Análisis de Algoritmos, Sem: 2018-2, 3CV1, Práctica 6, 10 de Octubre de 2018

Práctica 6: Algoritmo de Strassen

Reyes Valenzuela Alejandro, Gutiérrez Povedano Pablo.

Escuela Superior de Cómputo
Instituto Politécnico Nacional, México
*areyesv*11@*gmail.com*, *gpovedanop*@*gmail.com*

Resumen: En este reporte se analizará el algoritmo de producto de matrices de Strassen, donde calcularemos el orden de complejidad de manera teórica y lo comprobaremos de manera gráfica.

Palabras Clave: Matriz,Producto,Bloques,Straessen.

1 Introducción

Cuando se tiene que realizar productos de matrices, existen bastantes factores a considerar, como el tama no de las mismas y el número de operaciones que se realizarán.

Es por ello que el algoritmo convencional tiende a tener complejidad $\Theta \epsilon n^3$, ya que se recorre el arreglo para obtener la suma de una posición mediante un ciclo for y se usan otros dos para repetir dicho proceso para todos los elementos de la matriz. En esta práctica analizaremos un algoritmo alternativo, el cual a pesar de no ser el óptimo, mejora bastante el tiempo de ejecución.

2 Conceptos Básicos

El producto de matrices se define con dos matrices A y B de dimensiones $m*n$ y $n*p$, respectivamente, se propone su producto $A*B$ como la matriz de dimensión $m*p$ tal que el elemento de la posición fila i y columna j es el resultado de los vectores fila i de A y columna j de B. Para que se pueda realizar se tienen que cumplir las siguientes condiciones:

Si las matrices son:

$$A = (a_{ij}),\ 1 \leq i \leq m\ ,\ 1 \leq j \leq n$$
$$B = (b_{ij}),\ 1 \leq i \leq n\ ,\ 1 \leq j \leq p$$

Entonces el producto es:

$$A * B = (m_{ij}),\ 1 \leq i \leq m\ ,\ 1 \leq j \leq p$$

Donde:

$$m_{ij} = \sum_{k=1}^{n} a_{i,k} * b_{i,k},\ 1 \leq i \leq m\ ,\ 1 \leq j \leq p$$

El Algoritmo de Schonhage-Strassen es un algoritmo que consiste en la multiplicación de Matrices. Es asintóticamente más rápido que el algoritmo de multiplicacián de matrices estándar, pero más lento que el algoritmo más rápido conocido, y es útil en la práctica para matrices grandes. Para mostrar el algoritmo propondremos dos matrices cuadradas A y B de tama no n igual a una potencia de Z y lo compararemos con un algoritmo similar para calcular el orden de complejidad del propio algoritmo de Strassen:

$$\text{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \text{ B} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \text{ C} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

Donde:

$$c_{11} = a_{11}{*}b_{11}{+}a_{12}{*}b_{21}$$
$$c_{12} = a_{11}{*}b_{12}{+}a_{12}{*}b_{21}$$
$$c_{21} = a_{21}{*}b_{11}{+}a_{22}{*}b_{11}$$
$$c_{22} = a_{21}{*}b_{12}{+}a_{22}{*}b_{22}$$

Ahora se mostrará un algoritmo recursivo que nos ayudará a calcular el orden de complejidad.

$PMR(A[...], B[...], n)$
$if(n == 2)$
    Regresa el producto usual de matrices

$else$
    $c_{11} = PMR(A_{11}, B_{11}, n/2){+}PMR(A_{12}, B_{12}, n/2)$
    $c_{12} = PMR(A_{11}, B_{12}, n/2){+}PMR(A_{12}, B_{22}, n/2)$
    $c_{21} = PMR(A_{21}, B_{11}, n/2){+}PMR(A_{22}, B_{11}, n/2)$
    $c_{22} = PMR(A_{21}, B_{12}, n/2){+}PMR(A_{22}, B_{22}, n/2)$

Donde proponemos la variable a = #Número de productos y b=#Número de sumas.

Calculando la recurrecncia:

$$T(n) = \begin{cases} a+b & si \quad n=2 \\ \\ a*T(\frac{n}{2}) + b*(\frac{n}{2})^2 & si \quad n>2 \end{cases}$$

Resolviendo mediante recursividad hacia adelante (Suponiendo sólo matrices cuadradas en potencias de dos):

$$T(n=2)=a+b$$

$$T(n=4)=a*T(4/2)+b*(\tfrac{4}{2})^2=a(a+b)+b*(\tfrac{4}{2})^2$$

$$T(n=8)=a*T(8/2)+b*(\tfrac{8}{2})^2=a^3+a^2*b+a*b*(\tfrac{4}{2})^2+b*(\tfrac{8}{2})^2$$

$$...$$

$$T(n=2^k)=a^k+a^{k-1}*b+a^{k-2}*b*(\tfrac{4}{2})^2+b*(\tfrac{4}{2})^2+...+b*(\tfrac{2^k}{2})^2$$

Siendo el último resultado igual a:

$$= a^k + b*\sum_{i=0}^{k=1} a^i * (\tfrac{2^{k-i}}{2})$$

Desarrollando la sumatoria (nótese que se obtiene una serie geométrica, por lo que se trabaja con ella):

$$= a^k + \tfrac{b}{4} * \sum_{i=0}^{k=1} a^i * (\tfrac{4^k}{4^i})$$

$$= a^k + \tfrac{b*4^k}{4} * \sum_{i=0}^{k=1} (\tfrac{a}{4})^i$$

$$= a^k + \tfrac{b*4^k}{4} * (\tfrac{(\frac{a}{4})^k - 1}{(\frac{a}{4})-1})$$

$$= a^k + \tfrac{b*4^k}{4} * (\tfrac{4*(a^k-4^k)}{4^k*(a-4)})$$

$$= a^k + b * (\tfrac{a^k}{a-4} - \tfrac{4^k}{a-4})$$

Acontando (para $a-4>1$):

$$= a^k + b * (\tfrac{a^k}{a-4} - \tfrac{4^k}{a-4}) \leq a^k + b * (\tfrac{a^k}{a-4})$$

Por lo que tenemos que la complejidad de este algoritmo es $\Theta(a^k)$.

Ahora regresaremos a términos de n:

$$\Theta(a^k) = \Theta(a^{\log n}) = \Theta(n^{\log a}) = \Theta(n^{\log_2 a})$$

Podemos observar que la complejidad está dada por el númeroo de productos que realice la función. En el caso de la función original son ocho, por lo que tenemos que $\Theta(n^{\log_2 a}) = \Theta(n^{\log_2 8}) = \Theta(n^3)$.

Retomando nuestro propósito original, el algoritmo de Straessen nos permite definir los elementos de la matriz C de la siguiente manera:

$$\begin{aligned} C_{11} &= S_1 + S_2 - S_4 + S_6 \\ C_{12} &= S_4 + S_5 \\ C_{21} &= S_6 + S_2 \\ C_{22} &= S_2 - S_3 + S_5 - S_7 \end{aligned}$$

Donde $S_{1,\ldots,7}$ es igual a:

$$\begin{aligned} S_1 &= (A_{12} - A_{22})(B_{21} + B_{22}) \\ S_2 &= (A_{11} - A_{22})(B_{11} + B_{22}) \\ S_3 &= (A_{11} - A_{21})(B_{11} + B_{12}) \\ S_4 &= (A_{11} + A_{12})(B_{22}) \\ S_5 &= (A_{11})(B_{12} + B_{22}) \\ S_6 &= (A_{22})(B_{21} - B_{11}) \\ S_7 &= (A_{21} + A_{22})(B_{11}) \end{aligned}$$

Podemos observar que en este caso se realizan sólo siete productos, por lo que si retomamos el resultado del algoritmo original y sustituimos con el valor de productos de este tenemos que $\Theta(n^{\log_2 a}) = \Theta(n^{\log_2 7})$, donde el sustituimos el valor truncado del logaritmo, por lo que finalmente $\Theta(n^{2.8})$.

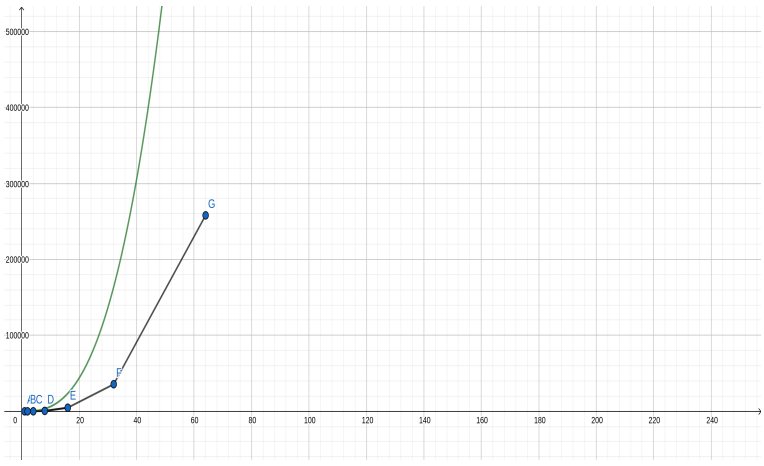
Este algoritmo tiene complejidad menor que los convencionales debido a la reducción de operaciones.

3 Experimentación y Resultados

```
[alejandro@alejandro-pc ~]$ python3 straessen.py
-----Matriz 1-----
[2, 2, 5, 4, 1, 5, 1, 2]
[5, 3, 5, 3, 3, 1, 4, 4]
[4, 5, 4, 2, 5, 2, 2, 4]
[2, 3, 1, 2, 2, 4, 2, 1]
[2, 4, 5, 4, 3, 3, 2, 3]
[2, 3, 2, 1, 2, 3, 2, 2]
[4, 2, 3, 4, 2, 4, 5, 4]
[4, 2, 3, 1, 1, 3, 1, 2]
-----Matriz 2-----
[5, 5, 5, 3, 5, 1, 5, 5]
[5, 2, 4, 2, 5, 5, 1, 2]
[1, 2, 2, 3, 4, 4, 3, 1]
[3, 4, 3, 5, 5, 3, 1, 5]
[1, 4, 4, 4, 2, 3, 1, 3]
[1, 5, 2, 1, 2, 3, 5, 5]
[4, 2, 2, 1, 3, 2, 1, 2]
[2, 4, 4, 2, 4, 1, 3, 3]
-----Producto-----
[51, 79, 64, 59, 83, 66, 64, 75]
[82, 94, 94, 76, 111, 73, 70, 85]
[78, 96, 98, 76, 107, 80, 68, 85]
[48, 62, 54, 41, 61, 50, 45, 60]
[67, 87, 82, 72, 100, 79, 62, 80]
[47, 59, 55, 40, 62, 49, 45, 54]
[79, 100, 88, 70, 105, 70, 74, 95]
[48, 63, 57, 42, 66, 45, 55, 58]
-----DATOS ADICIONALES-----
Tamaño matriz:8
Contador: 609
[alejandro@alejandro-pc ~]$
```

Ejecución de programa.

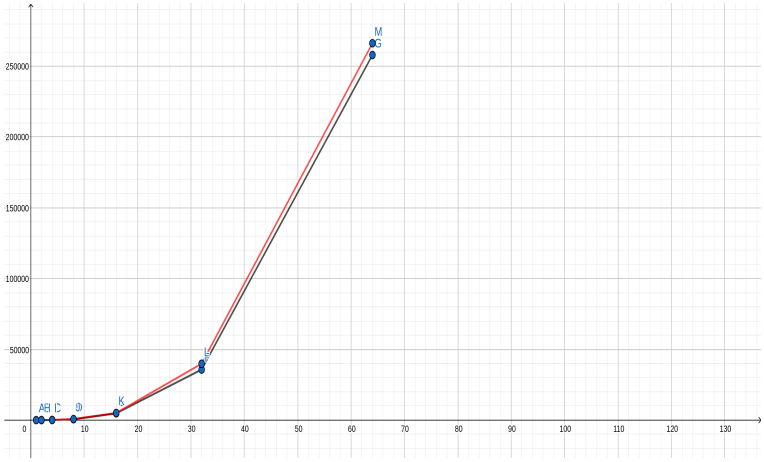
Gráfica de la función de producto de Strassen acotando con la función $10n^{2.8}$ (mostrada en color verde).



Valores de función Strassen

Tama *n*o	**Valor**
2	1
4	65
8	609
16	4801
32	35681
64	257985

Gráfica de la función de producto de Strassen acotando con la función del algoritmo convencional (mostrada en color rojo).



Valores de función tradicional (retomamos la gráfica generada anteriormente)

Tama *n*o	**Valor**
2	15
4	85
8	731
16	5021
32	39825
64	266305

Podemos notar que es muy poca la diferencia de complejidad de ambos algoritmos, tanto en la primera gráfica como en la segundo, esto se debe a que este problema no depende de los datos que maneja, si no de cuántas veces tendrá que realizar tanto las sumas como los productos necesarios para llegar a un resultado. A pesar de realizar la misma función, uno es mejor que el otro, por muy poco.

4 Conclusiones

Conclusión General: El producto de matrices es una operación que puede ser tan compleja o sencilla dependiendo de las propias matrices en sí, es por ello que mientras que un algoritmo se centra en únicamente ir realizando las operaciones, el otro divide matrices y busca ir reduciendo el tama no de estas para ir realizando cada vez menos operaciones, es por ello que a pesar de que en algún momento si se utiliza el método tradicional, se usa menos en el de Strassen.

Conclusión Gutiérrez Povedano: Al término de la práctica podemos concluir que el algoritmo de strassen tiene un orden de complejidad menor a la multiplicación de matrices normal, pero al estar limitado a matrices cuadradas de tama no potencia de 2 su utilidad se ve bastante reducida.

Conclusión Reyes Valenzuela: Tuve problemas al codificar el programa, ya que existían ciertas operaciones que requerían utilizar las submatrices para obtener otra y a su vez volver a llamar el algoritmo para continuar, sin embargo, si pude entender el porqué de la complejidad de ambos algoritmos, al igual que se demostró tanto de manera teórica como de manera práctica.

5 Bibliografía

Matesfacil.com (2018). [online] Available at: https://www.matesfacil.com/matrices/resueltos-matrices-producto.html [Accessed 10 Oct. 2018].

Wikipedia.org (2018). [online] Available at: https://es.wikipedia.org/wiki/Algoritmo-de-Schonhage-Strassen. [Accessed 10 Oct. 2018].