Sprinklers Editorial

Subtask 1

We cannot water all flowers unless all of them lie on the same side from the only sprinkler as sprinkler cannot spray both ways. Note that flowers with exactly the same position as the sprinkler will always be watered. If it is possible to water all flowers, the required spraying power is $\max_i |f_i - s_1|$.

Subtask 2

In this subtask, we can pretend that each group of three sprinklers is one two-sided sprinkler, i. e. a sprinkler which waters flowers on both sides. Using this simplification, we can find the sprinkler closest to each flower — the required spraying power will be the maximum across these distances — $\max_i \min_j |f_i - s_j|$.

One of the possible correct sprinkler configurations is `LRRLRRLRR...` — having both directions in one group will not be worse than any other configuration.

Subtask 3

There are only $2^N \leq 1024$ possible sprinkler configurations so we can try them all. For each configuration we find the minimum spraying power required to water all flowers. We then choose the configuration with the lowest spraying power.

To get the required spraying power for a given configuration, we can reuse the formula from Subtask 2 — $\max_i \min_j |f_i - s_j|$. However, this time we only consider the j-th sprinkler, if it is turned toward the i-th flower. The time needed is $\mathcal{O}(2^N NM)$.

A different approach uses binary search to find the minimum spraying power. For each configuration we ask if we can water all flowers with the given spraying power X. If it is possible for at least one configuration, then we know that the answer is at most X. The time needed by the binary search method is $\mathcal{O}(2^N NM \log K)$, where K is the correct minimum spraying power.

Subtask 4

If we know that the required spraying power is K, than for each flower there are at most $K-1$ other sprinklers between the given flower (denoted F) and the closest sprinkler (denoted S) that waters it. If there were more sprinklers, some of them would have to have exactly the same position as flower F or another sprinkler. If there were multiple sprinklers at the same position, we could configure them to spray in opposite directions and at least one of them would water flower F and be closer than sprinkler S. Similarly for the case, when a sprinkler shares its location with flower F.

Now, given a spraying power K, we can find a satisfying configuration using dynamic programming. For the i-th sprinkler and each of the 2^K configurations of the last K sprinklers, we determine if it is possible to water all flowers to the left of the i-th sprinkler. To determine if it is possible to water all flowers, we pretend that there is a an additional sprinkler located at positive infinity. The time needed for this is $\mathcal{O}(2^K(N+M))$.

(Note: There are multiple correct ways to do the dynamic programming. For example, we could look at each flower and the configuration of the sprinklers around it.)

Subtask 5

We say sprinklers $a_1, a_2, \ldots$ are in range (with respect to spraying power K), if and only if $\max_{i,j} |s_{a_i} - s_{a_j}| \leq K$. (In other words, they can water each other in some configuration.)

We observe the following: If there are three consecutive sprinklers within range (with respect to spraying power), they will not spray in the same direction. If they did, we could turn the middle sprinkler, without risk of any flower becoming unwatered.

We can use this observation to do dynamic programming with a constant number of states. For the i-th sprinkler and a configuration of the last few sprinklers, we determine the minimum spraying power required to water all flowers to the left of the i-th sprinkler with the given end configuration. (And, similarly to the previous subtask, we pretend there is an additional sprinkler located at positive infinity.)

There are multiple ways to pick which end configurations we will consider. For example, we could check all possible configurations of the last three sprinklers — which is enough thanks to the previous observation. Or we could check the following:

- `R` the i-th sprinkler is turned right.
- `RL` the i-th sprinkler is turned left and the $(i-1)$-th sprinkler is turned right.
- `LL` the i-th sprinkler is turned left and the $(i-1)$-th sprinkler is turned left.

It is also possible to do dynamic programming by looking at each flower and the surrounding sprinklers. We could also use binary search to find the spraying power and use the dynamic programming to determine if a given spraying power is sufficient.

The time needed is $\mathcal{O}(N+M)$ for the dynamic programming, and $\mathcal{O}((N+M)\log K)$ if we do binary search. The choice of end configurations determines the multiplicative constant.