OCPC 2025 Winter Day 8: Zagreb Contest 2 Analysis

2025.2.18

A. Arrow Array

Description

Construct a string of left and right arrows to satisfy the constraints of there being at least a_i arrows pointing towards the i-th gap between 2 arrows if $a_i > 0$ or at least $-a_i$ arrows pointing away from it if $a_i < 0$.

- $2 \leq n \leq 3 \cdot 10^5$

Solution

- Fix the total number of right arrows, let it be k.
- Now each constraint from the statement is either a lower or upper bound on the amount of right arrows in each prefix, where increasing k by 2 increases the bound by 1. So, let's look at odd and even k separately.
- There is a solution for this k if we can start at the lower and upper bound being 0 at the prefix of size 0, and each time we increase the size of the prefix by 1 we increase the upper bound by 1 and update it by the lower or upper bound mandated by the constraint, we never end up with the upper bound being lower than the lower bound, and k is inside the bounds at the end.

Solution

- Increasing all bounds by the same number(let's call it s) doesn't affect whether some lower bound affects some later upper bound, so we can just check if this happens with the initial bounds by starting with a lower bound of $-\infty$ and an upper bound of ∞.
- After this, we just need to check if the initial lower and upper bound of 0 makes the answer impossible or not and if k is inside the bounds at the end.

Solution

- To check if k is inside the bounds at the end, look at the bounds at the end with the initial bounds and starting with a lower bound of $-\infty$ and an upper bound of ∞, and check if adding s to both would make k inside the bounds.
- To check if the initial lower and upper bound of 0 makes the answer impossible or not, we just check that the minimum upper bound is $\geq -s$ and that the maximum (lower bound)-i is $\leq -s$.
- The total time complexity is $O(n)$.

B. Big Balls

Description

Given n balls in a line, sorted by their numbers from left to right. Each ball has a weight w_i and a direction (left or right).

Balls move simultaneously. When two balls collide, they merge into a big ball with weight $w_i + w_j$, and the direction of the heavier ball (if they are equal, it goes right).

Find the sums of balls that end up going left, and right.

Solve for q independent queries: we consider only balls l, $l + 1$, ..., r.

- $1 \leq n, q \leq 10^6$, $1 \leq w_i \leq 10^9$

Solution

- Simulate the process for the entire array. Create a binary tree (forest) of merge events. We can force this to be a tree by adding dummy balls on array endpoints with infinite weight.
- The set of events that happen if we consider a subarray is a subset of events if we consider the entire array.
- Every event can be represented as $L \leq p < R$, meaning that ball containing $[L, p]$ merges with $[p + 1, R]$.

Solution

- Let u be the LCA of $l-1$ and $r+1$. It was created by merging $[L,p]$ with $[p+1,R]$.
- Every initial ball from $[L,p]$ (that is within the query interval) will end up moving to left.
- This is because every ball from $[L,p]$ will at some point want to collide with a ball coming from its left side, but that ball won't exist if we only consider the query range $[l,r]$.
- All other balls will move to right.
- $\mathcal{O}((n+q)\log n)$

C. Crooked Cycles

Description

Given m unordered triples (a_i, b_i, c_i) check whether there exists a cyclic sequence

$$v_1, \ldots, v_k$$

such that every three consecutive numbers form a triple Nežmah likes.

- $1 \leq m \leq 2 \cdot 10^5$

Solution

- Let's build a graph with ordered pairs as node.
- We will connect two pairs (u, v) and (v', w') with a directed edge if $v = v'$ and there exists a triple (u, v, w') Mr. Nežmah likes.
- Now every cycle forms an actual directed cycle in this graph.
- To check whether there exists a crooked cycle, we can try labeling the directed graph with numbers 0, 1, 2 so that every directed edge corresponds to adding one modulo 3.
- If we succeed, there is no crooked cycle. Otherwise, there exists at least one. The total complexity is $O(m)$.

D. Dangerous Data

Description

This is a run-twice problem. Given a set of $n + k$ numbers S, construct some array b of length k, so that when given any subset of S with size n, we can reconstruct the missing k numbers using array b.

- $1 \leq n \leq 10^5, 1 \leq k \leq 10$
- $1 \leq x \leq 10^6$ for all $x \in S$
- $0 \leq b_i < 2^{20}$

Solution

- Notice the gap between the bound on $x \in S$ and the bounds on b_i. Let p be a prime such that $10^6 < p < 2^{20}$.
- Motivated by Vietta's formulas, let's try sending the following numbers, calculated with dynamic programming modulo p.

$$\sum_{x \in S} x$$

$$\sum_{x,y \in S, x<y} x \cdot y$$

$$\sum_{x,y,z \in S, x<y<z} x \cdot y \cdot z$$

Solution

- Now on the second run, let B denote the given set of n numbers and $A := S \setminus B$. By calculating the same sums as in the first run but for B, and doing a bit of dp, we can extract the same Vietta sums for only the set A.
- Now with appropriate signs, these numbers form the coefficients of a polynomial P whose roots are exactly the elements of A.
- As 10^6 is small enough, we can evaluate $P(x)$ for all x in $1, \ldots, 10^6$ and take the roots.
- Bonus: try solving for $0 \leq x < 2^{20}, x \in S$ and $k \leq 10^5$.

E. Exchange Error

Description

Given an array a of size n and q values of x_i, find what the maximum subarray sum would be if all elements were increased by x_i for all i.

- $1 \leq n, q \leq 2 \cdot 10^5$, $-10^8 \leq a_i, x_i \leq 10^8$

Solution

- Divide the array into blocks of size K and sort the values of x_i. Let ps_i be the sum of the first i elements of a.
- To consider the subarray sums that are completely inside a block, we find the maximum one of each length, put $y = lx + s$ into a convex hull where l is the length and s is the maximum subarray sum of that length for all l from 1 to k, and use 2 pointers to find the maximum value for each x_i. This takes $O(K^2 + q)$ time per block, so $O(nK + \frac{nq}{K})$ time overall.
- The subarray sums which aren't completely inside a block are equal to $ps_i - ps_j + x(i - j)$ for some i in a block and some j in an earlier block.

Solution

- Let's fix the block that i is in. We can put all $ps_i + xi$ from this block into a convex hull to find the maximum value for each query using 2 pointers in $O(q + K)$ time per block.
- We can put all $-ps_j - xj$ from all the previous blocks into a convex hull to find the maximum value for each query using 2 pointers in $O(n + q)$ time per block.
- Overall, the complexity is $O(n(\frac{n+q}{K} + K))$.
- Take $K = \sqrt{n+q}$ for a time complexity of $O(n\sqrt{n+q})$.

F. Flipping Frenzy

Description

You are given an $n \times m$ grid of 0s and 1s. and k pairs (a_i, b_i). In one operation, you can take a rectangle with height a_i and width b_i for any i that touches the left or top order of the grid and flip all values inside that rectangle. Find a sequence of operations to turn all values into 0s.

- $2 \leq n, m \leq 1000, 1 \leq k \leq 20$

Solution

- Let's look at a different basis for the operations.
- If we denote by $C(i, j)$ flipping the first a_j tiles in the i-th column and by $R(i, j)$ flipping the first b_j tiles in the i-th row, then obviously all operations we can do can be represented in terms of these row and column operations.
- They aren't all linearly independent; to obtain a basis, we need to exclude one of $C(b_i, j)$ or $R(a_i, j)$ for all i, j. Let's exclude all $R(a_i, j)$.
- We can uniquely find which elements of this basis need to be done and which ones don't.

Solution

- Let's now consider all of the row and column operations, including the ones which aren't in the basis.
- The operations we can do can now be represented as doing $C(1,i), C(2,i), \ldots, C(b_i,i)$ for some i(flipping the top-left rectangle), doing $C(x,i)$ and $C(x+b_i,i)$ for some x,i(moving a rectangle on the top by 1 column to the right), doing $R(x,i)$ and $R(x+a_i,i)$ for some x,i(moving a rectangle on the left by 1 row down), and doing $C(1,i), C(2,i), \ldots, C(b_j,i), R(1,j), R(2,j), \ldots, R(a_i,j)$ for some i,j(putting the operations $R(a_i,j)$ into the basis)

Solution

- We now need to check if a bitstring can be made a linear combination of other bitstrings, most of which have exactly 2 ones.
- Let's make a variable for the $k(k+1)$ bitstrings which don't have exactly 2 ones, and for the ones which do let's look at the 2 ones they have as an edge.
- Now each connected component gives us one equation for those $k(k+1)$ variables.
- After solving the system of $O(k(n+m))$ equations with $k(k+1)$ variables in $O(\frac{k^5(n+m)}{w})$, we now have everything we need to reconstruct the operations.

G. Grid Gradient

Description

You are given a n and m. How many ways are there to fill up an n times m grid with numbers $1, 2, 3$ and 4 so that the absolute difference of numbers written in cells sharing a side is exactly one.

- $1 \leq n, m \leq 24$

Solution

- Let's notice that neighbours have to be of different parity, so the parity forms a chess coloring. WLOG let the upper left corner contain an even number.
- Now let's observe the grid with only numbers 2 and 3. It obviously forms a valid grid. For a valid grid with the same parities, let S be the set of cells which are different from the $2, 3$ grid.
- Notice that no two cells in S can share a side, and that this is a complete characterization of S.
- Now we only need to count independent sets of the grid graph.

Solution

- We can do a broken profile dp. Our state will include the current cell, and a bit mask representing cells in S which are in the prefix of the current row, or the suffix of the previous row.
- Notice that there can be at most two consecutive elements in the bit mask, so the number of possible bit masks is bounded by $F_{<+1}$.
- This gives a total complexity of $O(nm \cdot F_{m+1})$.

H. Halfway Hawser

Description

You are given a string s. Out of the $|s|$ strings you can obtain by removing 1 character from it, find the lexicographical median.

- $2 \leq |s| \leq 10^6$

Solution

- In a block of consecutive equal characters, all of the strings you obtain by removing one are equal. Let's now assume that every 2 adjacent characters are different.
- If the first character is smaller than the second character, removing it will give the lexicographically highest string.
- If it's larger, removing it will give the lexicographically smallest string.
- Using this, we can find the sorted order of the strings we obtain, from which we can easily find the median.
- The total time complexity is $O(n)$.

I. Intricate Instrument

Description

A 0-indexed array s is good if it's strictly increasing and there exist a, b, c, d such that $s_{a+bi \mod n} = c + di \mod m$. You're given a good array with some elements replaced by -1, find a, b, c, d that could create it.

- $3 \leq n \leq 3 \cdot 10^5, n \leq m \leq 10^9$, n is prime

Solution

- If the array is an arithmetic progression with step $\frac{m}{n}$ or if there is at most 1 value which isn't -1, the solution is obvious.
- Let's say that we know that $s_u = v$, then we can transform s_i into $s_{i+u \mod n} - v \mod m$ which only changes a and c, now we have $s_0 = 0$ but we need to make sure $s_{n-u} \geq m - v$ and $s_{n-u-1} < m - v$. This just changes the values of a and c.
- For a strictly increasing array, being a good array is equivalent to it either being an arithmetic progression with step $\frac{m}{n}$, or, for each $1 \leq k < n$, there being exactly 2 different values of $s_{i+k \mod n} - s_i \mod m$.

Solution

- Let's denote the 2 different values for $k = 1$ as x_1 and y_1, where $x_1 < y_1$, and let's say that x_1 appears c_1 times.
- All values of $\frac{ns_i - mi}{y_1 - x_1}$ are distinct integers between $-t$ and $n - t - 1$ for some $0 \leq t < n$, and their values are equal to $-c_1 i$ modulo n.
- If we take the GCD of the known values of $ns_i - mi$, if there are o known values of s_i other than s_0, the number we have to divide the GCD by is $\leq \frac{n-1}{o}$, since dividing $y_1 - x_1$ by some number multiplies all $\frac{ns_i - mi}{y_1 - x_1}$ by the same amount, and we know that the range of that is $\leq n - 1$. Let's try all options.

Solution

- If we know $y_1 - x_1$, then we know $\frac{ns_i - mi}{y_1 - x_1} \equiv -c_1 i \pmod{m}$ for some i, so we know c_1.
- Now the known values of s_i give us some bounds on t, and so do the conditions of $s_{n-u} \geq m - v$ and $s_{n-u-1} < m - v$.
- If the bounds are nonempty, we can take any t inside the bound to determine all values of $\frac{ns_i - mi}{y_1 - x_1}$, which gives us the entire array s, from which we can easily find a, b, c, d by looking at $k = c_1^{-1} \mod n$.
- All options take $O(o)$ time to check, so the complexity is $O(n)$.

J. Jolly Janes

Description

There are n girls and two bachelors.

Each bachelor has given roses to some girls, represented with binary strings s and t.

The first bachelor has A, and the second one has B roses left to give. Both of them will hand out all of their roses to girls who haven't yet received a rose from them.

A girl passes to the next round if she receives at least one rose. Find the expected number of girls that pass if each bachelor distributes his remaining roses randomly.

- $1 \leq n \leq 10^5$

Solution

- If a girl has already received at least one rose, her probability of passing is 1.
- Every other girl has an equal probability of passing.
- $p_A = \frac{A}{\#zeroes\ in\ s} \qquad p_B = \frac{B}{\#zeroes\ in\ t}$
- It equals $p_A + p_B - p_A \cdot p_B$
- By linearity of expectation we can sum this probability through all girls.
- Special case: the string has no zeroes. We cannot divide by 0, so just output 1.
- $\mathcal{O}(n)$

K. Kempt Kale

Description

Mr. Nežmah wants to grow n pieces of kale over $2 \cdot n$ days.

On day i he can plant kale with tastiness a_i, or harvest a piece of kale that was previously planted.

Find the maximum sum of tastiness over all harvested kales.

- $1 \leq n \leq 2 \cdot 10^5$, $1 \leq a_i \leq 10^9$

Solution

- Equivalent to finding a regular bracket sequence with maximum value, where value is the sum of a_i, where i-th bracket is open.
- Greedily try adding an open bracket on the positions from largest to lowest value.
- This follows from the matroid structure.
- Proof is left as an exercise to the reader.
- The total complexity is $\mathcal{O}(n \log n)$.

L. Loop Land

Description

We are given a puzzle with different types of blocks. Some blocks are empty, other allows some sets of ”arm” configurations.

We need to solve the puzzle.

- $1 \leq n, m \leq 1000$

Solution

- Let's build a graph with a node for every pair of cells sharing a side. Every node represents a place where an arm can be.
- A configuration of arms is equivalent to choosing a set of nodes S.
- Notice that all block types create three different types of restrictions on the set S.
  - Node u is in S.
  - Node v is not in S.
  - Exactly one of nodes u and v is in S

Solution

- For restrictions of the third type, add an edge between u and v. Now the graph is bipartite because a solution exists.
- For a given component, if there exists a restriction of the first or second type on any of the nodes, then this uniquely defines which nodes must be in S. Otherwise, choose an arbitrary side of the bipartite component.
- The total complexity is $O(nm)$.

M. Maximum Mex

Description

We are given a weighted tree with n nodes. Each value appears on at most six edges. We need to find the maximum mex across all paths in the tree.

- $1 \leq n \leq 10^5$

Solution

- Let us binary search the answer. For a given x we need to check whether there exists a path such that for each value in $0, \ldots, x-1$ there is at least one edge with that value on that path.
- As $x = 0$ is trivial, the path always includes an edge with value 0, let us fix one of them, for instance (u, v).
- We will ignore all edges with value $\geq x$. Now we are only interested in whether there exists a path with x distinct values.
- This edge splits the tree into two subtrees, we will do a dfs on one subtree, and maintain a segment tree over the other subtree sorted in the dfs order.

Solution

- The segment tree will hold the number of distinct values on the path from u to that particular node.
- As we are doing the dfs over the other subtree, we want to "turn off" some values in the segment tree. If we encounter an edge with value y, then edges with value y in the other subtree do not add new anything to our paths, so we need to subtract one from them. As there are at most five other edges, this can be done with five updates on the segment tree.
- With careful analysis, this turns out to be $O(n \log^2 n \cdot k)$ where k is the maximum number of appearances of a value on the edges, 6 in this case.

Thank you!