ShanghaiTech University

CS240 Algorithm Design and Analysis

Fall 2025

Problem Set 1

Due: 23:59, Oct. 8, 2025

1. Submit your solutions to the course Gradescope.
2. If you want to submit a handwritten version, scan it clearly.
3. Late homeworks submitted within 24 hours of the due date will be marked down 25%. Homeworks submitted more than 24 hours after the due date will not be accepted unless there is a valid reason, such as a medical or family emergency.
4. You are required to follow ShanghaiTech's academic honesty policies. You are allowed to discuss problems with other students, but you must write up your solutions by yourselves. You are not allowed to copy materials from other students or from online or published resources. Violating academic honesty can result in serious penalties.

Problem 1:

Arrange the following functions in order of increasing asymptotic growth rate.

- $f_1(n) = n^4 + \log n$
- $f_2(n) = n + [\log(n)]^4$
- $f_3(n) = n \log n$
- $f_4(n) = C_n^3$
- $f_5(n) = C_n^{n/2}$
- $f_6(n) = 2^n$
- $f_7(n) = n^{\log n}$

Hint: Use Stirling's approximation for factorials:

$$n! \sim \sqrt{2\pi n} \left(\frac{n}{e}\right)^n .$$

Problem 2:

Determine the time complexity of the pseudocode below and explain your reasoning.

```
def count_triples(arr, threshold):
    n = len(arr)
    count = 0

    def quicksort(arr, left, right):
        if left >= right:
            return
        pivot = arr[right]
        i = left
        for j in range(left, right):
            if arr[j] <= pivot:
                arr[i], arr[j] = arr[j], arr[i]
                i += 1
        arr[i], arr[right] = arr[right], arr[i]
        quicksort(arr, left, i - 1)
        quicksort(arr, i + 1, right)
    quicksort(arr, 0, n - 1)

    def upper_bound(arr, left, right, target):
        result = left - 1
        while left <= right:
            mid = (left + right) // 2
            if arr[mid] <= target:
                result = mid
                left = mid + 1
            else:
                right = mid - 1
        return result

    for i in range(n):
        for j in range(i + 1, n):
            target = threshold - (arr[i] + arr[j])
            k = upper_bound(arr, j + 1, n - 1, target)
            if k > j:
                count += (k - j)

    return count
```

Problem 3:

At ShanghaiTech University, there is a group of students, and the energy value of each student is represented by a positive integer in the array nums.

To accomplish a large team project, at least three students must work together. However, not every combination of students can successfully complete the project:

If you choose k students ($k \geq 3$), and arrange their energy values in non-decreasing order as

$$a_1 \leq a_2 \leq \cdots \leq a_k,$$

then the following condition must hold:

$$a_1 + a_2 + \cdots + a_{k-1} > a_k.$$

In other words, the most capable student's energy value cannot exceed the total energy values of all the other students.

If a team satisfies this condition, then they can successfully complete the project. In this case, the team's *total combat power* is defined as the sum of all members' energy values:

$$\text{Total Combat Power} = a_1 + a_2 + \cdots + a_k.$$

Return the maximum *total combat power* of a team that can be formed from nums.

Problem 4:

At noon-peak hour, a food-delivery platform still has n unassigned orders. Each order can be delivered by exactly one of two on-line riders:

- Rider A (a new employee) handling order i brings the platform a profit of profit1$[i]$ yuan;
- Rider B (a veteran) handling order i brings the platform a profit of profit2$[i]$ yuan.

To satisfy a company regulation, **Rider A must complete exactly k orders** today, while the remaining orders are all assigned to Rider B.

If a total of n orders are dispatched in this way, the platform's *total revenue* is defined as the sum of the profits generated by the two riders:

$$\text{Total Revenue} = \sum_{\text{orders to A}} \text{profit}_1[\cdot] + \sum_{\text{orders to B}} \text{profit}_2[\cdot].$$

Return the maximum *total revenue* the platform can obtain under the above requirement.

Problem 5:

ShanghaiTech University Library is hosting a reading-sharing event. Each book on the shelf has a *difficulty index*, and students want to select books that best match their reading ability.

Given:

- n: the number of books on the shelf
- $d = [d_1, d_2, \ldots, d_n]$: the difficulty indices of the books in shelf order
- a: the student's reading ability
- m: the number of books the student must select

Goal: Select m books such that:

1. Each selected book's difficulty is as close as possible to a.
2. The selection preserves the original shelf order.
3. If multiple books are equally close, choose the ones that appear earlier.
4. Need to be done with **Divide and Conquer**.

Problem 6:

Whispers tell of an ancient city gate protected by two formidable rows of runic stones. There are n stones in each row, $2n$ stones in total, each inscribed with a unique number from 1 to $2n$. The current arrangement of stones is called the *initial state*, while the inscription from the old manuscripts describes the *target state*. Only when the stones are rearranged into the target state can the sealed gate be opened.

Yansen can change the position of the stones through this operation:

"Select any adjacent 2×2 *block of stones and cast the ancient spell; the selected block will rotate as a whole by* $180°$*."*

Since casting a spell consumes energy, Yansen needs to cast as few spells as possible to open the gate.

Here is an example with $n = 3$:

$$\text{Initial State: } \begin{matrix} 3 & 2 & 1 \\ 4 & 5 & 6 \end{matrix} \qquad \text{Target State: } \begin{matrix} 5 & 6 & 3 \\ 2 & 1 & 4 \end{matrix}$$

Operation 1: Rotate the left 2×2 block by $180°$.

$$\begin{matrix} \mathbf{3} & \mathbf{2} & 1 \\ \mathbf{4} & \mathbf{5} & 6 \end{matrix} \xRightarrow{\text{rotate}} \begin{matrix} \mathbf{5} & \mathbf{4} & 1 \\ \mathbf{2} & \mathbf{3} & 6 \end{matrix}$$

Operation 2: Rotate the right 2×2 block by $180°$.

$$\begin{matrix} 5 & \mathbf{4} & \mathbf{1} \\ 2 & \mathbf{3} & \mathbf{6} \end{matrix} \xRightarrow{\text{rotate}} \begin{matrix} 5 & \mathbf{6} & \mathbf{3} \\ 2 & \mathbf{1} & \mathbf{4} \end{matrix}$$

The minimum number of operations required is 2.

Here is another example with $n = 2$:

$$\text{Initial State: } \begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix} \qquad \text{Target State: } \begin{matrix} 3 & 4 \\ 2 & 1 \end{matrix}$$

In this case, the initial state cannot be transformed into the target state.

Adventurer Yansen wishes to know:

1. Under what conditions can the initial state be transformed into the target state through a sequence of such operations?
2. If it can be transformed, what is the minimum number of operations required?

Constraint: Your algorithm must run in time complexity no worse than $O(n \log n)$.