Week 1 D

> [!WARNING] Direct quote from solution notes contain words of venting

```
/*
 * Day 1 must be 0
 * Each day a maximum of 1 extra mark is made
 * When water level reaches 0, it either
 *  remains on the top mark; or
 *  makes a new top mark
 *
 * Subproblem:
 *  prev_max
 *  prev_max_idx
 *  max_mark
 *  for i = 0 -> n
 *  if (curr > max_mark)
 *      // max mark has been updated
 *      // recall that only way to grow is either 0 or max_mark
 *      // hence we can reimbue 0s in the smallest fashion
 *      agg_max = max_mark - 1
 *      for (j = i -> prev_max_idx)
 *          // greedily remove 1 per 0 occurence
 *          if (marks[j] == 0)
 *              sum += agg_max
 *              agg_max -= 1
 *          else
 *              sum += (agg_max - marks[j])-
 *      // assert(agg_max = prev_max)
 *      prev_max_id = i
 *      prev_max = curr
 *      max_mark = curr
 *  else
 *      // if no new limit, do nothing
`*/

/*
    The above is wrong
    It has ignored the possibility of new `max_value`
    actually <= the number of marks created in total
    Counter this by adding
        if (max_marks[j] == marks[j] && max_marks[j - 1] < max_marks[j])
            agg_max -= (agg_max == max_marks[j]);
 */
```

As observed from the pseudocode above, I have misunderstood the question at first. The number of marking above the waterline does not necessarily represent the quantitative height of the water, but really the number of different water positions.

The greedy strategy here is to make the water rise as late as possible, therefore we could, as described above, ensuring that the number of total marking is at least the number of markings above the water + 1, equivalence taken on the day with “maximum markings”. The maximum markings per day could be precomputed as we take the input. A backward pass with such calculations would yield the least number of total markings under the water across all days. This solution’s difference with my original understanding is that my original solution will only decrement on the days where either marking is 0 or a new maximum, which have me treating it as a convex hull problem instead and the solution is a lot more complicated.

However the total runtime of the two algortihms are the same, both being O(n + n) = O(n).