ECE 374 B - Fall 2024

Homework 4

- **Submit your solutions electronically on the course Gradescope site as PDF files.** If you plan to typeset your solutions, please use the LaTeX solution template on the course web site. If you must submit scanned handwritten solutions, please use a black pen on blank white paper and a high-quality scanner app (or an actual scanner, not just a phone camera). We will mark difficult to read solutions as incorrect and move on.
- **Every homework problem must be done *individually*.** Each problem needs to be submitted to Gradescope before 6AM of the due data which can be found on the course website: https://ecealgo.com/fa24/homeworks.html.
- For nearly every problem, **we have covered all the requisite knowledge required to complete a homework assignment prior to the "assigned" date.** This means that there is no reason not to begin a homework assignment as soon as it is assigned. Starting a problem the night before it is due a recipe for failure.

Policies to keep in mind

- **You may use any source at your disposal**—paper, electronic, or human—but you ***must*** cite ***every*** source that you use, and you ***must*** write everything yourself in your own words. See the academic integrity policies on the course web site for more details.
- **Being able to clearly and concisely explain your solution is a part of the grade you will receive.** Before submitting a solution ask yourself, if you were reading the solution without having seen it before, would you be able to understand it within two minutes? If not, you need to edit. Images and flow-charts are very useful for concisely explain difficult concepts.

See the course web site (https://ecealgo.com/fa24) for more information.

If you have any questions about these policies,
please don't hesitate to ask in class, in office hours, or on Piazza.

1. Solve the following recurrence relations. For parts (a) and (b), give an exact solution. For parts (c) and (d), give an asymptotic one. In both cases, justify your solution.

   (a) $A(n) = A\left(\frac{1}{2}n\right) + n^3$

   (b) $B(n) = 2B\left(\frac{1}{4}n\right) + \sqrt{n}$

   (c) $C(n) = C\left(\frac{1}{3}n\right) + C\left(\frac{2}{3}n\right) + O(n)$

   (d) $D(n) = D\left(\frac{1}{15}n\right) + D\left(\frac{1}{10}n\right) + 2D\left(\frac{1}{6}n\right) + \sqrt{n}$

2. Suppose you are given a sorted sequence of distinct integers $a = [a_1, a_2, \ldots, a_n]$ drawn from 1 to m where $n < m$. Give an $O(\lg(n))$ algorithm to find the *smallest* integer $\leq m$ that is not present in a.

3. Let $\boldsymbol{A}, \boldsymbol{B}$ be two $n \times n$ matrices of real numbers. In your Linear Algebra class, you might've learned the brute-force algorithm to compute $\boldsymbol{AB}$ in $O(n^3)$ time. However, using a similar strategy to Karatasuba's algorithm adapted to matrices, a mathematician named Volker Strassen invented a divide-and-conquer algorithm to perform the multiplication in $O(n^{2.8074})$ time, dutifully named **Strassen's Algorithm**. From now on, assume you're given a black box, Strassen($\boldsymbol{A}, \boldsymbol{B}$), which computes the matrix product of two matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n \times n}$. We assume that multiplication and addition of two real numbers is constant.

   Note: No Linear Algebra knowledge should be used to solve this problem

   (a) Suppose you needed to hand-compute the "power of 2" matrix power $\boldsymbol{A}^{2^k}$ for some matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ and $k \in \mathbb{N}$. Give an efficient algorithm to solve this problem and determine it's time complexity in terms of $m = 2^k$, that is in terms of the power on the matrix $\boldsymbol{A}$.

   (b) How would you modify this to compute $\boldsymbol{A}^m$ for any $m \in \mathbb{N}$, that is compute any positive integer matrix power? Would this change the time complexity?

   (c) Say your friend George was able to magically calculate the matrix product AB using 6 matrix multiplications of $\frac{n}{2} \times \frac{n}{2}$ matrices and 37400 additions of $n \times n$ matrices. What's the runtime of his algorithm? (*Note*: It takes $O(n^2)$ to calculate $\boldsymbol{A} + \boldsymbol{B}$).

   (d) George coded a correct version of the algorithm in Python, but when he benchmarked it against Strassen's algorithm he found his implementation was much slower. Why might that be?

4. We are given a array of n steel rods of integer length where the i^{th} piece has length $L[i]$. We seek to cut them so that we end up with k pieces of exactly the same length, in addition to other fragments. And we want this collection of k pieces to be as large as possible.

   For instance, the largest collection ($k = 4$) pieces you can get from the collection: $L = \{10, 6, 5, 3\}$ is 5.

   Give a correct and efficient algorithm that, for a given L and k, returns the maximum possible length of the k equal pieces cut from the initial n sticks.