

Exercise 1 (4 points). Justifiquen si las siguientes proposiciones son verdaderas o falsas

- (1 points) $n^2 - 10n + 25 \neq \omega(n^2)$ $\leftarrow T$
- (2 points) $2025n^2 + 20n + 25 = o(n^3)$ $\leftarrow T$
- (1 points) $\left\lfloor \frac{\lfloor \frac{n}{1.5} \rfloor}{1.5} \right\rfloor = \left\lfloor \frac{n}{2.25} \right\rfloor$ $\leftarrow F$

$$\lim_{n \rightarrow \infty} f(n) = 3$$

$$\forall \varepsilon > 0, \exists n_0 > 0 \text{ t.q.}$$

$$n > n_0 \rightarrow |f(n) - 3| < \varepsilon$$

Definición
del
límite

$$(1) \quad n^2 - 10n + 25 \neq \omega(n^2)$$

Supongamos por $\rightarrow \leftarrow$

$$\forall c > 0, \exists n_0 > 0 \text{ t.q. } \forall n \geq n_0 \quad n^2 - 10n + 25 > c n^2$$

Para $c = 1$

$$(n-5)^2 > n^2 \text{ esta premisa es falsa a partir de } n > 6$$

$$\therefore \forall n > n_c \rightarrow \underbrace{(n-5)^2 > n^2}_{n < 2,5} \text{ cuando } n < 6 \quad (\rightarrow \leftarrow)$$

Tenemos que todo $n \geq n_c$, debería cumplir $n < 2,5$

$$n = \lceil n_c + 2,5 + 1 \rceil \quad \text{---} \quad \text{---} \quad \text{---}$$

En la contradicción, propones una constante de modo que no cumpla $\forall n > n_0$

$$(2) \quad 2025n^2 + 20n + 25 = o(n^3)$$

$$\forall c > 0, \exists n > 0 \text{ t.q. } 2025n^2 + 20n + 25 < c n^3$$

$$2025n^2 \leq 2025n^2$$

$$20n \leq 20n^2 \quad \text{---} \quad \text{---} \quad \text{---}$$

$$25 \leq 25n^2$$

$$f(n) \leq 2070n^2$$

$$2070n^2 < cn^3$$

$$\frac{2070}{c} < n$$

$$\Rightarrow n_0 \geq \frac{2070}{c} + 1 \quad \text{cumple}$$

③ $\left\lfloor \frac{2 \left\lfloor \frac{2n}{3} \right\rfloor}{3} \right\rfloor = \left\lfloor \frac{4n}{9} \right\rfloor$ es falso por contraejemplo

Exercise 2 (4 points). Resolver la relación de recurrencia **explícitamente**, usando los métodos vistos en clase. Deberás brindar la **solución exacta** para infinitos valores de n . Usar esta información para encontrar y probar el crecimiento asintótico de $T(n)$.

$$T(n) = \begin{cases} n^2 & 1 \leq n < 45 \\ 2025T(\lfloor n/45 \rfloor) + n^2 - 2024 & \text{otherwise} \end{cases}$$

Puedes asumir que la función $T(n)$ es creciente.

$$T(45^k) = k45^{2k} + 1$$

$$3k \geq k+1 \rightarrow k \geq 1 \quad n \geq 45$$

Exercise 3 (3 points). Sea T una función de recurrencia definida de la siguiente manera:

$$T(n) = \sum_{k=1}^{n-1} (T(k) + T(n-k) + 2)$$

Para todo entero $n > 1$, y con $T(1) = 0$. Probar que $T(n) = \Omega(2^n)$

$$T(n) \geq c2^n$$

$$T(n) \geq 2n+2 + \sum_{k=2}^{n-1} T(k)$$

$$\geq 2n+2 + 1 + c \sum_{k=2}^{n-1} T(k)$$

$$\geq 2n+2T_1 + 1 + c \sum_{k=2}^{n-1} 2^k$$

$$\geq 2n+2T_1 + 2$$

Exercise 4 (3 points). Considere el siguiente problema:

Input: Un array $A[1 \dots n]$ de enteros positivos, ordenados de forma estrictamente decreciente.

Output: La cantidad de índices i tal que $A[i] + i \neq n + 1$.

Por ejemplo, trabajando con el array $A = [8; 6; 5; 4; 2; 1]$, el output el programa debería ser 4, porque hay 4 índices que no cumplen esa igualdad.

- Escribir el pseudocódigo de un algoritmo de división y conquista, con complejidad $O(n)$.
- Describa la recurrencia de su algoritmo y justifique su complejidad utilizando el teorema maestro.

Algo (A, l, r, n) // cantidad de índices tq $A_i + i \neq n + 1$

if $l == r$

return $(A[l] + l \neq n + 1)$

$m = \lfloor \frac{l+r}{2} \rfloor$

if $A[n] + n \neq n + 1$

return $(n - l + 1) + \text{Algo}(A, m+1, r, n)$

else

return $\text{Algo}(A, l, m, n)$

Exercise 5 (3 points). Las matrices de Hadamard son matrices H_0, H_1, \dots , definidas de la siguiente manera

- H_0 es la matriz unitaria $[1]$.
- Para $k > 0$, H_k es la matriz de dimensiones $2^k \times 2^k$:

$$H_k = \left[\begin{array}{c|c} H_{k-1} & H_{k-1} \\ \hline H_{k-1} & -H_{k-1} \end{array} \right]$$

Considere el siguiente problema

Input: Un arreglo $A[1..n]$ of integers, donde $n = 2^k$

Output: El arreglo que representa la transpuesta del producto matricial $H_k \cdot A^T$

Diseñe un algoritmo $\Theta(n \log n)$ para resolver el problema anterior. Puede considerar que todos los números envueltos en el producto son lo suficientemente pequeños para que cada producto numérico tome tiempo constante.

Algo(A, n)

if (n==1)

return A[1:1]

else:

X = A[1:n/2]

Y = A[n/2+1:n]

B = Algo(X, n/2)

C = Algo(Y, n/2)

D = B + C

E = B - C

A[1:n/2] = D

A[n/2+1:n] = E

return A

INPUT

V_{2^k}



OUTPUT

$H_k \cdot V_{2^k}^T$

↓
 $H_{k-1} \cdot V_{2^{k-1}}^T$

$$\begin{bmatrix} a & a \\ a & -a \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + ay \\ ax - ay \end{bmatrix}$$

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$$

$n \lg n$

Input [1 3 -2 5]

$H_k \begin{bmatrix} 1 \\ 3 \\ -2 \\ 5 \end{bmatrix}$

$H_1 \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$

$H_0 \begin{bmatrix} 1 \\ -2 \\ 1 \\ 5 \end{bmatrix}$

$H_1 \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ -2 \end{bmatrix} = \begin{bmatrix} 1+(-2) \\ 1-(-2) \end{bmatrix} = \begin{bmatrix} -1 \\ 3 \end{bmatrix}$

$H_1 \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 3 \\ 5 \end{bmatrix} = \begin{bmatrix} 1+5 \\ 1-5 \end{bmatrix} = \begin{bmatrix} 6 \\ -4 \end{bmatrix}$

$H_2 \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 3 \\ 6 \\ -4 \end{bmatrix} = \begin{bmatrix} -1+3+6-4 \\ -1-3+6+4 \\ -1+3-6-4 \\ -1-3-6+4 \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \\ -10 \\ -4 \end{bmatrix}$

Algo(A[1:2], 2)

Algo(A[3:4], 2)

Exercise 6 (3 points). Esta parte es electiva. *Responder solo 1 de las siguientes preguntas (Si respondes más de dos, no te calificaré).*

- Dado un heap de n nodos, n_l nodos en el subarbol izquierdo y n_r nodos en el subarbol derecho, Probar que se cumple la siguiente desigualdad: $n_r \leq n_l \leq 2n_r + 1$
Hint: En este item puede usar sin probar la relación entre la altura de un heap y su cantidad de nodos sin necesidad de probarlo.
- Probar que la cantidad de nodos en un heap de n nodos, que tengal altura como mínimo 3, es exactamente igual a $\lfloor \frac{n}{8} \rfloor$.

} cormen

} cantidad de nodos de altura h