

## EDITORIAL OF PROBLEM BOUQUETS

### Subtask 1

Let us consider a given order and denote by  $cnt$  the number of flowers that can be used for it, and by  $ans$  – the required number of flowers in each bouquet. We need to find the smallest number  $ans$  for which  $C_{cnt}^{ans} \geq K$ . If this inequality has no solution, the order cannot be fulfilled. The constraints in this subtask are small enough that we can check all possible values for  $ans$  using the formula  $C_n^k = \frac{n!}{k!(n-k)!}$ .

Complexity:  $O(M \times N^2)$ .

### Subtask 2

When calculating combinations using the standard formula, very large intermediate values arise. For this reason, we will instead use the following recurrence relation:

$$C_n^k = C_{n-1}^{k-1} + C_{n-1}^k$$

In this way, we can construct Pascal's Triangle and obtain all combinations that fit within a 64-bit integer variable.

Complexity:  $O(N^2 + M \times N)$ .

### Subtask 3

Due to the larger constraints, standard integer variables cannot be used. Solving this subtask requires only implementing the addition and comparison operations for big numbers.

Complexity:  $O((N^2 + M \times N) \times \log_{10} K)$ .

### Subtask 4

A major drawback of the previous solution is that it uses too much memory. To address this issue, we can properly store all orders in advance and process them sorted by the number of all flowers that can be used for them. This way, there is no need to store the entire Pascal's triangle, and it is sufficient to keep only one of its rows – the one needed for processing the current order. Additionally, we can determine the number of flowers that can be used for a given order in constant time using a prefix array.

Complexity:  $O(M \times \log_2 M + (N^2 + M \times N) \times \log_{10} K)$ .

### Subtask 5

One of the properties of Pascal's triangle is that its rows are symmetric. We can consider only its first half, which is non-decreasing. This allows us to optimize the process of finding the answer for each order using binary search. Alternatively, we can apply the two pointers technique if, after sorting the orders by the number of flowers that can be used for each of them, we also sort by  $K$  in case of equality.

Complexity when binary search is used:  $O((N^2 + M \times \log_2 N) \times \log_{10} K)$ .

Complexity when pointers are used:  $O((N^2 + M \times \log_2 M) \times \log_{10} K)$ .

### Subtask 6

In this subtask, a single bouquet must be made for each order. An order can be fulfilled if the shop has at least one type of flowers whose number of petals falls within the required interval.

Complexity:  $O(N + M)$ .

### Subtask 7

We should optimize the operations with big numbers in the solution of subtask 5. For this purpose, we can represent the numbers in a numeral system with a base greater than 10. The author's implementation uses base  $10^{18}$ . In this way, resources are saved for 18 digits that fit into a single variable.

Complexity:  $O((N^2 + M \times \log_2 N) \times \log_{base} K)$  or  $O((N^2 + M \times \log_2 M) \times \log_{base} K)$ .

*Author: Georgi Petkov*