Kick Start 2021 - Round C

Analysis: Rock Paper Scissors

Test Set 1

We could observe that it is easy to calculate the expected value of a certain strategy by iterating through all the choices you make every day and adjust your friend's probabilities of picking different choices accordingly. Therefore, we could start from some naive strategy to maximize our chance of winning, one strategy would be something like `RSPRSPRSP...`, and calculate its expected value. If such strategies cannot reach our expected bonus value, we could try to create a random strategy instead, calculate its expected value of bonus, and do so repeatedly until we find a strategy of which the expected bonus is at least $\mathbf{X}$. This approach is enough to solve the first test set.

Test Set 2

For the second test set, since the maximum value of $\mathbf{X}$ is raised, the method for Test Set 1 will exceed the time limit. Therefore we should find a more efficient way to look for a better strategy other than randomly generating them. We could first utilize what we have tried in the first test set, generating some random strategies, and pick the one with the best expected bonus as our base strategy S. We then randomly pick a day in S, and try to change the decision of that day and get a new strategy S'. If the expected bonus of the new strategy $E(\mathrm{S'})$ is higher than that of the original $E(\mathrm{S})$, we could substitute S with S' as our best strategy. By keep performing this substitution process, it is obvious the expected value of our strategy will be either increasing or reach a local maximum. Thus we could utilize this method, also known as hill climbing, to find a strategy which has an expected value larger than $\mathbf{X}$. Note that since what we find here is a local maximum, it is possible that we would eventually find out that the best stratgy derived from S cannot reach the value of $\mathbf{X}$. In that case we have to reselect the initial S and perform another substitutiton process.

Test Set 3

For the last test set, the methods we applied in the previous test sets could not find us a strategy in the given time limit. However we could try to solve the problem in a different angle. In this problem, we are targeting an expected bonus of $\mathbf{X}$. To simplify the problem, instead of finding a strategy to reach the targeted bonus, we can try to maximize the expected value we could get by using the best possible strategy we could have. This value will always be at least as large as $\mathbf{X}$ because the solution is guaranteed to exist in the problem statement. We could then use dynamic programming to solve this problem.

Let $v[r][p][s]$ denote the maximum expected value you could get if you made exactly r rocks, p papers, and s scissors in you strategy on the N-th day. We could first observe $r+p+s=N$, and that this value will be affected by the last decision you made in your strategy. For example, if you decide to choose rock on the last day, the expected value will be

$$v[r-1][p][s]+\frac{p}{n-1}\times\mathbf{W}+\frac{s}{n-1}\times\mathbf{E}.$$

Similarly, this can be applied to choosing paper or scissors as the last decision. Thus we could conclude that

$$\begin{aligned} v[r][p][s] =& \max(v[r-1][p][s] + \frac{p}{n-1} \times \mathbf{W} + \frac{s}{n-1} \times \mathbf{E}, \\ & v[r][p-1][s] + \frac{s}{n-1} \times \mathbf{W} + \frac{r}{n-1} \times \mathbf{E}, \\ & v[r][p][s-1] + \frac{r}{n-1} \times \mathbf{W} + \frac{p}{n-1} \times \mathbf{E}) \end{aligned}$$

We then get the function for state transfomation. As for the starting condition, since on the first day our friend's decision is totally random, we could easily have

$$v[1][0][0] = v[0][1][0] = v[0][0][1] = \frac{1}{3}\mathbf{W} + \frac{1}{3}\mathbf{E}$$

Finally, to get the whole strategy to achieve the maximum bonus, we could record the last decision we made with another array $step[i][j][k]$, and backtrace it when we found the final value at $\max(v[i][j][k])$ where $i + j + k = 60$.

We could do some brief analysis on the time and space complexity here. Notice that since in our dynamic programming process, if we want to find the optimal strategy for the N-th day, we will need to calculate all the states where $i + j + k \leq N$, which means that the space we are using will be $O(\binom{N}{3}) = O(N^3)$. As for the time complexity, since our transformation is constant, the dynamic programming process will be $O(\binom{N}{3}) = O(N^3)$. In our problem, where $N = 60$, such complexity is sufficient for our solution to find out the strategy in the given time limit.