

Análisis de Algoritmos, Sem: 2019-1, 3CV1, Práctica 6, 10/10/2018

PRÁCTICA 6: ALGORITMO DE STRASSEN

Hernández Castellanos César Uriel, Aguilar Garcia Mauricio

Escuela Superior de Cómputo
Instituto Politécnico Nacional, México

uuriel12009u@gmail.com, mauricio.aguilar.garcia,90@gmail.com

Resumen: Se implementa el algoritmo Strassen el cual reduce la complejidad del algoritmo utilizado para la multiplicación de matrices de $O(n^3)$ a $O(n^{2.8})$.

Palabras Clave: Algoritmo, Complejidad, Matriz, Strassen.

1. Introducción

Uno de los problemas computacionales que más se ha estudiado y optimizado son la multiplicación de matrices. La definición de la operación de matrices es muy sencilla de describir, lo cual simplifica su comprensión.[1]

Los métodos mas comunes son el método iterativo de la multiplicación de matrices, que ocupa la definición posteriormente descrita (en la sección 2; y el método recursivo para matrices cuadradas de orden 2, que partiona una matriz en 4 submatrices de mismo tamaño. Ambos con orden de complejidad $\Theta(n^3)$.[2]

El algoritmo de Strassen, al igual que el algoritmo recursivo que resuelve el mismo problema de multiplicación de matrices cuadradas de orden 2, implementa el paradigma divide y vencerás. Pero a diferencia de este ocupa 7 multiplicaciones (llamadas recursivas) en lugar de 8. Para el algoritmo de Strassen se particiona cada matriz en 4 submatrices (11,12,21,22) del mismo tamaño y se expresa la matriz $C = A * B$ con base en sus submatrices. A continuación se describe el algoritmo de Strassen:[2]

Llamadas recursivas a Strassen	Elementos de la Matriz resultado C.
$s_1 = (a_{11} + a_{22})(b_{11} + b_{22})$	$c_{11} = s_1 + s_4 - s_5 + s_7$
$s_2 = (a_{21} + a_{22})(b_{11})$	$c_{12} = s_2 + s_4$
$s_3 = (a_{11})(b_{12} - b_{22})$	$c_{21} = s_3 + s_5$
$s_4 = (a_{22})(b_{21} - b_{11})$	$c_{22} = s_1 + s_3 - s_2 + s_6$
$s_5 = (a_{11} + a_{12})(b_{22})$	
$s_6 = (a_{21} - a_{11})(b_{11} + b_{12})$	
$s_7 = (a_{12} - a_{22})(b_{21} + b_{22})$	

2. Conceptos Básicos

2.1. Cota ajustada asintótica

En análisis de algoritmos una cota ajustada asintótica es una función que sirve de cota tanto superior como inferior de otra función cuando el argumento tiende a infinito. Usualmente se utiliza la notación $\theta(g(x))$ para referirse a las funciones acotadas por la función $g(x)$. [2]

2.2. Cota inferior asintótica

En análisis de algoritmos una cota inferior asintótica es una función que sirve de cota inferior de otra función cuando el argumento tiende a infinito. Usualmente se utiliza la notación $\Omega(g(x))$ para referirse a las funciones acotadas inferiormente por la función $g(x)$. [2]

2.3. Cota superior asintótica

En análisis de algoritmos una cota superior asintótica es una función que sirve de cota superior de otra función cuando el argumento tiende a infinito. Usualmente se utiliza la notación de Landau: $O(g(x))$, Orden de $g(x)$, coloquialmente llamada Notación O Grande, para referirse a las funciones acotadas superiormente por la función $g(x)$. [2]

2.4. Algoritmos

2.4.1. Strassen

En la disciplina matemática del álgebra lineal, el algoritmo de Strassen, llamado así por Volker Strassen, es un algoritmo usado para la multiplicación de matrices. Es asintóticamente más rápido que el algoritmo de multiplicación de matrices estándar, pero más lento que el algoritmo más rápido conocido, y es útil en la práctica para matrices grandes.

Volker Strassen publicó el algoritmo de Strassen en 1969. Pese a que su algoritmo es sólo ligeramente más rápido que el algoritmo estándar para la multiplicación de matrices, fue el primero en señalar que el enfoque estándar no es óptimo. Su artículo comenzó la búsqueda de algoritmos aún más rápidos, como el complejo algoritmo de Coppersmith–Winograd de Shmuel Winograd en 2010 (que utiliza

20 multiplicaciones binarias, pero utiliza 155 sumas binarias en lugar de las 18 del algoritmo de Strassen), publicado en 2000.[3]

3. Experimentación y Resultados

3.1. Strassen

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

Las siguientes imagenes (figura 1,2,3 y 4) son el código implementando el algoritmo Strassen:

```
int main(){
    ofstream al;
    al.open("out.txt");
    srand (time(NULL));
    for(int NUM = 1; NUM<=MAX; NUM*=2){
        cont = 0;
        int n=pow(2,NUM);
        double **a = creaMatriz(n,1);
        double **b = creaMatriz(n,1);
        double **c = creaMatriz(n,0);
        strassen(a,b,c,n);
        //Imprimir matrices
        /*imprimeMatriz(a,n);
        cout << "-----" << endl;
        imprimeMatriz(b,n);
        cout << "-----" << endl;
        imprimeMatriz(c,n);*/
        cout << NUM << endl;
        al << n << "," << cont << '\n';
    }
    al.close();
    return 0;
}
```

Figura 1: Código de main

```

void strassen(double **a, double **b, double **c, int tam) {

    // cuando matriz es 1 x 1:
    if (tam == 1) {
        c[0][0] = a[0][0] * b[0][0];
        cont++;
        return;
    }

    // otros casos:
    else {
        int newTam = tam/2;
        double **a11, **a12, **a21, **a22;
        double **b11, **b12, **b21, **b22;
        double **c11, **c12, **c21, **c22;
        double **p1, **p2, **p3, **p4, **p5, **p6, **p7;

        // creaci n de matrices:
        a11 = creaMatriz(newTam, -1);
        a12 = creaMatriz(newTam, -1);
        a21 = creaMatriz(newTam, -1);
        a22 = creaMatriz(newTam, -1);

        b11 = creaMatriz(newTam, -1);
        b12 = creaMatriz(newTam, -1);
        b21 = creaMatriz(newTam, -1);
        b22 = creaMatriz(newTam, -1);

        c11 = creaMatriz(newTam, -1);
        c12 = creaMatriz(newTam, -1);
        c21 = creaMatriz(newTam, -1);
        c22 = creaMatriz(newTam, -1);

        p1 = creaMatriz(newTam, -1);
        p2 = creaMatriz(newTam, -1);
        p3 = creaMatriz(newTam, -1);
        p4 = creaMatriz(newTam, -1);
        p5 = creaMatriz(newTam, -1);
        p6 = creaMatriz(newTam, -1);
        p7 = creaMatriz(newTam, -1);

        double **aResultado = creaMatriz(newTam, -1);
        double **bResultado = creaMatriz(newTam, -1);

        int i, j;

        //divide las matrices en 4 matrices:
        for (i = 0; i < newTam; i++) {
            for (j = 0; j < newTam; j++) {
                a11[i][j] = a[i][j];
                a12[i][j] = a[i][j + newTam];
                a21[i][j] = a[i + newTam][j];
                a22[i][j] = a[i + newTam][j + newTam];

                b11[i][j] = b[i][j];
                b12[i][j] = b[i][j + newTam];
                b21[i][j] = b[i + newTam][j];
                b22[i][j] = b[i + newTam][j + newTam];
            }
        }
    }
}

```

Figura 2: Primera parte del c digo del algoritmo strassen

```

// Calculando p1 a p7:

suma(a11, a22, aResultado, newTam); // a11 + a22
suma(b11, b22, bResultado, newTam); // b11 + b22
strassen(aResultado, bResultado, p1, newTam); // p1 = (a11+a22) * (b11+b22)

suma(a21, a22, aResultado, newTam); // a21 + a22
strassen(aResultado, b11, p2, newTam); // p2 = (a21+a22) * (b11)

resta(b12, b22, bResultado, newTam); // b12 - b22
strassen(a11, bResultado, p3, newTam); // p3 = (a11) * (b12 - b22)

resta(b21, b11, bResultado, newTam); // b21 - b11
strassen(a22, bResultado, p4, newTam); // p4 = (a22) * (b21 - b11)

suma(a11, a12, aResultado, newTam); // a11 + a12
strassen(aResultado, b22, p5, newTam); // p5 = (a11+a12) * (b22)

resta(a21, a11, aResultado, newTam); // a21 - a11
suma(b11, b12, bResultado, newTam); // b11 + b12
strassen(aResultado, bResultado, p6, newTam); // p6 = (a21-a11) * (b11+b12)

resta(a12, a22, aResultado, newTam); // a12 - a22
suma(b21, b22, bResultado, newTam); // b21 + b22
strassen(aResultado, bResultado, p7, newTam); // p7 = (a12-a22) * (b21+b22)

// calculando c21, c21, c11 e c22:

suma(p3, p5, c12, newTam); // c12 = p3 + p5
suma(p2, p4, c21, newTam); // c21 = p2 + p4

suma(p1, p4, aResultado, newTam); // p1 + p4
suma(aResultado, p7, bResultado, newTam); // p1 + p4 + p7
resta(bResultado, p5, c11, newTam); // c11 = p1 + p4 - p5 + p7

suma(p1, p3, aResultado, newTam); // p1 + p3
suma(aResultado, p6, bResultado, newTam); // p1 + p3 + p6
resta(bResultado, p2, c22, newTam); // c22 = p1 + p3 - p2 + p6

// Se juntan las matrices resultantes en una sola:
for (i = 0; i < newTam; i++) {
    for (j = 0; j < newTam; j++) {
        c[i][j] = c11[i][j];
        c[i][j + newTam] = c12[i][j];
        c[i + newTam][j] = c21[i][j];
        c[i + newTam][j + newTam] = c22[i][j];
    }
}

```

Figura 3: Segunda parte del código del algoritmo strassen

```

/*-----*/
// funcion para sumar dos matrices
void suma(double **a, double **b, double **Resultado, int tam) {
    int i, j;
    for (i = 0; i < tam; i++) {
        for (j = 0; j < tam; j++) {
            Resultado[i][j] = a[i][j] + b[i][j];
        }
    }
}

/*-----*/
// funciones para restar dos matrices
void resta(double **a, double **b, double **Resultado, int tam) {
    int i, j;
    for (i = 0; i < tam; i++) {
        for (j = 0; j < tam; j++) {
            Resultado[i][j] = a[i][j] - b[i][j];
        }
    }
}

```

Figura 4: Código de la suma y resta de matrices

Como salida del código tenemos a la n del tamaño de la entrada y la t la cual es el contador del algoritmo en la figura 5. Si graficamos los valores de la salida respecto a la función de complejidad pro-

2,7
4,49
8,343
16,2401
32,16807
64,117649
128,823543

Figura 5: Salida del programa Strassen

puesta que es $O(n^{2,81})$ entonces obtenemos la gráfica de la figura 6.

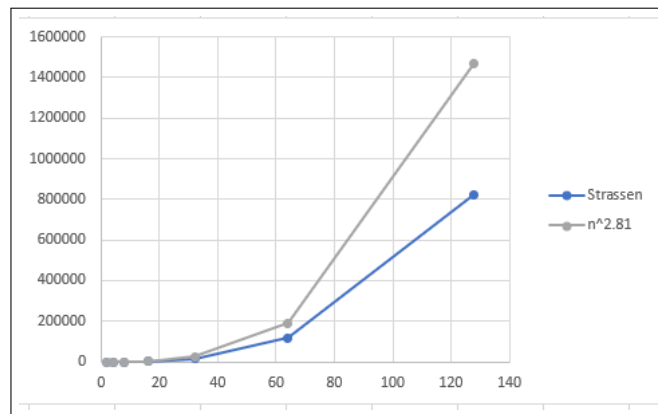


Figura 6: Gráfica de la salida del programa strassen junto con la función $O(n^{2,81})$

(ii). Para valores muy grandes de matrices compare el algoritmo de Strassen con el producto usual de matrices (comparar la complejidad). Usando como entradas grandes las primeras 10 potencias de 2, se tiene que las salidas del programa son las que aparecen en la Figura 1.


n_1		t_1
2		8
4		64
8		512
16		4096
32		32768
64		262144
128		2097152
256		16777216
512		134217728
1024		1073741824

Figura 7: Salida del programa en su versión iterativa, que es $\Theta(n^3)$.

De manera análoga, usando como entradas grandes las primeras 10 potencias de 2, se tiene que las salidas del programa usando el Algoritmo de Strassen son las que aparecen en la Figura 2.


n_2		t_2
2		7
4		49
8		343
16		2401
32		16807
64		117649
128		823543
256		5764801
512		40353607
1024		282475249

Figura 8: Salida del programa Strassen $\Theta(n^{2.8})$.

Con los datos que nos arrojaron ambos algoritmos se llega a la siguiente gráfica, que no permite ver como difiere el tiempo entre ambos algoritmos, como se puede ver en la Figura 3.

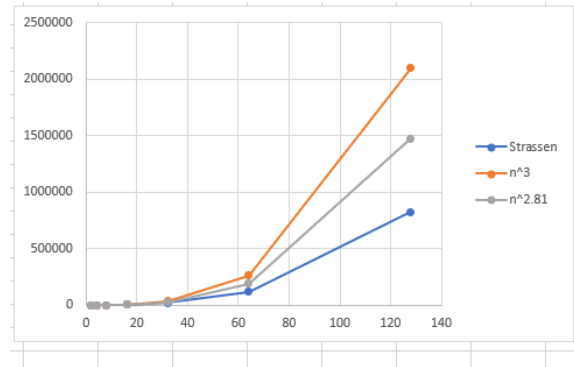


Figura 9: Gráfica comparativa de la versión iterativa y la de Strassen.

4. Conclusiones

En esta práctica pudimos verificar que hay algoritmos en los que su complejidad puede variar por muy poco, pero cuando se trata de entradas muy grandes esa pequeña diferencia se convierten en bastante tiempo, y también nos muestra la importancia de buscar siempre el algoritmo más óptimo, para que en casos como éste, en el que la complejidad solo cambia por muy poco, para que en entradas grandes se mantenga el tiempo a lo mínimo posible.

4.1. Mauricio Aguilar Garcia

En ésta práctica pudimos implementar el algoritmo strassen el cual demostró que la manera en que se estaba realizando la multiplicación de matrices no era la optima y dió cabida a que se buscaran maneras más rápidas de realizarlas, el algoritmo toma ayuda de la manera de resolver problemas el cual es divide y vencerás aplicandolo de manera recursiva y aunque el algoritmo se note mucho más complejo de implementar si se llega a notar una diferencia de operaciones realizadas respecto a su contra parte.

4.2. Hernández Castellanos César Uriel

El algoritmo de Strassen mejora el algoritmo tradicional del producto de dos matrices, realiza las sub-soluciones de una manera exacamente igual, pero con la peculiaridad de hacer un producto menos, resultando que nuestra complejidad será mejorada.

Se podría decir que el algoritmo de Strassen con respecto a los algoritmos tradicionales de producto de matrices no significan un cambio importante en el caso de matrices de un tamaño pequeño, pero al momento de ingresar matrices de una mayor dimensión el algoritmo de Strassen empieza a tener un efecto bastante significativo.

Bibliografía

- [1] 'Algoritmo divide y vencerás' Disponible en https://es.wikipedia.org/wiki/Algoritmo_divide_y_vencerás consultado 12 de Septiembre del 2017
- [2] Introduction to Algorithms, Second Edition by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein
- [3] Strassen, Volker, La eliminación gaussiana no es óptima, Numer. Math. 13, p. 354-356, 1969