Universidad de Chile
Departamento de Ciencias de la Computación
CC4102 - Diseño y Análisis de Algoritmos

Auxiliar 2 - "Técnicas de Análisis y Diseño"

Profesor: Pablo Barceló
Auxiliar: ~~Manuel~~ Ariel Cáceres Reyes

31 de Marzo del 2017

Dividir para reinar (DCC)

P1. **Inversiones**

Una inversión en un arreglo corresponde a un par de elementos $a[i] > a[j]$ con $i < j$.
Considere el problema de contar el número de inversiones de un arreglo.

a) Diseñe un algoritmo que haga $\mathcal{O}(n^2)$ comparaciones.

b) Diseñe un algoritmo que haga $\mathcal{O}(I+n)$ comparaciones (donde I es el número de inversiones).
Hint: Piense en ordenar.

c) Diseñe un algoritmo que haga $\mathcal{O}(n \log n)$ comparaciones.
Hint: Piense en ordenar.

P2. **Multiplicación de Naturales**

Considere el problema de multiplicar 2 números Naturales de n dígitos (en este problema las operaciones básicas serán de sumas, restas y multiplicaciones de números de 1 dígito).

a) Diseñe un algoritmo que haga $\mathcal{O}(n^2)$ operaciones básicas.

b) Diseñe un algoritmo del tipo "DCC" que resuelva el problema. ¿Cuál es su costo? ¿Se puede mejorar?

P3. **Multiplicación de Matrices**

Considere el problema de multiplicar 2 Matrices de dimensiones $n \times n$.

a) Diseñe un algoritmo de costo $\mathcal{O}(n^3)$.

b) Diseñe un algoritmo del tipo "DCC" que resuelva el problema. ¿Cuál es su costo? ¿Se puede mejorar?

Programación dinámica

P4. Subsecuencia palindrómica más larga

Diseñe un algoritmo para encontrar el largo de la subsecuencia más larga de un String $X[1, \ldots, n]$ que es un palíndrome. ¿Cuál es su costo?

P5. Árbol binario de búsqueda óptimo

Un árbol binario de búsqueda T que tiene como elementos los números $\{1, \ldots, n\}$ cada uno de los cuales tiene probabilidad de acceso $w_1, \ldots, w_n$, es óptimo si minimiza el valor:

$$\sum_{i=1}^{n} w_i(profundidad_T(i) + 1)$$

Diseñe un algoritmo que encuentre el costo del árbol binario de búsqueda óptimo. ¿Cuál es su costo?

Soluciones

P1. Inversiones

a) El algoritmo que mantiene un contador y revisa todos los pares de elementos en el arreglo aumentando en 1 el contador si corresponde a una inversión realiza $\mathcal{O}(n^2)$ comparaciones.

b) Si pensamos en el algoritmo InsertionSort, cada *swap* del algoritmo repara exactamente una inversión (pues se hacen entre elementos consecutivos) y considerando que cada inserción puede comparar a lo más un par de elementos no invertidos, el algoritmo hace $\mathcal{O}(I+n)$ comparaciones.

c) Usaremos MergeSort, pero le agregaremos que este cuente y retorne la cantidad de inversiones del arreglo que está procesando. Para esto recordemos que MergeSort divide el arreglo en dos mitades de igual tamaño:

- Recursivamente ordenaremos estas mitades y obtenemos las inversiones en ellas, digamos i_i e i_d.
- Hacemos el *merge* entre las dos mitades iniciando al mismo tiempo un contador $i_c \leftarrow 0$ que se encargará de guardar el número de inversiones cruzadas entre ambas mitades. Al hacer el *merge* si elegimos un elemento de la mitad izquierda dejamos i_c tal cual, sin embargo, si escogemos un elemento de la mitad derecha, significa que este elemento es menor a todos los elementos que quedan por escanear en la mitad izquierda por lo que aumentamos el contador en $i_c \leftarrow i_c + \textit{elementos no mergeados en mitad izquierda}$.
- Retornamos el arreglo ordenado y el número de inversiones $i_i + i_d + i_c$.

P2. Multiplicación de Naturales

a) El algoritmo de colegio que para multiplicar dos números toma cada dígito de uno y lo multiplica por el otro número (para esto lo multiplica con cada dígito del otro) ,sumando luego estos resultados toma $\mathcal{O}(n^2)$.

b) Si los números a multiplicar con x e y, y la primera mitad de dígitos de x es a y la segunda mitad es b y del mismo modo que c y d son de y, entonces se cumple que:

$$x \cdot y = (10^{n/2}a + b) \cdot (10^{n/2}a + b) = 10^n ac + 10^{n/2}(ad + bc) + bd$$

Notemos que el problema de multiplicar 2 números de n dígitos ($T(n)$) se transforma en un problema de multiplicar números de $n/2$ dígitos y sumar estos resultados ($\mathcal{O}(n)$), es decir:

$$T(n) = 4T(n/2) + \mathcal{O}(n)$$

Por teorema maestro concluimos que $T(n) \in \mathcal{O}(n^2)$.

Si ahora hacemos las siguientes multiplicaciones:

$$A = ac$$
$$B = bd$$
$$C = (a+b)(c+d)$$

Se cumple que:

$$10^n ac + 10^{n/2}(ad+bc) + bd = 10^n A + 10^{n/2}(C - A - B) + B$$

Y por lo tanto ahora tenemos que:

$$T(n) = 3T(n/2) + \mathcal{O}(n)$$

Concluyendo por teorema maestro que $T(n) \in \mathcal{O}(n^{\sim 1.59})$.

P3. Multiplicación de Matrices

a) El algoritmo de multiplicación de matrices que por cada elemento de la nueva matriz hace el producto punto entre la fila y la columna correspondiente de las matrices multiplicadas realiza $\mathcal{O}(n^3)$ operaciones. ⚡

b)
- Si ahora escribimos las matrices originales como la unión de 4 submatrices de la mitad de las dimensiones originales:

$$X = \begin{pmatrix} A & B \\ C & D \end{pmatrix}, \quad Y = \begin{pmatrix} E & F \\ G & H \end{pmatrix}$$

, se cumple que

$$X \cdot Y = \begin{pmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{pmatrix}$$

, es decir, reducimos nuestro problema de tamaño n a 8 problemas de tamaño $n/2$ más el trabajo de sumar matrices de $\mathcal{O}(n^2)$. Esto queda modelado por la ecuación:

$$T(n) = 8T(N/2) + cn^2$$

Finalmente por teorema maestro se obtiene $T(n) \in \mathcal{O}(n^3)$
- Volker Strassen publica en 1969 una manera alternativa de calcular el producto de las matrices. Teniendo en cuenta la misma división en submatrices del punto anterior calcula las siguientes matrices:

- $P_1 = A(F - H)$
- $P_2 = (A + B)H$
- $P_3 = (C + D)E$
- $P_4 = D(G - E)$
- $P_5 = (A + D)(E + H)$
- $P_6 = (B - D)(G + H)$
- $P_7 = (A - C)(E + F)$

De modo que

$$X \cdot Y = \begin{pmatrix} P_3 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{pmatrix}$$

, por lo que en este caso solo se realizan 7 problemas de tamaño $n/2$ y un trabajo de sumas y restas de $\mathcal{O}(n^2)$. Lo que se modela con:

$$T(n) = 7T(N/2) + cn^2$$

, que por teorema maestro es $\mathcal{O}(n^{\log_2 7}) = \mathcal{O}(n^{\approx 2.81})$

P4. Subsecuencia palindrómica más larga

Sea $L(i, j)$ el largo de la subsecuencia palindrómica más larga para $X[i \dots j], i \leq j$, estamos interesados en $L(1, n)$ y para calcularla lo haremos llenando una tabla de $L(i, j) \forall 1 \leq i \leq j \leq n$, lo que es posible pues el problema tiene subestructura óptima regida por la siguiente ecuación:

$$L(i,j) = \begin{cases} 1 & if\ i = j \\ 2 & elif\ i = j - 1 \wedge X[i] = X[j] \\ 2 + L(i+1, j-1) & elif\ X[i] = X[j] \\ max(L(i+1, j), L(i, j-1)) & else \end{cases}$$

Finalmente como el número de subproblemas es $\mathcal{O}(n^2)$ y resolver cada subproblema teniendo los mas chicos es $\mathcal{O}(1)$ el algoritmo derivado de la ecuación basado en programación dinámica es de costo $\mathcal{O}(n^2)$.

P5. Árbol binario de búsqueda óptimo

Definimos $c(i, j)$ como el costo del árbol binario de búsqueda óptimo entre los elementos i y j y lo que queremos obtener finalmente es $c(1, n)$.

Identificamos luego que c tiene subsestructura óptima (pues el árbol binario de búsqueda óptimo estará compuesto por una raíz y 2 subárboles binarios de búsqueda óptimos), dada por la siguiente ecuación:

$$c(i,j) = \begin{cases} w_i & i = j \\ \min_{i \leq r \leq j}\{c(i, r-1) + c(r+1, j) + \sum_{k=i}^{j} w_k\} & i < j \end{cases}$$

, donde el costo del árbol con solo un elemento es simplemente su probabilidad de acceso, y el costo del árbol entre los elementos i y j es el mínimo eligiendo r como la raíz, calculando el costo de los subárboles restantes y sumando las probabilidades de acceso de todos los elementos (por la raíz y porque los demás elementos descienden un nivel).

Finalmente podemos resolver este problema con "memoization" (tabla de caché) o construyendo las soluciones bottom-up (programación dinámica).
El costo del algoritmo estará dado entonces por la multiplicar el número de subproblemas por el tiempo para resolver cada subproblema estando los más chicos resueltos ($\mathcal{O}(n)$, pues el mínimo se calcula entre i y j que pueden ir de 1 a n), es decir $\mathcal{O}(n^2) \cdot \mathcal{O}(n) = \mathcal{O}(n^3)$.