MCC 2023 Problem 5: Rectangles Solution

Problem idea by: Neo Yong Li

Problem prepared by: Neo Yong Li

Definitions

We say that a red rectangle is *covered* by a blue rectangle if and only if the entire red rectangle is inside the blue rectangle.

Task 1

This task is meant to be solved by hand.

Task 2 ($K = 1$)

When $K = 1$, the minimum area to include all N rectangles is the sum of the widths multiplied by the maximum height of all the red rectangles.

Task 3 ($K = 2$)

Unless $N = 1$, it is best to use two blue rectangles. We can try all possible *breakpoints* i and assume that our first blue rectangle includes the first i red rectangles (and the second blue rectangle includes the remaining ones). Once we fix i, to find the total area of blue rectangles needed, we use Task 2's solution to solve the $K = 1$ case for the first i blue rectangles and the last $n - i$ blue rectangles, and sum up their answers. The answer is **minimum** area out of all breakpoints.

This has a time complexity of $O(N^2)$, but it can also be done in $O(N)$ with prefix sum, prefix max, suffix sum and suffix max.

Task 4 ($K = 998244353$)

When $K > N$, we can observe that we do not need to use all K blue rectangles. We can cover each of the N red rectangles using one blue rectangle. This means that the answer is the sum of the area of all the N red rectangles.

Task 5 (No other constraints)

This task requires the observation from Task 2. First of all, the K blue rectangles always cover a range of rectangles. With this, we can use dynamic programming to solve the problem.

Let $dp_{i,j}$ = minimum area of blue rectangles after covering i red rectangles using **exactly** j blue rectangles.

DP transition: For $t > i$, if $dp_{i,j} + sum(w_{i+1}, ..., w_t) \cdot \max(h_{i+1}, ..., h_t)$ is smaller than $dp_{t,j+1}$, assign it to $dp_{t,j+1}$.

Initial state: $dp_{i,j} = \infty$ for all i and j except $dp_{0,0} = 0$. The answer will be $dp_{N,K}$.

The time complexity is $O(N^2K)$. From Task 4, we can observe that we need at most N blue rectangles, hence we can set $K = \min(K, N)$ before running the program. The final time complexity is $O(N^2 \min(K, N))$. This solution can solve all tasks.