PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE
ESCUELA DE INGENIERIA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACION

Diseño y Análisis de Algoritmos - IIC2283
Rúbrica Interrogación 1

1. Recuerde que las funciones que representan el tiempo de ejecución de un algoritmo son de la forma $f : \mathbb{N} \to \mathbb{R}_0^+$. Responda las siguientes preguntas sobre este tipo de funciones.

   a) [0.7 puntos] Demuestre que $f(n) + g(n) \in \Theta(\text{máx}\{f(n), g(n)\})$

   Correción: El puntaje de esta pregunta se asigna considerando que hay dos direcciones que demostrar formalmente: $f(n) + g(n) \in O(\text{máx}\{f(n), g(n)\})$ y $\text{máx}\{f(n), g(n)\} \in O(f(n) + g(n))$.

   - [0.4 puntos] Solo una de las direcciones fue demostrada correctamente.
   - [0.7 puntos] Ambas direcciones fueron demostradas correctamente.

   b) [0.8 puntos] Demuestre que $3^n \notin O(2^n)$

   Correción: El puntaje de esta pregunta se asigna de la siguiente forma.

   - [0.4 puntos] La idea de la demostración es correcta, pero hay errores en su formalización matemática.
   - [0.8 puntos] La idea de la demostración y su formalización matemática están correctas.

2. [1.5 puntos] Considere la siguiente ecuación de recurrencia:

$$T(n) = \begin{cases} 1 & n = 1 \\ 2 \cdot T(\lfloor \frac{n}{2} \rfloor) + \lfloor n \cdot \log_2(n) \rfloor & n > 1 \end{cases}$$

   Demuestre que $T(n) \in O(n \cdot (\log_2(n))^2)$.

   Correción: El puntaje de esta pregunta se asigna de la siguiente forma.

   - [0.8 puntos] La idea de la demostración es correcta, pero hay errores en su formalización matemática. En particular, se utiliza inducción constructiva.
   - [1.5 puntos] La idea de la demostración y su formalización matemática están correctas.

3. [1.5 puntos] Dadas dos matrices $A_{m\times n}$ y $B_{n\times r}$ de números enteros, el algoritmo usual para calcular $A \cdot B$ debe realizar $m \cdot n \cdot r$ multiplicaciones de números enteros. Además, dadas tres de estas matrices $A_{m\times n}$, $B_{n\times r}$ y $C_{r\times s}$, podemos calcular su multiplicación como $(A \cdot B) \cdot C$ realizando $m \cdot n \cdot r + m \cdot r \cdot s$ multiplicaciones de números enteros, o como $A \cdot (B \cdot C)$ realizando $m \cdot n \cdot s + n \cdot r \cdot s$ multiplicaciones de números enteros.

   Dada una matriz A, usamos filas(A) y col(A) para denotar el número de filas y columnas de A. En esta pregunta, queremos utilizar programación dinámica para construir un algoritmo **NumMult** que tenga como entrada una secuencia de matrices A_1, A_2, ..., A_ℓ tal que $\ell \geq 1$ y col(A_i) = filas(A_{i+1}) para cada $i \in \{1, \ldots, \ell - 1\}$, y retorne el número mínimo de multiplicaciones de números enteros que debe realizar el algoritmo usual para calcular la multiplicación de la secuencia de matrices. Para este objetivo, utilizamos la siguiente definición recursiva para **NumMult**$(A_1, \ldots, A_\ell)$:

$$\textbf{NumMult}(A_1, \ldots, A_\ell) \quad = \quad \begin{cases} 0 & \ell = 1 \\ \min\limits_{1 \leq i \leq \ell - 1} \big\{ \textbf{NumMult}(A_1, \ldots, A_i) + & \\ \qquad \textbf{NumMult}(A_{i+1}, \ldots, A_\ell) + & \ell > 1 \\ \qquad \text{filas}(A_1) \cdot \text{col}(A_i) \cdot \text{col}(A_\ell) \big\} & \end{cases}$$

   En esta pregunta usted debe explicar por qué se puede aplicar programación dinámica en este caso, y por qué la recursión anterior es correcta. Además, debe implementar un algoritmo que utilice esta recursión para calcular **NumMult** en tiempo polinomial, suponiendo que la operación básica a contar es la multiplicación de números enteros.

   Correción: El puntaje de esta pregunta se asigna de la siguiente forma.

   - [0.5 puntos] Está explicado por qué la recursión es correcta.
   - [0.9 puntos] Está explicado por qué la recursión es correcta y por qué se puede aplicar programación dinámica en este caso.
   - [1.5 puntos] Está explicado por qué la recursión es correcta y por qué se puede aplicar programación dinámica en este caso, y se realiza una implementación de tiempo polinomial de la recursión usando un enfoque bottom-up o memoización.

4. [1.5 puntos] Dado un polinomio

$$p(x) \quad = \quad \sum_{i=0}^{n-1} a_i x^i$$

   representando por la tupla de coeficientes $\bar{a} = (a_0, \ldots, a_{n-1})$, defina la transformada alternativa de Fourier de la siguiente forma:

$$\textbf{TAF}(\bar{a}) \quad = \quad [p(\omega_{2n}^1),\, p(\omega_{2n}^3),\, \ldots,\, p(\omega_{2n}^{2n-1})].$$

   Muestre que las mismas ideas utilizadas para desarrollar el algoritmo **FFT** pueden ser utilizadas para desarrollar un algoritmo para calcular **TAF**$(\bar{a})$ en tiempo $O(n \cdot \log(n))$, considerando la suma y multiplicación de números complejos como la operación básica a contar.

Correción: El puntaje de esta pregunta se asigna de la siguiente forma.

- [0.6 puntos] Se mencionan las propiedades que debe cumplir la secuencia de raíces ω_{2n}^1, ω_{2n}^3, ..., ω_{2n}^{2n-1} para que las ideas de **FFT** puedan ser utilizadas para desarrollar el algoritmo pedido de tiempo $O(n \cdot \log(n))$. En particular, se menciona que cuando n es par, se tiene que $(\omega_{2n}^1)^2$, $(\omega_{2n}^3)^2$, ..., $(\omega_{2n}^{2n-1})^2$ es igual a ω_n^1, ω_n^3, ..., ω_n^{n-1}.
- [1.2 puntos] Se menciona lo descrito en el punto anterior, y se explica por qué estas propiedades son válidas.
- [1.5 puntos] Además de los dos puntos anteriores, se indica que el algoritmo solo funciona si n es una potencia de 2, y se explica cómo extenderlo a cualquier valor de n manteniendo la complejidad $O(n \cdot \log(n))$.