Course Name: Algorithm Design and Analysis

Greedy

For problems 1-4, you should do at least the following things:

1. Modeling: how you analyse the problem;
2. Algorithm description: describe your algorithm in **natural language**;
3. Time complexity: provide the time complexity and explain the reasoning behind it;
4. Space complexity: provide the space complexity and explain the reasoning behind it.

---

1. Distributing Candy Game

   One day, the teacher decided to play a fun reward game with the n children in the class. Each child writes an integer on their left hand and an integer on their right hand, and the teacher also recorded integers on his own left and right hands. The teacher and the children then lined up, with the teacher standing at the front of the line.

   Once lined up, the number of candies each child receives is the product of the number on the left hand of all the people in front of him divided by the number on his own right hand, then rounded down.

   However, the teacher does not want any child to receive too many candies, so they asked for your help in rearranging the order of the line to minimize the number of candies received by the child who receives the most. Note that the teacher must always remain at the front of the line.

2. Array Partition

   Given an integer array nums of $2n$ integers, group these integers into n pairs $(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)$ such that the sum of $min(a_i, b_i)$ for all i is **maximized**. Return the maximized sum.

3. Delete Number Game

   Given a non-negative integer of length n (this number may have leading zeros), delete any $k(k < n)$ numbers of them, and the remaining numbers can form a new integer in order from left to right (the new number can have leading zeros). For the given n and k, find a way to delete the numbers so that the new number composed of the remaining numbers is minimal.

4. Ex. Container Balancing Operations

   There are n containers arranged in a row, each initially holding a certain number of items (possibly zero). You can perform multiple operations to equalize the number of items in all containers.

   In each operation, you can select any number of containers and move one item from each selected container to its adjacent container.

   Given an integer array *containers*, where $containers[i]$ represents the number of items in the i-th container, the task is to calculate the **minimum number of operations** required to make the number of items in all containers equal. If it is impossible to make all the containers have the same number of items through the operations, return -1.