2da Lista de Problemas de Análisis y Diseño de Algoritmos

(Otoño-Invierno 2022)

Cristhian Alejandro Ávila-Sánchez

0. Considere un espacio D-dimensional. Plantee un algoritmo, de complejidad $O(NlogN)$ utilizando la estrategia de divide y vencerás para encontrar la distancia mínima que puede existir entre 2 puntos pertenecientes a un conjunto de N puntos.

1. Obtenga la función de recursión $T(N) = aT\left(\frac{N}{b}\right) + f(N)$ para los algoritmos de ordenamiento de Quicksort, Merge Sort y Heap Sort. Demuestre que la complejidad de los 3 algoritmos es $O(NlogN)$. ¿En cuales situaciones es más conveniente elegir cada uno de estos algoritmos sobre los demás?

2. Demuestre que la complejidad del algoritmo de multiplicación de Karatsuba es $O\left(N^{log_2 3}\right) \approx O(N^{1.5849})$. (*Tip:* reduzca 4 llamadas recursivas a la función de multiplicación a solamente 3 invocaciones).

3. Muestre, mediante series de Taylor, que un número complejo $z = x + iy$ puede representarse polarmente como $z = re^{i\theta}$. (*Tips:* Obtenga la serie de Taylor de la función exponencial y evalúe la serie cuando el parámetro es igual a cero; separe la serie por sus partes real e imaginaria y note que estas sumatorias corresponden a las series de las funciones $\cos \quad y \ \sin$, respectivamente).

4. Encuentre las soluciones, para el caso general, de la ecuación diferencial de segundo orden correspondiente al oscilador armónico:

$$a\frac{d^2y}{dx^2} + b\frac{dy}{dx} + cy = 0$$

(*Tip:* Sustituya la solución de prueba $y = e^{\lambda x}$ en la ecuación diferencial, obtenga un polinomio de segundo grado y calcule los valores reales, imaginarios y complejos de λ).

5. Pruebe que las N raíces de la función $f(z) = z^N$ correspondiente al círculo unitario: $f(z) = 1$ son:

$$z_k = e^{i2\pi k/N} \qquad k = 0, \dots, N-1$$

Además, probar que para el caso general $f(z) = c$, donde $c = re^{i\phi}$, las raíces son:

$$z_k = (r)^{1/N} e^{i(\phi+2\pi k)/N} \qquad k = 0, \dots, N-1$$

6. Muestre que la *Transformada Discreta de Fourier*:

$$F[m] = \sum_{n=0}^{N-1} f[n] e^{-i2\pi mn/N}$$

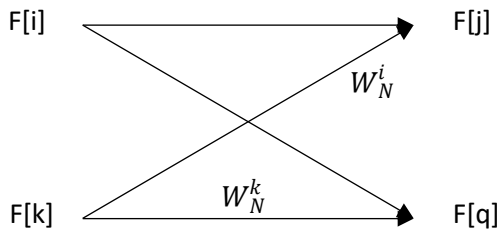
se puede descomponer a su vez en la suma de dos transformaciones:

$$F[m] = \sum_{k=0}^{\frac{N}{2}-1} f[2k] {W_N}^{2km} + {W_N}^{m} \sum_{k=0}^{\frac{N}{2}-1} f[2k+1] {W_N}^{2km}$$

Utilizando esta definición recursiva, demuestre que la complejidad del algoritmo de la *Transformada Rápida de Fourier* es $O(NlogN)$.

7. Provea un algoritmo iterativo para la *Transformada Rápida de Fourier*, utilizando *Mariposas*:

$$F_j = F_i + W_N^i F_k$$
$$F_q = F_i + W_N^k F_k$$



8. Con base al algoritmo de la *Transformada Rápida de Fourier*, proponga dos algoritmos de complejidad $O(NlogN)$ para calcular:

a) La versión discreta, dentro de un intervalo finito de tiempo T, de la *Transformada (Bilateral) de Laplace*, con variable compleja $s = \sigma + i\omega$:

$$F(s) = \int_{-\infty}^{\infty} f(t) e^{-st} dt$$

b) La *Transformada Z*, con variable compleja $z = re^{i\omega}$, para una señal finita de N muestras

$$F(z) = \sum_{k=0}^{N-1} f[k] z^{-k}$$

9. Muestre que el tiempo de ejecución del algoritmo de multiplicación de matrices de $N \times N$ de *Strassen* está representado por la ecuación de recurrencia $T(N) = 7T(N/2) + cN^2$. Demuestre que la complejidad del algoritmo es $O\left(N^{log_2 7}\right) \approx O(N^{2.8073})$.

10. Diseñe un algoritmo, mediante heurísticas voraces, para cambiar una cantidad X por una cantidad equivalente en monedas de denominaciones 1, 2, 3, 5, 7, 11, 13, 17, 19, 21 y 23 unidades, empleando la menor cantidad de monedas posible. Calcule la complejidad temporal y demuestre la optimalidad de su algoritmo.

11. Considere *el Problema de la Mochila* (*Knapsack*) y encuentre el tamaño del espacio de potenciales soluciones si se explorasen todas las posibles combinaciones. Diseñe un algoritmo voraz para encontrar soluciones "eficientes" aproximadas al problema.

12. Para el problema de *Knapsack*, incluya como restricciones adicionales el que se especifique si 2 objetos pueden o no colocarse uno junto a otro. ¿Esto reduce o aumenta el espacio de soluciones? Diseñe un algoritmo voraz para resolver el problema.

13. Diseñe un algoritmo para resolver, mediante heurísticas voraces, el problema de planificación de tareas con pesos. Calcule su complejidad.

14. Plantee un algoritmo voraz para llevar la logística de entrega de documentos en un servicio de mensajería, utilizando estrategias voraces. Considere recursos tanto en tiempo como espacio.

15. Demuestre que la complejidad del algoritmo de *Dijkstra* para encontrar la ruta mínima entre dos vértices de un grafo $G = (V, E)$ (cuyos conjuntos de vértices y aristas son V y E, respectivamente) es del orden $O(|V|^2)$.

16. Demuestre que la complejidad del algoritmo de *Kruskal* para encontrar el árbol recubridor mínimo de un grafo $G = (V, E)$ (cuyos conjuntos de vértices y aristas son V y E, respectivamente) es del orden $O(|V|^2 logV)$.

17. Modele una *memoria caché* de L niveles, especificando los criterios para traer datos de la memoria principal hacia la memoria caché y quitar datos de esta última, así como para mover datos entre niveles (considere una estrategia voraz). Calcule la complejidad espacial y temporal de su algoritmo.

18. Considere un tablero de $N \times N$ casillas y la pieza de ajedrez de caballo colocándose en una casilla arbitraria. Proponga un algoritmo voraz para encontrar un circuito que recorra todas las casillas hasta regresar a la casilla de partida, sin pasar más de una vez por una casilla; salvo la casilla inicial y final, las cuales coinciden. (Tip: considere la regla heurística de Warnsdorff y demuestre que puede encontrarse un recorrido del caballo con una complejidad $O(N^2)$).

19. Sea $a_0, a_1, a_2, \dots, a_{N-1}$ una secuencia de N números. Plantee un algoritmo de programación dinámica para encontrar la subsecuencia ascendente más larga $a_{i_0}, a_{i_1}, \dots, a_{i_k}$, tal que $0 \leq i_0 \leq i_1 \leq \cdots \leq i_k < N$ y $a_{i_0} \leq a_{i_1} \leq \cdots \leq a_{i_k}$. Calcule la complejidad de su algoritmo.

20. Utilizando programación dinámica, proponga un algoritmo para resolver el problema de planificación con pesos y demuestre que la función de optimización es

$$OPT(k) = max\{v_k + OPT(p(k)), OPT(k-1)\}$$

En donde k es la k-ésima tarea, $p(k)$ es la función que indica cual tarea es la más próxima al inicio de la tarea k, de tal forma que no haya traslapes de ejecución entre ambas. Demuestre que la complejidad del algoritmo es polinomial.

21. Sea P un conjunto de N puntos discretos provenientes del muestreo de puntos de una curva. Escriba un algoritmo para particionar el conjunto de puntos en segmentos lineales tal que se ajusten de la mejor manera a la curva original (inclusive puede uno desconocer la curva de donde provienen). Calcule la complejidad de su algoritmo. Tip: considere la siguiente función de optimización múltiple para los segmentos i y j:

$$OPT(j) = min_{1 \leq i \leq j}\{error_{ij} + COSTO(j) + OPT(i-1)\}$$

22. Diseñe un protocolo eficiente, mediante programación dinámica, para enviar y recibir mensajes óptimamente en una red el cual encuentre la distancia mínima entre un nodo fuente u y uno destino v. Escriba la función de optimización y el algoritmo correspondiente que utiliza memoización. Determine la complejidad de su algoritmo.

23. Resuelva el *Problema de la Mochila* utilizando programación dinámica. Considere que la mochila tiene una capacidad máxima W y todos los artículos suman un peso N. Demuestre que el problema puede resolverse con un algoritmo cuya complejidad es de orden $O(NW)$ (nota: todas las cantidades están indicadas por números enteros positivos).

24. Considere un muro de escalada artificial representado por una matriz de $M \times N$ celdas, en donde cada una tiene un número entero positivo que indica el grado de dificultad de asirse a un peldaño del muro. Diseñe un algoritmo, utilizando programación dinámica, para encontrar las rutas de mayor y menor dificultad para ascender y descender del muro, considerando que solamente puede desplazarse hacia adelante y de forma lateral, sin retrocesos, entre celdas contiguas. Indique la función de optimización correspondiente y calcule la complejidad de su algoritmo.

25. Sea $a_0, a_1, a_2, \dots, a_{N-1}$ una cadena de *ADN* de N nucleótidos $a_k \, \epsilon\{A, C, G, T\}$ que ha sido segmentada y traducida en R cadenas $b_0, b_1, b_2, \dots, b_{M-1}$ de *ARN mensajero*, $b_k \, \epsilon\{A, C, G, U\}$ (reemplazando $T \rightarrow U$), de longitud M. Utilizando programación dinámica, proponga un algoritmo que encuentre la *estructura secundaria* $S =$

$\{(b_i, b_j)|0 < i, j < M\}$ de cada subsecuencia de ARN con el mayor número de pares de base (b_i, b_j), acorde a las siguientes condiciones:

a) No tiene dobleces súbitos: $i < j - 4$.

b) Solamente se encuentran formadas por nucleótidos complementarios $A - U$ y $C - G$.

c) Los emparejamientos son únicos (i.e una base puede unirse a lo más con otra base).

d) No existen cruces entre pares de bases: si hay dos pares (b_i, b_j) y (b_k, b_l), no se puede suscitar que $i < k < j < l$.

Determine el orden de complejidad de su algoritmo.