




## CSIE 2136: Algorithm Design and Analysis (Fall 2023)

### Midterm

Time: 14:\_\_\_ 17:\_\_\_ (150 minutes), November 2 2023

## Instructions

- This is a 150-minute closed-book exam. There are 8 problems worth a total of 123 points. If your raw score exceeds 100, it will be capped at 100.
- Please write clearly and concisely; avoid giving irrelevant information.
- You are allowed to use basic data structures (limited to (dynamic) array, stack, queue, deque, heap, balanced search tree, and (doubly) linked list) without writing their implementation details. In addition, you can assume that sorting  $N$  numbers runs in  $O(N \log N)$ .
- Please use the assigned answer sheets for each problem. For each page, please write down **your name** and **student ID** on every paper. You will get **4 points** for complying with this policy.
- You **do not** need to prove the correctness and time complexity of your algorithms unless a problem requires you to do so.
- Happy Halloween! The symbol  represents a hint, or sometimes just a fun fact.
- The symbol  represents an **auto-scoring** relationship. For example,  $3 \rightarrow 2$  means you will automatically get the score of subproblem 2 if you complete subproblem 3.
- The symbol  shown on the answer sheet means that it may be graded in a fancy way like auto judging by AI. Therefore, please write down your answer clearly with a sufficiently large font size.

## Problem Outline

- Problem A - In-class Questions (20 points)
- Problem B - Jack of All Trades (15 points)
- Problem C - Keep Your Words (10 points)
- Problem D - Michael's Magical Chocolate (15 points)
- Problem E - Lineland (20 points)
- Problem F - Colonel BlottO (15 points)
- Problem G - Power of Hanoi (20 points)
- Problem H - Never Gonna Give You Up (4 points)
- Name and ID on each page (4 points)

## Problem A - In-class Questions (20 pt, 4 pt each)

In this problem, no further explanation about your answers is needed.

**For multiple choices, please write one capital character (A, B, C ...) on the answer sheet, other answers will not be accepted.**

**For True or False questions, please write O(true) or X(false) on the answer sheet, other answers will not be accepted.**

1.

Use the master theorem to solve  $T(n) = 4T(n/2) + n$ . Which of the following is true?

- (A)  $T(n) = \Theta(n^2)$
- (B)  $T(n) = \Theta(n)$
- (C)  $T(n) = \Theta(n \log n)$
- (D)  $T(n) = \Theta(n^2 \log n)$

2.

If we want to prove  $f(n)$  is not in  $O(g(n))$ , what should we prove?

- (A) There exist  $c, n_0 > 0$  such that for all  $n > n_0$ ,  $f(n) > cg(n)$
- (B) For all  $c > 0$ , there exist  $n > 0$  such that  $f(n) > cg(n)$
- (C) For all  $n > 0$ , there exist  $c > 0$  such that  $f(n) > cg(n)$
- (D) There exist  $c, n_0 > 0$  such that  $f(n_0) > cg(n_0)$
- (E) None of the above.

3.

Recall that we've learned to use Dynamic Programming to compute the Fibonacci sequence. Alternatively, one can implement it using pure recurrence as follows. What's the issue in the code below?

```
F(n):  
    if n < 2 :  
        return 1  
    return F(n - 1) + F(n - 2)
```

- (A) Computing overlapping subproblems many times.
- (B) Did not define the base case.
- (C) Computing optimal substructures many times.
- (D) Did not combine the solution in the subproblems.

4.

(O/X) Divide-and-conquer algorithms will always output incorrect answers if the problem has overlapping subproblems.

5.

Your friend presents an incorrect proof of the optimal substructure of the longest simple path problem. Point out which step is wrong in your friend's argument: (If you think there are multiple errors, choose the earliest step.)

- (A) Proof by the Cut-and-Paste argument
- (B) Suppose  $P_{ac} + P_{cb}$  is a longest path between two nodes  $A$  and  $B$  (through an intermediate node  $C$ )
- (C) Suppose there exists another path  $P'_{ac}$  between  $A$  and  $C$  that is longer than  $P_{ac}$ .
- (D) Cut  $P_{ac}$  and paste  $P'_{ac}$  to form a longer simple path  $P'_{ac} + P_{cb}$  between  $A$  and  $B$
- (E) The existence of a simple path longer than  $P_{ac} + P_{cb}$  leads to a contradiction.
- (F) Thus,  $P_{ac}$  must be a longest simple path between  $A$  and  $C$ . In other words, the longest simple path problem has the optimal-substructure property.

## Problem B - Jack of All Trades (15 pt, 5 pt each)

1.

If  $T(n) = 2T(n-1) + 1$ , prove that  $T(n) = \Theta(2^n)$ .

2.

Analyze the time complexity of the code below. You can assume that `max_crossing_subary` runs in  $\Theta(n)$  time.

🐞 Do you know how many times `max_subary` is called in this function?

```
max_subary(left, right, arr):
    if left == right : return arr[left]
    mid = (left + right)/2
    left_max = max_subary(left, mid)
    right_max = max_subary(mid + 1, right)
    dc_max = max(max_subary(left, mid),
                 max_subary(mid + 1, right))
    return max(max_crossing_subary(left, mid, right),
               dc_max)
```

3.

If  $T(n) = nT(\frac{n}{2}) + n$ , prove that  $T(n) = O(2^n)$ .

🐞 I heard that there's a method called substitution method.

## Problem C - Keep Your Words (10 pt, 5 pt each)

Given an integer  $K$  (possibly being negative) and an array containing  $n$  integers  $s_1, s_2, \dots, s_n$ . we call an integer pair  $(i, j)$  a **good pair** if  $1 \leq i < j \leq n$  and  $s_j - s_i \geq K$ .

### 1. Déjà Vu

Design an algorithm to find the number of good pairs that runs in  $O(n \log n)$ .

For example, if  $s = [4, 8, 7, 6, 3]$  and  $K = 2$ , the good pairs are  $(1, 2), (1, 3), (1, 4)$ , thus the number of good pairs is 3.

🐭 you only have to count the number of good pairs instead of listing out all of them.

### 2. Sense of Familiarity

Design an algorithm to find the largest  $j - i$  among all the good pairs that runs in  $O(n \log n)$ .

For example, if  $s = [4, 8, 7, 6, 3]$  and  $K = 2$ , the good pairs are  $(1, 2), (1, 3), (1, 4)$ , thus the largest  $j - i$  among all the good pairs is  $4 - 1 = 3$

## Problem D - Michael's Magical Chocolate (7+8 pt)

Michael has  $n$  magical chocolate beans, which are placed on a table. The  $i$ -th chocolate bean is at  $(x_i, y_i)$ .  $x_i, y_i \in \mathbb{R}, \forall n \in [1 \dots n]$ .

For any two chocolate beans, they will generate a **magical value**, which can be computed by

$$|x_i - x_j|^3 + |y_i - y_j|^3.$$

Can you help him find the two chocolate beans that **minimizes** the magical value?

🐭 As far as I know, *Michael Ian Shamos* and *Dan Hoey* first proposed the closest pair algorithm.

**For every subproblem, you can use the results from all previous subproblems even if you didn't manage to solve them.**

### 1. Golden Ratio

Suppose that no two points have a magical value smaller than  $r^3$ .

Prove that there exists some constant  $c \in \mathbb{N}$ , such that for any square with side length  $r$ , there are at most  $c$  magical chocolate beans inside the square.

### 2. M&M's Chocolate

Design an algorithm that finds the two chocolate beans that **minimizes** the magical value in  $O(n \log n)$ . Briefly explain why the time complexity is correct.

## Problem E - Lineland (20 pt, 4 pt each)

Lineland is a one-dimensional world. **Sphere**, the lord of the world, wants to summon some **segments**. However, segments are territorial animals, more precisely,

- a segment  $(x_i - r_i, x_i + r_i)$  is represented by a **center**  $x_i$  and **radius**  $r_i > 0$
- any two summoned segments cannot overlap (**excluding endpoint**)

Additionally, in this problem, assume that

- $x_1 < x_2 < \dots < x_n$

Sphere can choose to summon some segments, and she (note: sphäre is feminine in German) wants to maximize the number of segments she can summon. Can you help her?

For example, If there are three segments  $(x_i, r_i) = [(3, 1), (4, 2), (7, 3)]$ , then the maximum number of segments you can choose is the first segment and the third segment, 2 segments in total.

**For every subproblem, you can use the results from all previous subproblems even if you didn't manage to solve them.**

♠♠♠ Some of the characters here are inspired by *Flatland*.

### 1. Geometry

Prove that if  $i < j$  and  $x_i - r_i \geq x_j - r_j$ , then the maximum number of segments can be summoned remains unchanged if we remove (i.e. never choose) segment  $j$ .

♠♠♠ Also, if  $x_i + r_i \geq x_j + r_j$ , then we'll never choose segment  $i$ .

### 2. Post Meridiem

Following the previous problem, design an algorithm to remove some elements so that for all  $i < j$ ,  $x_i - r_i < x_j - r_j$ , without hurting the optimality. Your algorithm should run in  $O(n)$ .

### 3. Metric System

Assume that for all  $i < j$ , both  $x_i - r_i < x_j - r_j$  and  $x_i + r_i < x_j + r_j$  holds. (that is, the sequences  $\{x_i - r_i\}$  and  $\{x_i + r_i\}$  are both strictly increasing)

Define  $f(i) < i$  as the maximum index such that segment  $f(i)$  and  $i$  do not overlap. Prove that  $f(i) \leq f(i + 1)$  for all  $i < n$ .

### 4. Teleport

Follow the previous problem. Design an algorithm to calculate  $f(i)$  for all  $i$  within  $O(n)$ .

### 5. Happy Ending

Please design an  $O(n)$  algorithm to calculate the maximum number of segments that can be summoned. Briefly explain why your algorithm is correct.

## Problem F - Colonel Blotto (5+10 pt)

👤 auto scoring: 2 → 1

One day, Blotto is transported into an isekai (異世界) and becomes a Colonel (上校) of an imaginary army. In this world, a **war** can be formulated as below:

- There are 2 players and  $k$  **battlefields**.
- Each player has  $k$  groups of soldiers. A player can assign **one** group to **each** battlefield.
- For the  $i$ -th battlefield, the player that has **strictly more** soldiers than the other will score  $p_i$  points.

However, as a protagonist (主角), Blotto has a superpower: he can predict how his opponents would place their soldiers!

🐭 In reality, Blotto game is a problem from game theory, which is also quite interesting.

### 1. Novice

- Blotto has  $k$  groups of soldiers (with size)  $a_1, a_2, \dots, a_k$ .
- Each battlefield is worth the same, i.e.  $p_1 = p_2 = \dots = p_k = 1$ .
- It is known that the opponent will put  $b_i$  soldiers on the  $i$ -th battlefield.

Please design an algorithm that calculates the maximum score Blotto can get in  $O(k \log k)$ .

🐭 A sample testcase is given below:

- $k = 5$
- $a = [2, 4, 6, 9, 8]$
- $b = [1, 2, 3, 6, 9]$

The answer is 4, which can be achieved by sending  $[2, 4, 6, 9, 8]$  soldiers.

### 2. Adventure

- Blotto has  $k$  groups of soldiers (with size)  $a_1, a_2, \dots, a_k$
- the  $i$ -th battlefield is worth  $p_i$
- it is known that the opponent will put  $b_i$  soldiers on the  $i$ -th battlefield.

Please design an algorithm that calculates the maximum score Blotto can get in  $O(k \log k)$ .

🐭 you will get 3 points if you give an  $O(k^2)$  algorithm.

🐭 A sample testcase is given below:

- $k = 5$
- $a = [2, 4, 6, 9, 8]$
- $b = [3, 3, 7, 3, 7]$
- $p = [5, 4, 3, 2, 1]$

The answer is 14, which can be achieved by sending  $[4, 6, 9, 8, 2]$  soldiers.

## Problem G - Power of Hanoi (5+5+10 pt)

In Hanoi, there are  $n$  towers on the road. The king of Hanoi wants to build some altars, and he decided to turn exactly  $k \leq n/2$  ( $k$  is a given number) towers into altars. However, he cannot build adjacent altars due to an ancient taboo.

If the  $i$ -th tower is turned into an altar, it will produce  $a_i$  power. The power of the kingdom is the **minimum** power produced among all altars. People want to maximize the power of the kingdom.

For example, if  $k = 3$  and  $p = [2, 7, 9, 5, 4, 6]$ , then by selecting  $[7, 5, 6]$ , we'll maximize the power of the kingdom, which is  $\min(7, 5, 6) = 5$ .

### 1. Hard Work

Define  $\text{dp}[i][j]$  as the power of kingdom if we can only build exactly  $j$  altars chosen from  $1, 2, \dots, i$ . Can you give the **base case** and the **transition function** (should be  $O(1)$ ) of  $\text{dp}$ ? Set the value to  $-\infty$  if it's not possible to build such number of altars.

### 2. Being Fast

The king of Hanoi wants to know if it's possible to make the power of the kingdom at least  $p$ . Can you design an algorithm that checks if it's possible in  $O(n)$ ?

### 3. Aggregating Power

Back to the original problem. Can you design an algorithm that maximizes the power of the kingdom in

- (a)  $O(n \log n)$  (5 points)
- (b)  $O(n)$  (5 points, bonus)

**Please specify which time complexity you intended to achieve.**

🐞 We recommend trying to work on the bonus problem after all other problems are solved.

## Problem H - Never Gonna Give You Up (4 pt)

How's your ADA experience so far? We have the tradition of letting students write feedback about the course in the exam.

Please write down at least **2** opinions about this course, which could be either

- things you like about this course
- things that you would like to see some changes
- any other opinions or words

🐞 You will **not** get full marks if you wrote fewer than 2 opinions, or your opinions are very random or contain almost no information (for example, if you wrote "classes are great"). More opinions are super welcome, although it might not grant you extra points.

# Appendix

## Asymptotic Notations

### 1. $\Theta$ -notation:

$$\Theta(g(n)) = \{f(n) : \text{there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ such that} \\ 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for all } n \geq n_0\}$$

### 2. $O$ -notation:

$$O(g(n)) = \{f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that} \\ 0 \leq f(n) \leq c g(n) \text{ for all } n \geq n_0\}$$

### 3. $o$ -notation:

$$o(g(n)) = \{f(n) : \text{for any positive constant } c > 0, \text{ there exists a constant} \\ n_0 > 0 \text{ such that } 0 \leq f(n) < c g(n) \text{ for all } n \geq n_0\}$$

**Master Theorem** Let  $a > 0$  and  $b > 1$  be constants, and let  $f(n)$  be a driving function that is defined and nonnegative on all sufficiently large reals. Let  $T(n)$  be defined on the non-negative integers by the recurrence

$$T(n) = aT(n/b) + f(n),$$

where  $aT(n/b)$  actually means  $a'T(\lfloor n/b \rfloor) + a''T(\lceil n/b \rceil)$  for some constant  $a', a'' \geq 0$  satisfying  $a = a' + a''$ , then  $T(n)$  has the following asymptotic bounds:

1. If  $f(n) = O(n^{\log_b a - \epsilon})$  for some constant  $\epsilon > 0$ , then  $T(n) = \Theta(n^{\log_b a})$ .
2. If there exist a constant  $k \geq 0$  such that  $f(n) = \Theta(n^{\log_b a} \log^k(n))$ , then  $T(n) = \Theta(n^{\log_b a} \log^{k+1}(n))$ .
3. If  $f(n) = \Omega(n^{\log_b a + \epsilon})$  for some constant  $\epsilon > 0$  and if  $af(n/b) \leq cf(n)$  for some constant  $c < 1$  and all sufficiently large  $n$ , then  $T(n) = \Theta(f(n))$ .