

Ejercicio 1. [1 punto]

Demostrar: $n \ln n = \mathcal{O}(n^{1+a})$, donde $0 < a < 1$.

Solución:

Podemos demostrar que n^{1+a} ($0 < a < 1$) es cota superior de $n \ln n$ si se verifica:

$$\lim_{n \rightarrow \infty} \frac{n \ln n}{n^{1+a}} = 0$$

Procedemos a calcular el límite:

$$\lim_{n \rightarrow \infty} \frac{n \ln n}{n^{1+a}} = \lim_{n \rightarrow \infty} \frac{\ln n}{n^a} = \frac{\infty}{\infty}$$

Aplicando L'Hopital (derivando numerador y denominador) se obtiene:

$$\lim_{n \rightarrow \infty} \frac{1/n}{an^{a-1}} = \lim_{n \rightarrow \infty} \frac{1}{an^a} = 0$$

NOTA: se ha utilizado un logaritmo neperiano, pero el resultado es válido independientemente de la base del logaritmo.

Ejercicio 2. [1 punto]

Simplificar: $\mathcal{O}(3m^3 + 2mn^2 + n^2 + 10m + m^2)$

Solución:

Para simplificar la expresión, en primer lugar, podemos eliminar las constantes, quedando:

$$\mathcal{O}(m^3 + mn^2 + n^2 + m + m^2)$$

En segundo lugar, resulta trivial ver que los términos m y m^2 son de menor orden que m^3 , por tanto también se pueden cancelar:

$$\mathcal{O}(m^3 + mn^2 + n^2)$$

Finalmente n^2 también se puede eliminar dado que el término mn^2 es de mayor orden. Esto puede comprobarse hallando los siguientes límites:

$$\lim_{n \rightarrow \infty} \frac{n^2}{mn^2} = \frac{1}{m} \neq \infty \quad \lim_{m \rightarrow \infty} \frac{n^2}{mn^2} = 0$$

Como uno es igual a 0 y otro distinto de 0 se verifica que mn^2 es de mayor orden n^2 . Finalmente la expresión no puede simplificarse más, quedando:

$$\boxed{\mathcal{O}(m^3 + mn^2)}$$

NOTA: los dos últimos términos no se pueden eliminar. No se puede afirmar que m^3 sea de mayor orden que mn^2 ya que:

$$\lim_{n \rightarrow \infty} \frac{mn^2}{m^3} = \infty \quad \lim_{m \rightarrow \infty} \frac{mn^2}{m^3} = 0$$

Es decir, un límite es 0 y el otro infinito. Análogamente, tampoco se puede afirmar que mn^2 sea de mayor orden que m^3 dado que:

$$\lim_{n \rightarrow \infty} \frac{m^3}{mn^2} = 0 \quad \lim_{m \rightarrow \infty} \frac{m^3}{mn^2} = \infty$$

Ejercicio 3. [1 punto]

Demostrar utilizando la definición de θ : $n^2/4 + 3n - 1 \in \theta(n^2)$

Solución:

Para hacer la demostración debemos comprobar que n^2 es tanto cota inferior como superior. Es decir, $n^2/4 + 3n - 1 \in \Omega(n^2)$ y $n^2/4 + 3n - 1 \in \mathcal{O}(n^2)$.

Ω :

Debemos encontrar un par de constantes positivas c_1 y n_1 tal que se cumpla:

$$n^2/4 + 3n - 1 \geq c_1 n^2$$

para todo $n \geq n_1$. Por ejemplo, podemos elegir $c_1 = 1/4$. En ese caso tenemos

$$n^2/4 + 3n - 1 \geq n^2/4$$

$$3n - 1 \geq 0$$

Lo cual se cumple para todo $n \geq 1$. Por tanto, simplemente escogemos $n_1 = 1$. Finalmente, hemos visto que existen esas constantes y queda demostrado.

\mathcal{O} :

Debemos encontrar un par de constantes positivas c_2 y n_2 tal que se cumpla:

$$n^2/4 + 3n - 1 \leq c_2 n^2$$

para todo $n \geq n_2$. Por ejemplo, podemos elegir $c_2 = 1$. En ese caso tenemos

$$n^2/4 + 3n - 1 \leq n^2$$

$$3n - 1 \leq 3n^2/4$$

El orden del termino cuadrático naturalmente es superior al lineal, con lo cual la desigualdad se va a cumplir a partir de un n dado. En ese caso, se cumple a partir de $n \geq 4$, y podemos elegir $n_2 = 4$. Una vez más, hemos visto que existen esas constantes y queda demostrado.

NOTA: podíamos haber elegido otras constantes. Lo importante es que existan, no sus valores concretos.

Ejercicio 4. [2 puntos]

Analizar la complejidad θ del problema de las Torres de Hanoi para n discos. Es decir, hallar la expresión no recursiva del número de movimientos de discos T_n , para n discos, y describir su cota ajustada θ . (Ayuda: ver el enunciado del siguiente ejercicio). El pseudocódigo del algoritmo es:

```
1 hanoi(int n, int destino, int origen, int auxiliar)
2 {
3   if (n > 0)
4   {
5     hanoi(n-1, auxiliar, origen, destino);
6     << Mover disco n desde origen a destino >>
7     hanoi(n-1, destino, auxiliar, origen);
8   }
9 }
```

Solución:

La expresión recursiva del algoritmo es la siguiente:

$$T_n = 2T_{n-1} + 1 \quad T_0 = 0$$

Para resolver la recurrencia y hallar una expresión no recursiva de T_n se puede proceder de dos maneras:

1. Expansión de recurrencias:

$$\begin{aligned} T_n &= 2T_{n-1} + 1 \\ &= 2(2T_{n-2} + 1) + 1 = 2^2T_{n-2} + 2 + 1 \\ &= 2(2(2T_{n-3} + 1) + 1) + 1 = 2^3T_{n-3} + 4 + 2 + 1 \\ &= 2(2(2(2T_{n-4} + 1) + 1) + 1) + 1 = 2^4T_{n-4} + 8 + 4 + 2 + 1 \end{aligned}$$

Se puede ver, por tanto, que la expresión general es:

$$T_n = 2^i T_{n-i} + \sum_{j=0}^{i-1} 2^j$$

Se llega al caso base cuando el parámetro de la T es cero. Es decir, cuando $n - i = 0$. Despejando la i , se obtiene $i = n$. Sustituyendo en la expresión obtenemos:

$$T_n = 2^n T_0 + \sum_{j=0}^{n-1} 2^j$$

Viendo que $T(0) = 0$ y resolviendo la serie geométrica queda finalmente:

$$T_n = 2^n - 1 \in \theta(2^n)$$

2. Método general de resolución de recurrencias: La recurrencia se puede expresar como:

$$T_n - 2T_{n-1} = 1 \cdot n^0 \cdot 1^n$$

La cual es una recurrencia no homogénea, cuya ecuación característica es:

$$(x - 2)(x - 1) = 0$$

Por tanto, la solución final tiene la forma

$$T_n = c_1 2^n + c_2 1^n = c_1 2^n + c_2$$

Teniendo en cuenta dos casos base o condiciones iniciales podemos hallar el valor de las constantes resolviendo el siguiente sistema:

$$\left. \begin{array}{l} c_1 + c_2 = 0 = T_0 \\ 2c_1 + c_2 = 1 = T_1 \end{array} \right\}$$

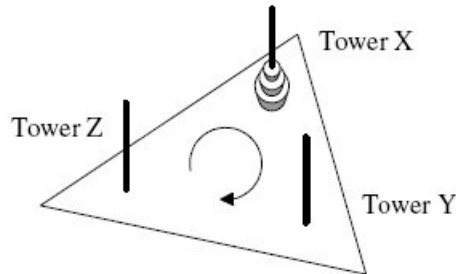
De donde obtenemos $c_1 = 1$ y $c_2 = -1$. Finalmente, concluimos que la solución es:

$$\boxed{T_n = 2^n - 1 \in \theta(2^n)}$$

NOTA: No es necesario resolver el ejercicio usando los dos métodos. Uno es suficiente. Por otro lado, también se podía haber escogido el caso base $T_1 = 1$ en el método de expansión de recurrencias.

Ejercicio 5. [2 puntos]

En este ejercicio se pide analizar la complejidad θ del problema de las Torres de Hanoi Cíclicas para n discos. El algoritmo es:



<pre> Clock(n,X,Y,Z) if(n>0) AntiClock(n-1,X,Z,Y) Move disk n from X to Y AntiClock(n-1,Z,Y,X) </pre>	<pre> AntiClock(n,X,Z,Y) if(n>0) AntiClock(n-1,X,Z,Y) Move disk n from X to Y Clock(n-1,Z,X,Y) Move disk n from Y to Z AntiClock(n-1,X,Z,Y) </pre>
--	---

Si a la función `AntiClock` la denominamos A , y `Clock` C , entonces tenemos las siguientes definiciones recursivas del número de operaciones (movimientos de discos), para n discos:

$$A_n = \begin{cases} 0 & \text{si } n = 0 \\ 2A_{n-1} + C_{n-1} + 2 & \text{si } n > 0 \end{cases} \quad (1)$$

$$C_n = \begin{cases} 0 & \text{si } n = 0 \\ 2A_{n-1} + 1 & \text{si } n > 0 \end{cases} \quad (2)$$

Se pide demostrar:

$$A_n = \frac{1}{4\sqrt{3}} \left[(1 + \sqrt{3})^{n+2} - (1 - \sqrt{3})^{n+2} \right] - 1$$

$$C_n = \frac{1}{2\sqrt{3}} \left[(1 + \sqrt{3})^{n+1} - (1 - \sqrt{3})^{n+1} \right] - 1$$

Solución:

En este ejercicio el primer paso consiste en eliminar la recursividad mutua, y expresar las funciones A_n y C_n en términos de sí mismas.

Dado que $C_n = 2A_{n-1} + 1$, decrementando el índice una unidad obtenemos $C_{n-1} = 2A_{n-2} + 1$. Sustituyendo en (1) logramos una definición recursiva que

sólo depende de A , a la que añadimos los dos casos base necesarios:

$$A_n = \begin{cases} 0 & \text{si } n = 0 \\ 2 & \text{si } n = 1 \\ 2A_{n-1} + 2A_{n-2} + 3 & \text{si } n \geq 2 \end{cases}$$

Se puede observar que $A_1 = C_0 + 2 = 2$. Además $A_2 = 7$.

También podemos hallar una expresión recursiva de C solamente en términos de sí misma. Por ejemplo, de (2) obtenemos las igualdades $2A_{n-1} = C_n - 1$, y $A_n = (C_{n+1} - 1)/2$, que podemos sustituir en (1) para obtener:

$$\frac{C_{n+1} - 1}{2} = C_n - 1 + C_{n-1} + 2$$

$$C_{n+1} = 2C_n + 2C_{n-1} + 3$$

Finalmente, decrementando el índice y añadiendo los casos base, se obtiene:

$$C_n = \begin{cases} 0 & \text{si } n = 0 \\ 1 & \text{si } n = 1 \\ 2C_{n-1} + 2C_{n-2} + 3 & \text{si } n \geq 2 \end{cases}$$

Se puede observar que $C_1 = A_0 + 1 = 1$. Además $C_2 = 5$.

En primer lugar, desarrollamos la fórmula no recursiva para A resolviendo la recurrencia no homogénea:

$$A_n - 2A_{n-1} - 2A_{n-2} = 3$$

La ecuación característica es:

$$(x^2 - 2x - 2)(x - 1) = 0$$

Las raíces del primer polinomio son:

$$x = \frac{2 \pm \sqrt{4 + 8}}{2} = 1 \pm \sqrt{3}$$

Por tanto, la ecuación característica es:

$$(1 + \sqrt{3})(1 - \sqrt{3})(x - 1) = 0$$

Y la solución tiene la siguiente forma:

$$A_n = c_1(1 + \sqrt{3})^n + c_2(1 - \sqrt{3})^n + c_3 1^n$$

Para hallar los valores de las constantes resolvemos el siguiente sistema de ecuaciones, usando 3 casos base:

$$\left. \begin{aligned} c_1 + c_2 + c_3 &= 0 = A_0 \\ c_1(1 + \sqrt{3}) + c_2(1 - \sqrt{3}) + c_3 &= 2 = A_1 \\ c_1(1 + \sqrt{3})^2 + c_2(1 - \sqrt{3})^2 + c_3 &= 7 = A_1 \end{aligned} \right\}$$

De la primera ecuación obtenemos

$$c_3 = -c_1 - c_2 \quad (3)$$

que sustituimos en las dos restantes ecuaciones para obtener:

$$\left. \begin{aligned} c_1(1 + \sqrt{3}) + c_2(1 - \sqrt{3}) - c_1 - c_2 &= 2 \\ c_1(1 + \sqrt{3})^2 + c_2(1 - \sqrt{3})^2 - c_1 - c_2 &= 7 \end{aligned} \right\}$$

De la primera ecuación se obtiene la expresión:

$$c_1 = \frac{2}{\sqrt{3}} + c_2 \quad (4)$$

que sustituyendo en la segunda se puede verificar que el valor de la constante c_2 es:

$$c_2 = \frac{\sqrt{3} - 2}{2\sqrt{3}} = -\frac{2 - \sqrt{3}}{2\sqrt{3}} = -\frac{4 - 2\sqrt{3}}{4\sqrt{3}} = -\frac{(1 - \sqrt{3})^2}{4\sqrt{3}}$$

Sustituyendo el valor de c_2 en (4) se obtiene:

$$c_1 = \frac{2}{\sqrt{3}} + \frac{\sqrt{3} - 2}{2\sqrt{3}} = \frac{2 + \sqrt{3}}{2\sqrt{3}} = \frac{4 + 2\sqrt{3}}{4\sqrt{3}} = \frac{(1 + \sqrt{3})^2}{4\sqrt{3}}$$

Finalmente, sustituyendo c_1 y c_2 en (3) se obtiene $c_3 = -1$, con lo cual la definición de A resulta ser:

$$A_n = \frac{1}{4\sqrt{3}} \left[(1 + \sqrt{3})^{n+2} - (1 - \sqrt{3})^{n+2} \right] - 1 \in \theta((1 + \sqrt{3})^n) \quad (5)$$

tal y como se había pedido.

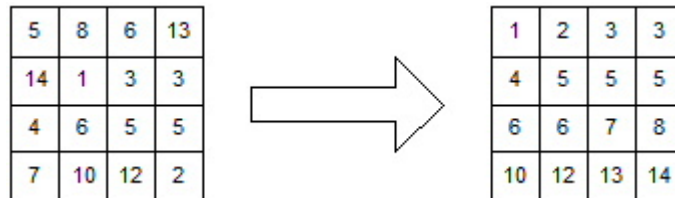
La expresión no recursiva de C_n se puede hallar de forma análoga, resolviendo un sistema de ecuaciones lineales. Sin embargo, dado que ya se ha calculado una expresión no recursiva para A_n , podemos combinar (2) y (5) para hallar C_n :

$$C_n = 2A_{n-1} + 1 = 2 \left[\frac{1}{4\sqrt{3}} \left[(1 + \sqrt{3})^{n+1} - (1 - \sqrt{3})^{n+1} \right] - 1 \right] + 1$$

$$C_n = \frac{1}{2\sqrt{3}} \left[(1 + \sqrt{3})^{n+1} - (1 - \sqrt{3})^{n+1} \right] - 1 \in \theta((1 + \sqrt{3})^n)$$

Ejercicio 6. [1 punto]

Determinar la complejidad θ del siguiente algoritmo: se trata de ordenar los elementos de una matriz cuadrada, por filas, utilizando una variante del algoritmo de ordenación Bubble-sort (Burbuja).



Solución:

La variante del algoritmo podría consistir en transformar la matriz en un vector auxiliar, ordenarlo, y volver a convertir el vector ordenado en una matriz. Otra posibilidad sería realizar la ordenación sin el vector auxiliar. En cualquier caso, la solución a este ejercicio depende de la medida que usemos del tamaño de la matriz cuadrada. De esta manera podemos considerar dos posibilidades:

- Si suponemos que la matriz cuadrada tiene n elementos (tendría \sqrt{n} filas y columnas), entonces el algoritmo tendría complejidad cuadrática:

$$\boxed{\theta(n^2)}$$

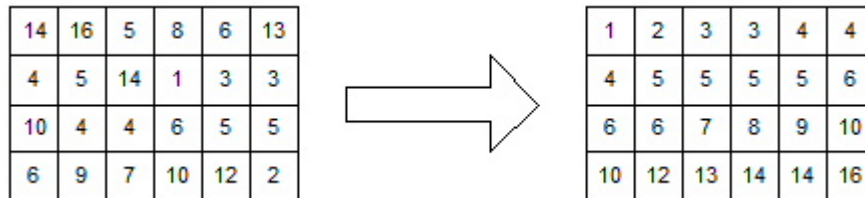
ya que el algoritmo de ordenación Bubble-sort tiene complejidad $\theta(n^2)$, tanto en el peor como el mejor caso, para un vector de tamaño n .

- Si suponemos que la matriz cuadrada tiene n filas y columnas entonces el número de elementos de la matriz es n^2 , y el algoritmo de ordenación sobre n^2 elementos tiene complejidad:

$$\boxed{\theta(n^4)}$$

Ejercicio 7. [1 punto]

Determinar la complejidad θ del siguiente algoritmo: se trata de ordenar los elementos de una matriz rectangular, por filas, utilizando una variante del algoritmo de ordenación Bubble-sort (Burbuja).



Solución:

Este ejercicio es idéntico al anterior, salvo que en este caso es necesario considerar dos parámetros a la hora de definir el tamaño del problema: el número de filas (n) y columnas (m). De esta manera, la matriz tiene nm elementos, y el algoritmo de ordenación Bubble-sort siempre necesita del orden de $(nm)^2$ operaciones. Por tanto, la medida de complejidad θ para este algoritmo es:

$$\theta(m^2n^2)$$

Ejercicio 8. [2 puntos]

Considérese el siguiente algoritmo, que determina si una matriz cuadrada A de tamaño $N \times N$ es simétrica ($A = A^T$):

```
1 es_traspuesta = TRUE;
2 c = 1;
3 while (c<=N) AND es_traspuesta
4     f = N;
5     while (f>c) AND es_traspuesta
6         if A(f,c)~A(c,f)
7             es_traspuesta = FALSE;
8         end
9         f = f-1;
10    end
11    c = c+1;
12 end
13 return es_traspuesta
```

Se pide hallar la complejidad θ usando el método de “contar pasos” (se valorará el uso de sumatorios en la respuesta).

Solución:

Analizamos la complejidad del algoritmo en el mejor y peor caso:

- **Mejor caso:** Si los dos primeros elementos comparados de la matriz son diferentes entonces el algoritmo devuelve el valor FALSE en un número constante de pasos, independientemente del tamaño del problema. Por tanto la complejidad en este caso es:

$$\theta(1)$$

- **Peor caso:** El peor caso ocurre cuando la matriz es simétrica, y la variable `es_traspuesta` siempre toma el valor verdadero. Para usar la técnica de “contar pasos” podemos dividir el algoritmo en los siguientes bloques:



Considerando que cada bloque tarda una unidad de tiempo (podríamos usar diferentes constantes para cada bloque), la fórmula para el tiempo de ejecución suponiendo que la matriz es de tamaño N es:

$$T(N) = 1 + \sum_{c=1}^N \left[1 + 1 + 1 + \sum_{f=N}^{c+1} [1 + 1] + 1 \right] + 1 + 1$$

Simplificando y resolviendo los sumatorios se obtiene:

$$T(N) = 3 + \sum_{c=1}^N \left[4 + \sum_{f=N}^{c+1} 2 \right] = T(N) = 3 + \sum_{c=1}^N [4 + 2(N - c)]$$

$$T(N) = 3 + 4N + 2N^2 - 2 \sum_{c=1}^N c = 3 + 4N + 2N^2 - 2 \frac{N(N + 1)}{2}$$

$$T(N) = N^2 + 3N + 3 \in \theta(N^2)$$

NOTA: La agrupación de bloques podría hacerse de varias maneras, lo cual origina diferentes polinomios de grado 2. Por ejemplo, los bloques 3 y 6 se ejecutan el mismo número de veces, con lo cual podrían haberse representado mediante un sólo bloque. Por otro lado, aunque no es relevante de cara a la complejidad del algoritmo, en la fórmula se han tenido en cuenta las últimas comparaciones de los bucles en las que la condición es falsa.

Ejercicio 9. [5 puntos]

Considérese un algoritmo que procesa una serie de números en base 2 distintos del 0. Para cada número empieza a procesar cada bit (realiza una operación por bit), empezando por el bit menos significativo, hasta el bit más significativo, pero para en el momento que encuentra el primer 1. La siguiente figura ilustra las operaciones (sombreadas) que haría con todos los números de 4 bits distintos de 0, es decir, procesa desde el 1 hasta el 15.

0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

←

Si se procesan todos los números el algoritmo habrá realizado 26 operaciones (que es el número de bits sombreados).

Se pide hallar la complejidad θ para este algoritmo si procesa todos los números:

- a) Contando operaciones (**1 punto**)
- b) Resolviendo la recurrencia no homogénea mediante la técnica de expansión de recurrencias (**2 puntos**)
- c) Resolviendo la recurrencia no homogénea mediante el método de resolución de relaciones en recurrencias (**2 puntos**)

Solución:

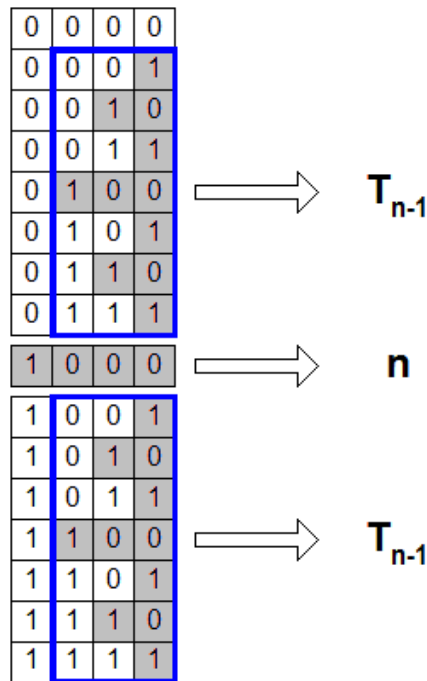
- a) Analizando las operaciones que se realizan por cada columna, empezando desde el bit más significativo al menos, podemos ver que la secuencia es 1,3,7,15... Es decir, para n bits el número de operaciones que se realizan se puede expresar como:

$$T_n = \sum_{i=1}^n (2^i - 1) = \sum_{i=1}^n 2^i - n$$

Por tanto, resolviendo la serie geométrica:

$$T_n = 2^{n+1} - 2 - n \in \theta(2^n)$$

- b) En primer lugar, es necesario obtener una fórmula recursiva del número de operaciones que se realizan. Podemos descomponer el número de operaciones de la siguiente manera:



Para n bits contamos n operaciones (correspondientes a la palabra central 2^{n-1}), además de resolver dos veces el mismo problema para $n - 1$ bits, la fórmula recursiva es:

$$T_n = 2T_{n-1} + n \quad T_0 = 0 \quad (6)$$

Una vez hallada la fórmula recursiva procedemos a realizar la expansión de recurrencias:

$$\begin{aligned}
 T_n &= 2T_{n-1} + n \\
 &= 2(2T_{n-2} + n - 1) + n = 2^2T_{n-2} + 2(n - 1) + n \\
 &= 2(2(2T_{n-3} + n - 2) + n - 1) + n \\
 &= 2^3T_{n-3} + 2^2(n - 2) + 2(n - 1) + n \\
 &= 2(2(2(2T_{n-4} + n - 3) + n - 2) + n - 1) + n \\
 &= 2^4T_{n-4} + 2^3(n - 3) + 2^2(n - 2) + 2(n - 1) + n
 \end{aligned}$$

Se puede ver, por tanto, que la expresión general es:

$$T_n = 2^i T_{n-i} + \sum_{j=0}^{i-1} 2^j (n - j) = 2^i T_{n-i} + \sum_{j=0}^{i-1} n 2^j - \sum_{j=0}^{i-1} j 2^j$$

El caso base se alcanza en $T_0 = 0$. Es decir, cuando $i = n$. Realizando las correspondientes sustituciones obtenemos

$$T_n = \sum_{j=0}^{n-1} n 2^j - \sum_{j=0}^{n-1} j 2^j = n(2^n - 1) - \sum_{j=0}^{n-1} j 2^j \quad (7)$$

La dificultad en este ejercicio radica en hallar una expresión para el sumatorio de la fórmula anterior:

$$\sum_{j=0}^{n-1} j 2^j = 1 \cdot 2^1 + 2 \cdot 2^2 + 3 \cdot 2^3 + \dots + (n - 1) \cdot 2^{n-1} \quad (8)$$

Para hallarla partimos de una serie geométrica general:

$$S = 1 + x + x^2 + \dots + x^{m-1} = \frac{x^m - 1}{x - 1}$$

derivando S con respecto a x obtenemos la siguiente igualdad:

$$\begin{aligned}
 1 + 2x + 3x^2 + \dots + (m - 1)x^{m-2} &= \frac{mx^{m-1}(x - 1) - (x^m - 1)}{(x - 1)^2} = \\
 &= \frac{mx^m - mx^{m-1} - x^m + 1}{(x - 1)^2}
 \end{aligned}$$

Multiplicando por x a ambos lados obtenemos:

$$x^1 + 2x^2 + 3x^3 + \dots + (m - 1)x^{m-1} = \frac{mx^{m+1} - mx^m - x^{m+1} + x}{(x - 1)^2} \quad (9)$$

Por tanto, podemos ver que el sumatorio que queremos calcular en (8) es idéntico al descrito en (9) realizando los cambios $x = 2$ y $m = n$. De esta manera, tenemos:

$$\sum_{j=0}^{n-1} j2^j = \frac{n2^{n+1} - n2^n - 2^{n+1} + 2}{(2-1)^2} = n2^{n+1} - n2^n - 2^{n+1} + 2$$

Finalmente, sustituyendo en (7) obtenemos la expresión final de T_n :

$$\boxed{T_n = n2^n - n - n2^{n+1} + n2^n + 2^{n+1} - 2 = 2^{n+1} - n - 2 \in \theta(2^n)}$$

c) La ecuación característica de asociada a (6) es:

$$(x-2)(x-1)^2 = 0$$

Por tanto, la solución tiene la siguiente forma:

$$T_n = c_1 2^n + c_2 1^n + c_3 n 1^n \quad (10)$$

Para hallar el valor de las constantes debemos resolver el siguiente sistema de ecuaciones (teniendo en cuenta que $T_0 = 0$, $T_1 = 1$, y $T_2 = 4$):

$$\left. \begin{array}{l} c_1 + c_2 = 0 = T_0 \\ 2c_1 + c_2 + c_3 = 1 = T_1 \\ 4c_1 + c_2 + 2c_3 = 4 = T_2 \end{array} \right\}$$

Resolviendo el sistema se obtiene $c_1 = 2$, $c_2 = -2$, y $c_3 = -1$. Por tanto, sustituyendo en (10), la expresión final del número de operaciones en función del número de bits n es:

$$\boxed{T_n = 2^{n+1} - 2 - n \in \theta(2^n)}$$

NOTA: El algoritmo tiene un coste exponencial dado el número de bits de las palabras. También podríamos haber calculado la complejidad en función del número total de palabras procesadas (m). En ese caso, dado que $m = 2^n - 1$, bastaría con realizar la sustitución ($n = \log_2(m+1)$) en la fórmula del número de operaciones $T_n = 2^{n+1} - 2 - n$:

$$T'(m) = 2 \cdot 2^{\log_2(m+1)} - \log_2(m+1) - 2 = 2m - \log_2(m+1) \in \theta(m)$$

Ejercicio 10. [4 puntos]

NOTA: no es necesario realizar el siguiente ejercicio. Se ha descartado dado que la mayor parte del trabajo consiste en realizar cálculos de manipulación algebraica. En este documento simplemente se enuncia el problema y se dan pistas para poder resolverlo.

Considérese la siguiente función general:

$$G_i = \begin{cases} a & \text{si } i = 1 \\ b & \text{si } i = 2 \\ c + G_{i-1} + G_{i-2} & \text{si } i \geq 3 \end{cases}$$

Se pide demostrar (resolviendo la relación de recurrencia), y conociendo que

$$F_n = \frac{\phi^n - (1 - \phi)^n}{\sqrt{5}} = \frac{\phi^n - \hat{\phi}^n}{\sqrt{5}} \quad (11)$$

donde F_n es un número de Fibonacci ($F_n = F_{n-1} + F_{n-2}$, $F_1 = 1$, $F_2 = 1$), con

$$\phi = \frac{1 + \sqrt{5}}{2} \quad \text{y} \quad \hat{\phi} = \frac{1 - \sqrt{5}}{2}$$

que se verifica:

$$G_i = \begin{cases} a & \text{si } i = 1 \\ b & \text{si } i = 2 \\ aF_{i-2} + bF_{i-1} + cF_i - c & \text{si } i \geq 3 \end{cases}$$

Solución:

Se dan unas pistas para poder resolverlo. En primer lugar, se parte de la expresión $G_i - G_{i-1} - G_{i-2} = c$, que es una recurrencia no homogénea. La ecuación característica es $(x^2 - x - 1)(x - 1) = 0$. Las raíces de $(x^2 - x - 1)$ son ϕ y $\hat{\phi}$. De esta manera la fórmula tiene la forma:

$$G_i = c_1\phi^n + c_2\hat{\phi}^n + c_3$$

A continuación tendríamos que crear un sistema de 3 ecuaciones para hallar los valores de las constantes. Podemos usar G_1 , G_2 , y $G_0 = b - a - c$.

Posteriormente resolveríamos el sistema, donde debemos expresar los valores de las constantes en términos de ϕ y $\hat{\phi}$. Para hacer las oportunas simplificaciones es importante tener en cuenta las siguientes identidades:

$$\phi = 1 - \hat{\phi} \quad \hat{\phi} = 1 - \phi \quad \phi^2 = \phi + 1 \quad \hat{\phi}^2 = \hat{\phi} + 1$$

Por último, usando (11) y la relación $F_n = F_{n-1} + F_{n-2}$ es posible llegar al resultado final.