Informe práctica 2

Introducción

Strassen es un algoritmo recursivo que reduce a siete los subproblmeas para multiplicar dos matrices. Lo que hace que tenga una complejidad mucho menor que el algoritmo tradicional.
Si tenemos:

$$\mathbf{C} = \mathbf{A}\mathbf{B} \qquad \mathbf{A}, \mathbf{B}, \mathbf{C} \in R^{2^n \times 2^n}$$

Los pasos son los siguientes*:*
Descomponemos nuestras matrices en 4 subM:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} \mathbf{B}_{1,1} & \mathbf{B}_{1,2} \\ \mathbf{B}_{2,1} & \mathbf{B}_{2,2} \end{bmatrix}, \mathbf{C} = \begin{bmatrix} \mathbf{C}_{1,1} & \mathbf{C}_{1,2} \\ \mathbf{C}_{2,1} & \mathbf{C}_{2,2} \end{bmatrix}$$

Definimos que sus productos son:

$$\mathbf{C}_{1,1} = \mathbf{A}_{1,1}\mathbf{B}_{1,1} + \mathbf{A}_{1,2}\mathbf{B}_{2,1}$$
$$\mathbf{C}_{1,2} = \mathbf{A}_{1,1}\mathbf{B}_{1,2} + \mathbf{A}_{1,2}\mathbf{B}_{2,2}$$
$$\mathbf{C}_{2,1} = \mathbf{A}_{2,1}\mathbf{B}_{1,1} + \mathbf{A}_{2,2}\mathbf{B}_{2,1}$$
$$\mathbf{C}_{2,2} = \mathbf{A}_{2,1}\mathbf{B}_{1,2} + \mathbf{A}_{2,2}\mathbf{B}_{2,2}$$

Con esto tenemos 8 subproblemas por lo que no mejora el algoritmo tradicional de $\mathbf{O}^3$
La parte que añadió Strassen es reducir esto 8 productos a 7 de la siguiente forma:

$$\mathbf{M}_1 := (\mathbf{A}_{1,1} + \mathbf{A}_{2,2})(\mathbf{B}_{1,1} + \mathbf{B}_{2,2})$$
$$\mathbf{M}_2 := (\mathbf{A}_{2,1} + \mathbf{A}_{2,2})\mathbf{B}_{1,1}$$
$$\mathbf{M}_3 := \mathbf{A}_{1,1}(\mathbf{B}_{1,2} - \mathbf{B}_{2,2})$$
$$\mathbf{M}_4 := \mathbf{A}_{2,2}(\mathbf{B}_{2,1} - \mathbf{B}_{1,1})$$
$$\mathbf{M}_5 := (\mathbf{A}_{1,1} + \mathbf{A}_{1,2})\mathbf{B}_{2,2}$$
$$\mathbf{M}_6 := (\mathbf{A}_{2,1} - \mathbf{A}_{1,1})(\mathbf{B}_{1,1} + \mathbf{B}_{1,2})$$
$$\mathbf{M}_7 := (\mathbf{A}_{1,2} - \mathbf{A}_{2,2})(\mathbf{B}_{2,1} + \mathbf{B}_{2,2})$$

PseudoCodigo

```
int[][] StrassenMultiply(int[][] A, int[][] B) {
       int n = A.length;
       int[][] res = new int[n][n];
       // if the input matrix is 1x1
       if (n == 1) {
           res[0][0] = A[0][0] * B[0][0];
       } else {
           // first matrix
           int[][] a,b,c,d = new int[n / 2][n / 2];;
           // second matrix
           int[][] e, f, g, h = new int[n / 2][n / 2];
           // dividing matrix A into 4 parts
           divArr(A, a, 0, 0);
           divArr(A, b, 0, n / 2);
           divArr(A, c, n / 2, 0);
           divArr(A, d, n / 2, n / 2);
           // dividing matrix B into 4 parts
           divArr(B, e, 0, 0);
           divArr(B, f, 0, n / 2);
           divArr(B, g, n / 2, 0);
           divArr(B, h, n / 2, n / 2);

           int[][] p1 = StrassenMultiply(addM(a, d), addM(e, h));
           int[][] p2 = StrassenMultiply(addM(c,d),e);
           int[][] p3 = StrassenMultiply(a, subM(f, h));
           int[][] p4 = StrassenMultiply(d, subM(g, e));
           int[][] p5 = StrassenMultiply(addM(a,b), h);
           int[][] p6 = StrassenMultiply(subM(c, a), addM(e, f));
           int[][] p7 = StrassenMultiply(subM(b, d), addM(g, h));

           int[][] C11 = addM(subM(addM(p1, p4), p5), p7);
           int[][] C12 = addM(p3, p5);
           int[][] C21 = addM(p2, p4);
           int[][] C22 = addM(subM(addM(p1, p3), p2), p6);

           // adding all subarray back into one
           cpySubArr(C11, res, 0, 0);
           cpySubArr(C12, res, 0, n / 2);
           cpySubArr(C21, res, n / 2, 0);
           cpySubArr(C22, res, n / 2, n / 2);
       }
       return res;
   }
```

Analizando el código

La fórmula de recursión que obtenemos es la siguiente:

T(n) = {
Θ(1) if n = 1,
$7T(n/2) + \Theta(n^2)$ if n > 1
}

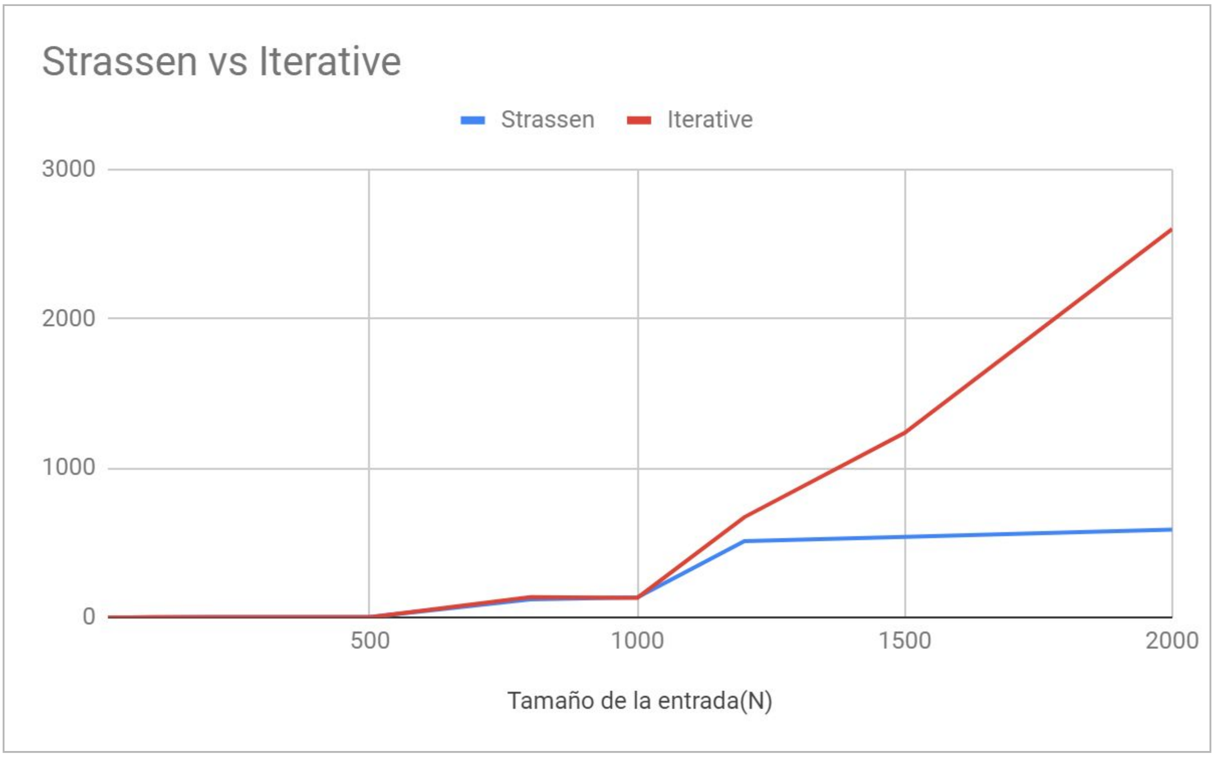
Para nuestro caso el número de subproblemas a = 7 En nuestro caso el tamaño de los subproblemas b = n/2 El exponente de combinación d = 2 El tiempo en dividir el problema D(n) es 1 El tiempo en combinar el problema C(n) es n^2
Utilizando el [Teoréma méstro] Estamos en el caso 1: El algoritmo de Strassen pertenece a: **$\Theta(n \log^2(7))$.** Asi que es **$\Theta(n^{2.81})$.**

Analizando los resultados experimentales

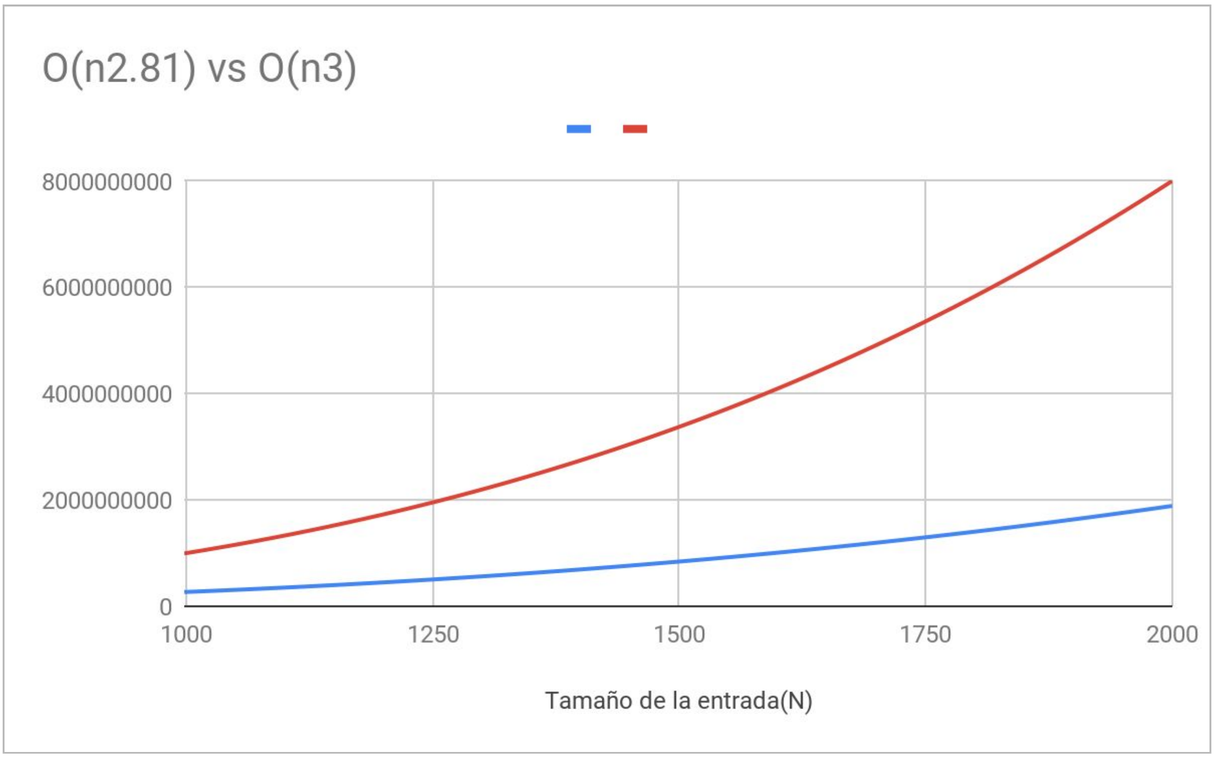
Tabla datos de los experimentos y la función calculada

Tamaño de la entrada(N)	Strassen	Iterative
10	0.002333333333	0.006333333333
20	0.01533333333	0.06333333333
50	0.04633333333	0.005666666667
100	0.272	0.6396666667
200	1.941333333	0.7136666667
300	1.894	0.7533333333
500	1.963	0.8473333333
800	120.095	135.4166667
1000	134.1946667	130.9193333
1200	510.147	670.8986667
1500	538.445	1236.147667
2000	587.089	2600.333333

Gráfica del algoritmo Strassen y del algoritmo Iterativo experimental



Gráfica del algoritmo Strassen y del algoritmo Iterativo esperada



Conclusiones

Las conclusiones que sacamos del análisis es que tiene algunos errores ya que los tiempo de ejecución variaba mucho de unas pruebas a otras por lo tanto debe haber algo interfiriendo en el proceso. Aún así con algunos datos de prueba vemos que efectivamente la relación se cumple bastante acertadamente. Ya que en al segunda gráfica podemos apreciar el parecido a la primera. Strassen es un método que aunque a priori no parece reducir mucho el exponente nos damos cuenta que en tamaños de **N** muy grandes la diferencia es abismal.

Referencias

Algoritmo de Strassen, (s. f). En Wikipedia. Recuperado el 5 de Marzo de 2018 de https://es.wikipedia.org/wiki/Algoritmo_de_Strassen

Strassen algorithm, (s. f). En Wikipedia. Recuperado el 5 de Marzo de 2018 de https://en.wikipedia.org/wiki/Strassen_algorithm

Grupo Guíame. (31 de diciembre de 2017). Multiplicación de Karatsuba y Matrices de Strassen [Video de Youtube]. Recuperado de https://www.youtube.com/watch?v=6cBSAkzzQzU