Solución del ejercicio 2 de la práctica 3

Cálculo analítico de la complejidad temporal

Tal y como se especifica en el enunciado, la complejidad ha de calcularse en función del parámetro m (complejidad paramétrica). En cualquier caso, no tendría sentido expresarla en función del tamaño del problema ya que en este algoritmo, el tamaño del problema es constante.

La función no presenta caso mejor y peor dado que solo existe una instancia por cada valor de m. Por lo tanto, la expresión de complejidad que se obtenga corresponderá a un *coste exacto* que vendrá dado por el número total de pasos de programa que realiza el bucle interno, o lo que es lo mismo en este caso, el número total de iteraciones que hace dicho bucle interno (ya que el coste de cada iteración es constante).

La siguiente tabla representa una cuenta de dichas iteraciones mediante un desglose en cada una de las k iteraciones del bucle externo

Iteración bucle externo	i	número de iteraciones del bucle interno
$1^{\underline{a}}$	1	1
$2^{\underline{a}}$	3	3
$3^{\underline{a}}$	9	9
...	...	...
k-ésima	3^{k-1}	3^{k-1}

En la segunda columna de la tabla se muestran los valores de i al comienzo de cada iteración del bucle externo (puede tomarse cualquier punto del bucle, no necesariamente al comienzo, pero ha de seguirse siempre el mismo criterio). Se ha escogido la variable i porque es la que determina el número de iteraciones que hará dicho bucle externo (dato que se necesita conocer) y, ademas, de ella depende el número de iteraciones del bucle interno.

El término general de la segunda columna se obtiene siguiendo la progresión numérica y relacionándola con cada iteración k; es decir, ha de expresar el valor que toma i en función de k.

En la tercera columna se desglosa, para cada iteración del bucle externo, el número de iteraciones que hace el interno para cada valor de i concreto. Claramente, el número de pasos (o de iteraciones) que realiza el último bucle coincide con el valor de i con el que comienza. El caso general de esta columna ha de ponerse también en función de k, para obtener sumatorios coherentes.

Puesto que la complejidad viene dada por la suma de todos los elementos de la tercera columna, para obtenerla se necesita conocer el número de iteraciones

que hace el bucle externo, es decir, el mayor valor de k que se va a alcanzar, que depende en última instancia de m. Para conocerlo, hemos de relacionar k con m, algo que resulta sencillo hacer a través de la última iteración, en la que se cumple la ecuación (2.1) (de no ser así no sería la última iteración):

$$\frac{m}{3} < i \leq m \tag{2.1}$$

Por otra parte, a través de la segunda columna de la tabla, se dedujo que en cualquier iteración k, se cumple:

$$i = 3^{k-1} \tag{2.2}$$

Combinando (2.1) y (2.2) se tiene:

$$\frac{m}{3} < 3^{k-1} \leq m \tag{2.3}$$

Donde, por la forma de obtener (2.1), k corresponde a la última iteración, o lo que es lo mismo, al número de iteraciones que hace el bucle externo.

Operando adecuadamente en (2.3), se tiene:[1]

$$m < 3^k \leq 3m$$

$$\lfloor \log_3 m \rfloor < k \leq \lfloor \log_3 m \rfloor + 1$$

Siendo k un número natural, el único que cumple ambas desigualdades es:

$$k = \lfloor \log_3 m \rfloor + 1 \tag{2.4}$$

Conociendo ya el número de iteraciones del bucle externo, la expresión de complejidad temporal de la función algorítmica, viene dada por la suma de todos los elementos de la tercera columna de la tabla (notar que se trata de una serie geométrica de razón 3; con $\lfloor \log_3 m \rfloor + 1$ elementos, siendo 1 el primero):[2]

$$c_e(n) = \sum_{k=1}^{\lfloor \log_3 m \rfloor + 1} 3^{k-1} = \frac{3^{\lfloor \log_3 m \rfloor + 1} - 1}{3 - 1} = \frac{1}{2}(3m - 1) \in \Theta(m).$$

[1] El operador $\lfloor x \rfloor$ devuelve la parte entera de x.

[2] Por simplicidad y asumiendo que k es un número natural, tomar $k = \lfloor \log_3 m \rfloor$ o simplemente $k = \log_3 m$, también puede ser válido siempre que no afecte al orden de complejidad resultante (como ocurre en este caso).

Solución del ejercicio 3 de la práctica 3

Cálculo analítico de la complejidad temporal

Puesto que no se especifica lo contrario en el enunciado, la complejidad temporal ha de expresarse en función del tamaño del problema, al que se le llamará n. Viene dado por el número de elementos que tiene el vector.

El bucle `for` no presenta casos mejor y peor puesto que el número de iteraciones que realiza (todas con coste constante con n) no depende de valores concretos del vector v. Por el contrario, el número de iteraciones que hace el bucle `while` sí que depende de dichos valores.

Por lo tanto, la complejidad del algoritmo depende del contenido del vector. En estos casos, para expresarla se distingue entre el mejor de los casos (el más favorable) y el peor (el más desfavorable).

Complejidad temporal en el mejor de los casos:

En el mejor de los casos están todos los vectores, de cualquier tamaño, para los cuales el bucle `while` solo hace una iteración. Para que esto ocurra, la variable `swaped` debe quedar con valor `false` en su primera iteración; sucederá siempre que el vector de entrada esté ordenado de manera ascendente.

Por lo tanto, todas las instancias siguientes pertenecen al caso mejor:[3]

$$v \in \mathbb{Z}^n \,:\, v[i] \geq v[i-1],\ \forall i : 1 \leq i \leq n-1$$

Si asumimos que el bucle externo termina tras la primera iteración, en la que $i = 1$, se tiene:

$$c_i(n) = \sum_{j=1}^{n-1} 1 = n - 1 \in \Omega(n) \tag{3.1}$$

La serie (3.1) expresa la cuenta de pasos realizados por el algoritmo solo en la primera iteración del `while`, donde $i = 1$. Los índices del sumatorio expresan las iteraciones del bucle `for`; cada una de ellas contribuye con un coste constante, de ahí que solo se suma un paso de programa. Al tratarse de un bucle decreciente, se han permutado los límites superior e inferior del sumatorio. Notar que se ha ignorado cualquier otra contribución de pasos de programa cuyo tiempo de ejecución está acotado por una constante (es decir, no depende de n).

[3] Notar que son instancias de cualquier tamaño n.

Complejidad temporal en el peor de los casos:

En el peor de los casos están todos los vectores, de cualquier tamaño, para los cuales el bucle `while` realiza el máximo número de iteraciones. Esto ocurre cuando la variable `swaped` se queda con valor `false` por el hecho de que no se entra en el `for`. A su vez, esto solo puede ocurrir cuando la variable i alcanza el valor n. Notar que si la variable `swaped` se queda con valor `false` antes de que i alcance el valor máximo, entonces el bucle `while` no realiza el máximo número de iteraciones y por lo tanto, no sería una instancia del caso peor.

Para que i llegue hasta n, el primer elemento del vector debe ser estrictamente mayor que cualquiera de los otros, sin importar el orden que pueda haber entre los elementos del vector.[4] Por lo tanto, todas las instancias siguientes pertenecen al caso peor:

$$v \in \mathbb{Z}^n \ : \ v[0] > v[i], \ \forall i : 1 \leq i \leq n-1$$

La complejidad temporal viene dada por el acumulado total de pasos de programa que realiza el `for` para todas y cada una de las iteraciones del `while`. Puesto que el coste de cada iteración del bucle interno es constante, la cantidad de pasos de programa que consume coincide con el número de iteraciones que realiza. La siguiente tabla representa una cuenta de dichas iteraciones mediante un desglose en cada una de las k iteraciones del bucle externo.

Iteración bucle externo	i	número de iteraciones del bucle interno
$1^{\underline{a}}$	1	$n-1$
$2^{\underline{a}}$	2	$n-2$
$3^{\underline{a}}$	3	$n-3$
...	...	...
k-ésima	k	$n-k$

Para escoger lo que se representa en cada columna se ha seguido el mismo criterio de el ejercicio anterior. Notar que la primera fila de la tabla corresponde al coste en el mejor de los casos, en el que el bucle externo solo hace una iteración.

Por la segunda columna, tenemos $i = k$ y, según se ha explicado anteriormente, en la última iteración del `while` se cumple $i = n$.

Por lo tanto, el número de iteraciones que hace el `while` en el peor de los casos es n. Sumando los elementos de la tercera columna obtenemos la expresión de complejidad buscada:

$$c_s(n) = \sum_{k=1}^{n}(n-k) = n\frac{n-1+0}{2} = \frac{1}{2}(n^2-n) \in O(n^2)$$

[4]Notar que los vectores ordenados de manera descendente son un caso particular del más desfavorable, pero hay muchos más vectores que están en el caso peor.