Divide and Conquer: Strassen's Matrix Multiplication

Session No.: 13
Course Name: Design and analysis of algorithm
Course Code: R1UC407B
Instructor Name: Mili Dhar
Duration: 50 Min.
Date of Conduction of Class:

Review of the key concepts

1. Review of previous session

Q: Which takes more time: matrix addition or matrix multiplication?

Learning Outcome

Explain the core idea behind Strassen’s algorithm

Analyse the time complexity of the algorithm.

Session Outline

1 Introduction to KMP approach

2 Step by Step demonstration of KMP approach

3 Apply KMP approach to find a pattern in a given text

4 Analyse its complexity

Basic Matrix Multiplication

Suppose we want to multiply two matrices of size N x N:

for example A x B = C.

$$\begin{vmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{vmatrix} = \begin{vmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{vmatrix} \begin{vmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{vmatrix}$$

$$\mathbf{C_{11} = A_{11}B_{11} + A_{12}B_{21}}$$

$$\mathbf{C_{12} = A_{11}B_{12} + A_{12}B_{22}}$$

$$\mathbf{C_{21} = A_{21}B_{11} + A_{22}B_{21}}$$

$$\mathbf{C_{22} = A_{21}B_{12} + A_{22}B_{22}}$$

Basic Matrix Multiplication

```
void multiply(int A[][N], int B[][N], int C[][N])
{
    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 analysis

The time Complexity of the above method is $O(N^3)$.

Matrix Multiplication using Divide and Conquer

Following is simple Divide and Conquer method to multiply two square matrices.

1. Divide matrices A and B in 4 sub-matrices of size N/2 x N/2 as shown in the below diagram.
2. Calculate the following values recursively.

Divide and Conquer Matrix Multiply

$$A \times B = C$$

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} A_{11}\times B_{11}+A_{12}\times B_{21} & A_{11}\times B_{12}+A_{12}\times B_{22} \\ A_{21}\times B_{11}+A_{22}\times B_{21} & A_{21}\times B_{12}+A_{22}\times B_{22} \end{bmatrix}$$

- Divide matrices into sub-matrices: A_{11} , A_{12}, A_{21} etc
- Use blocked matrix multiply equations
- Recursively multiply sub-matrices

Divide and Conquer Matrix Multiply

Divide-and conquer is a general algorithm design paradigm:

- Divide: divide the input data $\boldsymbol{S}$ in two or more disjoint subsets $\boldsymbol{S}_1$, $\boldsymbol{S}_2$, …
- Recur: solve the subproblems recursively
- Conquer: combine the solutions for $\boldsymbol{S}_1$, $\boldsymbol{S}_2$, …, into a solution for $\boldsymbol{S}$

The base case for the recursion are subproblems of constant size

Analysis can be done using **recurrence equations**

Divide and Conquer Matrix Multiply

$$A \times B = C$$

$$\boxed{a_0} \times \boxed{b_0} = \boxed{a_0 \times b_0}$$

- Terminate recursion with a simple base case

Divide and Conquer Matrix Multiply

Array A =>

1	1	1	1
2	2	2	2
3	3	3	3
2	2	2	2

Array B =>

1	1	1	1
2	2	2	2
3	3	3	3
2	2	2	2

Result Array =>

8	8	8	8
16	16	16	16
24	24	24	24
16	16	16	16

Matrix Multiplication using Divide and Conquer

MMult(A, B, n)
1. If n = 1 Output A × B
2. Else
3. Compute A11, B11, . . ., A22, B22 % by computing m = n/2
4. X1 ← MMult(A11, B11, n/2)
5. X2 ← MMult(A12, B21, n/2)
6. X3 ← MMult(A11, B12, n/2)
7. X4 ← MMult(A12, B22, n/2)
8. X5 ← MMult(A21, B11, n/2)
9. X6 ← MMult(A22, B21, n/2)
10. X7 ← MMult(A21, B12, n/2)
11. X8 ← MMult(A22, B22, n/2)
12. C11 ← X1 + X2
13. C12 ← X3 + X4
14. C21 ← X5 + X6
15. C22 ← X7 + X8
16. Output C
17. End If

Matrix Multiplication using Divide and Conquer

Analysis:

The operations on line 3 take constant time. The combining cost (lines 12–15) is $\Theta(n^2)$ (adding two $n/2 \times n/2$ matrices takes time $n^2/4 = \Theta(n^2)$). There are 8 recursive calls (lines 4–11). So let $T(n)$ be the total number of mathematical operations performed by MMult(A, B, n),

then $T(n) = 8T(n/2) + \Theta(n^2)$

The Master Theorem gives us $T(n) = \Theta(n^{\log_2(8)}) = \Theta(n^3)$

So this is not an improvement on the "obvious" algorithm given earlier (that uses n^3 operations).

Strassens’s Matrix Multiplication

Strassen showed that 2x2 matrix multiplication can be accomplished in 7 multiplication and 18 additions or subtractions. .**($2^{\log_2 7} = 2^{2.807}$)**

This reduction can be done by Divide and Conquer Approach.

Strassen’s method, the four sub-matrices of the result are calculated using the following formulae.

Strassens's Matrix Multiplication

$$A \times B = C$$

$$\begin{array}{|c|c|}\hline A_{11} & A_{12} \\ \hline A_{21} & A_{22} \\ \hline \end{array} \times \begin{array}{|c|c|}\hline B_{11} & B_{12} \\ \hline B_{21} & B_{22} \\ \hline \end{array} = \begin{array}{|c|c|}\hline \mathbf{P_5+P_4-P_2+P_6} & \mathbf{P_1+P_2} \\ \hline \mathbf{P_3+P_4} & \mathbf{P_1+P_5-P_3-P_7} \\ \hline \end{array}$$

- Divide matrices into sub-matrices: A_{11} , A_{12}, A_{21} etc
- Use blocked matrix multiply equations
- Recursively multiply sub-matrices

Strassens's Matrix Multiplication

$$\begin{vmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{vmatrix} = \begin{vmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{vmatrix} \begin{vmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{vmatrix}$$

$P_1 = (A_{11} + A_{22})(B_{11} + B_{22})$
$P_2 = (A_{21} + A_{22}) * B_{11}$
$P_3 = A_{11} * (B_{12} - B_{22})$
$P_4 = A_{22} * (B_{21} - B_{11})$
$P_5 = (A_{11} + A_{12}) * B_{22}$
$P_6 = (A_{21} - A_{11}) * (B_{11} + B_{12})$
$P_7 = (A_{12} - A_{22}) * (B_{21} + B_{22})$

$C_{11} = P_1 + P_4 - P_5 + P_7$
$C_{12} = P_3 + P_5$
$C_{21} = P_2 + P_4$
$C_{22} = P_1 + P_3 - P_2 + P_6$

Computation

$$
\begin{aligned}
C_{11} &= P_1 + P_4 - P_5 + P_7 \\
&= (A_{11}+ A_{22})(B_{11}+B_{22}) + A_{22} * (B_{21} - B_{11}) - (A_{11} + A_{12}) * B_{22}+ \\
&\quad (A_{12} - A_{22}) * (B_{21} + B_{22}) \\
&= A_{11} B_{11} + A_{11} B_{22} + A_{22} B_{11} + A_{22} B_{22} + A_{22} B_{21} – A_{22} B_{11} - \\
&\quad A_{11} B_{22} -A_{12} B_{22} + A_{12} B_{21} + A_{12} B_{22} – A_{22} B_{21} – A_{22} B_{22} \\
&= A_{11} B_{11} + A_{12} B_{21}
\end{aligned}
$$

Strassen Algorithm

Strassen(A, B)
1. If n = 1 Output A × B
2. Else
3. Compute A11, B11, . . ., A22, B22 % by computing m = n/2
4. P1 ← Strassen(A11, B12 − B22)
5. P2 ← Strassen(A11 + A12, B22)
6. P3 ← Strassen(A21 + A22, B11)
7. P4 ← Strassen(A22, B21 − B11)
8. P5 ← Strassen(A11 + A22, B11 + B22)
9. P6 ← Strassen(A12 − A22, B21 + B22)
10. P7 ← Strassen(A11 − A21, B11 + B12)
11. C11 ← P5 + P4 − P2 + P6
12. C12 ← P1 + P2
13. C21 ← P3 + P4
14. C22 ← P1 + P5 − P3 − P7
15. Output C
16. End If

Complexity Analysis: Strassen Algorithm

Analysis: The operations on line 3 take constant time. The combining cost (lines 11–14) is $\Theta(n^2)$. There are 7 recursive calls (lines 4–10). So let $T(n)$ be the total number of mathematical operations performed by Strassen(A, B), then

$$T(n) = 7T(n/2) + \Theta(n^2)$$

The Master Theorem gives us

$$T(n) = \Theta(n^{\log_2(7)}) = \Theta(n^{2.8})$$

In the next class we will going through Greedy Approaches

Thank You