Licenciatura en ciencia de la computación

Algoritmo de Strassen

Complejidad

Profesor:
Nicolas Thériault

Autor:
Sergio Salinas
Danilo Abellá

Contents

1 Introducción

Este informe trata sobre el algoritmos de multiplicación de Strassen, se verá su demostración, su complejidad y la de la multiplicación normal y se verá su eficiencia.

Los programas fueron escritos en lenguaje C usando el compitalador GCC versión 5.4.

2 Análisis teorico

2.1 Demostración Strassen

Dada la multiplicación de dos matrices A y B

$$AB = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix} = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

Se calculan los S y los T de la técnica de Strassen.

$$\begin{aligned} S_1 &= A_{21} + A_{22} \\ S_2 &= A_{21} + A_{22} - A_{11} \\ S_3 &= A_{11} - A_{21} \\ S_4 &= A_{12} - A_{21} - A_{22} + A_{11} \end{aligned}$$

$$\begin{aligned} T_1 &= B_12 - B_{11} \\ T_2 &= B_{22} - B_{12} + B_{11} \\ T_3 &= B_{22} - B_{12} \\ T_4 &= B_{22} - B_{12} + B_{11} - B_{21} \end{aligned}$$

Se calculan los P.

$$\begin{aligned} P_1 &= A_{11}B_{21} \\ P_2 &= A_{12}B_{21} \\ P_3 &= A_{12}B_{22} - A_{21}B_{22} - A_{22}B_{22} + A_{11}B_{22} \\ P_4 &= A_{22}B_{22} - A_{22}B_{12} + A_{22}B_{11} - A_{22}B_{21} \\ P_5 &= A_{21}B_{11} - A_{21}B_{11} + A_{22}B_{12} - A_{22}B_{11} \\ P_6 &= A_{21}B_{22} - A_{21}B_{12} + A_{21}B_{11} + A_{22}B_{22} - A_{22}B_{12} + A_{22}B_{11} - A_{11}B_{22} + A_{11}B_{22} \\ &\quad +A_{11}B_{12} - A_{11}B_{11} \\ P_7 &= A_{11}B_{22} - A_{11}B_{12} - A_{21}B_{22} + A_{21}B_{12} \end{aligned}$$

Se calculan los U

$$\begin{aligned}
U_1 &= A_{11}B_{11} + A_{12}B_{21} \\
U_2 &= A_{21}B_{22} - A_{21}B_{12} + A_{21}B_{11} + A_{22}B_{22} - A_{22}B_{12} + A_{22}B_{11} - A_{11}B_{22} + A_{11}B_{12} \\
U_3 &= A_{21}B_{11} + A_{22}B_{22} - A_{22}B_{12} + A_{22}B_{11} \\
U_4 &= A_{21}B_{22} + A_{22}B_{22} - A_{11}B_{22} + A_{11}B_{12} \\
U_5 &= A_{11}B_{12} + A_{12}B_{22} \\
U_6 &= A_{21}B_{11} + A_{22}B_{21} \\
U_7 &= A_{22}B_{22} + A_{21}B_{12}
\end{aligned}$$

De está forma se obtiene que

$$A \cdot B = \begin{pmatrix} U_1 & U_5 \\ U_6 & U_7 \end{pmatrix} = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix} = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

Con lo que se demuestra que el algoritmo de strassen es equivalente a la multiplicación de matrices.

2.2 Análisis complejidad

2.2.1 Strassen

Strassen hace en total 7 multiplicaciones más 14 sumas (y restas, aunque se tratan como su fueran la misma operación) y además divide el orden de la matriz en dos por cada recurrencia por lo su relación de recurrencia es

$$T(n) = 7 \cdot T\left(\frac{n}{2}\right) + 14n^2$$

Dado el teorema maestro se obtiene que $a = 7$, $b = 2$ y $d = 2$, obteniendo que $7 > 2^2$ por que la complejidad del algoritmos es

$$T(n) \in O(n^{\log_2 7}) = O(n^{2.80735})$$

2.2.2 Multiplicación clásica

La multiplicación clásica de matrices cuadradas de orden 2 debe hacer un total de 8 multiplicaciones escalares y 4 sumas de coste n^2, por lo que si quiere trabajar con bloques de matrices cuadradas de orden $n = 2^k$ el total de multiplicaciones tendría como relación de recurrencia:

$$T(2^k) = 8 \cdot T(2^{k-1}) + 4(2^k)^2$$

Usando el teorema maestro se obtiene que la complejidad de la multiplicaciones es

$$T(n) \in O(n^{\log_2 8}) = O(n^3)$$

Lo que es consistente con el costo $n^3M + (n^3 - n^2)A$.

2.3 Valor de $\mathbf{n_0}$

Después de probar con varias potencias de 2 se pudo concluir que el punto en donde Strassen lleva ventaja por sobre la multiplicación clásica es en $n_0 = 16$.

Esto es debido a que en los valores menores y mayores a 16 se puede apreciar dos rectas continuas con crecimiento distinto, mientras que cuando n_0 vale 16 se pueden ver los resultados como una sola recta.

3 Algoritmos Desarrollados

Algorithm 1: Strassen. Multiplica dos matrix A y B

```
Data: Dos matrices A y B de orden n, n_0
Result: Matriz C
if n ≤ n_0 then
    A·B;
else
    Dividir A en A_11, A_12, A_21 y A_22;
    Dividir B en B_11, B_12, B_21 y B_22;
    S_1 ← A_21 + A_22;  T_1 ← B_12 − B_11;
    S_2 ← S_1 − A_11;  T_2 ← B_22 − T_1;
    S_3 ← A_11 − A_21; T_3 ← B_22 − B_12;
    S_4 ← A_12 − S_2;  T_4 ← T_2 − B_21;
    if n/4 ≤ n_0 then
        P_1 ← A_11 · B_11; P_2 ← A_12 · B_21;
        P_3 ← S_4 · B_22; P_4 ← A_22 · T_4;
        P_5 ← S_1 · T_1; P_6 ← S_2 · T_2;
        P_7 ← S_3 · T_3;
    else
        P_1 ← strassen(A_11, B_11, n, n_0);
        P_2 ← strassen(A_12, B_21, n, n_0);
        P_3 ← strassen(S_4, B_22, n, n_0);
        P_4 ← strassen(A_22, T_4, n, n_0);
        P_5 ← strassen(S_1, T_1, n, n_0);
        P_6 ← strassen(S_2, T_2, n, n_0);
        P_7 ← strassen(S_3, T_3, n, n_0) ;
    end
    U_1 ← P_1 + P_2;
    U_2 ← P_1 + P_6;
    U_3 ← U_2 + P_7;
    U_4 ← U_2 + P_5;
    U_5 ← U_4 + P_3;
    U_6 ← U_3 − P_4;
    U_7 ← U_3 + P_5;
end
```

return $C \leftarrow \begin{pmatrix} U_1 & U_5 \\ U_6 & U_7 \end{pmatrix}$

Algorithm 2: Multiply Matrix. Multiplica dos matrices A y B

Data: Dos matrices A y B de largo n

Result: La matrix C con la multiplicación de A y B

```
for i ← 0 … n-1 do
    for j ← 0 … n-1 do
        C_ij ← 0;
        for k ← 0 … n-1 do
            C_i j ← C_ij + A_ik · B_kj
        end
    end
end
```

Algorithm 3: Sum Matrix. Suma dos matrices A y B

Data: Dos matrices A y B de largo n

Result: La matrix C con la suma de A y B

```
for i ← 0 … n-1 do
    for j ← 0 … n-1 do
        C_i j ← A_ij + B_ij
    end
end
```

Algorithm 4: Sub Matrix. Resta dos matrices A y B

Data: Dos matrices A y B de largo n

Result: La matrix C con la resta de A y B

```
for i ← 0 … n-1 do
    for j ← 0 … n-1 do
        C_i j ← A_ij − B_ij
    end
end
```

4 Información de Hardware y Software

4.1 Notebook - Danilo Abellá

4.1.1 Software

- SO: Xubuntu 16.04.1 LTS
- GMP Library
- Mousepad 0.4.0

4.1.2 Hardware

- AMD Turion(tm) X2 Dual-Core Mobile RM-72 2.10GHz
- Memoria (RAM): 4,00 GB(3,75 GB utilizable)
- Adaptador de pantalla: ATI Raedon HD 3200 Graphics

4.2 Notebook - Sergio Salinas

4.2.1 Software

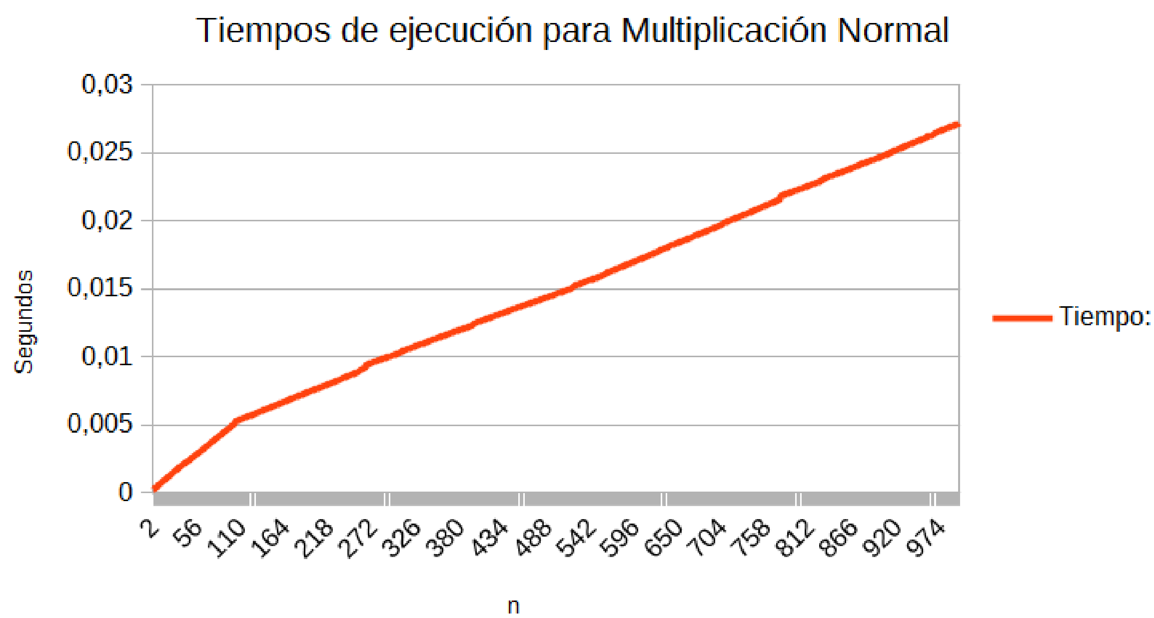
- SO: ubuntu Gnome 16.04 LTS
- Compilador: gcc version 5.4.0 20160609
- Editor de text: Atom

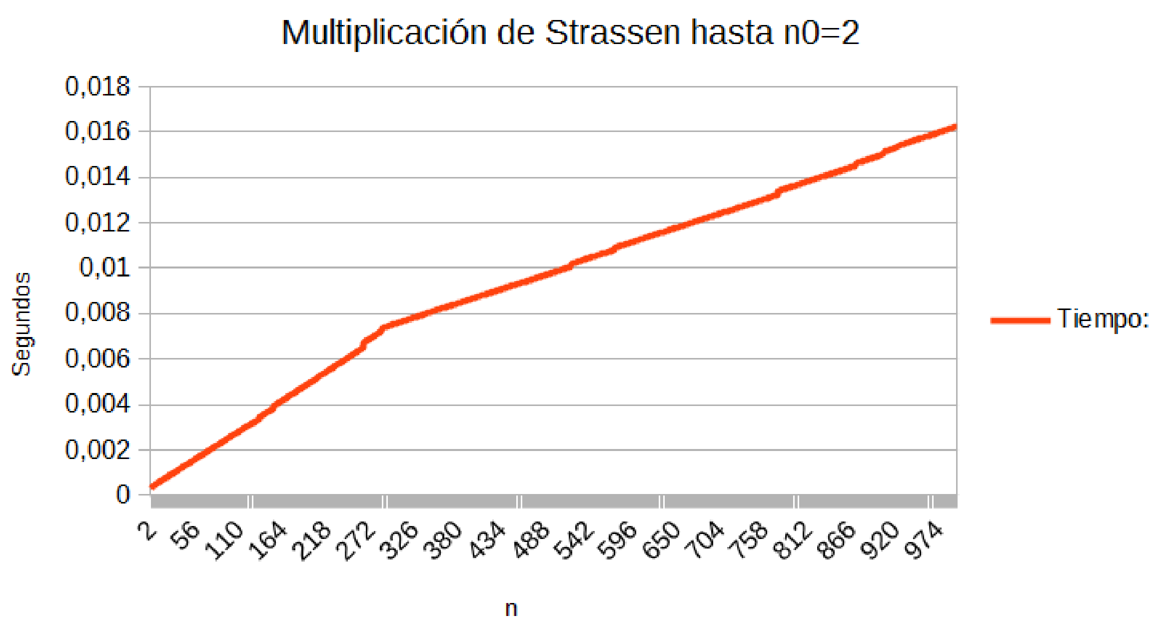
4.2.2 Hardware

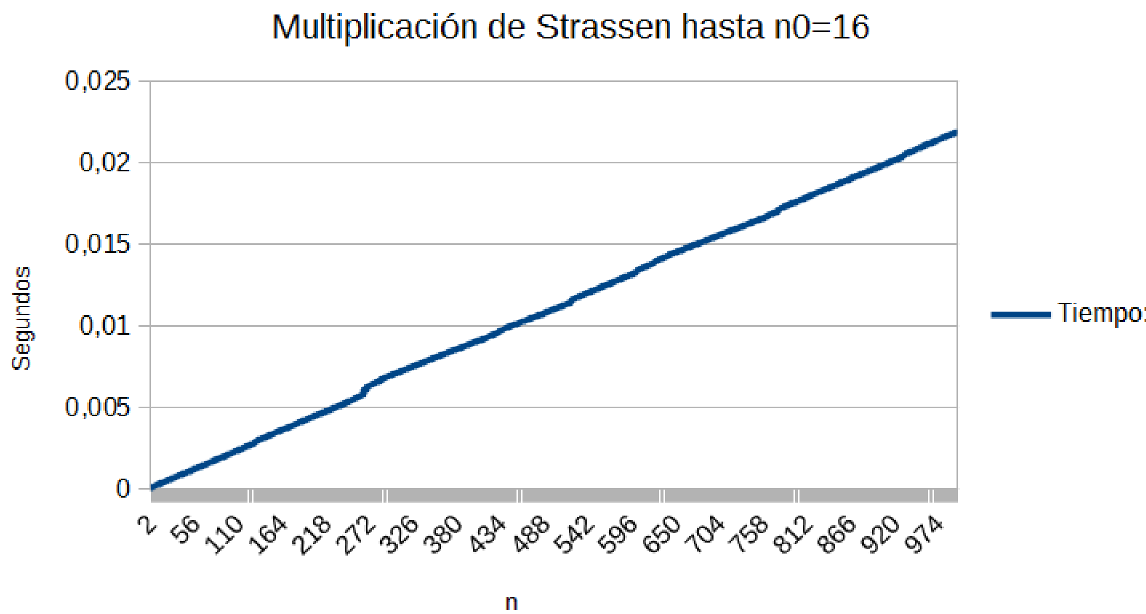
- Procesador: Intel Core i7-6500U CPU 2.50GHz x 4
- Video: Intel HD Graphics 520 (Skylake GT2)

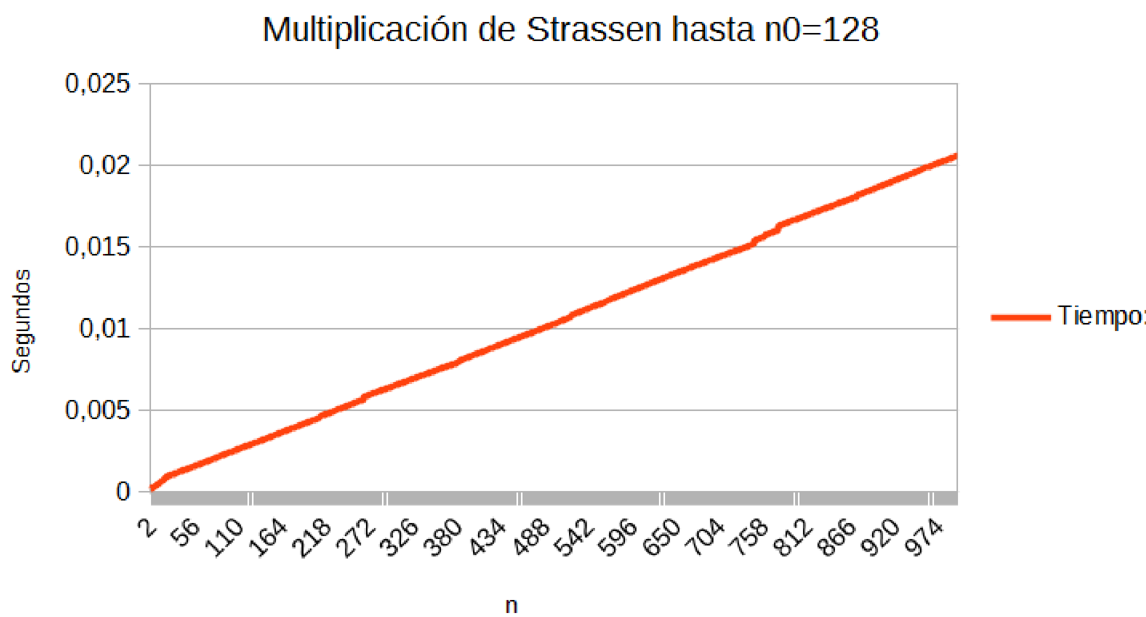
5 Curvas de desempeño de resultados

5.1 Multiplicación clasica vs distintos valores de n_0

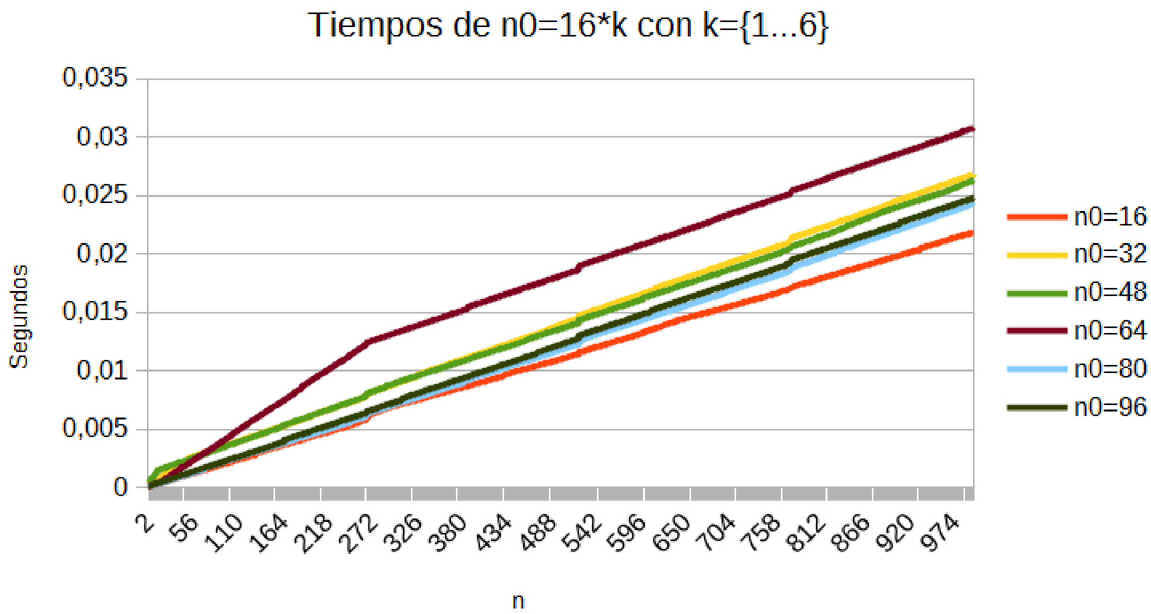


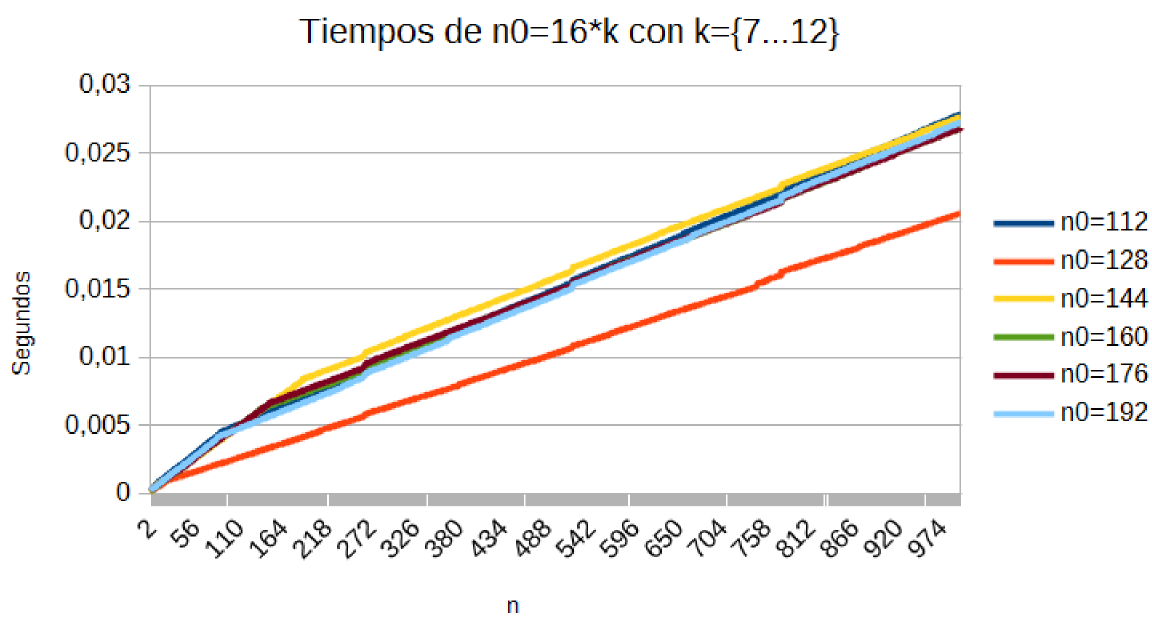


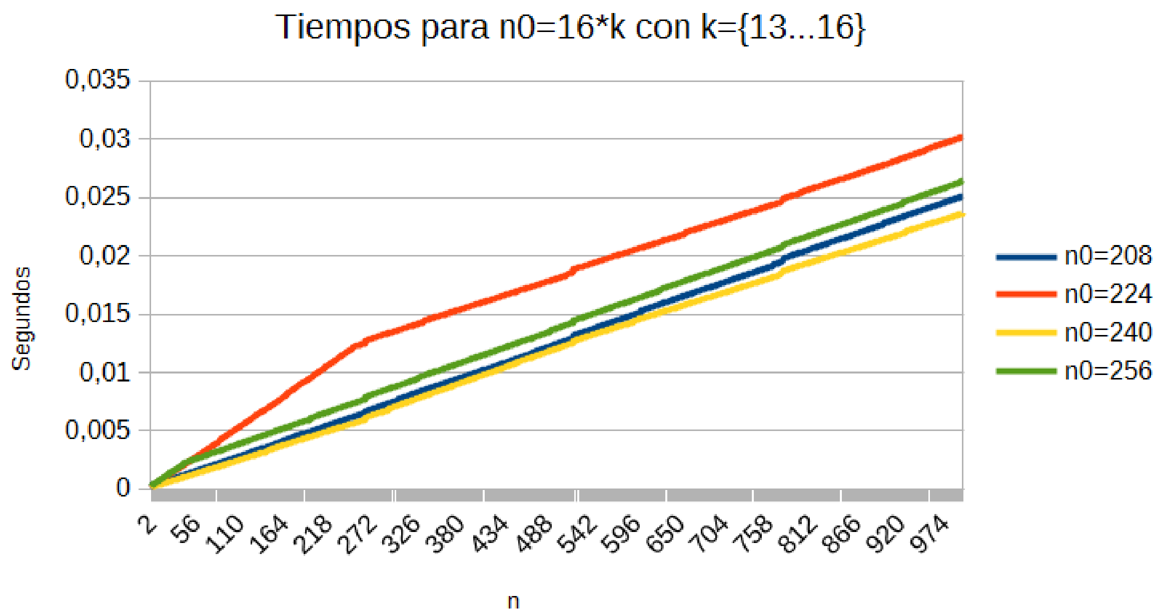

Multiplicación de Strassen hasta n0=16
0,025
0,02
0,015
0,01
0,005
0
Segundos
2
56
110
164
218
272
326
380
434
488
542
596
650
704
758
812
866
920
974
n
Tiempo:


Multiplicación de Strassen hasta n0=128
0,025
0,02
0,015
0,01
0,005
0
Segundos
2
56
110
164
218
272
326
380
434
488
542
596
650
704
758
812
866
920
974
n
Tiempo:

5.2 n_0 = 16*k, k={1,2 ... 16}






Tiempos para n0=16*k con k={13...16}
0,035
0,03
0,025
0,02
0,015
0,01
0,005
0
Segundos
2
56
110
164
218
272
326
380
434
488
542
596
650
704
758
812
866
920
974
n
n0=208
n0=224
n0=240
n0=256

6 Conclusiones

Se puede concluir que la eficiencia de la multiplicación de Strassen ($O(n^{2.801})$) por sobre la clásica ($O(n^3)$) es indiscutible, llegando está última incluso a tomar el doble de tiempo que Strassen cuando aunque la última es más eficiente cuando se trata de matrices de orden estrictamente menores a 16.