Escritura del problema del problema de multiplicación matricial

María José Niño Rodríguez[1] David Santiago Quintana Echavarria[1]

[1]Departamento de Ingeniería de Sistemas, Pontificia Universidad Javeriana
Bogotá, Colombia
{ma.nino, quintanae-david}@javeriana.edu.co

1 de septiembre de 2022

Resumen

En este documento se presenta la formalización del problema del cálculo de parentización que minimice la cantidad de multiplicaciones escalares en una composición de matrices implementando la estrategia de "programación dinámica". **Palabras clave:** programación dinámica, matrices, Julia, algoritmo, optimización, minimizar, formalización, complejidad.

Índice

1. Introducción

Los algoritmos de programación dinámica nos ayudan a optimizar procesos, evita el recalculo de operaciones ya calculadas previamente, evita la concurrencia y se implementa en problemas comúnmente de optimización, ya sea maximizando o minimizando. En este documento se expone un problema que busca minimizar la cantidad de multiplicaciones escalares en una composición de matrices, con el objetivo de mostrar: la formalización del problema (sección 2), la escritura formal de este algoritmo (sección 3) junto con su análisis de complejidad e invariante.

2. Formalización del problema

A partir de una secuencia ($\mathbb{N}$), se quiere obtener otra secuencia con los mismos valores de la secuencia original, pero en un orden de multiplicación, separado por paréntesis cuyas cantidades de multiplicación sea la mínima. Implementando memorización y composición de matrices se puede llegar a la solución de optimización.

2.1. Definición del problema de "minimizar multiplicaciones escalares en una composición de matrices"

Así, el problema se define a partir de:

1. una secuencia de naturales ($\mathbb{N}$)

para producir una nueva secuencia de números naturales reagrupados.

- Entradas:
  - Una secuencia $D \in \mathbb{N}$.
- Salidas:
  - Una secuencia $r' \in \mathbb{N}$ con paréntesis entre los valores de la secuencia, representando el orden de operación entre ellos.

3. Algoritmo de solución

3.1. Algoritmo de programación dinámica

La idea de este algoritmo es: a partir de la secuencia de matrices como entrada, encontrar la secuencia de multiplicación entre las matrices que al multiplicarse requiera un menor numero de multiplicaciones escalares. Se usa memorización para almacenar los valores ya calculados, optimizar el proceso y por medio de una cadena matricial, encontrar la secuencia esperada.

3.1.1. Análisis de complejidad

En el caso del algoritmo de programación dinámica, se encuentran con tres ciclos anidados por lo que su complejidad es $O(n^3)$, sin embargo para analizar la composición de matriz y llegar a la secuencia minimizada, se implementa un algoritmo de dividir y vencer, su complejidad se calcula por medio del teorema maestro, este cuenta con dos concurrencia, se hace una partición a la mitad y el resto del algoritmo cuenta con una complejidad de $O(1)$ por lo que se llega a:

$$
\begin{aligned}
f(n) \in O(n^{\log_b(a-\varepsilon)}) n^{\log_b(a-\varepsilon)} &= 1 \\
n^{\log_2(2-\varepsilon)} &= 1 \\
\varepsilon &= 1 \\
n^{\log_2(2-1)} &= 1 \\
n^{\log_2(1)} &= 1 \\
n^0 &= 1
\end{aligned}
$$

Como el primer caso cumple se obtiene que la complejidad es

$$
\begin{aligned}
T(n) \in \Theta(n^{\log_a(b)}) \\
\Theta(n^{\log_2(2)}) \\
\Theta(n^1) \\
T(n) = \Theta(n)
\end{aligned}
$$

La complejidad total del algoritmo resulta siendo la suma de ambos algoritmos, por lo que la complejidad total es: $O(n^3) + O(1)$, por lo que se llega a una complejidad de $O(n^3)$.

Algoritmo 1 Minimización de multiplicaciones escalares

Require: $D \in \mathbb{N}$

Ensure: El resultado será una secuencia $n' \in \mathbb{N}$ encadenada entre sus valores con paréntesis que encapsulan prioritariamente las operaciones que debe tomar las matrices para minimizar las operaciones escalares de su multiplicación.

```
1: procedure MINIMIZAR MULTIPLICACIONES ESCALARES(D)
2:     n ← |D|
3:     for i ← 1 to n do
4:         m[i][i] ← 0
5:     end for
6:     L ← 2
7:     for L ← 2 to n do
8:         i ← 1
9:         while i¡n-L+1 do
10:            j ← i + L − 1
11:            m[i][j] ← ∞
12:            k ← i
13:            while k¡=j-1 do
14:                q ← m[i][k] + m[k + 1][j] + D[i − 1] ∗ D[k] ∗ D[j]
15:                if q < m[i][j] then
16:                    m[i][j] ← q
17:                    r[i][j] ← k
18:                end if
19:                k + +
20:            end while
21:            i + +
22:        end while
23:    end for
24:    return GENERARPARENTESIS(r, 1, n − 1, n, D)
25: end procedure
```

Algoritmo 2 Generar secuencia parentesis, dividir y vencer

Require: $r, b, e, n, D \in \mathbb{N}$

Ensure: El resultado será una secuencia $n' \in \mathbb{N}$ encadenada entre sus valores con paréntesis que encapsulan prioritariamente las operaciones que debe tomar las matrices para minimizar las operaciones escalares de su multiplicación.

```
1: procedure GENERARPARENTESIS(r, b, e, n, D)
2:     if i = j then
3:         printD[i]
4:     end if
5:     print(
6:     GENERARPARENTESIS(r, b, r[(b ∗ n)][e]], n, D)
7:     GENERARPARENTESIS(r, r[(b ∗ n)][e + 1]], e, n, D)
8:     print)
9: end procedure
```

3.1.2. Invariante

A medida que se recorren las posibles maneras de hacer las combinaciones entre las operaciones matriciales, se van almacenando de manera de matriz los resultados ya calculados, usando memorización para evitar re calculos, de igual manera se almacenan las posiciones en una matriz resultante que indica los caminos para llegar al resultado. Por medio de la estrategia de dividir y vencer se recorre la matriz de manera inversa imprimiendo los valores donde se encuentra la minimización con sus respectivos paréntesis.

4. Conclusiones

1. A partir de la escritura de un algoritmo de programación dinámica se logro entender, dividir, analizar una problemática de optimización, en este caso de minimización donde se optimizaba la cantidad de multiplicaciones escalares.

2. Aunque se implementa un algoritmo de programación dinámica, es viable como parte de la solución también usar la estrategia de dividir y vencer para analizar la matriz resultante y llegar al resultado esperado.