2025 Pacific Northwest Division 1 Solutions

The Judges

November 15, 2025

Bouquet of Balloons

Problem

- You're competing in a contest where problem i takes s_i minutes to solve. You get a balloon with one liter of helium that deflates at the rate $\frac{1}{d}$ liters per minute. If your goal is to have balloons that aggregate to at least m liters of helium, compute the minimum number of problems to solve to attain this, or report it is impossible.

Bouquet of Balloons

Observations

- If it is possible to do it by solving k problems, you can do it by solving the problems with the k smallest solve times.
- You should solve problems in decreasing time order.

Solution

- Sort the problems in increasing order of solve time. The amount of deflated air obtained by solving k problems can be done by computing the sum of the first k prefix sums of solve times. We can use this to determine whether the goal is attainable by solving k problems.

Bus Seating

Problem

- m people want to sit on a bus with n rows, each with k seats. Each person has a target row r_x they want to sit in, and they'll derive utility $\frac{C-|r_x-r_y|}{2^p}$ if they sit in row r_y that already has p people in it. Compute which row every person ends up in.

Bus Seating

Observation

- Because $C \leq 10^9$, it is never valid to consider rows that have 31 or more people than the row with the fewest number of people.

Solution

- We maintain, for a count of number of occupied seats in a row, an ordered set of the rows that have that many people.
- We therefore only need to consider at most 30 occupied row counts.
- Among all such rows, we pick the optimal one and update our data structure accordingly.

Closest Equal Pair

Problem

- Define $f(a)$ on an array a to be zero if all elements in a are distinct. Otherwise, define it to be the minimum value of $j - i$ where $i < j$ and $a_i = a_j$. Given an array of n integers, compute the sum of f over all subarrays.

Solution

- Sweep over the indices from left to right. Maintain a monotonic stack of $(\texttt{score}, \texttt{index})$ where `score` is active for all indices less than or equal to `index`. We can also maintain the running sum for all f with a fixed rightmost endpoint. When we append a new index, we may need to pop off elements from this stack that get subsumed and update the running sum.

Farthest City

Problem

- Given a connected weighted graph of n vertices and n edges, compute the distance of the farthest vertex from each vertex.

Solution

- A connected graph with n vertices and n edges is a graph with exactly one cycle.
- We start by computing, for each vertex on the cycle, the length from each vertex to the deepest leaf in the subtree hanging off the cycle. We then compute, for each vertex on the cycle, the length from each vertex to the farthest leaf. We finally propagate to each subtree distances from the cycle vertex downward.
- The second step is the most complicated part and can be done in linear time by looping over all indices in cycle order and keeping a max queue. $\mathcal{O}(n \log n)$ solutions can also pass.

Fractal Painting

Problem

- An infinite fractal is set up where a segment has two outgoing segments drawn out of it, and each of the outgoing segments recursively has similar outgoing segments going out of it. Determine if the infinite fractal fits in a finite size rectangle.

Solution

- The infinite fractal extends infinitely in the following cases:
  - Either of the two recursive segments is longer than the original.
  - One of the recursive segments points in the same direction as the original segment and the same length.
  - Both segments are the same length as the original segment.
- We can show that, in all other cases, the fractal is bounded.

Friendships

Problem

- n children are incrementally given toys and become friends with each other.
- Quickly answer queries of the form - compute the maximum number of toys someone who is not the friend of a given child has.
- Each child may have at most 50 toys.

Solution

- Because the maximum number of toys is small, we can maintain for each child, the number of their friends that have a specific toy count.
- To quickly answer queries, we iterate from 50 to 0 and see all friends can account for the number of people with a specific toy count.

Honkai Stress Relief

Problem

- Given an interval (a, b) and n intervals (s_i, e_i), compute the probability that if we sample a value independently and uniformly at random from (a, b) for each of the given n intervals, at least one of the sampled values will be outside its corresponding intervals.

Honkai Stress Relief

Solution

- We compute the complementary probability - the probability that none of the sampled values are outside the corresponding intervals, or that all sampled values are inside their corresponding intervals.
- We therefore only need to solve whether a sampled value in (a, b) is in (s_i, e_i).
- Compute the intersection of the two intervals, the length of the intersection divided by $b - a$ is the probability of the sampled value being in the interval.
- The final answer is therefore 1 minus the product of all the above probabilities.

Kth King

Problem

- We define an operation on an element of the array as incrementing or decrementing the element by 1.
- For a constant k, we define $f(a)$ over an array a to be the minimum number of operations such that all subarrays b of the array a of length greater than or equal to k have the same k^{th} largest element.

Kth King

Solution

- We first prove that for a given k, it is enough to modify the array such that all the elements besides the largest $k-1$ must be the same.
  - Suppose we have greater than or equal to k largest elements that are not the same as the rest of the array.
  - Take the subarray b as the entire array a. This subarray will have a k^{th} largest element equal to the k^{th} largest element of the entire array.
  - However, there exist a subarray c of length k such that the k^{th} largest element is not in the k largest elements.
- Thus, for each k, we want to change every element besides the largest $k-1$ to the same value. This is a known problem where setting all elements to the median value is minimal.

Kth King

Solution part 2

- We can speed up the calculations for finding the number of operations to set a range of elements using prefix sums.
- First sort all of the elements and then calculate prefix sums over the array.
- The cost of setting the first x elements to the median can be done:
  - Let $m = \text{array}[x/2]$ be the median.
  - Take the sum of the elements to the right of the median minus the number of elements on the right times the median to calculate the cost to move all elements above the median to the median.
  - We can use the same idea for the elements below the median.
- The ending time complexity is $\mathcal{O}(n \log n)$.

Pair Linked Mokepon

Problem

- You and your friend are each walking along a personalized linked list. Items can be in certain nodes on the linked list that make it possible for someone to enter a specific node on a given linked list. Compute the number of distributions of items that make it possible for both people to make it to the end of their linked list.

Solution

- For a given distribution, imagine that we greedily make your friend do all their moves if both of you are able to advance. We will do a DP based on this state along with the person who moved last.
- If your friend moves after you move, then you must have discovered that item in your most recent move.
- In all other transitions, the item can be discovered among any of the reachable nodes.

Server Room

Problem

- Given a ternary grid, compute the maximum number of 1's that can be flipped to 2's where no 2's are adjacent. Furthermore, compute the number of distinct grids that can be obtained flipping that many 1's to 2's.

Solution

- We transpose the grid so the number of columns is less than or equal to the number of rows.
- Sweep the grid in row-major order. We maintain a bitmask of the last c entries of the grid, and track both the maximum number of 1's flipped along with the number of ways to get to that state.

Shh

Problem

- Given a string, compute the minimum number of characters to change such that it has exactly k instances of the substring `shh`. Furthermore, compute the number of distinct strings with exactly k instances of `shh` that can be obtained by changing exactly that many characters.

Solution

- We solve this with dynamic programming. The state to maintain is the length of the prefix, the number of full occurrences of `shh`, and the length of the longest suffix that is a strict prefix of `shh`, and we maintain both the minimum number of characters needed to change to get to this state along with the number of ways to get to this state.

Training, Round 4

Problem

- You are given n lattice points (x_i, y_i). You can start at any point (X, Y). On turn i, both $X \geq x_i$ and $Y \geq y_i$ must be true. Then, you must increment exactly one of X and Y. Determine the minimum possible value of $X + Y$ such that this condition is true over all n turns.

Solution

- We maintain a lower bound of X and Y over all turns, along with a count of increments we have not used yet.
- When we see a new point, we consume as many increments as possible, and then update X and Y if needed.
- The final answer is therefore the current sum of X, Y, and any unused increments, minus n.

Triangle of Triangles

Problem

- Given a triangle T and two triangles T_1 and T_2, is it possible to draw a line segment on T that splits the triangle into two subtriangles, both of which are similar to either T_1 or T_2?

Solution

- We solve the reverse problem - is it possible to combine two triangles to form T? We can then use this primitive to the three distinct combinations.
- To combine two triangles, one angle from each must sum to 180 degrees.
- We brute force over all such combinations of angles.