AAD Question Bank - 1

Part I

August 2025

Introduction

This problem set is ungraded and meant purely for practice. Each problem carries 100 points, though some may be split into parts. A few problems include optional bonus follow-ups to make them more interesting.

The problems have been shuffled and do not include topic tags, so that you can approach them without a bias. If you encounter any doubts, feel free to reach out during office hours. In case of typos or unclear statements, please let us know.

In exam, you might be asked to "give an algorithm" to solve a problem. Your write-up should take the form of a short essay. Start by defining the problem you are solving and stating what your results are. Then provide:

1. A description of the algorithm in English and, if helpful, pseudo-code.
2. A proof (or proof sketch) for the correctness of the algorithm.
3. An analysis of the running time.

We will give full credit only for correct solutions that are described clearly.

Computability Theory

Decidability and Recognizability

Prove the following questions:

1. Prove that $A_{DFA} = \{L \mid M(L) \text{ accepts } L\}$ is decidable
2. Is $A_{TM} = \{\langle M, w\rangle \mid M \text{ accepts } w\}$ decidable? Is it recognizable?
3. Denote the complement of a language L as $\overline{L}$, where $\overline{L}$ rejects whatever L accepts and vice-versa. If both L and $\overline{L}$ are recognizable, prove that L is decidable.
4. Is $L = \{\langle M\rangle \mid M \text{ accepts the empty string } \epsilon\}$ decidable? Is it recognizable?
5. Prove that $L = \{\langle M\rangle \mid M \text{ accepts } \langle M\rangle\}$ is Turing-recognizable, but $\overline{L}$ is not Turing-recognizable.

Reducibility

Prove the following questions:

1. Reduce A_{TM} to the language $X = \{\langle M\rangle \mid L(M) \text{ is finite}\}$ to show that X is undecidable.
2. Reduce A_{TM} to language $\text{PAL} = \{\langle M\rangle \mid M \text{ accepts some palindrome}\}$ to show PAL is undecidable.
3. Reduce A_{TM} to language $L_{1000} = \{\langle M\rangle \mid M \text{ accepts a string of length } 1000\}$ to show L_{1000} is undecidable.

Problem 1

Use the divide-and-conquer integer multiplication algorithm to multiply the two binary integers `10011011` and `10111010`.

Problem 2

Suppose you are choosing between the following three algorithms:

- Algorithm A solves problems by dividing them into five subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.
- Algorithm B solves problems of size n by recursively solving two subproblems of size $n-1$ and then combining the solutions in constant time.
- Algorithm C solves problems of size n by dividing them into nine subproblems of size $n/3$, recursively solving each subproblem, and then combining the solutions in $O(n^2)$ time.

What are the running times of each of these algorithms (in big-O notation), and which would you choose?

Problem 3

Solve the following recurrence relations and give a Θ bound for each of them.

1. $T(n) = 2T\left(\frac{n}{3}\right) + 1$
2. $T(n) = 5T\left(\frac{n}{4}\right) + n$
3. $T(n) = 7T\left(\frac{n}{7}\right) + n$
4. $T(n) = 9T\left(\frac{n}{3}\right) + n^2$
5. $T(n) = 8T\left(\frac{n}{2}\right) + n^3$
6. $T(n) = 49T\left(\frac{n}{25}\right) + n^{3/2}\log n$
7. $T(n) = T(n-1) + 2$
8. $T(n) = T(n-1) + n^c, \quad$ where $c \geq 1$ is a constant
9. $T(n) = T(n-1) + c^n, \quad$ where $c > 1$ is some constant
10. $T(n) = 2T(n-1) + 1$
11. $T(n) = T(n-1) + 1$

Problem 4

What is the FFT of $(1, 0, 0, 0)$? What is the appropriate value of ω in this case? And of which sequence is $(1, 0, 0, 0)$ the FFT. Repeat for $(1, 0, 1, -1)$.

Problem 5

Suppose that you have a $\Theta(n^\alpha)$-time algorithm for squaring $n \cdot n$ matrices, where $\alpha \geq 2$. Show how to use that algorithm to multiply two different $n \cdot n$ matrices in $\Theta(n^\alpha)$ time.

Problem 6

This problem illustrates how to do the Fourier Transform (FT) in modular arithmetic, for example, modulo 7.

(a) There is a number ω such that all the powers $\omega, \omega^2, \dots, \omega^6$ are distinct (modulo 7). Find this ω, and show that

$$\omega + \omega^2 + \cdots + \omega^6 = 0.$$

(Interestingly, for any prime modulus there is such a number.)

(b) Using the matrix form of the FT, produce the transform of the sequence $(0, 1, 1, 1, 5, 2)$ modulo 7; that is, multiply this vector by the matrix $M_6(\omega)$, for the value of ω you found earlier. In the matrix multiplication, all calculations should be performed modulo 7.

(c) Write down the matrix necessary to perform the inverse FT. Show that multiplying by this matrix returns the original sequence. (Again all arithmetic should be performed modulo 7.)

(d) Now show how to multiply the polynomials $x^2 + x + 1$ and $x^3 + 2x - 1$ using the FT modulo 7.

Problem 7

A $k - way$ merge operation. Suppose you have k sorted arrays, each with n elements, and you want to combine them into a single sorted array of kn elements. (a) Here's one strategy: merge the first two arrays, then merge in the third, then merge in the fourth, and so on. What is the time complexity of this algorithm, in terms of k and n? (b) Give a more efficient solution to this problem, using divide-and-conquer.

Problem 8

Write pseudocode for Strassen's algorithm

Problem 9

What is the largest k such that if you can multiply $3 \cdot 3$ matrices using k multiplications, then you can multiply $n \cdot n$ matrices in $o(n^{lg 7})$ time? What is the running time of this algorithm.

Problem 10

CLRS Problem 4-6 Chip Testing.

Problem 11

Professor F. Lake tells his class that it is asymptotically faster to square an n-bit integer than to multiply two n-bit integers. Should they believe him?

Problem 12

The square of a matrix A is its product with itself, AA.

1. Show that five multiplications are sufficient to compute the square of a $2 \cdot 2$ matrix.
2. What is wrong with the following algorithm for computing the square of an n $\times$ n matrix?

   "Use a divide-and-conquer approach as in Strassen's algorithm, except that instead of getting 7 subproblems of size $n/2$, we now get 5 subproblems of size $n/2$ thanks to part 1. Using the same analysis as in Strassen's algorithm, we can conclude that the algorithm runs in time $O(n^{log_2 5})$."

Problem 13

An array $A[1 \dots n]$ is said to have a *majority element* if more than half of its entries are the same. Given an array, the task is to design an efficient algorithm to tell whether the array has a majority element, and, if so, to find that element. The elements of the array are not necessarily from some ordered domain like the integers, and so there can be no comparisons of the form "is $A[i] > A[j]$?". (Think of the array elements as GIF files, say.) However you *can* answer questions of the form: "is $A[i] = A[j]$?" in constant time.

(a) Show how to solve this problem in $O(n \log n)$ time. *(Hint: Split the array A into two arrays A_1 and A_2 of half the size. Does knowing the majority elements of A_1 and A_2 help you figure out the majority element of A? If so, you can use a divide-and-conquer approach.)*

(b) Can you give a linear-time algorithm? *(Hint: Here's another divide-and-conquer approach:*

- Pair up the elements of A arbitrarily, to get $n/2$ pairs
- Look at each pair: if the two elements are different, discard both of them; if they are the same, keep just one of them

Show that after this procedure there are at most $n/2$ elements left, and that they have a majority element if and only if A does.)

Problem 14

You are given an array of n elements, and you notice that some of the elements are duplicates; that is, they appear more than once in the array. Show how to remove all duplicates from the array in time $O(n \log n)$.

Problem 15

You are given an infinite array $A[\cdot]$ in which the first n cells contain integers in sorted order and the rest of the cells are filled with ∞. You are *not* given the value of n. Describe an algorithm that takes an integer x as input and finds a position in the array containing x, if such a position exists, in $O(\log n)$ time. (If you are disturbed by the fact that the array A has infinite length, assume instead that it is of length n, but that you don't know this length, and that the implementation of the array data type in your programming language returns the error message ∞ whenever elements $A[i]$ with $i > n$ are accessed.)

Problem 16

Given a sorted array of distinct integers $A[1, \dots, n]$, you want to find out whether there is an index i for which $A[i] = i$. Give a divide-and-conquer algorithm that runs in time $O(\log n)$.

Problem 17

Consider the task of searching a sorted array $A[1 \dots n]$ for a given element x: a task we usually perform by binary search in time $O(\log n)$. Show that any algorithm that accesses the array only via comparisons (that is, by asking questions of the form "is $A[i] \leq z$?"), must take $\Omega(\log n)$ steps.

Problem 18

You are given two sorted lists of size m and n. Give an $O(logm + logn)$ time algorithm for computing the k^{th} smallest element in the union of the two lists.

Bonus(+50 Points): Can we do better than this?

Problem 19

You are given two multisets of integers:

- A set of *apples* $A = \{a_1, a_2, \dots, a_n\}$,
- A set of *bananas* $B = \{b_1, b_2, \dots, b_m\}$,

where each element lies in the range $[1, k]$.
Define the *pair sum frequency function* $f : \{2, 3, \dots, 2k\} \to \mathbb{N}$ as

$$f(w) = \left|\{(a, b) \in A \times B \mid a + b = w\}\right|.$$

That is, $f(w)$ counts the number of pairs consisting of one apple and one banana whose weights sum to exactly w.

Problems

1. Prove that computing $f(w)$ reduces to a convolution of two sequences.
2. Design an $O(k \log k)$-time algorithm for computing $f(w)$ for all $w \in \{2, \dots, 2k\}$.
3. Compare the efficiency of the FFT-based approach with the naïve $O(nm)$ algorithm under the given constraints.

Constraints.

- $1 \leq k, n, m \leq 2 \cdot 10^5$,
- $1 \leq a_i, b_j \leq k$.

Naïve Approach. A brute-force algorithm that enumerates all nm pairs is too slow for the given bounds.

Hint 1. Use FFT.

Hint 2. Let $F[i]$ denote the number of apples of weight i, and $G[i]$ the number of bananas of weight i. Then

$$f(w) = \sum_{i=1}^{k} F[i] \cdot G[w - i].$$

This corresponds to the *discrete convolution* $F * G$, which can be computed in $O(k \log k)$ time using the Fast Fourier Transform (FFT).

Problem 20

You are given a string $s = s_1 s_2 \dots s_n$ consisting only of the characters $\{A, B\}$, where n is the length of the string.

Problems

1. Give a correct algorithm to compute the number of k-inversions for all $k \in \{1, \dots, n - 1\}$ in a string s.
2. Analyze the running time of your algorithm.
3. Can your algorithm be improved asymptotically compared to the naive $O(n^2)$ solution? Prove your answer.

For an integer k, a pair of indices (i, j) with $1 \leq i < j \leq n$ is called a *k-inversion* if and only if

$$s[i] = B, \quad s[j] = A, \quad \text{and } j - i = k.$$

For example, the string $s =$ `BABA` has:

- two 1-inversions,
- one 3-inversion,
- and no 2-inversions.

Constraints.

- $1 \leq n \leq 10^6$,
- s contains only the characters `A` and `B`.

Problem 21

In this problem we will confirm that Fibonacci sequence grows exponentially fast and obtain some bounds on its growth.

(a) Use induction to prove that $F_n \geq 2^{0.5n}$ for $n \geq 6$.
(b) Find a constant $c < 1$ such that $F_n \leq 2^{cn}$ for all $n \geq 0$. Show that your answer is correct.
(c) What is the largest c you can find for which $F_n = \Omega(2^{cn})$?
Where F_n is the n^{th} fibonacci number in the sequence defined by the recurrence $F_n = F_{n-1} + F_{n-2}$.

Problem 22

Part One - 10 points

You are given a number $1 \leq N \leq 100$, Design an algorithm to find the n^{th} Fibonacci number.

Part Two - 90 points

For $1 \leq N \leq 10^{100,000}$ Design an algorithm to find the N^{th} Fibonacci number, which is efficient[1].

[1]runs in < 5 seconds of time limit on a standard intel architecture.