Universidad del Bío-Bío

ICINF

INFORME

Análisis de Algoritmos de Multiplicación de Matrices

Integrantes :
Bastian Rodriguez
Marcelo Paz
Nicolás Gómez
Juan Henriquez
Víctor López

Docente :
Rodrigo Torres

16 de octubre de 2023

1. Comportamiento Asintótico

1.1. Algoritmo Tradicional

Algorithm 1 Algoritmo Tradicional

```
function AlgoritmoTradicional(A, B, C, n)
    for i = 0 to n-1 do
        for j = 0 to n-1 do
            for k = 0 to n-1 do
                C[i][j] += A[i][k] × B[k][j]
            end for
        end for
    end for
    return C
end function
```

El algoritmos tradicional para multiplicar matrices de $N \times N$ tiene un comportamiento asintótico:

$$T(n) = \Theta(n^3) \tag{1}$$

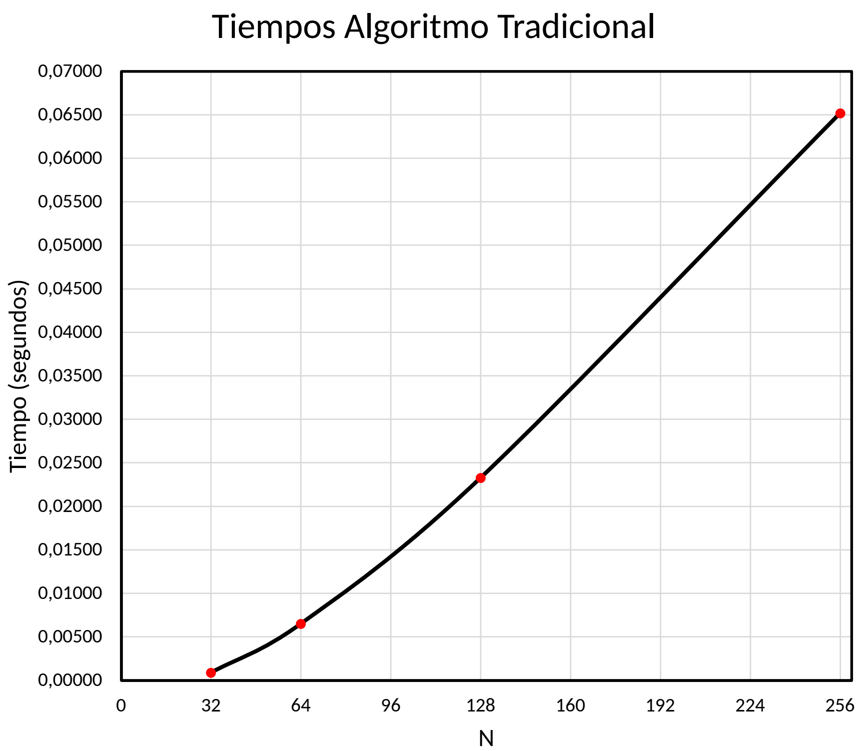

Figura 1: Gráfico Algoritmo Tradicional $T(n) = \Theta(n^3)$

1.2. Algoritmo DR1

Algorithm 2 Algoritmo DR1

function DR1(A, B)
  $n \leftarrow dim(A)$
  $C \leftarrow$ new n $\times$ n matrix
  if n $==$ 1 **then**
    $C \leftarrow (a_{11} \cdot b_{11})$
    return C
  else
    partition A and B into four (n/2) $\times$ (n/2) quadrants
  end if
  $C_{11} \leftarrow \text{DR1}(A_{11}, B_{11}) + \text{DR1}(A_{12}, B_{21})$
  $C_{12} \leftarrow \text{DR1}(A_{11}, B_{12}) + \text{DR1}(A_{12}, B_{22})$
  $C_{21} \leftarrow \text{DR1}(A_{21}, B_{11}) + \text{DR1}(A_{22}, B_{21})$
  $C_{22} \leftarrow \text{DR1}(A_{21}, B_{12}) + \text{DR1}(A_{22}, B_{22})$
  $C \leftarrow \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$
  return C
end function

El algoritmo DR1, base su lógica en los algoritmos de divide y conquista pero de una manera menos efectiva, utiliza 8 multiplicaciones y debido a la recursividad obtenemos que: $T(n) = 8T(\frac{n}{2}) + n^2$

Y esto por teorema maestro simplificado obtenemos que el comportamiento asintótico es $\Theta(n^{log_b(a)})$ y eso nos lleva a que $\Theta(n^{log_2(8)})$ lo que resulta que su comportamiento corresponde a $\Theta(n^3)$ en otras palabras no hay mucha mejorar en comparación con el tradicional bajo este análisis.

$$T(n) = 8T(\frac{n}{2}) + n^2 = \Theta(n^{\log_2 8}) = \Theta(n^3) \tag{2}$$

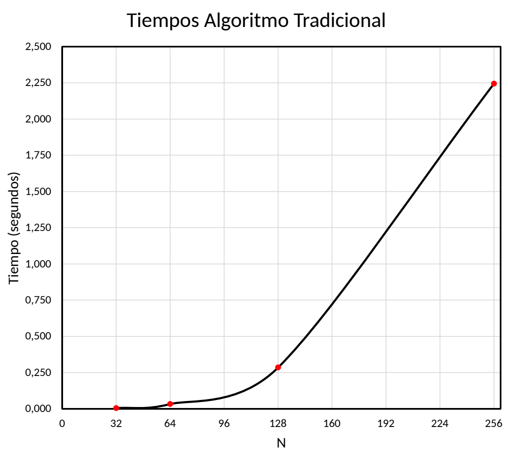

Figura 2: Gráfico DR1 $T(n) = \Theta(n^3)$

1.3. Algoritmo DR2 (Strassen)

Algorithm 3 Algoritmo DR2

function STRASSEN(A, B)
  $n \leftarrow dim(A)$
  $C \leftarrow$ new n $\times$ n matrix
  if n == 1 **then**
    $C \leftarrow (a_{11} \cdot b_{11})$
    return C
  else
    partition A and B into four (n/2) $\times$ (n/2) quadrants
  end if
  $M_1 \leftarrow Strassen(A_{11} + A_{22}), B_{11} + B_{22})$
  $M_2 \leftarrow Strassen(A_{21} + A_{22}), B_{11})$
  $M_3 \leftarrow Strassen(A_{11}, B_{12} - B_{22})$
  $M_4 \leftarrow Strassen(A_{22}, B_{21} - B_{11})$
  $M_5 \leftarrow Strassen(A_{11} + A_{21}, B_{22})$
  $M_6 \leftarrow Strassen(A_{21} - A_{11}, B_{11} + B_{12})$
  $M_7 \leftarrow Strassen(A_{12} - A_{22}, B_{21} + B_{22})$
  $C_{11} \leftarrow M_1 + M_4 - M_5 + M_7$
  $C_{12} \leftarrow M_3 + M_5$
  $C_{21} \leftarrow M_2 + M_4$
  $C_{22} \leftarrow M_1 - M_2 + M_3 + M_6$
  $C \leftarrow \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$
  return C
end function

El algoritmo DR2 (Algoritmo de Strassen) funciona muy similar al DR1 pero este reduce la cantidad de multiplicaciones a 7 realizando simplificaciones algebraicas, por lo que, reduce la expresión a: $T(n) = 7T(\frac{n}{2}) + n^2$

Y esto por teorema maestro simplificado obtenemos que el comportamiento asintótico es $\Theta(n^{log_b(a)})$ y eso nos lleva a que $\Theta(n^{log_2(7)})$, lo que resulta en el valor de Θ sea de $\Theta(n^{2,807})$ por lo que este algoritmo debería ser el mas eficiente al momento de resolver multiplicación de matrices de tamaño NxN con n siendo potencia de 2.

$$T(n) = 7T(\frac{n}{2}) + n^2 =^{T.M.S.O3} \Theta(n^{\log_2 7}) = \Theta(n^{2,80735}) \tag{3}$$

2. Comparación de algoritmos

Tiempos			
N	AT	DR1	DR2
32	0.00088	0.0005	0,01441
64	0.00651	0.002	0.10292
128	0.02328	0.0135	0.66786
256	0.06519	0.105	4.82923
512	0.74843	0,833	35.22188
1024	5.03849	6,75	238.44549
2048	65.86131	53,4915	1522.54312
4096	625.83509	426,9955	12415.32413

Cuadro 1: Tiempos de todos los algoritmos

Tiempos		
N	DR2 b2	DR2 b16
32	0.01958	0.00128
64	0.08476	0.00912
128	0.18530	0.03821
256	1.13591	0.11093
512	8.19686	0.58102
1024	56.20033	3.83984
2048	316.42530	26.62368
4096	2635.54724	188.79235

Cuadro 2: Tiempos de todos los algoritmos

Figura 3: Comparación de tiempos 256 max

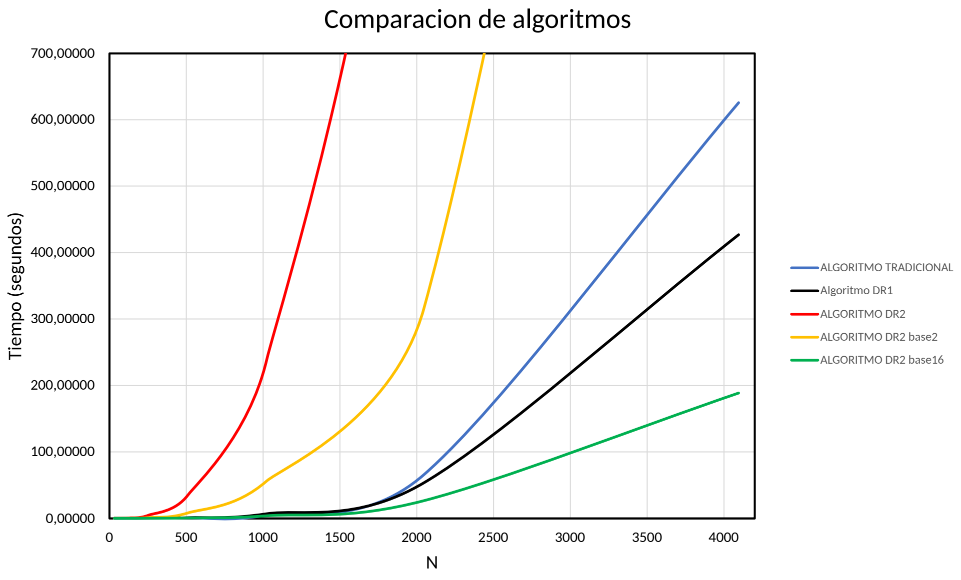

Figura 4: Comparación de tiempos 4096 max

Se puede observar que los algoritmos DR2 y DR2 en base 2, tiene un comportamiento asintótico exagerado, esto por temas de tiempo, pues tuvimos miedo de correr los códigos para los casos de N = 4096 y pensamos unos valores aproximados.

3. Conclusiones

El algoritmo tradicional de multiplicación de matrices tiene una complejidad temporal de $\Theta(n^3)$ lo que significa que su tiempo de ejecución aumenta significativamente con el tamaño de las matrices. Aunque es fácil de implementar, puede volverse ineficiente para matrices grandes debido a esta alta complejidad.

En contraste, el algoritmo de Strassen reduce la complejidad temporal en aproximadamente $\Theta(n^{2,81})$, una mejora considerable en comparación con el método tradicional. Esto debido a que Strassen logra va dividiendo las matrices de entrada en submatrices más pequeñas utilizando operaciones algebraicas y operaciones recursivas. Los algoritmos DR1 y Strassen pueden volverse ineficientes si no se maneja cuidadosamente la memoria, dado que los tiempos de ejecución de los algoritmos pueden verse comprometidos si no se tiene precaución al manipular la memoria.

En resumen, el algoritmo tradicional es preferible para matrices pequeñas o de tamaño moderado debido a su menor costo constante. Para matrices grandes, el algoritmo de Strassen puede ser más eficiente, especialmente si se optimiza utilizando el método tradicional como caso base y se maneja cuidadosamente la memoria para evitar ineficiencias. La elección entre estos algoritmos depende del tamaño de las matrices y de la habilidad para optimizar el manejo de memoria en una implementación específica.

4. Información de Hardware y Software

4.1. Notebook que proceso el Algoritmo Tradicional, DR2, DR2 base 2 y DR2 base 16

1. **SO:** Windows 11 Pro 22H2
2. **Compilador:** gcc (GCC) 11.4.0 Cygwin
3. **Procesador:** Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz
4. **Ram:** 24,0 GB (23,8 GB utilizable)

4.2. Notebook que proceso el Algoritmo DR1

1. **SO:** Windows 11 Pro 22H2
2. **Compilador:** gcc (Built by MinGW-W64 project) 8.1.0
3. **Procesador:** AMD Ryzen 5 3600 6-Core Processor @3.59 GHz
4. **Ram:** 16,0 GB