Felipe Augusto Morais Silva / Maratona

Vou anotando o nome das classes dos problemas que eu conhecer

Problema A

Brute force

Gerar e salvar todas as combinações positivo/negativo (que vão ser no máximo 8 eu acho, mas de qualquer forma vai ser um N pequeno) dos 3 numeros (exemplo: 22 –5 22, -22 –5 22, 22, 5, -2) e salvar em um array essas combinações, depois, usando cada array a gente gera todas as combinações possíveis ({[5], [5,22]… com backtracking fica == O(2^n), mas como o N é pequeno não vai fazer mal), depois a gente checa se em alguma das possibilidades geradas dentro de cada array encaixa no que a gente quer (O(n)).

Talvez dê pra otimizar na hora de armazenar a gente possa armazenar em um hashmap a sequência como chave e o resultado da sequência como valor, e depois só checar o hashmap.

Talvez tenha um jeito ainda melhor e mais fácil de fazer com uma matemática simples também, mas como só tenho 1h30 aqui vai ser isso mesmo.

Problema B

Questão de Range Queries

Como vai até N vai ate 10^5 tem que ser melhor que O(N^2).

A gente pode usar uma priority_queue<pair<int,int>> pq; pra armazenar o par {papel, posição} e evitar os pares menores que X. A gente vai removendo os elementos da pq até até encontrar algum que esteja dentro do nosso intervalo.
O problema que parando pra pensar, esse guloso não funciona em alguns casos...

Talvez dê pra fazer com grafos checando rota por path. A gente faz um bfs pra ver se existe uma rota do elemento x da query {L, R} até algum elemento < X, se n'ao existir achamos a resposta, se existir a gente checa todos e pega o menor.

Geralmente esses problemas de range query tem algum algoritmo ou uma ideia ja conhecida pra otimizar, tipo kadane e soma de prefixo.

Problema C

Acho que esse eh mais ADhoc, mas não consegui pensar em nada só de bater o olho não... mas uma coisa que talvez tenha que ser feita independente da implementação é armazenar a soma dos containers adjascentes em algum lugar...

Problema D

Esse eu já vi antes,

Tem um limite pequeno pro número de divisores, então mesmo sendo de pra números que vão até 10^9, a quantidade de divisores de cada número ainda é pequena, e aí da pra checar todos.

https://codeforces.com/blog/entry/14463

Problema E

Só pensei em bruteforce, nem idéia de como fazer isso rodar em menos de O(assustador)

Problema F

O número mínimo de nodes vai ser o tamanho da menor arvore mais número máximo de nodes centrais entre a arvore a e b menos o número minimo entre as duas arvores mais os nodes opostos (se a menor arvore for a canhota a gente soma os nodes destros e vice versa), + : minTreeNodes + (max(a,b) - min(a,b)) + diffNodes.

Problema G

Com certeza não é só isso, mas um approach inicial seria:

Salvar os pares referentes às posições ocupadas por pretas e brancas em um hashmap e ir checando as somas de cada sub-área 1 por 1

Problema H

Kkk, essa nem tem o que falar né? Só fazer um palíndromo pras vogais.

Problema I

Nem idéia

Problema J

Testar pra todos os atletas em todos os tempos e pegar os com melhores rendimentos em cada tempo. Com hashmap da pra passar em menos de O(N^2), que é o necessário pra dar tudo certo no problema.

Problema k

Conheço 2 algorítmos de geometria computacional só por nome: Convex hull e line sweep, mas acho que nenhum deles serve aqui, deve ser só uma matemática com aquela formula de tronco de pirâmide. Mas não deu tempo de pensar.

Problema L

Não deu tempo de ler.