



Escuela Superior de Cómputo.
Instituto Politécnico Nacional, México.



Práctica 7: Multiplicación de una secuencia de matrices.

Blancas Pérez Bryan Israel
orionmunecaycanica@gmail.com

Resumen: Implementar un algoritmo para encontrar el orden óptimo de multiplicar las matrices de una secuencia de matrices.

Palabras Clave: Optimizar, Multiplicación de matrices, Secuencia de matrices, Complejidad Computacional.

1. Introducción

En esta práctica se implementará un algoritmo que de como salida, la mejor configuración de paréntesis para efectuar la multiplicación de una secuencia de matrices. Esto es con el fin de no multiplicar la secuencia conforme están ordenadas las matrices, ya que esto puede hacer que se realicen más operaciones escalares entre los elementos de las matrices. En este caso, lo óptimo, es realizar todas las multiplicaciones entre las matrices haciendo el menor número de productos escalares.

2. Conceptos Básicos

Multiplicación de una secuencia de matrices.

Supongamos que tenemos una secuencia de 3 matrices que queremos multiplicar.

Sea $A_{10 \times 100}$, $B_{100 \times 5}$ y $C_{5 \times 50}$, y supongamos dos configuraciones de paréntesis diferentes.

Configuración 1.

$$(AB)_{10 \times 5} \rightarrow 10 \times 100 \times 5 = 5000$$

$$(AB)C_{10 \times 50} \rightarrow 10 \times 5 \times 50 = 2500$$

Suma total de productos escalares = 7500.

Configuración 2.

$$(BC)_{100 \times 50} \rightarrow 100 \times 5 \times 50 = 25000$$

$$A(BC)_{10 \times 50} \rightarrow 10 \times 100 \times 50 = 50000$$

Suma total de productos escalares = 75000.

Entonces, como se aprecia en el ejemplo anterior, en realidad se puede reducir el número de productos escalares que se realizan, para multiplicar una misma secuencia de matrices, mejorando así, el uso de los recursos del computador.

En el caso del ejemplo anterior, podemos concluir que multiplicar las matrices con la configuración 1: $((AB)C)$, es más conveniente que con la configuración 2: $A(BC)$.

El algoritmo que se implementará más abajo en este reporte, muestra como salida, la mejor configuración de paréntesis de entre todas las posibles.

3. Experimentación y Resultados

Ejercicio 1.

Implementar el algoritmo de la multiplicación de una secuencia de matrices.

- i) Como entrada, el algoritmo tendrá n matrices A_i de tamaño $P_{i-1} \times P_i$.
- ii) Como salida, se mostrará la configuración de matrices generada por el algoritmo.

Pseudocódigo del algoritmo:

```
1  secuencia_matrices(P)
2
3  n=P.length-1
4  M[1,...,n][1,...,n] y S[1,...,n-1][2,...,n]
5
6  for i = 1 to n do
7      m[i][i] = 0
8
9  for l = 2 to n do
10     for i = 1 to n-l+1 do
11         j = i+l-1
12         m[i][j] = infinito
13         for k = i to j-1 do
14             q = m[i][k] + m[k+1][j] + p[i-1]p[k]p[j]
15             if q < m[i][j] then
16                 m[i][j] = q
17                 s[i][j] = k
18
19  return m y s
20
21 imprime_parentizacion_optima(s,i,j)
22
23 if i = j then
24     imprime Ai
25 else
26     imprime "("
27     imprime_parentizacion_optima(s,i,s[i][j])
28     imprime_parentizacion_optima(s,s[i][j]+1,j)
29     imprime ")"
```

Como vemos, el algoritmo hace uso de la función "secuencia de matrices", la cual crea dos matrices que llenará para luego retornarlas. La función `imprime_parentizacion_optima`, es la que calcula la configuración de paréntesis óptima, y utiliza la matriz `s` creada en la primer función.

Resultados.

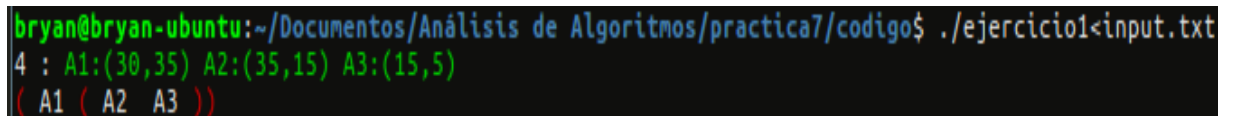
Ejecución 1.

Input.

$$P = [30, 35, 15, 5]$$

Output.

$$(A_1(A_2A_3))$$



```
bryan@bryan-ubuntu:~/Documentos/Análisis de Algoritmos/practica7/codigo$ ./ejercicio1<input.txt
4 : A1:(30,35) A2:(35,15) A3:(15,5)
( A1 ( A2 A3 ) )
```

Figura 1: 1ra ejecución.

Configuraciones posibles para la ejecución 1.

1. $(A_1(A_2A_3))$. Con un total de $(30 \times 35 \times 5) + (35 \times 15 \times 5) = 7875$ operaciones escalares.
2. $((A_1A_2)A_3)$. Con un total de $(30 \times 35 \times 15) + (30 \times 15 \times 5) = 18000$ operaciones escalares.

Por lo tanto concluimos que la salida mostrada por el algoritmo en la figura 1, es en realidad la configuración óptima para multiplicar las matrices.

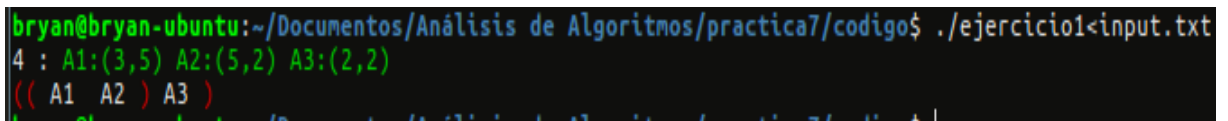
Ejecución 2.

Input.

$$P = [3, 5, 2, 2]$$

Output.

$$((A_1 A_2) A_3)$$



```
bryan@bryan-ubuntu:~/Documentos/Análisis de Algoritmos/practica7/codigo$ ./ejercicio1<input.txt
4 : A1:(3,5) A2:(5,2) A3:(2,2)
(( A1 A2 ) A3 )
```

Figura 2: 2da ejecución.

Configuraciones posibles para la ejecución 2.

1. $(A_1(A_2A_3))$. Con un total de $(3 \times 5 \times 2) + (5 \times 2 \times 2) = 50$ operaciones escalares.
2. $((A_1A_2)A_3)$. Con un total de $(3 \times 5 \times 2) + (3 \times 2 \times 2) = 42$ operaciones escalares.

Por lo tanto concluimos que la salida mostrada por el algoritmo en la figura 1, es en realidad la configuración óptima para multiplicar las matrices.

4. Conclusiones

Esta práctica se me hizo muy sencilla, puesto que los algoritmos eran fáciles de implementar en un lenguaje de programación, pero lo difícil es la lógica implementada. El llenado de las matrices en la función "secuencia_matrices", es muy complejo de comprender rápido, y me costo entenderle a la perfección.

Tuve algunos problemas al momento de implementar el algoritmo, hasta que decidí hacerlo con matrices de doble apuntador, y memoria dinámica.

5. Anexo

Problema: Utilizando método por sustitución, muestre:

$$P(n) = \sum_{k=1}^{n-1} P(k)P(n-k) \text{ si } n \geq 2 \text{ y } P(n=1) = 1 \text{ es } \Omega(2^n).$$

Utilizando Sustitución hacia adelante.

$$P(n=1) = 1$$

$$P(n=2) = P(1)P(1) = 1$$

$$P(n=3) = P(1)P(2) + P(2)P(1) = 2$$

$$P(n=4) = P(1)P(3) + P(2)P(2) + P(3)P(1) = 5$$

$$P(n=5) = P(1)P(4) + P(2)P(3) + P(3)P(2) + P(4)P(1) = 14$$

$$P(n=6) = P(1)P(5) + P(2)P(4) + P(3)P(3) + P(4)P(2) + P(5)P(1) = 42$$

Tabla de valores	
n	P(n)
1	1
2	1
3	5
4	5
5	14
6	42
10	4862
15	2674440

Cuadro 1: Tabla de valores de P(n).

Por lo tanto vemos que la función tiene un comportamiento exponencial.

Por lo tanto $P(n) \in \Omega(2^n)$.

6. Bibliografía

https://es.wikibooks.org/wiki/Optimizaci%C3%B3n_del_Producto_de_Matrices