Spring 2024

CS 412 (Algorithms: Design and Analysis)

Weekly Challenge 05

Announced: Friday, February 16, 2024.
Deadline: Friday, February 23, 2024 (11:59 pm PKT).
Total marks: 1.

Instructions: Submit **individually** your solution as a PDF with the file name as your *studentID.pdf*; typeset in LaTeX. You must submit your solution on Canvas.

---

1. (1 point) Applying the standard mathematical method to define an algorithm for matrix multiplication gives an $O(n^3)$ solution. Given two nxn matrices, a brute force solution requires n^3 arithmetic operations. Just like fast integer multiplication, a solution exists to reduce the time complexity of matrix multiplication. For example, Strassen's algorithm completes matrix multiplication in $O(n^2.81)$. Start by identifying a fast matrix multiplication algorithm with complexity better than $O(n^3)$. Use the algorithm to compute the matrix product:

$$\begin{pmatrix} 1 & 3 \\ 6 & 5 \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

Clearly list all the steps of the solution.

Next, compare the dimension of matrix where your algorithm outperforms the brute force algorithm.

Solution: We will go by the hint of Strassen's algorithm [1] which has the complexity $O(n^{2.81})$

By following the steps below, we will implement the algorithm:

1 Splitting the matrices A and B where the two 2x2 matrices mentioned above are A and B respectively.

$$A = \begin{pmatrix} 1 & 3 \\ 6 & 5 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

$$B = \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

2 Then we compute the matrices by breaking the two nxn matrices into smaller sub-multiplications of $(n/2)x(n/2)$ matrices products:

$$P_1 = (1+5) \times (a+d) = 6(a+d)$$
$$P_2 = (6+5) \times a = 11a$$
$$P_3 = 1 \times (b-d) = b-d$$
$$P_4 = 5 \times (c-a) = 5(c-a)$$
$$P_5 = (1+3) \times d = 4d$$
$$P_6 = (6-1) \times (a+b) = 5(a+b)$$
$$P_7 = (3-5) \times (c+d) = -2(c+d)$$

3 By combining the sub-products following the algorithm, we will compute the final product matrix C:

$$c_{11} = P_1 + P_4 - P_5 + P_7 = 6(a+d) + 5(c-a) - 4d - 2(c+d)$$

$$c_{12} = P_3 + P_5 = (b-d) + 4d$$

$$c_{21} = P_2 + P_4 = 11a + 5(c-a)$$

$$c_{22} = P_1 - P_2 + P_3 + P_6 = 6(a+d) - 11a + (b-d) + 5(a+b)$$

4 Now by combining the resulting submatrices to obtain the final matrix C in the form:

$$C = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}$$

$$C = \begin{pmatrix} 6(a+d) + 5(c-a) - 4d - 2(c+d) & (b-d) + 4d \\ 11a + 5(c-a) & 6(a+d) - 11a + (b-d) + 5(a+b) \end{pmatrix}$$

Simplifying the matrix C leads to

$$C = \begin{pmatrix} a + 3c & b + 3d \\ 6a + 5c & 6b + 5d \end{pmatrix}$$

If we compare the Strassen's algorithm with brute force, the Strassen algorithm will have a step ahead of brute force when $n = 2$ for the nxn matrices since the time complexity of Strassen is $O(n^{log_2 7})$ while the brute force has complexity $O(n^3)$. However, we see more clear difference when $n \geq 30$ and the difference between the two algorithms is vast, and it seems more efficient as the value of n increases.

References

[1] Tuan Nguyen, Alex Adamson, and Andreas Santucci. Lecture 3: Distributed algorithms and optimization. Course lecture notes, April 2016. Scribed for CME 323: Distributed Algorithms and Optimization, Stanford University.