Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2283 - Diseño y Análisis de Algoritmos - 2^{do} semestre 2022

INTERROGACIÓN 1

Preguntas en blanco: Items de preguntas entregadas en blanco se evaluarán con 0.5 puntos de 6.

Pregunta 1

Para esta pregunta asuma que las funciones son de tipo $f : \mathbb{N} \to \mathbb{R}_0^+$. Responda lo siguiente:

1. Demuestre que si $f_1 \in \mathcal{O}(g_1)$ y $f_2 \in \mathcal{O}(g_2)$ luego $f_1 + f_2 \in \mathcal{O}(\max\{g_1(n), g_2(n)\})$.
2. Se define:
$$\omega(f) = \{g : \mathbb{N} \to \mathbb{R}_0^+ \mid \forall c \in \mathbb{R}^+.\ \exists n_0 \in \mathbb{N}.\ \forall n \geq n_0.\ g(n) \geq c \cdot f(n)\}$$
Demuestre que $3^n \in \omega(2^n)$.

Pregunta 2

Considere la siguiente ecuación de recurrencia:

$$T(n) = \begin{cases} 1 & n = 1 \\ T\left(\left\lfloor \frac{n\sqrt{3}}{2} \right\rfloor\right) + T\left(\lfloor \frac{n}{2} \rfloor\right) + n^2 & n > 1 \end{cases}$$

Demuestre que $T(n) \in \mathcal{O}(n^2 \cdot \log_2(n))$ usando inducción constructiva.

Pregunta 3

Sea Σ un alfabeto. Para una palabra $w \in \Sigma^*$ se tiene que una partición palindrómica de w es una secuencia de palabras $u_1, \ldots, u_\ell \in \Sigma^*$ tal que $u_1 \cdots u_\ell = w$ y u_i es un palíndromo para cada $1 \leq i \leq \ell$ (recuerde que un palíndromo es una palabra que es igual a su reverso).

En esta pregunta, queremos utilizar programación dinámica para construir un algoritmo **MinPart** que reciba una palabra $w = a_1 \cdots a_n$ y retorne el largo mínimo ℓ posible para una partición palindrómica de w. Para este objetivo utilizamos la siguiente definición recursiva de **MinPart**$(a_1 \cdots a_n)$:

$$\textbf{MinPart}(a_1 \cdots a_n) = \begin{cases} 1 & a_1 \cdots a_n \text{ es un palíndromo} \\ \min\limits_{1 \leq k < n} \{\textbf{MinPart}(a_1 \cdots a_k) & \\ \qquad + \textbf{MinPart}(a_{k+1} \cdots a_n)\} & a_1 \cdots a_n \text{ NO es un palíndromo} \end{cases}$$

En esta pregunta usted debe explicar por qué se puede aplicar programación dinámica para resolver este problema, en particular, por qué la recursión anterior es correcta. Además, debe implementar un algoritmo que utilice esta recursión para calcular **MinPart** en tiempo polinomial, suponiendo que la operación básica a contar es la comparación entre símbolos pertenecientes a Σ.

Pregunta 4

Considere un polinomio $p(x) = \sum_{k=0}^{n-1} a_k x^k$ no nulo de coeficientes racionales con $n \geq 1$. En clases vimos que si n es par entonces podemos separar $p(x)$ en dos polinomios de grado $n/2 - 1$:

$$p(x) = \sum_{k=0}^{\frac{n}{2}-1} a_{2k} x^{2k} + x \cdot \sum_{k=0}^{\frac{n}{2}-1} a_{2k+1} x^{2k} = q(x^2) + x \cdot r(x^2)$$

Lo que permite confeccionar el algoritmo **FFT** que realiza 2 llamadas recursivas sobre los polinomios $q(x)$ y $r(x)$. Dicho eso, no es difícil ver que si n es divisible por 3 se va a cumplir que:

$$p(x) = \sum_{k=0}^{\frac{n}{3}-1} a_{3k} x^{3k} + x \cdot \sum_{k=0}^{\frac{n}{3}-1} a_{3k+1} x^{3k} + x^2 \cdot \sum_{k=0}^{\frac{n}{3}-1} a_{3k+2} x^{3k} = q(x^3) + x \cdot r(x^3) + x^2 \cdot s(x^3)$$

Es decir, podemos separar $p(x)$ en 3 polinomios de grado $n/3 - 1$.

Usando esta idea, explique como se podría modificar el algoritmo **FFT** de manera que se realicen 3 llamadas recursivas y se siga entregando correctamente $\mathbf{DFT}(a_0, \ldots, a_{n-1})$ en tiempo $\mathcal{O}(n \cdot \log n)$. Justifique su respuesta.

Formulario

Notación asintótica

Sea $f : \mathbb{N} \to \mathbb{R}_0^+$ una función, definimos:

$$\mathcal{O}(f) = \{g : \mathbb{N} \to \mathbb{R}_0^+ \mid \exists c \in \mathbb{R}^+ \exists n_0 \in \mathbb{N}. \forall n \geq n_0. \, g(n) \leq c \cdot f(n)\}$$

$$\Omega(f) = \{g : \mathbb{N} \to \mathbb{R}_0^+ \mid \exists c \in \mathbb{R}^+ \exists n_0 \in \mathbb{N}. \forall n \geq n_0. \, c \cdot f(n) \leq g(n)\}$$

$$\Theta(f) = \mathcal{O}(f) \cap \Omega(f)$$

Transformada de Fourier

Raíces de la unidad:

$$\omega_n^k = e^{\frac{2\pi i}{n} k}$$

Transformada discreta de Fourier:

$$p(x) = \sum_{k=0}^{n-1} a_k x^k$$

$$\mathbf{DFT}(a_0, \ldots, a_{n-1}) = [p(\omega_n^0), ..., p(\omega_n^{n-1})]$$

Transformada rápida de Fourier:

FFT$(a_0, \ldots, a_{n-1})$
    if $n = 2$ **then**
        $y_0 = a_0 + a_1$
        $y_1 = a_0 - a_1$
        return $[y_0, y_1]$
    else
        $[u_0, \ldots, u_{\frac{n}{2}-1}] :=$ **FFT**$(a_0, \ldots, a_{n-2})$
        $[v_0, \ldots, v_{\frac{n}{2}-1}] :=$ **FFT**$(a_1, \ldots, a_{n-1})$
        $\omega_n := e^{\frac{2\pi i}{n}}$
        $\alpha := 1$
        for $k := 0$ **to** $\frac{n}{2} - 1$ **do**
            $y_k := u_k + \alpha \cdot v_k$
            $y_{\frac{n}{2}+k} := u_k - \alpha \cdot v_k$
            $\alpha := \alpha \cdot \omega_n$
        return $[y_0, \ldots, y_{n-1}]$