Homework 5

Due: October 19th

This homework must be typed in LaTeX and handed in via Gradescope.

Please ensure that your solutions are complete, concise, and communicated clearly. Use full sentences and plan your presentation before you write. Except in the rare cases where it is indicated otherwise, consider every problem as asking you to prove your result.

Problem 1

Given two $n \times n$ square matrices A and B, we want to compute their product $C = A \times B$ efficiently. This is a fundamental problem in linear algebra, and the interest it has generated both in the Computer Science and Math research community cannot be overstated. For a long time, the best-known algorithms to compute this product had runtime $O(n^3)$. However, in 1971, Volk Strassen presented what is now known as the first Fast Matrix Multiplication algorithm.

We begin by partitioning A, B, C into equally sized block matrices:

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}, B = \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix}, C = \begin{bmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{bmatrix}$$

where $A_{i,j}, B_{i,j}, C_{i,j} \in \mathbb{R}^{2^{n-1} \times 2^{n-1}}$.

The naive $O(n^3)$ approach would compute the product through its block matrices as such:

$$\begin{aligned} C_{1,1} &= A_{1,1}B_{1,1} + A_{1,2}B_{2,1} \\ C_{1,2} &= A_{1,1}B_{1,2} + A_{1,2}B_{2,2} \\ C_{2,1} &= A_{2,1}B_{1,1} + A_{2,2}B_{2,1} \\ C_{2,2} &= A_{2,1}B_{1,2} + A_{2,2}B_{2,2} \end{aligned}$$

Strassen instead noticed that it was possible to compute C as follows:

$$\begin{aligned} M_1 &:= (A_{1,1} + A_{2,2})(B_{1,1} + B_{2,2}) \\ M_2 &:= (A_{2,1} + A_{2,2})B_{1,1} \\ M_3 &:= A_{1,1}(B_{1,2} - B_{2,2}) \\ M_4 &:= A_{2,2}(B_{2,1} - B_{1,1}) \\ M_5 &:= (A_{1,1} + A_{1,2})B_{2,2} \\ M_6 &:= (A_{2,1} - A_{1,1})(B_{1,1} + B_{1,2}) \\ M_7 &:= (A_{1,2} - A_{2,2})(B_{2,1} + B_{2,2}) \end{aligned}$$

$$\begin{aligned} C_{1,1} &= M_1 + M_4 - M_5 + M_7 \\ C_{1,2} &= M_3 + M_5 \\ C_{2,1} &= M_2 + M_4 \\ C_{2,2} &= M_1 - M_2 + M_3 + M_6 \end{aligned}$$

(a) Briefly explain the main difference between the naïve approach and Strassen's idea.

(b) Sketch a divide and conquer recursive algorithm for square matrix multiplication based on the naïve approach. Provide a bound to its runtime using the Master Theorem. Rather than using asymptotic notation, try to analyze the runtime (expecially the combine step) with an exact constant.

(c) Sketch a divide and conquer recursive algorithm for square matrix multiplication based on Strassen's approach. Provide a bound to its runtime using the Master Theorem.

(d) Compare the two algorithms. Which one is asymptotically faster? Can we conclude that one is better than the other for any input size?

Problem 2

The recurrence $T(n) = 7T(n/2) + n^2$ describes the running time of an algorithm A. A competing algorithm A' has a running time of $T'(n) = aT'(n/4) + n^2$. What is the largest integer value for a such that A' is asymptotically faster than A?

Problem 3

An $m \times n$ array A of real numbers is a *Gnome array* if for all i, j, k, l such that $1 \leq i \leq k \leq m$ and $1 \leq j \leq \ell \leq n$, we have

$$A[i,j] + A[k,l] \leq A[i,l] + A[k,j].$$

That is, whenever we pick two rows and two columns of a Gnome array and consider the four elements at the intersections of rows and columns, the sum of the upper-left and lower-right elements is less or equal than to the sum of the lower-left and upper-right elements.

(a) Prove that an array is Gnome iff for all $i = 1, 2, \ldots, m-1$ and $j = 1, 2 \ldots, n-1$, we have:

$$A[i,j] + A[i+1,j+1] \leq A[i,j+1] + A[i+1,j].$$

Hint: For the if part, use induction separately on rows and column

(a) Let $f(i)$ be the index of the column containing the leftmost minimum lement of row i. Prove that $f(1) \leq f(2) \leq \ldots, f(m)$ for an $m \times n$ Gnome array.

(b) Here is a description of a divide-and-conquer algorithm that computes the leftmost minimum element in each row of a $m \times n$ Gnome array

- Construct a sub-array A' of A consisting of the even-numbered rows of A.
- Recursively invoke the algorithm to find the leftmost minimum for each row of A'.
- Using the information on the leftmost minimum of the even-numbered rows of A you just computed, computed the leftmost minimum in the odd-rows of A.

Explain how to compute the leftmost minimum of the odd-numbered rows of A given that you have already computed the leftmost minimum for even-numbered rows of A. Prove that this can be done in $O(n+m)$ time.

(c) Write the recurrence describing the running time of the algorithm, and obtain a explicit closed-form solution for it,

Hint: if you get stuck on any of the points, you can work on the remaining assuming the results you were asked to prove in the previous ones hold.