Kick Start 2022 - Round D

Analysis: Maximum Gain

Test Set 1

One straightforward way to solve this problem is to simulate the process with a recursive brute force method. Let us define the following variables:

- Ai: The pointer to the start of array $\mathbf{A}$. The pointer moves one step ahead after answering the first question from $\mathbf{A}$, i.e. Ai becomes $Ai+1$.
- Aj: The pointer to the end of array $\mathbf{A}$. The pointer moves one step back after answering the last question from $\mathbf{A}$, i.e. Aj becomes $Aj-1$.
- Bi: The pointer to the start of array $\mathbf{B}$.
- Bj: The pointer to the end of array $\mathbf{B}$.
- k: The remaining number of questions to answer.

With these variables, we can write down the recursive equation as follows:

$$f(Ai,Aj,Bi,Bj,k)=\begin{cases} -\mathrm{inf}, & \text{if } (Ai-1)+(\mathbf{N}-Aj)>\mathbf{N} \text{ or } (Bi-1)+(\mathbf{M}-Bj)>\mathbf{M}. \\ 0, & \text{if } k=0. \\ \max\begin{pmatrix} f(Ai+1, & Aj, & Bi, & Bj, & k-1)+\mathbf{A}[Ai] \\ f(Ai, & Aj-1, & Bi, & Bj, & k-1)+\mathbf{A}[Aj] \\ f(Ai, & Aj, & Bi+1, & Bj, & k-1)+\mathbf{B}[Bi] \\ f(Ai, & Aj, & Bi, & Bj-1, & k-1)+\mathbf{B}[Bj] \end{pmatrix}, & \text{otherwise.} \end{cases}$$

The maximum points we can get by answering $\mathbf{K}$ questions from arrays $\mathbf{A}$ and $\mathbf{B}$ would be the return value of $f(1,\mathbf{N},1,\mathbf{M},\mathbf{K})$. Be careful that indexes Ai, Aj, Bi, and Bj may go out of bounds of arrays $\mathbf{A}$ and $\mathbf{B}$. The out-of-bounds values are defined to be zeros in the equations above.

However, the time complexity of the recursion above is $O(4^{\mathbf{K}})$. To improve the performance, we can use dynamic programming: keep the return values in a 4-dimensional array to avoid repeated computation. Notice that the memoization array does not need to keep parameter k since it can be derived from the other four parameters with $k=\mathbf{K}-(Ai-1)-(\mathbf{N}-Aj)-(Bi-1)-(\mathbf{M}-Bj)$. With memoization, the time complexity of this method is $O(\mathbf{K}^4)$, and space complexity is $O(\mathbf{K}^4)$.

Test Set 2

Notice that the order of questions to answer does not affect the points we get from them. Therefore, we can always answer questions in the first array $\mathbf{A}$ first, and then in the second array $\mathbf{B}$. This observation allows us to divide the problem into two sub-problems: find the maximum points P_A we can get by answering K_A questions in the first array $\mathbf{A}$, and the maximum points P_B by answering K_B questions in the second array $\mathbf{B}$. Enumerating all possible combinations of (K_A, K_B) where $K_A+K_B=\mathbf{K}$, the total maximum points we can get by answering $\mathbf{K}$ questions would be the maximum value of all P_A+P_B.

Now we want to find a way to get the maximum points P_A by answering K_A questions in the first array $\mathbf{A}$. Since the order of questions to answer does not matter, we can answer K_{AStart} questions from the start of the array first, and then K_{AEnd} questions from the end of the array, such that $K_{AStart}+K_{AEnd}=K_A$. The points we get would be:

$$
\begin{aligned}
P'_A &= (\mathbf{A}_1 + \mathbf{A}_2 + \cdots + \mathbf{A}_{K_{AStart}}) + (\mathbf{A}_{\mathbf{N}-K_{AEnd}+1} + \mathbf{A}_{\mathbf{N}-K_{AEnd}+2} + \cdots + \mathbf{A}_{\mathbf{N}}) \\
&= \mathrm{Prefix}(\mathbf{A}, K_{AStart}) \qquad\qquad + \mathrm{Suffix}(\mathbf{A}, K_{AEnd})
\end{aligned}
$$

Therefore, we can find the maximum points P_A by enumerating all combinations of (K_{AStart}, K_{AEnd}) where $K_{AStart} + K_{AEnd} = \min(K_A, \mathbf{N})$, and find the maximum values of all P'_A with the equation aforementioned. With the prebuilt prefix sum and suffix sum arrays, we can get P'_A in $O(1)$ time, and the maximum points P_A in $O(K_A)$ time.

We can apply the same algorithm to find the maximum points P_B by answering K_B questions in the second array $\mathbf{B}$. The time complexity for this method would be:

- $O(\mathbf{N} + \mathbf{M})$ to prebuild the prefix and suffix arrays for $\mathbf{A}$ and $\mathbf{B}$.
- $O(\mathbf{K}^2)$ to get the total maximum points.
    - $O(1)$ to get the points with a pair (K_{AStart}, K_{AEnd}) or (K_{BStart}, K_{BEnd}).
    - $O(\mathbf{K})$ to enumerate all (K_{AStart}, K_{AEnd}) pairs and get P_A — maximum points by answering questions from array $\mathbf{A}$. Same for array $\mathbf{B}$.
    - $O(\mathbf{K}^2)$ to enumerate all (K_A, K_B) pairs and get the total maximum points.
- Overall: $O(\mathbf{K}^2 + \mathbf{N} + \mathbf{M})$.