EGOI 2025 Editorial - IMO

Task Author: Eliška Macáková

July 18, 2025

This problem was inspired by a situation that happened in a certain local math competition. All the results, except one score of one contestant, who used an unusual solution for one of the problems, were published. When she requested the organizers to finally grade it, they argued it would not change her final rank anyway, as there was a big gap between her and the previous person in the ranklist.

The described solutions for the subtasks 1, 2, 3 are independent from the rest of the explanation. However, if you wish to understand the full solution, you should start reading from the solution of subtask 4, as it motivates the solutions for the remaining subtasks and defines some notions that will be reused.

Introduction

In this task, you are given the grading results of a competition per task and contestant, and want to reveal the minimum number of scores such that the correct ranking can be determined.

Test group 1: $N = M = 2, K = 1$

You can solve the subtask with a case analysis or a bruteforce.

Test group 2: $N = 2$

Fix how many scores are revealed for the first and second contestant. Notice that, for the contestant who is supposed to have better score, it is best to grade the problems where she has most points, while for the other contestant, it is best to grade the problems where she has the least amount of points. Check whether this results in a non ambigous comparison between the two contestants, and by checking all the possibilities for how many scores are revealed for the two contestants, find the optimal answer. This can be implemented in $\mathcal{O}(NM^2)$, but asymptotically slower implementations can pass as well.

Test group 3: $N \cdot M \leq 16$

Try all possibilities on which scores to hide and which scores to reveal, then compute the range of points each contestant can have, and check if their ordering is unique (so the ranges may overlap only at the endpoints, and even that only if the comparison according to the criteria in case of equal score evaluates correctly). Time complexity $\mathcal{O}(2^{N \cdot M} \cdot N^2 \cdot M)$.

Test group 4: $K = 1$

The scores of every contestant are either 0 or 1. Let $c_{i,0}$ be the number of 0s the ith contestant has and $c_{i,1}$ be the number of 1s the ith contestant has (observe that $c_{i,0} + c_{i,1} = M$ for every i). When you

decide to reveal y_i scores for the contestant i, the lower bound on her score, l_i, can be any of the numbers $\max(0, y_i - c_{i,0}) \ldots, \min(y_i, c_{i,1})$, and the upper bound on her score, r_i, is going to be equal to $l_i + (M - y_i)$.

This leads to a dynamic programming solution. Sort the contestants according to their final order, the states of the dynamic programming will be (i, S, x) where:

- i = first how many contestants (according to results) have we determined the revealed scores for
- S = lower bound on the score of ith contestant according to our revealed scores
- x = how many scores have we **hid** so far. The number of revealed scores is then simply $i \cdot M - x$.

and $dp[i][S][x] = 1$ if there's a way to do this and 0 otherwise. Transitions from $dp[i][S][x]$ are done by checking all possible l_{i+1}, r_{i+1} (according to the reasoning above), now if $r_{i+1} < S$ or $r_{i+1} = S$ and contestant that should be $i+1$st in the results had higher index original than the one who should be ith, you can update $dp[i+1][l_i][x + (r_i - l_i)]$ ($r_{i+1} - l_{i+1}$ is the number of scores that we should hide for $i+1$ contestant if we want to have the lower bound l_{i+1} and upper bound r_{i+1} on her score, according to the revealed information).

To speed this up, notice that for a fixed i, x, the higher S we have, the better, so we can instead calculate $dp[i][x] = S$, meaning, if we have chosen the scores for first i contestants, and we chose to hide x of them, what is the maximum possible lower bound on score of ith contestant we might have. Now for the transitions, we can first fix the number of scores to hide for $i+1$st contestant, h_{i+1}, and then calculate highest r_{i+1} we can have so that $i+1$ is certainly below i, and then try to update $dp[i+1][x + h_{i+1}]$ with $l_{i+1} = r_{i+1} - h_{i+1}$.

How many scores can we hide? Since we're hiding $r_i - l_i$ scores for contestant i and we know that $M \geq l_1 \geq r_1 \geq l_2 \geq r_2 \geq l_3 \geq \ldots \geq l_n \geq 0$ should hold, then we can get that the total number of hidden scores $= \sum_i r_i - l_i \leq M$. So the number of scores we can hide is asymptotically lower than the total number of scores, which is the motivation behind doing dynamic programming not on the number of revealed scores, but on the number of hidden scores instead.

So we have $N \cdot M$ states and M transitions for each, so the total time complexity is $O(N \log N + NM^2)$ (first factor comes from sorting the contestants by their final score first and original index second in the beginning).

Test group 5: $N \leq 10,000$ and $M, K \leq 10$

Similarly to previous subtask, the main idea is to precalculate for every contestant i the possible lower and upper bounds on their score and then do $dp[i][X]$.

Let's analyze the situation for one contestant, i, and find out what can be the lower and upper bounds on her score. First of all, observe that $r_i = l_i + h_i \cdot K$, where l_i, r_i are the lower and upper bound on the score of contestant i and h_i is the number of hidden scores she has (l_i represents the situation where we grade all the remaining h_i problems with 0 and r_i represents the situation where we grade all the remaining h_i problems with the full score, K).

We can introduce the following dynamic programming to help us find the possible l_i and r_i - let

- j be the prefix of her scores considered (for each score, we are trying to decide whether to hide or reveal it).
- S be the sum of the revealed scores so far
- x be the number of hidden scores so far

Then $t_i[j][S][x]$ is 1 if you can achieve these parameters and 0 otherwise. The transitions from a reachable state $t_i[j][S][x]$ can be done in the following manner - if she has s_{j+1} points on her $j+1$st problem, then if we hide this score, we go to the state $t_i[j+1][S][x+1]$ and otherwise, we go to the state $t_i[j+1][S+s_{j+1}][x]$.

In the end, the bounds l_i, r_i are reachable if and only if $t_i[M][l_i][\frac{r_i - l_i}{K}] = 1$. So we can iterate through all reachable $t_i[M][S][x]$ to find all possible l_i, r_i.

This precalculation can be done in the time complexity $O(M^3K)$ for each i, because we have $O(M \cdot MK \cdot M)$ states and $O(1)$ transitions from each.

Now to solve the original problem, we may adapt the dynamic programming from the previous subtask. The states will be still the pairs (i, x) of the length of the prefix and the number of hidden scores, the values will be the maximum lower bound S we may have, and to transition from $dp[i][x] = S$, fix h_{i+1} and l_{i+1} such that the corresponding r_{i+1} is not too high (less or less than equal to S) and then try to update $dp[i+1][x+h_{i+1}]$ with l_{i+1}. In the beginning, initialize $dp[0][0]$ with $M \cdot K$ and everything else with $-\infty$.

Once again, notice that since in the optimal solution it holds that $M \cdot K \geq l_1 \geq r_1 \geq l_2 \geq r_2 \geq l_3 \geq \ldots \geq l_n \geq 0$, and the number of hidden scores is equal to $\sum_{j \leq i} \frac{r_j - l_j}{K} \leq \frac{M \cdot K}{K}$, so the second dimension of the dynamic programming, x, can go only up to M.

Then the number of states is $\mathcal{O}(N \cdot M)$ and each of them leads to $\mathcal{O}(MK \cdot M)$ transitions, so the total time complexity of this solution is $\mathcal{O}(N \log N + NM^3K)$.

Test group 6: no further constraints

To speed up the solution for the test group 5, we need to make one more optimization. Let m_i be the score of ith contestant if all her scores were revealed. Notice that in the optimal solution, for all i, it must hold that $m_{i+1} \leq l_i$, otherwise, if contestant i would get all 0s for her remaining problems and contestant $i+1$ would get her true scores, they would be incorrectly ordered.

We can then modify the dynamic programming t_i - let

- j be the prefix of scores of contestant i considered (for each score, we are trying to decide whether to hide or reveal it).
- S **be** m_i **minus the sum of her hidden scores so far**
- x be the number of hidden scores so far

then $t_i[j][S][x]$ is 1 if and only if it is possible to reach this state. And from this state you can do transitions to $t_i[j+1][S - s_{j+1}][x+1]$ if you decide to hide the $j+1$st score with the value s_{j+1} and $t_i[j+1][S][x]$ otherwise. Initialize $t_i[0][m_i][0] = 1$ in the beginning and let everything else be equal to 0. The reachable pairs l_i, r_i can still be deduced from $t_i[M]$ just like before.

Now because of the condition that $m_{i+1} \leq l_i$ in any solution, it only makes sense to consider pairs l_i, r_i with $m_{i+1} \leq l_i$, so we don't have to consider states of t_i with $S < m_{i+1}$.

Therefore the time complexity of calculation of t_i is now $\mathcal{O}(M \cdot (m_i - m_{i+1}) \cdot M)$ with the right implementation, summing up to $\mathcal{O}(M^3K)$ for all i.

Now to finish the problem, you can do the same dynamic programming as in test group 5, but you won't consider l_i lower than m_{i+1} when updating for every i, so the total time complexity of this step will be $\mathcal{O}\left(\sum_i (m_i - m_{i+1}) \cdot M^2\right) = \mathcal{O}(M^3K)$.

The total time complexity of this solution is $\mathcal{O}(N \log N + M^3K)$, enough to get full points under the constraints $N \leq 20,000$, $M, K \leq 100$.