]+M[i-2][i-2];          // t_f
        }
    }
}
//------------------------------------------------------------
//Pre:   Irrelevante
//Post:  Irrelevante
template<typename T, int N>
void tricky (T M[N][N]) {
    trickyI (M, N-1);                                   // t_g
}
```

Una posible solución

Dado que $tricky$ invoca a $trickyI$, habrá que llevar a cabo el cálculo para ambas. Con los mismos argumentos del ejercicio anterior, no hay casos mejor y peor ni en tiempo ni en memoria.

Coste en tiempo

- El número de invocaciones recursivas de $trickyI$ depende del parámetro i, que elegimos como tamaño del problema. Por otro lado, todas las invocaciones ejecutan las mismas instrucciones, por lo que no hay casos mejor y peor. La ecuación en recurrencias que establece el coste es $t_{tI}(i) = t_{inv} + t_a + t_f + t_{tI}(i-1) + t_{tI}(i-2)$. Se trata de una ecuación lineal, de coeficientes constantes, no homogénea, que podemos escribir como $t_{tI}(i) - t_{tI}(i-1) - t_{tI}(i-2) = (t_{inv} + t_a + t_f) \cdot 1^i$.

  La ecuación característica es $(x^2 - x - 1)(x-1) = 0$, que da como raíces $x_1 = 1$ con multiplicidad 1 y $x_2 = \frac{1+\sqrt{5}}{2}$ y $x_3 = \frac{1-\sqrt{5}}{2}$, ambas con multiplicidad 1. La tercera no aporta nada a la función de coste, ya que daría valores negativos. Por lo tanto, la función de coste en tiempo es $t_{tI}(i) = K_1 \cdot 1^i + K_2 \cdot (\frac{1+\sqrt{5}}{2})^i$, siendo K_1 y K_2 constantes. Es decir $t_{tI}(i) \in O(1{,}6^i)$

- En el caso de $tricky$ el tamaño del problema es claramente N, y la función de coste será $t_t(N) = t_{tI}(N-1) \in O(1{,}6^N)$. Se trata de un coste exponencial.

Coste en memoria

- En el caso de $trickyI$, la memoria requerida viene determinada por el número de invocaciones recursivas, que depende de i. A diferencia del tiempo, la memoria requerida por las dos

invocaciones recursivas no se acumula, ya que la invocación $trickyI(M, i-2)$ deberá terminar (y, por tanto, liberar la memoria requerida) antes de ejecutar la $trickyI(M, i-1)$. Así pues, $mem_{tI}(i) = mem_{inv} + mem_{ref} + mem_{int} + mem_{tI}(i-1)$ que se puede escribir como $mem_{tI}(i) - mem_{tI}(i-1) = (mem_{inv} + mem_{ref} + mem_{int}) \cdot 1^i$. La ecuación característica es $(x-1)(x-1) = 0$, con raíz $x = 1$ de multiplicidad 2, por lo que la solución es $mem_{tI}(i) = (K_1 + K_2 \cdot i) \cdot 1^i \in O(i)$

- La función de coste en memoria para $tricky$ será la $mem_t(N) = mem_{ref} + mem_{tI}(N-1) \in O(N)$. Coste lineal.

Modo de entrega

Cada alumno entregará al profesor de su grupo un informe con las soluciones planteadas, en papel (ya sea manuscrito o impreso). No se admitirán trabajos presentados de otra forma, ni fuera de plazo o plagiados (Total o parcialmente).