MCC 2024 Problem 5: Exploding Arrow Solution

Problem idea by: Neo Yong Li

Problem prepared by: Loh Kwong Weng

Editorial by: Go Jun Xing

Slow $O(N^2 \log N)$ solution: Binary search and simulate

Note that if we can destroy all targets with power X, we can also do so with power $\geq X$. Thus, we can binary search on X. For a fixed X, we want to check whether we can destroy all targets.

Note that one optimal strategy is to always greedily shoot the target with the smallest index until it is destroyed. Thus, for a fixed X, we can simulate this in $O(N^2)$: For each target from small to large index, calculate the number of times we need to shoot the target, then update the HP of affected targets naively in $O(N)$ time.

The final time complexity is $O(N^2 \log N)$, where $\log N$ comes from binary search and N^2 comes from the simulation. However, this is too slow to solve all tasks (at least if the time limit is 2 seconds like in competitive programming contests).

Full solution: $O(N \log N)$

For an arrow shot at target i, the damage received by target j is $\max(0, XM - (j-i)^2)$. We can also read this as:

- For an arrow shot at target i, for j where $i \leq j \leq i + \lfloor\sqrt{XM}\rfloor$, target j receives $XM - (j-i)^2$ damage.

Furthermore, we can rewrite $XM - (j-i)^2 = XM - j^2 + 2ji - i^2$.

Let's go through all targets j from $j = 1$ to $j = n$ and destroy them. Fix the current j; we haven't shot target j and want to destroy it. Before this, we would have shot some arrows at targets $< j$. For convenience, call an arrow shot at target i "arrow i". Let arrow x_1, arrow x_2, ..., arrow x_m be the multiset of arrows we have shot that can reach target j (i.e. $x_i \leq j \leq x_i + \lfloor\sqrt{XM}\rfloor$). Then the total damage received by target j so far is

$$\sum_{p=1}^{m} XM - j^2 + 2jx_p - x_p^2 = mXM - mj^2 + 2j\sum_{p=1}^{m} x_p - \sum_{p=1}^{m} x_p^2$$

Let $s = \sum_{p=1}^{m} x_p$ and $t = \sum_{p=1}^{m} x_p^2$. Note that we only need s and t in the formula above (and not the entire set of x_p). Knowing the damage received, we can know how many more arrows to shoot at target j. We can also maintain s and t fast as j increases. Whenever we shoot arrows at target y, we will store

an "event" at target $i + \lfloor \sqrt{XM} \rfloor$ telling us to remove those arrows from the set of x_p when we reach that target.

Now that we know the total damage received by target j so far (call it d), the number of arrows we need to shoot at target j is $\lceil \max(0, \frac{a_i - d}{XM}) \rceil$. Add this to the number of arrows shot and update s and t.