Code Jam 2021 - Round 1C

Analysis: Closest Pick

Test Set 1

In Test Set 1, the limits are small enough that we can try every one of the $\mathbf{K}^2$ possible pairs of integers to pick. For each of those, we can calculate the probability of winning with each possible draw. There are $\mathbf{K}$ possible values for c. If for each one, we check the distance to each $\mathbf{P_i}$ and to our own two picks, we end up with an overall running time of $O(\mathbf{K}^3 \cdot \mathbf{N})$. This is fast enough to pass.

There are several possible optimizations. A simple one is to search for the "closest picked integer" using a sorted structure, reducing that phase to $O(\log \mathbf{N})$ and the overall running time to $O(\mathbf{K}^2 \cdot \mathbf{N} \log \mathbf{N})$. It is also possible to search for the distance from each c to each $\mathbf{P_i}$ only once, independently of our choices, which reduces the overall running time further to $O(\mathbf{K}^2)$.

Test Set 2

For Test Set 2, we can optimize all phases from the previous solution. If every integer is already picked at least once, then the answer is always 0 as the samples demonstrate. Otherwise, we can divide the non-purchased integers from $1, 2, \dots, \mathbf{K}$ into intervals between $\mathbf{P_i}$s. For example, if the $\mathbf{P_i}$ values are $3, 4, 8, 3$ and $\mathbf{K} = 8$, the intervals would be $[1, 2], [5, 7]$.

If we make only one of our picks within one of those intervals, we can get closest to all of it if the interval contains 1 or $\mathbf{K}$, by picking ours next to the only delimiting $\mathbf{P_i}$ (picking 2 from $[1, 2]$ in the example above). If the interval is surrounded by two $\mathbf{P_i}$s, we can get closest to at most half of the integers in the interval (rounding up), which is achievable by picking either end (picking either 5 or 7 from $[5, 7]$). If we make both of our picks within one of those intervals, we can get closest to all integers in the interval by picking both ends (5 and 7 from $[5, 7]$).

This shows that only integers that are next to an already purchased integer are worth picking for ourselves. At this point, we can go back to our last Test Set 1 solution and do the same but restricting our picks to integers next to a $\mathbf{P_i}$, which makes the $\mathbf{K}$s in the running time be $O(\mathbf{N})$, making it fast enough with the right implementation.

A few extra reasoning steps lead to a much more efficient solution. There are only $O(\mathbf{N})$ ways to make both picks within the same interval, and we can check each of them. If we choose two different intervals, it is always optimal to choose the two most valuable ones, so we do not need to check each combination separately. This leads to a single additional case that requires $O(\mathbf{N})$ time to compute to find the top two intervals. All in all, this solution takes only linear time (after sorting the array to find the intervals). This solution requires some care about corner cases, specifically the cases when no or just one integer from $[1, \mathbf{K}]$ is not a $\mathbf{P_i}$, and the case of the intervals list containing a single interval.