Code Jam to I/O for Women 2019 - Code Jam to I/O for Women

Analysis: Parcel Posts

Test set 1

Test set 1 has a small upper bound of only 10 for **K**, which leaves at most 9 places where a post can be placed. That means we can just try every one of the $2^9 = 512$ combinations. For each one, check whether all of the produced parcels are valid, and if they all are, update a result if the number of posts in this option is larger than it. To check whether a parcel is valid, notice the condition is equivalent to checking if the list of elevations is not sorted non-decreasingly nor non-increasingly. We can do that by either sorting it in both ways and compare whether either is equal to the original, or check there exist two consecutive marks with one strictly lower than the other, and also two consecutive marks with one stricly higher than the other.

Test set 2

We can improve upon the exponential time required for the solution in the previous paragraph with the following insight: if a parcel between marks i and j is valid, so is any extension of it to a parcel between marks i-d and j+e for non-negative d and e. So, let i be the westernmost mark such that the parcel between marks 0 and i is valid and the parcel between i and **K** is valid. If there is not such an i, then the result is just 0 (no posts can be added). We can then prove that there is a way to place a maximum number of posts that places a post at mark i and not at any mark west of i. If that's true, a greedy algorithm follows: find such i, place a post there, and recursively try to place more posts within the [i, **K**] parcel. This algorithm takes $O(\mathbf{K}^2)$ time if implemented as described, which is fast enough to solve this test set, but a linear time alternative is presented in the next paragraph.

To solve the problem in linear time, we scan the list of elevations from west to east, recording when we see consecutive marks in strictly increasing and strictly decreasing order. As soon as we have seen both, we found a potential i (we don't check if [i, **K**] is valid just yet, to save time). We add 1 to the result, reset whether we have seen markings in each order to false, and resume scanning (this is equivalent to the recursive step in the previous presentation). After the scan is complete, it's possible that the eastmost parcel is not valid (since we didn't check it), but we can check it by inspecting the value of the same two boolean variables showing whether we saw markings in each order since our last reset. If the westmost parcel is not valid, the last place we found for a post was not valid, so we subtract 1 from our running result. Notice that each time we find a place where we can place a new post, it means that all previously placed posts were valid. This takes a single linear pass and a constant number of updates and checks of 3 variables at each step (two boolean variables about having seen markings in each order, and the result), so the algorithm runs in linear time.

To prove the main proposition, notice first that the parcel between marks 0 and j for j < i is never valid by definition of i, so any valid placement of posts does not place a post at mark j < i. For the second part, since just i is a valid placement, there exists a non-empty valid placement. Let $L = i_1, i_2, ..., i_m$ be the list of mark positions where posts should be added in a valid placement of a maximum number m of posts, in increasing order. Then, $i \le i_1$ because of the first part, so we can define $L' = i, i_2, ..., i_m$. Now we can show L' is also such a list, and one that has i as its westernmost post. L' has m posts by definition. The validity of L' follows because the parcel between marks 0 and i is valid by definition of i, and the parcel between i and i_2 is an extension of the one between i_1 and i_2 (because $i \le i_1$ from the previous part). All other parcels in L' are also parcels in L, and are therefore also valid, so L' is valid.