Kick Start 2021 - Round D

Analysis: Cutting Intervals

We are given $\mathbf{N}$ intervals and are asked to find the maximum number of intervals we can obtain if we perform a maximum of $\mathbf{C}$ cuts. There are a few key observations required to solve this problem.

First, performing a cut at the same point X more than once will not result in any additional intervals. After the initial cut, all intervals which contained X will now be split into intervals that have X as an endpoint, and thus cannot be cut further at X.

Second, a cut at X will result in the same number of additional intervals regardless of the number of cuts performed earlier at other points. This implies that at most N additional intervals can be obtained with each cut.

Based on the aforementioned observations, we can solve this problem as follows. Let A_j represent the number of points at which performing a cut will result in j additional intervals ($0 \leq j \leq \mathbf{N}$). Iterating over A in reverse order, we perform the cuts greedily, adding $j \cdot \min(A_j, \mathbf{C})$ to our result, and decrementing $\mathbf{C}$ by $\min(A_j, \mathbf{C})$ until $\mathbf{C} = 0$. The final answer is the result.

Iterating over A can be done in $O(\mathbf{N})$. Now, we will go over how to populate A for the two test sets.

Test set 1

For this test set, we can iterate over each point coordinate and count the number of intervals it lies strictly within, i.e. for each $X \in [\min(\mathbf{L_i}), \max(\mathbf{R_i})]$, we count the number of intervals such that $\mathbf{L_i} < X < \mathbf{R_i}$. Let that number be k. Then, we increment A_k.

This can be performed in $O(\mathbf{N} \cdot \max(\mathbf{R_i}))$.

Test set 2

For this test set, the above solution would exceed the time limits.

We observe that the number of additional intervals obtained by a cut at two consecutive points X and $X + 1$ are the same, except, possibly, when some intervals start or end at these points. We can construct a sorted map M_X which maps the coordinate X to the number of *additional* intervals it lies *within* as compared to $X - 1$. That is, for each starting interval endpoint $\mathbf{L_i}$, we increment $M_{\mathbf{L_i}+1}$. And for each ending interval endpoint $\mathbf{R_i}$, we decrement $M_{\mathbf{R_i}}$.

Consider an example with the following intervals: $[3, 7], [1, 5], [4, 7]$. The mappings created are $M_2 = 1, M_4 = 1, M_5 = 0$, and $M_7 = -2$: because one interval begins at 1 (M_2 += 1), one interval begins at 3 (M_4 += 1), one interval begins at 4 (M_5 += 1), one intervals ends at 5 (M_5 -= 1) and two intervals end at 7 (M_7 -= 2).

Finally, we iterate over the map in sorted order of keys, keeping track of the number of overlapping intervals j, the previous key k_{prev}, and the current key k_{curr}. We increment A_j by $k_{curr} - k_{prev}$. Now A_j can be used to compute the final solution as described above.

In the above example, we iterate over the keys of M, and start with $j = 0$. All points smaller than the first key (2) will produce zero additional intervals.

We increment j by M_2 and go to the next key. Now $j = 1, k_{curr} = 4, k_{prev} = 2$. We increment $A_j = A_1$ by $k_{curr} - k_{prev} = 2$, because the 2 points (2, 3) will produce 1 additional interval if we perform a cut on them.
Now we increment j by M_4 and go to the next key. Now $j = 2, k_{curr} = 5, k_{prev} = 4$. We increment A_2 by 1, because there is 1 point (4) at which performing cuts will result in 2 additional intervals.
Then, we increment j by $M_5 = 0$ and go to the next key, so $j = 2, k_{curr} = 7, k_{prev} = 5$. Again we increment A_2 by 2, because the points 5, 6 also result in 2 additional intervals.
Finally we increment j by $M_7 = -2$ and end.
The final result is $A_1 = 2, A_2 = 3$, and we can start performing greedy cuts.

Constructing the map requires adding $2 \cdot \mathbf{N}$ endpoints to it, with each addition requiring $O(\log(\mathbf{N}))$. Therefore, the overall time complexity is $O(\mathbf{N} \cdot \log(\mathbf{N}))$.