Análisis de Algoritmos, Sem: 2018-1, 3CV2 Práctica 5, 12 de Octubre del 2017

Práctica 5: Algoritmo de Strassen

Luis Daniel Martinez Berumen

Escuela Superior de Computo
Instituto Politecnico Nacional
dany.berumen@gmail.com

Abstract

En esta practica vamos a demostrar la complejidad del algoritmo de Strassen, un algortimo de multiplicacion de matrices que se basa en la descomposicion en simples sumas, llevando acabo el analisis de manera grafica, nos ayudaremos sabiendo que el resultado a obtener es: $O(n^{2,8})$, como lo vimos en clase.
Se realizara de igual manera en analisis de el procedimiento normal para multiplicacion de matrices, cabe señalar que las matrices que utilizaremos para estos procesos son cuadradas con orden 2^n

Palabras Clave

- Algoritmo
- Fucion
- Recursividad
- Orden
- Matriz

1. Introducción

Divide y Vencerás es una frase quye hemos escuchado todos, al menos, una vez en nuestra vida, para nosotros es técnica de diseño de algoritmos, siendo de gran utilidad para nuestra carrera, ya que, los problemas a los que nos enfrentamos día con día son mas faciles de resolver si aplciamos una tecnica de este tipo. De hecho, suele ser considerada una filosofía general para resolver problemas, no solo del termino informatico, sino que también se utiliza en muchos otros ámbitos

2. Conceptos Básicos

Para la correcta comprension de este trabajo, es necesario definir algunos terminos tales como θ, O y Ω.
θ(n):
Sea g(n) una función. Se define θ (g(n)) como:

$\theta(\text{g(n)}) = \{f(n) \quad | \quad \exists c1, c2 > 0 \quad \& \quad n_0 > 0 \quad | \quad \forall n >= n_0 \quad 0 <= c1g(n) <= f(n) <= c2g(n)\}$

O(n):
Sea g(n) una función, O(n) (el pero de los casos) se define como:

$$\text{O(n)}=\{f(n) \quad | \quad \exists c > 0 \quad \& \quad n_0 > 0 \quad | \quad f(n) <= Cg(n) \quad \forall n >= n_0\}$$

Ω(n):
Sea g(n) una función. Se define Ω (g(n)) (el mejor de los casos) como:

$$\Omega(\text{g(n)}) =\{f(n) \quad | \quad \exists c > 0 \quad \& \quad n_0 > 0 \quad | \quad 0 <= cg(n) <= f(n) \quad \forall n >= n_0\}$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix}$$

Figura 1: Matriz Cuadrada

Para la correcta comprension de esta practica, debemos definir el concepto de una matriz cuadrada, siendo esta una matriz que tiene tantas filas como columnas, como podemos apreciar en la figura 1, la matriz se compone de mxn elementos, en el caso de una matriz cuadrada, se restringe esta propiedad al tener que ser m=n, por lo tanto la matriz quedara como de nxn elementos.

Tomando como ejemplo una amtriz caudrada de orden 2, ejemplificaremos el uso del algoritmo de Strassen:

$$m_1 = (a_{21} + a_{22} - a_{11})(b_{22} - b_{12} + b_{11})$$
$$m_2 = a_{11} b_{11}$$
$$m_3 = a_{12} b_{21}$$
$$m_4 = (a_{11} - a_{21})(b_{22} - b_{12})$$
$$m_5 = (a_{21} + a_{22})(b_{12} - b_{11})$$
$$m_6 = (a_{12} - a_{21} + a_{11} - a_{22}) b_{22}$$
$$m_7 = a_{22}(b_{11} + b_{22} - b_{12} - b_{21})$$

Entonces el producto AB queda:

$$\begin{vmatrix} m_2 + m_3 & m_1 + m_2 + m_5 + m_6 \\ m_1 + m_2 + m_4 - m_7 & m_1 + m_2 + m_4 + m_5 \end{vmatrix}$$

Figura 2: Operaciones Resultantes

Si reemplazamos cada elemento de A y B por una matriz de n x n, las fórmulas anteriores nos dan una forma de multiplicar dos 2n X 2n matrices. A partir de esto tenemos un método recursivo para calcular el producto de matrices con n Algoritmo de multiplicación de Strassen potencia de 2. Este método se puede generalizar también a matrices cuyas dimensiones no sean de la forma 2n

3. Experimentación y Resultados

3.1. Implemente el algortimo Stressen para producto de matrices.

i)Mediante graficas, muestre que el algoritmo de **Strassen** tiene complejidad $O(n^{2,8})$.

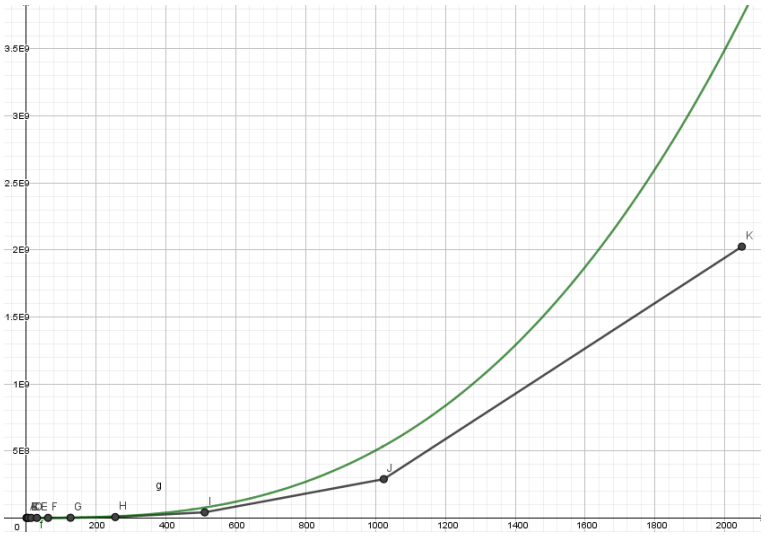

Figura 3: Resultados Strassen vs $n^{2,8}$

La grafica en color negro es la función f(n) que se propone, con base en los resultados obtenidos tras la implementacion del algoritmo, la grafica en color verde es T(n)=$\theta(n^{2,8})$, como observamos en la figura 3 la función representante del algoritmo de Strassen es muy similar a los resultados esperados, cabe aclarar que la variacion puede darse debido a que en la implementacion, se utilizaron unicamente matrices con orden 2^2, por eso los saltos tan grandes entre el punto "jz el punto "k", por lo que se concluye que cumple con los resultados esperados, por lo tanto, el tiempo computacional de la función Stressen es de tipo T(n)=$\theta(n^{2,8})$.

ii)Para valores muy grandes de matrices compare el algoritmo de **Strassen** con el producto usual de matrices (compara la complejidad).

Para este inciso, crei conveniente primero realizar la comparacion del algoritmo de multiplicacion tradicional y la funcion obtenida en el salon de clases, y despues comparar los resultados de Strassen con la multiplicacion normal.

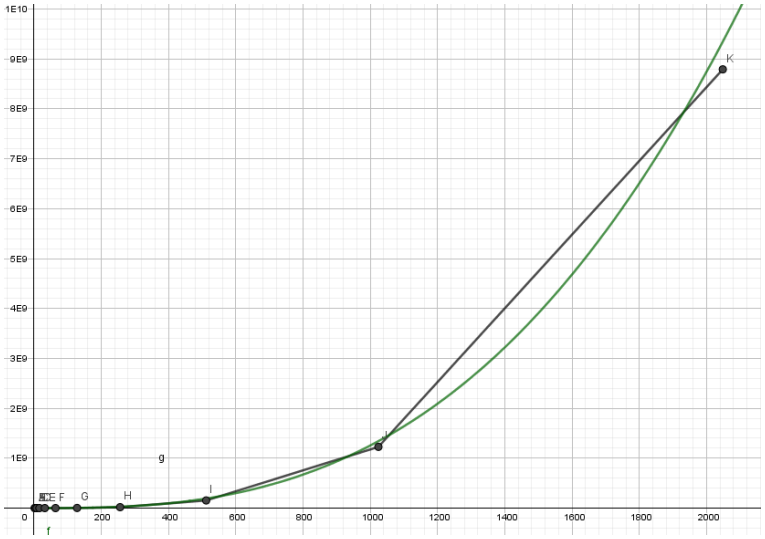

Figura 4: Resultados Multiplicacion Tradicional vs $n^{2,8}$

Como se puede observar en la figura 4, la grafica es de nuevo muy parecida, con los resultados obtenidos, se aprecia un cambio algo drastico en el salto del punto "J.ªl punto "K", esto se da, de igual manera que el caso anterior, a que las matrices utilizadas para este ejercicio son de orden 2^2, sin embargo, por el comportamiento que se puede apreciar en el trayecto de la recta verde que representa a nuestra funcion propuesta, se nota que esta crece de mayor tamaño que la grafica de color negro, que representa los resultados obtenidos, de igual manera, la grafica propuesta se esta multiplicando por un excponente para aprecair mejor su comportamiento.

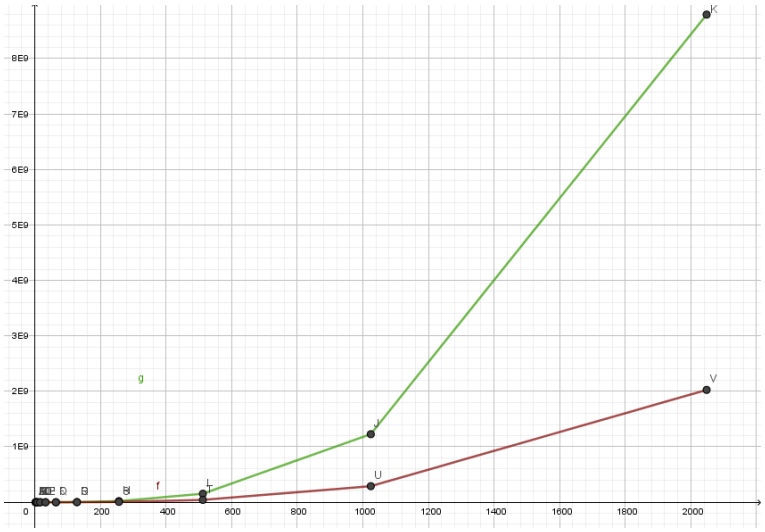

Figura 5: Resultados Multiplicacion Tradicional vs Strassen.

En la figura 5, podemos ver la comparacion de los resultados arrojados por strassen y los resultados arrojados por la multiplicacion tradicional. De color verde tenemos la funcion de Strassen, y de rojo la m.Tradicional.
Podemos observar que la grafica representada por Strassen crece de manera mas rapida que la otra, sin embargo, esto es solamenter aplicable para matrices de tamaño muy grande, no se si este algoritmo sea el mas eficiente para esta multiplicacion, pero al momento de realizar operaciones tan grandes, sin duda alguna es de gran ayuda, al arrojar mejores resultados, podemos concluir que el algoritmo de Strassen es mas eficiente que el algoritmo tradicional, al momento de realizar multiplicaciones de matrices grandes.

4. Conclusión

Esta practica fue muy entretenida, el analisis de algoritmos para determinar cual es mejor, es un tema que de verdad es muy fascinante, sin embargo, la realizacion de este algoritmo fue muy complicada, no pude realizarla en Python como lo venia haciendo, ya que Python no maneja matrices como tal, y el manejo de arreglo de arreglos es algo que aun me tiene con complicaciones, por lo que la realice en Java.
Regresando a la practica, el analisis de este algoritmo, mediante el metodo grafico fue sencillo, una vez completado, no conocia este algoritmo, y no se si sea el mejor, pero por ahora para la multiplicacion de matrices de gran tamaño, creo que es una opcion muy interesante.