Vacation – Analysis

Author: Viktor Kozhuharov

Subtask 1: $K = 0$

It is enough to compute $\max L_i$ and $\min R_i$. The overlap between these gives the maximum possible common interval.

Subtask 2: $K \leq 1$

In this subtask, we may move one interval by 1 which can be simulated for all intervals in both directions. A naive implementation is slow, but it can be optimized with prefix/suffix sums or sets.

Subtask 3: $K = 10^{18}$

Since K is large enough, we can align all L_i's to be equal. Therefore, the answer is simply the size of the smallest interval.

Subtask 4: $N \leq 10^4, L_i \leq 10, R_i \leq 10$

Because L and R are small, we can try all possibilities for the final interval $[\textit{start}, \textit{end}]$. For each candidate, it is straightforward to calculate the number of operations needed to make every friend available. The answer is the maximum length over all feasible intervals.

Subtask 5: $N \leq 10^3$

We can binary search on the length of the longest possible interval. Observation: it suffices to check all L_i and R_i as endpoints (although one of the endpoints might not be among this set). This is a common observation in tasks of this kind.

It can be proven by assuming the contrary – that there is no optimal solution where the final interval $[\textit{start}, \textit{end}]$ shares an endpoint with any L_i and R_i. Each interval that is moved has either $L_i > \textit{start}$ (and the movement is to the left so its starting point becomes *start*), or $R_i < \textit{end}$ (and the movement is to the right so its ending point becomes *end*). Let us denote their counts with c_l and c_r, respectively. Without loss of generality, we can assume that $c_l \geq c_r$ (meaning that a similar argument will hold in the other case). This means that there is at least one $L_i > \textit{start}$ (c_l cannot be 0). We can start moving the final interval to the right until we hit one of the endpoints. When we move 1 to the right, the cost decreases by c_l and increases by c_r, so the change is $-c_l + c_r \leq 0$, i.e., the cost remains the same or decreases. This is a contradiction because we will have an

optimal common interval sharing an endpoint with the intervals. So we have proven the observation.

For a candidate interval $[start, end]$, the cost is $\sum_{i=1}^{N} \max(L_i - start, 0) + \sum_{i=1}^{N} \max(end - R_i, 0)$. Additionally, the answer can't exceed the minimum interval length.

The complexity is $O(N^2 \log MAX)$, where MAX is the maximum possible day.

Subtask 6: $N \leq 10^5$

We can optimize the previous solution. It's needed to check for all L_i and $R_i - len + 1$ whether they can start a valid vacation interval. To do this, have the starts as events along as the original L_i-s and R_i-s and maintain how many operations are needed for each start. Each binary search step requires sorting the events, giving a complexity of $O(N \log MAX \log N)$

Subtask 7: Full problem

We can further optimize by moving the sorting outside the binary search. If L and R are sorted initially, the check function can be implemented using a merge-like procedure. This reduces the overall complexity to $O(N \log MAX)$.