07.04.2020

Strassen's Matrix Multiplication

Given 2 square matrices A & B of size n×n each, find their multiplication matrix.

> Naive method:

```
void multiply (int A[][], int B[][], int C[][])
{
    for (int i=0; i<N; i++)
        for(int j=0; j<N; j++)
        {
            C[i][j]=0
            for (int k=0; k<N; k++)
                C[i][j] += A[i][k] * B[k][j];
        }
}
```

Time complexity: $O(N^3)$

> Divide & Conquer method:

(i) Divide matrices A & B into 4 sub-matrices of size N/2 × N/2 each.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{bmatrix}$$

(ii) Calculate the values ae+bg, af+bh, ce+dg, cf+dh recursively.

In this method we do 8 multiplications for matrices of size $\frac{N}{2} \times \frac{N}{2}$ and 4 additions.

Addition of 2 matrices takes $O(N^2)$ time. So the time complexity is:

$$T(N) = 8T(N/2) + O(N^2)$$

Applying master's theorem, complexity is $O(N^3)$ → same as the Naive approach.

> Strassen's method

In the divide & conquer method, the main component for high time complexity is 8 recursive calls. The idea of strassen's method is to reduce the no. of recursive calls to 7.

Strassen's method is similar to the divide and conquer method, but the 4 sub-matrices of result are calculated using the following formulae.

$p1 = a(f-h)$ $\quad$ $p5 = (a+d)(e+h)$

$p2 = (a+b)h$ $\quad$ $p6 = (b-d)(g+h)$

$p3 = (c+d)e$ $\quad$ $p7 = (a-c)(e+f)$

$p4 = d(g-e)$

AxB can be calculated using the above 7 multiplications.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} p5+p4-p2+p6 & p1+p2 \\ p3+p4 & p1+p5-p3-p7 \end{bmatrix}$$

Time complexity : $T(N) = 7T(N/2) + O(N^2)$

Using master's theorem, time complexity = $O(N^{\log_2 7}) = O(N^{2.8074})$