Kick Start 2022 - Round F

Analysis: Scheduling a Meeting

Test Set 1

In Test Set 1, we can simply try all possible options to schedule a new meeting. We can try all possible start times for a new meeting, for each Tech Lead calculate how many of their scheduled meetings overlap with our proposed meeting time, and then pick $\mathbf{K}$ smallest of these values. Output the minimum for all the possible start times as the final answer.

This can be done in $O(\mathbf{D} \times (\mathbf{M} + \mathbf{N} \log \mathbf{N}))$ time complexity, or similar, which is fast enough for this test set.

Note that limits in Test Set 1 are very permissive, and thus it is possible to pass it with even slower solutions. For example, we can try all possible starting times and all possible subsets of Tech Leads who will attend the meeting, and for each such option count how many already scheduled meetings overlap with the given time slot and have to be canceled. This will result in a solution with $O(\mathbf{D} \times 2^{\mathbf{N}} \times \mathbf{M})$ overall time complexity.

Test Set 2

In Test Set 2, the limits are higher, so we will need a smarter solution.

There are many possible ways to solve this problem, and we list some of them below. All approaches in this analysis use the following techniques and observations:

1. We will again try all possible start times for a new meeting, but now in a sliding window fashion. This means that first we will try the range of $(0, \mathbf{X})$ hours, then $(1, \mathbf{X} + 1)$, and so on.
2. We will maintain the number of current meetings for each Tech Lead as meetings enter and exit our sliding window.
3. To minimize the number of cancelled meetings we need to meet with $\mathbf{K}$ Tech Leads having the smallest number of current meetings. Sum of their numbers of meetings will be the answer for the current window.

Approach #1. Data structures and binary search.

Let us maintain two helper arrays:

- Number of Tech Leads that have exactly i meetings in the current window, for each i. Note that if you take a prefix sum $SUM(0, j)$ over this array, you will find the total number of people with less than or equal to j meetings for some j.
- Sum of number of meetings for everyone with exactly i meetings in the current window, for all i. Note that if you take a prefix sum $SUM(0, j)$ over this array, you will find the total number of meetings for everyone with less than or equal to j meetings for some j.

To be able to quickly take prefix sums over these arrays, we will keep the values in a data structure like segment tree or Fenwick tree.

Armed with these data structures and helper arrays, in each window we can use binary search to find the biggest number of meetings among $\mathbf{K}$ Tech Leads we will meet with, and then the sum of the number of their meetings.

Time complexity of this solution will be $O(\mathbf{D} + \mathbf{N} + \mathbf{M} \log \mathbf{M} + \mathbf{D} \log^2 \mathbf{M})$ (which you can also optimize to $O(\mathbf{D} + \mathbf{N} + \mathbf{M} \log \mathbf{M} + \mathbf{D} \log \mathbf{M})$ if you replace binary search with some kind of smart segment tree descent).

Approach #2. Rebalancing sets.

Another elegant solution is to store the number of the current meetings for each Tech Lead in a data structure that allows to extract the minimum and the maximum value, and remove an arbitrary value from it, like a tree-based set. We will keep two of these data structures: one containing $\mathbf{K}$ Tech Leads with the minimum number of meetings in the current window, and another with $\mathbf{N} - \mathbf{K}$ other Tech Leads. We will update these data structures as the meetings come and go: as a meeting starts or ends, we remove an old entry for the corresponding Tech Lead in the data structure, insert a new entry with an updated meetings count, and rebalance two data structures so that one of them again has $\mathbf{K}$ smallest values (to achieve this, we may need to exchange the biggest value from one data structure with the smallest value in the second one). In addition, we will also maintain the sum of the number of meetings in the data structure that holds $\mathbf{K}$ smallest values.

It can be noted that with careful implementation each meeting will trigger only a constant number of operations to keep two data structures balanced, and the total time complexity of this solution will be $O(\mathbf{D} + \mathbf{N} + \mathbf{M} \log \mathbf{N})$.

Approach #3. Linear solution.

Finally, the "linear" $O(\max(\mathbf{D}, \mathbf{N}, \mathbf{M}))$ solution to this problem exists. This is possible if you carefully maintain the following values:

- Number of the current meetings for each Tech Lead.
- Number of Tech Leads with exactly i current meetings for all i.
- Value of maximum number of meetings among $\mathbf{K}$ Tech Leads we will meet with.
- Number of Tech Leads with that maximum number of meetings among $\mathbf{K}$ Tech Leads we will meet with.
- Sum of the meetings of all $\mathbf{K}$ Tech Leads we will meet with (which is also the sum of $\mathbf{K}$ smallest numbers of current meetings).

It can be noted that with each starting or ending meeting, all these values can be recalculated in $O(1)$ time (in fact, most of these values change at most by one), and we can achieve a solution with total linear time complexity.