

# Fall 24 Div Compete Week 1 - Solutions

(taken from editorials of original problems)

## Problem A

(Source: ETH Zurich Competitive Programming Contest Spring 2024 Problem L)

Solution

$\mathcal{O}(n)$

- ▶ Use station B as a stack
- ▶ Prioritize the A-C type operations

## Problem B

(Source: ETH Zurich Competitive Programming Contest Spring 2024 Problem B)

Solution

$\mathcal{O}(|V|)$

If there is a path with length more than 2, then we have a maximum matching with size more than 1.

Solution

$\mathcal{O}(|V|)$

For each vertex  $v$  count the number of subgraphs with maximum matching of size 1 it is in it, which is equal to  $2^{\deg(v)} - 1$ .  
By inclusion-exclusion answer is  $(\sum_{v \in V} 2^{\deg(v)}) - 2n - 1$ .

## Problem C

(Source: ETH Zurich Competitive Programming Contest Spring 2024 Problem H)

Solution	$\mathcal{O}(n \log n)$
<ol style="list-style-type: none"><li>Observe: <math>k</math> of each type possible <math>\rightarrow k - 1</math> of each type possible <math>k</math> of each type not possible <math>\rightarrow k + 1</math> of each type not possible <math>\Rightarrow</math> Binary Search</li><li>Solution will be between 0 and <math>n/m \Rightarrow \mathcal{O}(\log n)</math> checks</li><li>For a possible <math>k</math>: Take the <math>k</math> Mate of each type with highest expiration date, sort by expiration date, and check if every Mate is not expired when dispensed. <math>\Rightarrow \mathcal{O}(n)</math> per check (sort only at beginning) <math>\mathcal{O}(n \log n)</math> might also pass (sort in every check)</li></ol>	ETH Contest Spr

## Problem D

(Source: ETH Zurich Competitive Programming Contest Spring 2024 Problem A)

Solution	$\mathcal{O}(n \cdot \log(n) + q \cdot \log(n))$
<ol style="list-style-type: none"><li>Root the tree at vertex 0 and calculate the probability of a random walk from 0 to all vertices <math>v</math>, call it <math>P_v</math>.</li><li>Create a data structure to query the lowest common ancestor of two vertices <math>u, v</math>.</li><li>Let <math>c = LCA(u, v)</math>, we can now calculate the probability for <math>u, v</math> with <math>P_u, P_v</math> and <math>P_c</math> (With some special cases)</li><li>You cannot solve each query in linear time, TLE.</li></ol>	