INF221 – Algoritmos y Complejidad

Clase #11
Programación Dinámica

Aldo Berrios Valenzuela

Martes 6 de septiembre de 2016

1. Programación Dinámica

Queremos calcular el producto de n matrices, $A_1 \cdot A_2 \cdot \ldots \cdot A_n$, donde A_i es $n_i \times n_{i+1}$ (si no es $A_i \cdot A_{i+1}$ imposible)

La técnica tradicional de multiplicar una matriz de $r \times s$ por otra $s \times t$ toma rst multiplicaciones, y usaremos esto como medida de costo. Nuestro resultado no depende realmente del detalle de ésto.

Sabemos que la multiplicación de matrices es asociativa:

$$A \cdot (B \cdot C) = (A \cdot B) \cdot C$$

El trabajo total depende del orden:

$A : 2 \times 12$

$B : 12 \times 3$

$C : 3 \times 4$

Entonces, el costo de calcular $(A \cdot B) \cdot C$ es:

$$(A \cdot B) \cdot C \rightsquigarrow 2 \cdot 12 \cdot 3 + 2 \cdot 3 \cdot 4 = 96$$

Explicamos de donde vienen los números anteriores:

- Comenzamos multiplicando $A \cdot B$. Como ambas matrices son de 2×12 y 12×3 respectivamente. Es claro que la cantidad de multiplicaciones que tenemos que hacer para obtener la matriz resultante es $2 \cdot 12 \cdot 3 = 72$. Dado lo anterior, se tiene que la dimensión de $A \cdot B$ es 2×3.
- Llegados a este punto, consideramos la matriz $A \cdot B$ como una sola, la que llamaremos X. Entonces, siguiendo lo anterior, vemos que $X \cdot C$ nos toma una cantidad de $2 \cdot 3 \cdot 4 = 24$ multiplicaciones.
- Finalmente, sumamos la cantidad de multiplicaciones que nos llevó a obtener $A \cdot B = X$ y $X \cdot C$, lo que se traduce a:

$$(A \cdot B) \cdot C \rightsquigarrow 2 \cdot 12 \cdot 3 + 2 \cdot 3 \cdot 4 = 96 \tag{1.1}$$

Continuamos con el cálculo del costo de obtener $A \cdot (B \cdot C)$:

$$A \cdot (B \cdot C) \rightsquigarrow 2 \cdot 12 + 4 + 12 \cdot 3 \cdot 4 = 240$$

Consideremos "de afuera adentro": Si el último producto es:

$$\underbrace{(A_1 \cdot A_2 \cdot \ldots \cdot A_i)}_{\text{óptimo}} \underbrace{(A_{i+1} \cdot \ldots \cdot A_n)}_{\text{óptimo}} \tag{1.2}$$

Nótese que $(A_1 \cdot A_2 \cdot \ldots \cdot A_i)$ y $(A_{i+1} \cdot \ldots \cdot A_n)$ también deben calcularse de forma óptima, de lo contrario el cuento no sirve. En consecuencia, tendremos que dividir nuevamente lo que está entre paréntesis sucesivamente hasta obtener un producto de dos matrices. Es importante destacar que no conocemos i... así que tendremos que probar todas las opciones. Ojo, muchos subproblemas se repiten.

Idea de programación no recursiva:

- $T\left[i,j\right]$: costo de calcular el producto $A_i \cdot \ldots \cdot A_j$.

Inicialmente:

$$T[i,i] = 0$$

Sabemos que:

$$T\left[i,j\right] = \min_k \{T[i,k] + T\left[k+1,j\right] + \underbrace{n_i \cdot n_{k+1} \cdot n_{j+1}}_{\text{costo del producto}}\} \tag{1.3}$$

Donde k corresponde a la k-ésima matriz que hicimos el corte (donde dividimos con los paréntesis. Por ejemplo, en (1.2) se tiene que $k = i$) y que dio el valor mínimo de (1.3). Además, n corresponde al valor izquierdo de la dimensión de la matriz. Por ejemplo, si nuestra A_i tiene dimensión $a \times b$, $A_{k+1} : c \times d$ y $A_{j+1} : e \times f$, se tiene que $n_i = a$, $n_{k+1} = c$ y $n_{j+1} = e$ respectivamente.

Nos interesa: $T[1,n]$. Calculamos:

$$T[i,i]$$
$$T[i,i+1]$$
$$\vdots$$

Esta da sólo el costo... hay que registrar con cada $T\left[i,j\right]$ cuál fue el k que dio el mínimo. Siguiendo esos desde $T[1,n]$ da el orden óptimo.

Ejemplo 1.1. Supongamos que queremos calcular el producto de matrices:

$$A \cdot B \cdot C \cdot D \cdot E \cdot F$$

donde cada uno de los tamaños son:

- $A = A_1 : 1 \times 2$
- $B = A_2 : 2 \times 7$
- $C = A_3 : 7 \times 4$
- $D = A_4 : 4 \times 2$
- $E = A_5 : 2 \times 9$
- $F = A_6 : 9 \times 1$

Para la primera iteración, es decir, $T[i,i]$ es claro que intentamos calcular la cantidad de productos que son necesarios para hacer la multiplicación A_i. Cómo no lo estamos multiplicando con nada más, se tiene que la cantidad de multiplicaciones necesarias es $T[i,i] = 0$ para cualquier i. Agregamos esta información al Cuadro 1. Nótese que cada casilla del Cuadro 1 está configurado de la forma que se logra apreciar en el Cuadro 2 independientemente de la iteración en la que vayamos.

Para la segunda iteración, tenemos que calcular todos los $T[i,i+1]$, es decir, la mínima cantidad de multiplicaciones para obtener $A_i \cdot A_{i+1}$. Como solo tenemos dos matrices involucradas, es bastante fácil realizar este cálculo:

- $T[1,2] = 1 \cdot 2 \cdot 7 = 14$
- $T[2,3] = 2 \cdot 7 \cdot 4 = 56$
- $T[3,4] = 7 \cdot 4 \cdot 2 = 56$

	1	2	3	4	5	6
1	0 1					
2		0 2				
3			0 3			
4				0 4		
5					0 5	
6						0 6

Cuadro 1: Para la primera iteración no necesitamos realizar multiplicaciones.

	1	2	3	4	5	6
1	$T[1,1]$ k	$T[1,2]$ k	$T[1,3]$ k	$T[1,4]$ k	$T[1,5]$ k	$T[1,6]$ k
2		$T[2,2]$ k	$T[2,3]$ k	$T[2,4]$ k	$T[2,5]$ k	$T[2,6]$ k
3			$T[3,3]$ k	$T[3,4]$ k	$T[3,5]$ k	$T[3,6]$ k
4				$T[4,4]$ k	$T[4,5]$ k	$T[4,6]$ k
5					$T[5,5]$ k	$T[5,6]$ k
6						$T[6,6]$ k

Cuadro 2: En cada iteración, los $T[i,j]$ con $j > i$ se calculan a través de la ecuación (1.3). Por otro lado, k al corte de la k-ésima matriz con la cual obtuvimos el mínimo valor de $T[i,j]$.

	1	2	3	4	5	6
1	0 1	14 1				
2		0 2	56 2			
3			0 3	56 3		
4				0 4	72 4	
5					0 5	18 5
6						0 6

Cuadro 3: Se agregan los valores correspondientes a la segunda iteración. Es importante destacar, que el para el cálculo de $A_i \cdot A_{i+1}$ lo hacemos entre i e $i+1$, es decir, $k = i$ para cualquier valor de i.

- $T[4,5] = 4 \cdot 2 \cdot 9 = 72$
- $T[5,6] = 2 \cdot 9 \cdot 1 = 18$

Luego, agregamos estos valores al Cuadro 1. Los cambios se pueden apreciar en el Cuadro 3.

Seguimos con la tercera iteración. En esta ocasión, tendremos que calcular los $T[i, i+2]$, es decir, la cantidad de multiplicaciones mínima para obtener $A_i \cdot A_{i+1} \cdot A_{i+2}$. Para ello, comenzamos calculando $T[1,3]$, es decir:

$$T[1,3] = \min_k \{T[1,k] + T[k+1,3] + n_1 \times n_{k+1} \times n_4\} \tag{1.4}$$

Vamos por partes:

- Para $k = 1$:

$$\begin{aligned} T[1,3]\,(k=1) &= T[1,1] + T[2,3] + n_1 \cdot n_2 \cdot n_4 \\ &= 0 + 56 + 1 \cdot 2 \cdot 4 = 64 \end{aligned}$$

- Para $k = 2$:

$$\begin{aligned} T[1,3]\,(k=2) &= T[1,2] + T[3,3] + n_1 \cdot n_3 \cdot n_4 \\ &= 14 + 0 + 1 \cdot 7 \cdot 4 = 42 \end{aligned}$$

Por lo tanto, de acuerdo a lo anterior la ecuación (1.4) obtiene el mínimo cuando hacemos el corte en $k = 2$. Es decir, obtenemos el mínimo de productos para $A_1 \cdot A_2 \cdot A_3$ si hacemos un corte

$$(A_1 \cdot A_2) \cdot (A_3) \tag{1.5}$$

En otras palabras, si resolvemos primero $A_1 \cdot A_2$ y luego, la matriz resultante se la multiplicamos a A_3 obtendremos el mínimo para $T[1,3]$. Más adelante agregaremos esto a la tabla.

Para dejar más en claro cómo calcular (1.3) repetiremos los pasos anteriores, pero para calcular $T[2,4]$, el que se puede obtener a través de:

$$T[2,4] = \min_k \{T[2,k] + T[k+1,4] + n_2 \times n_{k+1} \times n_5\} \tag{1.6}$$

Vamos por partes:

- Para $k = 2$:

$$\begin{aligned} T[2,4]\,(k=2) &= T[2,2] + T[3,4] + n_2 \times n_3 \times n_5 \\ &= 0 + 56 + 2 \cdot 7 \cdot 2 = 84 \end{aligned}$$

- Para $k = 3$:

$$\begin{aligned} T[2,4]\,(k=3) &= T[2,3] + T[4,4] + n_2 \cdot n_4 \cdot n_5 \\ &= 56 + 0 + 2 \cdot 4 \cdot 2 = 72 \end{aligned}$$

Nuevamente, de acuerdo a la ecuación (1.6) el mínimo para $T[2,4]$ se obtiene para $k = 3$, lo que implica que la cantidad mínima de productos para $A_2 \cdot A_3 \cdot A_4$ se obtiene al hacer un corte en $k = 3$, es decir,

$$(A_2 \cdot A_3) \cdot (A_4) \tag{1.7}$$

Esto nos dice que si resolvemos primero el producto $A_2 \cdot A_3$ y luego la matriz resultante se la multiplicamos a A_4, vamos a hacerlo en la menor cantidad de operaciones posibles.

El objetivo de este ejemplo es mostrar cómo funciona el algoritmo de Programación Dinámica, por lo que no es necesario explicar paso a paso cómo obtener el resto de las casillas de la tabla. Así que agregamos los valores obtenidos al Cuadro 3, cuyo resultado se refleja en el Cuadro 4.

Nuestro principal interés en esto es obtener el último producto representado en (1.2). Para efectos de nuestra tablita, este último producto está ubicado en la casilla pintada marcada con un × del Cuadro 5.

■

	1	2	3	4	5	6
1	0 1	14 1	42 2			
2		0 2	56 2	72 3		
3			0 3	56 3		
4				0 4	72 4	
5					0 5	18 5
6						0 6

Cuadro 4: Se agregan algunos valores correspondientes de la tercera iteración. No es necesario explicar cómo rellenar el resto de la tabla.

	1	2	3	4	5	6
1	0 1	14 1	42 2			× k
2		0 2	56 2	72 3		
3			0 3	56 3		
4				0 4	72 4	
5					0 5	18 5
6						0 6

Cuadro 5: A final de cuentas, lo que nos interesa es obtener el "último producto". Este se obtiene en $T[1,6]$, ya que nos dice el valor de k para hacer el corte.

Demostración (programación dinámica). Como siempre, demostramos que cumple con:

- *Estructura inductiva:* Dada la selección k (última) se subdivide en problemas, cuyas soluciones viables $+k$ dan una solución viable para todo.
- *Subestructura óptima:* Con soluciones óptimas para $1\ldots k$ y $k+1\ldots n$ obtenemos la solución óptima para $1\ldots n$, *suponiendo* k.
- *Elección completa:* Elegimos aquel k que da el mejor "último paso" (en el fondo a nosotros no nos interesa obtener el valor del costo mínimo, si no que averiguar cómo obtener ese valor).

□