Candy

EGOI 2023

Problem author: Yann Viegas.

1 The Problem

You are given an array $a_1, a_2, \ldots, a_N$ and two integers F and T. In a single operation, you are allowed to swap any two adjacent elements of the array. Find the minimum number of operations required so that the first F elements of the array sum to at least T.

2 Subtask 1: $N \leq 2, a_i \leq 100, T \leq 10^9$

Here, N can only take two values: either 1 or 2.

- Case $N = 1$: We cannot apply any operation and as $1 \leq F \leq N$ we know that $F = 1$. Thus it is enough to check whether or not $a_1 \geq T$
- Case $N = 2$: We can either swap the two numbers, or not swap them. We can just consider the two cases and see if one of them gives a correct solution.

Overall complexity: $O(1)$

Expected score: 6.

3 Subtask 2: $a_i \leq 1$

In this subtask, a_i is either 0 or 1 for all i. First we will try to find if we can reach the objective without minimising the number of swaps. Let's say there are X ones in the array. The largest possible sum would be achieved by moving all the ones at the beginning of the array. The sum of the first F elements of the array will then be $s = \min(\text{F},\text{X})$. If $s < T$, the answer is `NO`. Else, we can construct an answer by reaching a state where there are at least T ones at the beginning of the array. It is optimal to chose ones in increasing order of their indices in the array.

Expected score: 19 for just subtask 2, $6 + 19 = 25$ for subtasks 1 and 2.

4 Subtask 3: $N \leq 20$

In this subtask, $N \leq 20$ which suggests some sort of backtracking/bitmask solution.

We can split the array in two parts:

- the left part (first F elements)
- the right part (everything except the first F elements)

Fixing the set of the elements that will end in the left part will be enough. Let's say that their indexes (in the original state of the array) are:

$$l_0 < l_1 < \ldots < l_{F-1}$$

The minimum required number of swaps to move them to the left part is:

$$(l_0 - 0) + (l_1 - 1) + \ldots (l_{F-1} - (F-1))$$

Indeed, it is never useful to swap l_i and l_j when $i < j$. Furthermore, as l_0 has to end up in the first position, it needs to be swapped with all the elements to its left, same argument for l_1 which need to be swapped with all the elements with indexes in

One way of implementing this would be by iterating over bitmasks to iterate through all the subsets of $[0 \,..\, (N-1)]$
Overall complexity: $O(2^N \cdot N)$.

Expected score: $6 + 16 = 22$ (it solves subtasks 1 and 3).

5 Subtask 4: $a_i \leq 100$

The constraints of the problem suggest that we use DP. Also, the solution of subtask 3 presents clearly a process of "chosing/not chosing" elements which is definitely something that could be solved with DP.

The states are:

- The index of the integer we are currently considering
- How many elements we already selected to be in the left part
- The sum of the elements currently in the left part

So $dp[i][j][s]$ will give the minimum number of swaps required to select the first j integers of the left part while having only used the first i integers and such that their sum is exactly s.

The transitions are pretty simple:

- Either we choose the ith integer to be in the left part
- Or we don't choose it to be in the left part

We use the formula of subtask 3 to update the number of swaps while doing our transitions.

Overall complexity: $O(N^2 \cdot \max_{0 \leq i \leq N-1}(a_i))$

Expected score: $6 + 19 + 30 = 55$.

6 Full Task

Let's try to optimise the DP solution we got for the previous subtask. The thing that makes it slow is that the sum of the integers of the left part can be extremely high. On the other hand, the value that our DP computes is small. Indeed, by the formula we found in subtask 3, the number of swaps required to move all the selected integers to the left part is bounded by N^2. Thus we can change a bit our states:

- The index of the integer we are currently considering
- How many elements we already selected to be in the left part
- The minimum number of swaps required to move the selected elements to the first part

Given such a state, we would like to maximise the sum of the integers selected to be in the left part. Thus, $dp[i][j][s]$ will give the maximal sum we can reach by selecting the first j integers of the left part while having only used the first i integers and such that the minimum number of swaps required to move them to the left part is exactly s.

Overall complexity: $O(N^4)$

Expected score: 100