CS207 Design and Analysis of Algorithms

Sajith Gopalan

Indian Institute of Technology Guwahati

sajith@iitg.ac.in

January 14, 2022

Divide and Conquer

Divide and Conquer

- solve a problem recursively
  - Divide the problem into a number of smaller instances of the same problem
  - For each of these subproblems, if its size is sufficiently large, then Conquer it recursively, else Conquer it directly
  - Combine the solutions to the subproblems into the solution for the original problem

Matrix Multiplication

Input: $n \times n$ matrices A and B

Algorithm 1 Matrix Multiplication

```
for i = 1 to n do
    for j = 1 to n do
        C[i, j] = 0
        for k = 1 to n do
            C[i, j] = C[i, j] + A[i, k] * B[k, j]
        end for
    end for
end for
```

This algorithm runs in $O(n^3)$ time

Matrix Multiplication

- Split the matrices horizontally and vertically into 4 $n/2 \times n/2$ matrices each
- $\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$ and $\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$
- Then $A * B = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} * \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} =$ $\begin{bmatrix} A_{11} * B_{11} + A_{12} * B_{21} & A_{11} * B_{12} + A_{12} * B_{22} \\ A_{21} * B_{11} + A_{22} * B_{21} & A_{21} * B_{12} + A_{22} * B_{22} \end{bmatrix}$
- That is, multiplication of $n \times n$ matrices can be achieved through 8 multiplications of $n/2 \times n/2$ matrices and 4 additions of $n/2 \times n/2$ matrices

Divide and Conquer Matrix Multiplication

- This suggests a Divide and Conquer algorithm
- Addition of two $n/2 \times n/2$ matrices involves $n^2/4$ elementary additions
- The recurrence relation for the time complexity function is, therefore, $T(n) = 8T(n/2) + cn^2$

Analysis

- $T(n) = 8T(\frac{n}{2}) + cn^2$
  $= 8[8T(\frac{n}{2^2}) + c\frac{n^2}{2^2}] + cn^2 = 8^2T(\frac{n}{2^2}) + cn^2[2+1]$
  $= 8^3T(\frac{n}{2^3}) + cn^2[2^2+2+1]$
  $= 8^kT(\frac{n}{2^k}) + cn^2[2^{k-1} + \ldots + 1]$
  $= 8^kT(\frac{n}{2^k}) + cn^2[2^k - 1]$
  $= 8^{\log n} + cn^2[n-1]$ (When $k = \log n$)
  $= O(n^3)$
- This is no improvement on the iterative algorithm

Strassen's Algorithm

Input: A and B, two 2×2 matrices

Algorithm 2 Strassen's Algorithm

1: $m_1 = (A_{12} - A_{22}) * (B_{21} + B_{22})$
2: $m_2 = (A_{11} + A_{22}) * (B_{11} + B_{22})$
3: $m_3 = (A_{11} - A_{21}) * (B_{11} + B_{12})$
4: $m_4 = (A_{11} + A_{12}) * B_{22}$
5: $m_5 = A_{11} * (B_{12} - B_{22})$
6: $m_6 = A_{22} * (B_{21} - B_{11})$
7: $m_7 = (A_{21} + A_{22}) * B_{11}$
8: $C_{11} = m_1 + m_2 - m_4 + m_6$
9: $C_{12} = m_4 + m_5$
10: $C_{21} = m_6 + m_7$
11: $C_{22} = m_2 - m_3 + m_5 - m_7$

Analysis

- As can be seen, the algorithm performs 7 elementary multiplications and 18 elementary additions
- This can be generalized into an algorithm for $n \times n$ multiplication
- That will perform 7 $n/2 \times n/2$ multiplications and 18 $n/2 \times n/2$ additions
- Its time complexity can be expressed as: $T(n) = 7T(n/2) + cn^2$

Analysis

- $T(n) = 7T(\frac{n}{2}) + cn^2$
  $= 7[7T(\frac{n}{2^2}) + c\frac{n^2}{2^2}] + cn^2 = 7^2 T(\frac{n}{2^2}) + cn^2[\frac{7}{4} + 1]$
  $= 7^3 T(\frac{n}{2^3}) + cn^2[(\frac{7}{4})^2 + \frac{7}{4} + 1]$
  $= 7^k T(\frac{n}{2^k}) + cn^2[(\frac{7}{4})^{k-1} + \ldots + 1]$
  $= 7^k T(\frac{n}{2^k}) + c\frac{4}{3}.n^2[(\frac{7}{4})^k - 1]$
  $= 7^{\log n} + c\frac{4}{3}.n^2[\frac{7^{logn}}{4^{logn}} - 1]$ (When $k = \log n$)
  $= O(n^{\log 7})$
- This is $o(n^3)$!!