Kick Start 2020 - Round F

Analysis: Metal Harvest

Consider given time intervals as an array. We will refer to the start time of i-th interval as $\mathbf{S_i}$ and to the end time of i-th interval as $\mathbf{E_i}$.

Let us sort the time interval array by start time in ascending order. To start the harvesting, we need to deploy our first robot at the start time of the first interval ($\mathbf{S_1}$) in the sorted array. It is not optimal to deploy the robot before $\mathbf{S_1}$ because we would be wasting robot hours. Per the problem statement, our first robot can be deployed from $\mathbf{S_1}$ to $\mathbf{S_1}$ + **K** before it returns for calibration. Let us define interval length of i-th interval as $\text{interval\_length}_i = \mathbf{E_i} - \mathbf{S_i}$.
The following cases are possible for our first robot:

1. $\text{interval\_length}_1 > \mathbf{K}$: In this case, as soon as our first robot finishes, we need to deploy another robot to cover the remaining part of the interval.
2. $\text{interval\_length}_1 \leq \mathbf{K}$: This can be further divided in two cases:
    a. Bring the first robot back as soon as the first interval finishes. We cannot re-deploy this robot since a robot can only harvest for consecutive units of time.
    b. Allow the first robot to be available until **K** units of time has elapsed. This is a better strategy since it allows us to utilize the robot for future harvesting.

Based on the above scenarios, the optimal strategy is to deploy a robot at the beginning of the first harvesting interval, for **K** hours. If after **K** hours the robot is inside the first or any subsequent harvesting interval, deploy another robot to replace it. Otherwise, deploy another robot as soon as the next harvesting interval begins. Repeat this for every interval and keep count of the number of total deployments.

Test Set 1

For this test set we can loop over sorted time intervals and then have another inner loop to go over each point in time of such interval.
At the i-th point in time, if there is no robot currently deployed for harvesting, we can deploy a robot and keep it harvesting for **K** units of time. If a robot is already currently deployed for harvesting then no action is needed. The total number of robots deployed is the required answer.
Since we are sorting the intevals array and then iterating over each point in time inside an interval, the worst case time complexity will be $O(\Sigma_{1 \leq i \leq \mathbf{N}} \text{interval\_length}_i + \mathbf{N} \log \mathbf{N})$, where **N** is the total number of intervals and $\text{interval\_length}_i \leq 200$.

Test Set 2

To solve for this test case we need to find a way to count the number of robots needed for an interval without iterating over each point in time.
Let us say we deploy a robot at $\mathbf{S_1}$. The number of robots needed for the first inteval can be calculated as $\lceil \text{interval\_length}_1 / \mathbf{K} \rceil$.
Let us define a variable last_harvest_time as the last point in time covered by all deployed robots and initialise it as 0. It is possible that the last robot deployed to cover the first interval will go beyond $\mathbf{E_1}$ hence the last_harvest_time after covering the first interval will be,
last_harvest_time = max(last_harvest_time , $\mathbf{S_1}$) + number of robots needed for the first interval * **K**.
For subsequent intervals:

1. If the interval is already fully covered with the last_harvest_time, no action is needed.
2. If the interval is not fully covered by the last_harvest_time then follow a similar strategy as first interval to calculate the number of new robots required and again update the last_harvest_time. Also increment the answer by total number of new robots.

Since we are sorting the intervals array and then iterating over all intervals at once, the worst case time complexity will be O(**N** log **N**), where **N** is the total number of intervals.