Coding Competitions Farewell Rounds - Round B

Analysis: Collecting Pancakes

Let X_a and X_b be the indices of the stacks chosen by Alice and Bob respectively. As per the definition in the question, $\mathbf{L_a} \leq X_a \leq \mathbf{R_a}$ and $\mathbf{L_b} \leq X_b \leq \mathbf{R_b}$.

The key observation to note here is that, while collecting pancakes, Alice and Bob cannot cross over any stack claimed by the other person at any point. Specifically, if $X_a > X_b$, then Bob cannot collect pancakes from any stacks with indices $> X_a$ and vice versa. This leads to the observation that only the pancake stacks in between Alice and Bob's initial choices are being contested for. All pancake stacks are guaranteed to be claimed by the person that started closer to them. Thus, the optimal strategy for either player would be to first start claiming stacks between X_a and X_b, then start claiming the other stacks, leaving each player with a contiguous range of stacks which is either a prefix or suffix of the overall line of stacks.

Test Set 1

The constraint $\mathbf{N} \leq 100$ allows a naive solution based on the observations above, which is to try out all valid combinations of X_a and X_b, and simulate the optimal strategy described.

The optimal strategy would be for Alice and Bob to first start moving towards each other's initial positions, claiming stacks along the way. Once all stacks between X_a and X_b have been claimed, each player would claim all the stacks closer to them. The final answer would thus be the maximum number of pancakes collected by Alice in any such simulation.

Since there are $O(\mathbf{N}^2)$ combinations of X_a and X_b, and simulating the strategy for any such X_a and X_b would take at most $\mathbf{N}$ steps, the overall time complexity of this approach is $O(\mathbf{N}^3)$.

Test Set 2

Another observation to be made is that once X_a is chosen, Bob will always try to set X_b as close to X_a as possible, to minimize the number of additional pancakes Alice can get by choosing stacks between X_a and X_b. This leads to three scenarios:

1. $X_a \leq \mathbf{L_b}$: In this case, Bob can only choose X_b to the right of X_a. This will result in Bob choosing $X_b = \mathbf{L_b}$ (or $X_b = \mathbf{L_b} + 1$ if $X_a = \mathbf{L_b}$). Then, once Alice and Bob start claiming stacks between X_a and X_b, Alice will end up claiming all the stacks from X_a to $\lfloor \frac{X_a + X_b}{2} \rfloor$. Finally, Alice would claim the stacks from 1 to $X_a - 1$, leaving her with all the stacks from 1 to $\lfloor \frac{X_a + X_b}{2} \rfloor$.
2. $X_a \geq \mathbf{R_b}$: Mirror image of Case 1.
3. $\mathbf{L_b} < X_a < \mathbf{R_b}$: Here, Bob can choose a position immediately adjacent to X_a (either $X_a - 1$ or $X_a + 1$). This leaves no stacks between Alice and Bob's initial positions, so both players would just claim all the stacks closer to them without crossing the other player. Bob would choose between $X_a - 1$ or $X_a + 1$ with the goal of minimizing the number of panackes Alice collects, so the final number of pancakes collected by Alice would be $\min($ sum of pancakes in stacks from 1 to X_a, sum of pancakes in stacks from X_a to $\mathbf{N})$.

Using these cases, for a given X_a, we can exactly predict the contiguous range of stacks that Alice will claim, assuming optimal play, in $O(1)$. Given a range, calculating the sum of pancakes in that range can be done in constant time if we precompute prefix sums. The precomputation

can be done in $O(\mathbf{N})$ time. Finally, if we do the constant time computation for each possible X_a and take the maximum, we find the answer in $O(\mathbf{N})$ time overall.