My USACO Silver Editorials

Target Practice II

We can approach the problem by first making the observation that given a target, the top right corner has to be hit by a negative slope, while the bottom left corner has to be hit by a positive slope. If there are not enough positive or negative slopes, we know that the problem is impossible.

For the left two points, we can observe that the most optimal strategies would have all the remaining negative slopes be given to the bottom-most points, while all the positive slopes would be given to the rest top points.

Given these observations, we have boiled the problem down to an allocation one, we now know which points should have a positive and negative, but we don't know exactly how to allocate slopes to our points. It's important to note that there is actually no way of comparing two points without a given slope, this is because let say we have p_1, p_2 be at $(a, b), (c, d)$ respectively. Assuming our slope, m, is positive, we get that the y intercepts will be $b - am, d - cm$. This clearly has issues, as given certain m's one can be bigger than the other.

Once given the positive and negative points, we can greedily go through the most positive/negative slopes, then allocate that to the point that yields the most/least y intercept. This done by brute force, will give us a time of $O(N^2)$ which is enough to pass the second subtask, but not good enough for a full score.

To get the last few cases, we have to optimize this step of allocating the slopes to the points to do this, we can utilize a binary search over the possible intercepts. For each intercept, we go through all the positive/negative points, testing the maximum/minimum slope each of them must have to reach the intercept. After this, we then run through all the given slopes, matching the greatest/least slopes to each other. In other words, we compare the just-generated slopes of the cows trying to reach the intercept, to the shooting trajectories given at the start of the problem. If there is ever a mismatch, in that the slope given at the start of the problem is greater than the one generated by the method, we then know that the intercept is unreachable.

This problem runs in roughly $O(N \log N \log R)$ time, where R is the range of y intercepts to check, given the problem, its about 10^{18}. It's more specifically in between the greatest y value of the rectangles, and the greatest x value multiplied by the greatest slope, added by the least x value.

Test Tubes

For starters, we realize that we can compress rows of colors into just one of its type. For example, we can compress "1112221121" into "12121". This proves rather useful, as all problems then start to look very similar and repetitive.

To take our idea to the next level, we can observe that for each move, we are able to remove at most one "block", as an example, "12121" has 5 blocks. Thus it should take $\ell_1 + \ell_2 - 2$ moves if we were to remove one block at a time. Now, we want to find a configuration that does exactly that.

We end up on the observation that we can do that, we can first make sure that

flasks 1 and 2 start off with the different first color(if not, just combine the first layer), then we can place one of the layers on to the bucket, then we can empty all but 1 blocks from one of the flasks. This is because if the flasks color matches with the vial, they pour it in the vial, effectively removing one block, if they match with the other vial, they combine, agian effectively removing one block. This goes on until one flask goes to length one, then its the others turn, doing the same thing. This way, we are dumping in the most efficient way.

Bessies Interview

The problem is split into two parts: finding the time in which bessie gets is interviewed, and finding which farmers are able to interview bessie.

For the first part, we can just use a priority queue to simulate the cows being tested. The key is to add the current simulation time to the priority queue, making it so that you can get the farmer finishing first without updating the entire list.

Now consider the ways that the farmers are able to interview bessie. We can very quickly make the observation that two farmers can be considered identitical/swapable(in terms of final result) if they both have stopped at the same time. This gets to a simple algorithm(not dfs, as I had previously mistaken, that doesnt work because if (1, 2) split, and 2 later splits into (2,3) that doesn't mean that 1 is tied to 3). We can simply store the instances where more than 1 farmer stops at the same time, then we can go backwards, keeping a set of possible-testing farmers, if for an instance one of the possible-testing farmers is in, then all of them are added to the set.

This will get us the answer.