Code Jam 2021 - Qualification Round

Analysis: Cheating Detection

Solutions for this problem are based on the fact that the cheater's advantage pumps up their numbers independently of the difficulty. Judged by their total number of correct answers, a cheater with base skill level B is going to look like a player with skill $B + \Delta$ for some significant Δ (Δ depends on B). However, this cheater does worse on easy problems than a player of actual $B + \Delta$ skill level, and better on hard problems, because the correct answers that are coming from cheating happen uniformly instead of more heavily on easier problems like skill-based improvements.

Test Set 1

There are multiple ways to get to 10% accuracy to pass Test Set 1. One such way is to estimate each question's difficulty as its number of correct answers, then sort by that difficulty, and check how uniform the distribution of corrects for each candidate is. The closer to uniform a candidate looks, the more they look like a cheater. One possible metric is the number of pairs of questions with different answers such that the incorrect answer is estimated to be for an easier question than the correct one. Using this metric is just enough to pass Test Set 1.

Test Set 2

The issue with counting inversions as the suggested metric for Test Set 1 is that it is a metric that is very susceptible to the contestant's strength. More concretely, a list that has few corrects or few incorrects has fewer opportunities to have inversions than one that is fairly evenly split. We can solve this by dividing the number of inversions by the expected number of inversions in a randomly arranged one. This reduces the noise enough to pass Test Set 2.

Another way to increase accuracy, of invesions or any other metric, is to check super-easy and super-hard questions, because the difference between a uniform distribution of correct answers and a heavily biased distribution is more pronounced in those. Exactly how much depends on the metric, but around the easiest and hardest 5% of questions seems like the right number experimentally for our solutions.

Other metrics are more accurate than inversions and also help solve the problem. We found two techniques that work well enough: sort the players by estimated skill level (i.e., by total number of correct answers) and compare the number of corrects only in the "extreme" questions with the number of corrects of other players with similar estimated skill level among those. Then, assume that the cheater will have the greatest difference with its neighbors. Another technique could be to estimate the actual skill level (by computing the inverse sigmoid of the accuracy of each player) and the actual difficulty of questions (again, the inverse sigmoid of its proportion of corrects). Then, using those two, estimate the expected number of correct answers for each player in the extreme questions. The player with the largest difference between that estimation and the real value is the cheater. Using this latest technique can get a solution above 90% accuracy.