Fall 23 Div I Week 2 - Solution Sketches

(taken from editorials of original problems)

Problem A

(Source: 2022-2023 ACM-ICPC Nordic Collegiate Programming Contest (NCPC 2022) Problem H)

Problem

Given a sequence of integers $h_1, h_2, \cdots, h_n$, a *hill* is defined as an interval where $h_i \leq h_{i+1} \leq \cdots \leq h_j \geq h_{j+1} \geq \cdots \geq h_k$. The height of a hill is defined as $\min(h_j - h_i, h_j - h_k)$. Find the highest hill.

Solution

1. For every index i, calculate $L(i)$, the minimum j such that $h_j \leq h_{j+1} \leq \cdots \leq h_i$.
2. Similarly, calculate $R(i)$, the maximum j such that $h_i \geq \cdots \geq h_j$.
3. The height of the hill with peak at i is $\min(h_i - h_{L(i)}, h_i - h_{R(i)})$. Take the maximum of these.

Problem B

(Source: 2022-2023 ICPC, NERC, Northern Eurasia Onsite Problem E)

This is an easy problem. First, you need to index all blocks with distinct consecutive integers, e.g. from 0 to $m-1$, where $m = \sum_1^n k_i$ — the total number of blocks. Then, you can compute the minimal number of required split operations s as the number of places in the initial towers where a block with index x is followed by a block with index y and $x + 1 \neq y$. These are the only split operations that are absolutely necessary. Now, after s split operations, the number of towers will become $n + s$, so in order to combine them into a single tower, you need to perform $c = n + s - 1$ combine operations.

Problem C

(Source: 2022-2023 Winter Petrozavodsk Camp, Day 2: GP of ainta Problem G)

It is enough to use only operation 2 and operation 3. Consider state of the grid M after executing operation 2 for all cells. If a cell's color is black in M, then changing its color to white without changing other cells could be done by replacing the operation 2 on the cell by operation 3. Therefore, after doing operation 2 on white cells in M and doing opration 3 in black cells in M, we could obtain a grid with every cell colored white.

Problem D

(Source: 2022-2023 Winter Petrozavodsk Camp, Day 4: KAIST+KOI Contest Problem B)

The quantity $strength(S)$ is a sum of $\binom{|S_i|}{2}$ where S_i are the connected components of S under the given tree. Naively, you can scan all tree edges where both endpoints belong to S and compute the components with graph search or disjoint sets, which requires $O(N)$ time per query and times out.

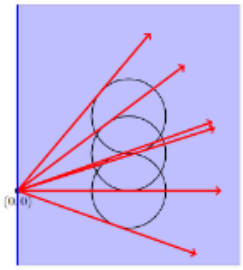
Performing DFS on the vertices of S may seem to work, but this approach may encounter many irrelevant edges, hence this also times out.

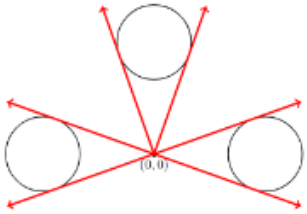
To improve the algorithm, root the tree arbitrarily, and compute the *parent* for all non-root vertices. An edge is relevant if and only if $par(v) \in S, v \in S$, therefore you can simply iterate through S and check if $par(v) \in S$ with a simple boolean array. The time complexity is $O(N + \sum K_i)$.

Problem E

(source: Osijek Competitive Programming Camp, Winter 2023, Day 9: Magical Story of LaLa (The 1st Universal Cup. Stage 14: Ranoa) Problem E)

- If any circles contain the origin in its interior or on its boundary, the answer is "YES".
- Otherwise, each circles have 2 tangent rays starting from the origin. The answer is "YES" if and only if no half-plane contains all $2N$ rays.





- It's safe to use doubles in computation due to the distance condition.

Time Complexity: $O(N \cdot \log N)$ from sorting rays.