Minimización multiplicación de matrices

Jorge Luis Esposito Albornoz[1] Juan Sebastián Herrera Guaitero[1]

[1]Departamento de Ingeniería de Sistemas, Pontificia Universidad Javeriana
Bogotá, Colombia
{jesposito,jsebastianherrera}@javeriana.edu.co

1 de septiembre de 2022

Resumen

En este documento se presenta la formalización del problema de la multiplicación de matrices. **Palabras clave:** minimización, multiplicación.

Índice

1. Formalización del problema

Cuando se habla de realizar una multiplicación de matrices, en primer lugar nos viene a la mente realizar el procedimiento clásico que consiste en realizar las operaciones escalares correspondientes. No obstante al pasar esta definición a una implementación en código, se hace evidente que la complejidad de dicho algoritmo es sumamente elevada, por esto se procede a dar solución a este problema por medio del uso de programación dinámica para evitar llenar la pila de ejecución.

1.1. Definición del problema del "Multiplicación de matrices"

La multiplicación matricial en cadena, es un problema de optimización relativo a la forma más eficiente de multiplicar una secuencia dada de matrices. El problema no es realmente realizar las multiplicaciones, sino simplemente decidir la secuencia de las multiplicaciones matriciales involucradas. El problema puede resolverse mediante programación dinámica.

Hay muchas opciones porque la multiplicación de matrices es asociativa. En otras palabras, no importa cómo se ponga el producto entre paréntesis, el resultado obtenido seguirá siendo el mismo. Por ejemplo, para cuatro matrices $A, B, C\ y\ D$, hay cinco opciones posibles:

$$((AB)C)D = (A(BC))D = (AB)(CD) = A((BC)D) = A(B(CD)).$$

Aunque no afecta al producto, el orden en que se ponen los términos entre paréntesis afecta al número de operaciones aritméticas simples necesarias para calcular el producto, es decir, a la complejidad computacional.

- Entradas: $S = < s_i : 0 \leq i \leq n \wedge s_i \in N >$, donde s_0 es la cantidad de las de la primera matriz, s_1 es la cantidad de columnas de la primera matriz y la cantidad de las de la segunda matriz y, en general, s_{ii}, es la cantidad de las de A_i y la cantidad de columnas de A_{i-1}, entonces se simplicaría:

$$m_{ikj} = s_{i-1} d_k d_j$$

- Salidas: R, cadena de texto de los corchetes ubicados entre matrices para expresar la multiplicación.

2. Algoritmos de solución

Algoritmo 1 BracketsChainMatrix.

```
1:  procedure MatrixChainBK(B, R, i, j)
2:      if i = j then
3:          R + ← "A" + string(i + 1)
4:      else
5:          R + ← "("
6:          q ← B_ij
7:          MatrixChainBk(B, R, i, q)
8:          MatrixChainBk(B, R, q + 1, j)
9:          R + ← ")"
10:     end if
11: end procedure
```

Algoritmo 2 Multiplicación de matrices.

```
1: procedure MATRIXCHAINB(S)
2:     M ← [|S|][|S|]                                  ▷ Inicializar matriz en 0
3:     B ← [|S|][|S|]                                  ▷ Inicializar matriz en −1
4:     for i ← |S| − 2 to − 1 step: − 1 do
5:         for j ← i + 1 to |S| do
6:             q ← INF                                 ▷ Asignar infinito a q porque se esta minimizando
7:             m ← 0
8:             for k ← i to j do
9:                 l ← M_ij
10:                r ← M_{k+1j}
11:                v ← l + r + (S_{i−1} ∗ S_k ∗ S_j)
12:                if v < q then
13:                    q ← v
14:                    n ← k
15:                end if
16:            end for
17:            M_ij ← q
18:            B_ij ← m
19:        end for
20:    end for
21:    return M, B
22: end procedure
23:
```

2.0.1. Análisis de complejidad

Para cada iteración del bucle exterior, el número total de las ejecuciones en los bucles interiores sería equivalente a la longitud de la matriz. En general, si la longitud de la matriz es N, la complejidad temporal total sería O(N * N * N) = O(N^3).

2.0.2. Invariante

La invariante es que la tabla se esté llenando correctamente.

2.1. Análisis experimental

En esta sección se presentara el análisis experimental de la multiplicación de matrices con entradas de 2 a 1000.

2.1.1. Protocolo

1. Definir el rango de la secuencia.
2. Correr el algoritmo para obtener el tiempo de ejecución entre cada entrada.
3. Generar la gráfica para ver el comportamiento de las multiplicaciones de matrices.

2.1.2. Procedimiento

1. El rango se definió entre 50 - 1000 con saltos de 50 .
2. El programa en cada ejecución guarda en un archivo CSV el tiempo gastado en cada entrada.
3. Se obtuvieron como resultados 20 de datos con los que se generará la gráfica para análizar el comportamiento.

2.1.3. Resultado

Tiempo	#matrices
0.0049863	100
0.0378988	150
0.1246669	200
0.2942135	250
0.6353025	300
1.2237322	350
1.8151438	400
2.8583634	450
3.9823508	500
5.5551524	550
7.7920892	600
10.310501	650
12.134605	700
17.434725	750
19.363056	800
24.019817	850
28.260456	900
34.396611	950
44.67953	1000

Figura 1: Resultados obtenidos en la experimentación

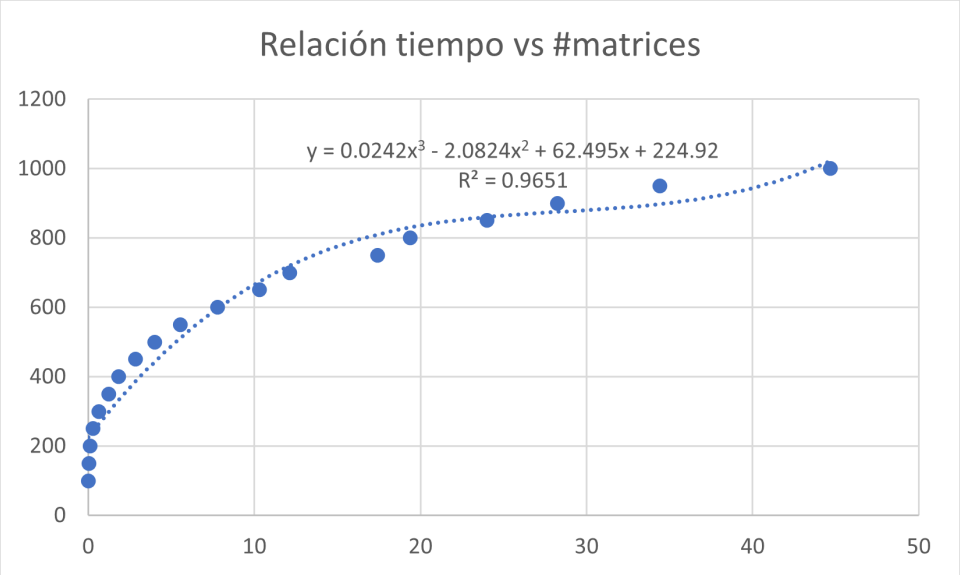

Figura 2: Gráfica de relación tiempo y número de matrices.

2.1.4. Análisis

Como se puede evidenciar en la gráfica el comportamiento de la gráfica es de un polinomio de grado 3, es decir que a medida en que se incrementa el tiempo de ejecución para la minimización aumentará a medida de un x^3. De igual modo, gracias al R^2 obtenido en el ajuste de regresión polinomial de orden 3, que fue 0,96 podemos deducir que el ajuste va tender a 1 cuando al predicirlo con IA. Asimismo, mientras R^2 este mas cercano a 1 muestra el ajuste perfecto de la linea de tendencia. Al tener tres bucles anidados podemos conseguir un $O(n^3)$ pero en realidad es mejor que eso porque los bucles internos no se ejecutan exactamente n veces. Sin embargo, se acasa apreciar el comportamiento polinomial de orden 3.

2.2. Conclusiones

1. No es difícil decir que el código se ejecuta en tiempo polinómico. Tenemos tres bucles anidados y lo peor que podemos conseguir es $O(n^3)$ pero en realidad es mejor que eso porque los bucles internos no se ejecutan exactamente n veces. Esto demuestra que la Programación Dinámica es una técnica poderosa si el problema en cuestión puede ser resuelto usando DP.

2. Como el número de columnas de la matriz A es igual al número de filas de la matriz E, se puede concluir que el producto de AE está definido.

3. Se puede concluir que al aumentar el número de matrices, el tiempo de ejecución tiene una tendencia n^3.