EjercicioComplejidad1

Estructuras de Datos

Tema 1: tipos abstractos de datos y algoritmia

1º Grado en Ingeniería de la Computación
© Profesor Dr. Carlos Grima Izquierdo (www.carlosgrima.com)
URJC (www.urjc.es)

Calcula el T(n) y a continuación su O(n) en los siguientes dos algoritmos (el propósito de los algoritmos no tiene importancia). Para ello, sigue los siguientes pasos:

1. Calcula primero el T(n) de cada línea, el número de operaciones elementales (OE) de cada línea en función del parámetro "n" de la función
2. Calcula el T(n) de cada iteración en cada bucle
3. Calcula el T(n) de cada bucle entero, a partir del T(n) de cada iteración
4. Calcula el T(n) total del algoritmo
5. Calcula el O(n) del algoritmo

La función comprobar() tiene un T(n)=n^2+3 y funcionCompleja() tiene un T(n)=2n^3+5n^2+2n+1.

```
1 void hacerTarea1(int n) {
2     int r;
3     int i = 10;
4     while (i >= 0 && comprobar(n, 0) == 1) {
5         int pos = 0;
6         pos = funcionCompleja(n) - 1;
7         i--;
8     }
9 }
```

Vamos paso a paso:

- 2 OE en la línea 1: salto hasta el comienzo de la función y declaración del parámetro
- 1 OE en la línea 2
- 2 OE en la línea 3
- De las líneas 4 a la 8 hay un total de 22n^3+67n^2+22n+196 OE, desglosados de la siguiente manera:
    - En cada iteración del bucle hay un total de 2n^3+6n^2+2n+17 OE, desglosado de la siguiente manera:
        - Comprobar la condición (línea 4) tiene n^2+8, desglosado de la siguiente manera:
            - 1 OE para comprobar i>=0

            - 1 OE por el &&
            - 1 OE por el ==
            - n^2+3 OE por la función comprobar()
            - 2 OE por la asignación de n y 0 a los parámetros de la función comprobar()
        - 2 OE en la línea 5
        - En la línea 6 hay $2n^3+5n^2+2n+4$ OE, desglosado de la siguiente manera:
            - 1 OE por la asignación
            - 1 OE por la resta
            - $2n^3+5n^2+2n+1$ OE por funcionCompleja()
            - 1 OE por la asignación de n al parámetro formal de funcionCompleja()
        - 2 OE en la línea 7 (suma y asignación)
        - 1 OE por el salto de la línea 8 a la 4
    - La comprobación de la última condición (la que no se cumple) es n^2+9, porque:
        - La comprobación de la condición es n^2+8 OE
        - Como la última condición no se cumple, hay que añadir 1 OE por el salto desde la línea 4 a la 9.
    - Para averiguar las vueltas que da el bucle, tenemos que recordar que siempre hay que hacerlo con el peor caso. El peor caso es dar el mayor número de vueltas, y para ello la función comprobar(), en el peor caso, devolverá siempre 1. Por tanto sólo nos fijamos en lo que hay antes del &&. Ésto último permitirá 11 iteraciones.
    - Por tanto, de las líneas 4 a la 8 hay un total de $11*(2n^3+6n^2+2n+17)+(n^2+9) = 22n^3+67n^2+22n+196$ OE
- En la línea 9 hay un salto (1 OE más) hasta el lugar en donde llamamos a hacerTarea1() (ej: desde el main o desde otra función).

Sumando todo, la función hacerTarea1() tiene un $T(n) = 22n^3+67n^2+22n+202$ OE.

Su complejidad espacial es $O(n^3)$, pues nos quedamos con el mayor sumando del polinomio, despreciando su constante multiplicativa. Decimos que $T(n) \in O(n^3)$.

```
1  void hacerTarea2(int n) {
2      int r;
3      int i = n - 1;
4      while (i >= 0 && comprobar(n, 1) == 1) {
5          int pos = 0;
6          pos = n * funcionCompleja(n) - 1;
7          i--;
8      }
9  }
```

De nuevo, vamos a paso a paso:

- 2 OE en la línea 1: salto hasta el comienzo de la función y declaración del parámetro
- 1 OE en la línea 2
- 3 OE en la línea 3 (declaración, asignación y resta)
- De las líneas 4 a la 8 hay un total de $2n^4+6n^3+3n^2+18n+9$ OE, desglosados de la siguiente manera:
    - En cada iteración del bucle hay un total de $2n^3+6n^2+2n+18$ OE, desglosado de la siguiente manera:
        - Comprobar la condición (línea 4) tiene n^2+8, por las mismas razones que en hacerTarea1()
        - 2 OE en la línea 5
        - En la línea 6 hay $2n^3+5n^2+2n+5$ OE, desglosado de la siguiente manera:
            - 1 OE por la asignación
            - 1 OE por la multiplicación
            - 1 OE por la resta
            - $2n^3+5n^2+2n+1$ OE por funcionCompleja()
            - 1 OE por la asignación de n al parámetro formal de funcionCompleja()
        - 2 OE en la línea 7 (suma y asignación)
        - 1 OE por el salto de la línea 8 a la 4
    - La comprobación de la última condición (la que no se cumple) es n^2+9, porque:
        - La comprobación de la condición es n^2+8 OE
        - Como la última condición no se cumple, hay que añadir 1 OE por el salto desde la línea 4 a la 9.
    - Para averiguar las vueltas que da el bucle, tenemos que recordar que siempre hay que hacerlo con el peor caso. El peor caso es dar el mayor número de vueltas, y para ello la función comprobar(), en el peor caso, devolverá siempre 1. Por tanto sólo nos fijamos en lo que hay antes del &&. Ésto último permitirá n iteraciones (son $(n-1)+1=n$ porque el 0 hay que contarlo también en este caso).

    - Por tanto, de las líneas 4 a la 8 hay un total de $n*(2n^3+6n^2+2n+18)+(n^2+9) = 2n^4+6n^3+3n^2+18n+9$ OE
- En la línea 9 hay un salto (1 OE más) hasta el lugar en donde llamamos a hacerTarea2() (ej: desde el main o desde otra función).

Sumando todo, la función hacerTarea2() tiene un $T(n) = 2n^4+6n^3+3n^2+18n+16$ OE.

Su complejidad espacial es $O(n^4)$, pues nos quedamos con el mayor sumando del polinomio, despreciando su constante multiplicativa. Decimos que $T(n) \in O(n^4)$.

Como hacerTarea2() tiene una complejidad temporal peor que hacerTarea1(), hacerTarea2() es peor algoritmo (tarda más tiempo en completarse) que hacerTarea1(), con un n grande y en el peor caso.