Kick Start 2017 - Round F

Analysis: Dance Battle

Dance Battle: Analysis

Even before solving the Small dataset, we need to reduce the number of options available to us, because the Delay action could allow a dance battle to go on forever! A critical initial insight is that we can use Delay to sort the opponents and face them in whatever order we want: we can delay against everyone else until we face our first desired opponent, then take some other non-delay action on that opponent, then keep delaying until we face our second desired opponent, and so on.

Small dataset

Once we know that we can use Delay to sort, the Small dataset's problem space allows us to use brute force. We will consider the worst case, **N** = 5. We can choose one of 5! possible orders in which to face the opponents; for each opponent, we choose one of the three other actions (Dance, Truce, or Recruit). That is a total of 5! × 3^5 = only 29160 possible scenarios. We can simulate each of them to make sure that our energy does not drop below 1, and our honor does not drop below 0. Then, we take the maximum honor value among all valid scenarios. Each simulation takes linear time, so the overall time complexity of this strategy is O(**N**! × $3^{\mathbf{N}}$ × **N**).

Large dataset

The brute force method above will not scale to 1000 opponents. Let us devise an alternate strategy. For starters, we can observe that we cannot use the Recruit action until we have defeated at least one opponent (and gained a point of honor) by using Dance. So, if our starting energy level does not let us defeat the opponent with the lowest dancing skill, then our best option is to use Truce on everyone and finish with 0 honor.

Suppose that we can defeat at least one opponent by using Dance. Then we have no reason not to choose the opponent with the lowest dancing skill, since all opponents give the same amount of honor when defeated via Dance. Moreover, we may as well continue dancing against the weakest remaining opponent as long as we have the energy to do so.

What happens when we do not have enough energy to Dance against any remaining opponent? We can either use Truce to send everyone else away, or Recruit some opponent. If we are going to recruit someone, we should pick the opponent with the highest energy, since the cost of recruiting any opponent is the same. But how do we decide whether to use Recruit or Truce?

As long as there are at least two opponents remaining, it cannot hurt us to recruit the one with the most energy, because after that, we can surely defeat at least the one with the least energy. Even if we can defeat only one, and that one has the same skill as the opponent we recruited, we have lost one honor and gained one honor, and we have had no net change in energy, so we are no worse off than if we had used Truce to remove the same two opponents.

So, we can sort the opponent list from lowest to highest energy, and start with one pointer at each end. Setting aside the case in which we cannot defeat anyone: first, we move our left pointer forward, defeating opponents by dancing until we no longer can. Then, as long as there are at least two step, recruiting the strongest opponent and gaining energy. We repeat this until the pointers meet, or we have one opponent left that we cannot defeat (in which case we should use Truce instead of wasting a point of honor by using Recruit). Whatever amount of honor we have at that point is our answer.

This algorithm has an O(**N** log **N**) sorting step followed by an O(**N**) execution step. Other O(**N**) execution steps are possible; for example, we can notice that the algorithm above will terminate with the pointers either zero or one opponent apart, so we can use partial sums and binary search to directly find that point.