Kick Start 2019 - Round E

Analysis: Code-Eat Switcher

Test set 1 (Visible)

Let's start with the brute force way of solving the problem. We can iterate through all possible coding units for slot 1, let's say X. Now, we can calculate the coding unit left for slot 2 ($\mathbf{A_i}$ - $\mathbf{X}$). Using this we can calculate eating unit we can achieve for both slot 1 and slot 2 and compare it with $\mathbf{B_i}$. Time complexity for each test case would be O($\mathbf{C_{max}}$*$\mathbf{D}$).

Test set 2 (Hidden)

Let's first solve for only 1 day. So assuming two time slots $\mathbf{S_i}$($\mathbf{E_i}$, $\mathbf{C_i}$) and $\mathbf{S_j}$($\mathbf{E_j}$, $\mathbf{S_j}$) we can see that to achieve every 1 unit of eating in $\mathbf{S_i}$ slot we are losing $\mathbf{C_i}$/$\mathbf{E_i}$ unit of coding, similarly for $\mathbf{S_j}$ slot. Hence, we can observe that it's always a better choice to choose time slot $\mathbf{S_i}$ if $\mathbf{C_i}$/$\mathbf{E_i}$ ≤ $\mathbf{C_j}$/$\mathbf{E_j}$. Now, we have the order in which we should achieve the required eating unit.

For calculating minimum coding requirement we can maintain prefix cumulative and suffix cumulative sum of maximum coding and eating unit that can be achieved in a time slot. Then using binary search on prefix cumulative sum array, we can find out which minimum indexed time slot will give us eating more than required and we can remove the excess fraction of time for use in coding. We can compute the maximum coding unit achieved from unused time slots by using the suffix cumulative sum. This will take O(**SlogS** + **S**) preprocessing time and O(**DlogS**) for each test case.