Code Jam 2022 - Round 1A

Analysis: Weightlifting

Test Set 1

The limits for Test Set 1 are small enough that we can afford to use brute force. Since there is no reason to have more than $\max_j \mathbf{X_{i,j}}$ weights of type i, the number of possible states of the stack to consider is very limited (in fact, it is $\sum_{0 \le a,b,c \le 3} \frac{(a+b+c)!}{a!b!c!} = 5248$). To solve for the minimum number of operations to finish all of the exercises, you can use a breadth-first search over all possible combinations of the stack states and the number of completed exercises. Specifically, the breadth-first search starts at the state of an empty stack with no completeted exercises, and the final answer is the distance from that to the state where the stack is empty but all exercises are completed.

Test Set 2

The first observation we can make to solve Test Set 2 of this problem is the following: if there are any weights that are used in every exercise, it is always optimal to put them at the bottom of the stack before the first exercise, and leave them there until after the last one. Moreover, if there are multiple such weights, putting them in any order is equivalent. Formally, if C is the multiset of weights that is common to all exercises, there is an optimal solution that has C at the bottom of the stack during all exercises.

Now, for a given full sequence of exercises, let us call A to the largest sequence of weights that appears at the bottom of the stack for all exercises (A could be empty). Since A is used for all exercises, $A \subseteq C$. If we use the first observation, then also $C \subseteq A$, so $A = C$. Therefore, we can make a second observation: if there is more than one exercise, there is at least one time in between exercises in which the stack contains exactly C. This points towards a divide and conquer solution: find that intermediate point and then recursively solve the problem of minimizing the operations before and after that point, disregarding C.

As it is common with divide and conquer approaches, we can either find the place to split without recursion, or we can use memoization to be able to simply try every possible split point, without incurring significant additional computation time.

To formalize the approach, let $C(\ell, r)$ be the multiset of weights that is the intersection of the multiset of weights needed for each exercise between ℓ and r, inclusive. Additionally, let $M(\ell, r)$ be the minimum number of operations to get from a stack containing $C(\ell, r)$, in any order, perform all operations between ℓ and r, inclusive, and leave the stack with $C(\ell, r)$ on it, in the same order. To calculate $M(\ell, r)$ we follow the strategy above: for some mid-point x we do this twice: load the additional weights for one part of the split, optimize according to a recursive result, then unload those additional weights. The additional weights for the left part of the split are $C(\ell, x) \setminus C(\ell, r)$ and for the right part of the split are $C(x+1, r) \setminus C(\ell, r)$ (notice that $C(\ell, r)$ is included in the other two multisets by definition). Therefore, the cost in addition to the recursion for the mid-point x is $2 \times (|C(\ell, x)| + |C(x+1, r)| - 2 \times |C(\ell, r)|)$. Putting it all together:

- $M(\ell, r) = 0$, if $\ell = r$;
- $M(\ell, r) = \min_{\ell \le x < r}\big(M(\ell, x) + M(x+1, r) + 2 \times (|C(\ell, x)| + |C(x+1, r)| - 2 \times |C(\ell, r)|)\big)$, otherwise.

Precomputing the $O(\mathbf{E}^2)$ values of C takes $O(\mathbf{E}^2 \times \mathbf{W})$ time. With memoization, we can compute the $O(\mathbf{E}^2)$ values M in $O(\mathbf{E}^3)$ overall time. This is fast enough to pass Test Set 2.