EGOI 2026 Editorial - Watering Plants

Task Author: Harry Zhang

Subtask 1

In this subtask, there is only one day and one query for some resident i. We have to output how often resident i comes home before $i-1$. Since there is only one day, this boils down to checking whether $t_i < t_{i-1}$. If yes, the answer is 1, else 0.

Subtask 2 - No ! type queries

Since there are no `!` type queries, the relative order of arrival times never changes: if resident i comes home before resident $i-1$ at the beginning, the same holds when we answer any `?` query.

So for a given day j, the answer will be - $j+1$, if $t[i] < t[i-1]$. - 0, if $t[i] \geq t[i-1]$.

Subtask 3 - N = 2

In this subtask, only resident 1 ever waters the plants for resident 0. When iterating through the days, we keep track of a number w (initially 0), the number of times resident 1 watered for resident 0 so far, and we keep track of the current t_0 and t_1. Each day, we first check whether $t_1 < t_0$ (does resident 1 water for resident 0 today?) and if yes, increment w by one. After that, in case we read an update event, we update t_0 or t_1. In case we read an update event, we output w.

Subtask 4 - $O(ND)$ Solution

For each day, we directly track and update for everyone how many times they have watered the plants for resident below them.

Maintain $w[i]$ for $1 \leq i \leq N-1$, the number of days resident i has watered resident $i-1$'s plants. We initialize $w[i] = 0$ for all i. For each day j from 0 to $D-1$: - For each i with $1 \leq i \leq N-1$: if $t[i] < t[i-1]$, increment $w[i]$ by 1. - Process the event at the end of day j: - For "! i t", set $t[i] = t$. - For "? i", output $w[i]$.

This runs in $\mathcal{O}(N \cdot D)$ time.

Subtask 5

The key observation for this subtask and the full solution is that the watering relationship for pair $(i-1, i)$ only changes when one of those two residents updates their arrival time. Between consecutive such updates, resident i either

waters every day or never, so we can process whole time segments between any changes without updating $w[i]$ after every step.

In this subtask, every resident changes their return time at most once. For each resident i, save their previous arrival time $prev[i]$ and current arrival time $cur[i]$ (initially both t_i) and the time when it changed $change[i]$ (initially 0). When some resident updates their arrival time, we update $prev[i]$, $cur[i]$ and $change[i]$. To answer a query i, we can compute the answer directly.

Let $t' = \min(change[i-1], change[i])$ denote the time of the earlier change, and let $t'' = \max(change[i-1], change[i])$ denote the time for the later change. There are three time segments we need to consider: - from time 0 to the first change t', where the number is $t'+1$ if $prev[i] < prev[i-1]$ or 0 otherwise - from the earlier change t' to the second change t'', where we need to distinguish two cases: - for $change[i-1] < change[i]$: the number is $(change[i] - change[i-1])$ if $prev[i] < cur[i-1]$ or 0 otherwise - otherwise, it is $(change[i-1] - change[i])$ if $cur[i] < prev[i-1]$ or 0 otherwise - from the later change t'' to time i, where the number is $(i+1-t'')$ if $cur[i] < cur[i-1]$ or 0

Summing all numbers gives us the solution.

This runs in $\mathcal{O}(N+D)$ time.

Full Solution

Using a similar approach from the previous subtask, we could accumulate the correct numbers for all time segments between consecutive changes for every `?` query. However, this is too slow since return times can now change often. Instead, we now update $w[i]$ continuously whenever one of the following events happen: 1. There is a query for i, 2. the schedule of resident i changes or 3. the schedule of resident $i-1$ changes.

For each resident i we keep: - $t[i]$, the current arrival time - $w[i]$, the number of days resident i has watered for resident $i-1$ - $lu[i]$, the last day when $w[i]$ was updated (initially 0)

When we need to *update* some $w[i]$ on some day d, this is done in the following way: We check whether currently, resident i comes home before $i-1$ ($t[i] < t[i-1]$). If yes, we add $d - lu[i]$ to $w[i]$ and set $lu[i] = d$.

The solution then works as follows:

When we process an update event `! i t` at the end of day j, we update both $w[i]$ and $w[i+1]$ as described above, and then adapt the arrival times $t[i]$ accordingly.

When we process a query event `? i` at the end of day j, we update $w[i]$ and then output the value of $w[i]$ as the result.

This solution runs in $\mathcal{O}(N+D)$ time.