Seletiva individual 2019: Terceira prova

Nathan Benedetto Proença

23 de julho de 2019

Isenbaev's Number

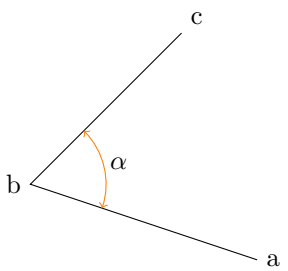
Construa um grafo com os nomes dados e com uma clique para cada time. Basta então fazer uma busca em largura a partir do Isenbaev para calcular as respostas.

Eu também tomei WA no caso em que não tem Isenbaev na entrada.

Divide an Island!

Quem diria, o único problema não resolvido!

O que mata o problema é observar que o triângulo dado já define a área e o perímetro da resposta. Seja $2l$ e $2A$ o perímetro e a área do triângulo dado. Vamos tentar construir uma resposta em torno de um dos lados do triângulo, como na figura abaixo.



Queremos andar uma distância l ao redor do ângulo em b e construir um triângulo de área A. Seja x a distância que cobrimos, a partir de b, no segmento $\overrightarrow{ba}$. A distância que cobrimos, a partir de b, no segmento $\overrightarrow{bc}$ é $l - x$. A área do triângulo formado é

$$\frac{x(l-x)\sin\alpha}{2}.$$

Como queremos que esse valor seja A, basta considerar as duas raízes (quando existem) do polinômio. Basta então repetir a mesma análise para os demais vértices.

O cuidado especial nesse problema é com precisão. As operações vetoriais como *cross product* e *inner product* funcionam bem, assim como a função `sqrt` do C. Precisa apenas pensar com carinho nos testes das desigualdades que surgem durante a implementação, e usar um `eps` de maneira que faça teu código de fato encontre as respostas.

Por exemplo, ao se considerar o discriminante do polinômio, caso ele seja muito próximo de zero, você deve garantir que teu código prossiga com o teste das duas raízes (que serão praticamente iguais), e não que diga que não é possível. Prosseguindo dessa maneira, não é complicado resolver o problema.

Vanya and Triangles

Todas as triplas de triângulos não colineares definem um triângulo. Basta então contar quantas triplas colineares existem. Para isso, seja p o ponto lexicograficamente menor. Ordene os demais por ângulo em torno de p. Ao fazer isso, pontos colineares estarão consecutivos na ordenação. Assim, basta percorrer ela de maneira linear e, ao se encontrar um conjunto de k pontos colineares, contabilizar

$$\binom{k}{2}$$

triplas de pontos degenerados — o ponto p e qualquer par dos pontos colineares.

Com isso, você contabilizou todos os "triângulos degenerados" que contém p em tempo $O(n \lg n)$. Remova p do conjunto e resolva recursivamente para ter uma solução $O(n^2 \lg n)$.

Halloween treats

Seja c a quantidade de crianças. Note que se existem $i < j$ tais que

$$\sum_{k=i}^{j} a_i \equiv 0 \mod c,$$

todos os índices entre $[i, j]$ são uma solução. Mas isto ocorre se somente se existem índices $i < j$ tais que

$$F_{i-1} = F_j,$$

onde F_{i-1} e F_j são as somas dos prefixos módulo c. Mas tal par de índices com certeza existe pelo *princípio da casa dos pombos*: temos n inteiros em $\mathbb{Z}_c$, com $c \leq n$.

The Child and Sequence

A observação que resolve o problema é a seguinte

Proposição 1. *Seja x um número natural, e $(a_i)_{i=1}^n$ uma sequência de números naturais. Defina a sequência $(x_i)_{i=0}^n$ como*

$$\begin{aligned} x_0 &\coloneqq x \\ x_i &\coloneqq x_{i-1} \mod a_i. \end{aligned}$$

Então o conjunto $\{x_i : i \leq n\}$ tem tamanho $O(\lg n)$.

A prova é simples, e deixo para vocês. Basta mostrar que, ao se considerar o resto da divisão de a por b, ou o resto é o próprio a ou o resto é menor ou igual à $a/2$.

Assim, podemos manter uma segtree que contém os máximos e a soma dos subintervalos. As consultas de soma e as modificações de elementos específicos podem ser feitas da maneira convencional.

Considere então as consultas de aplicar resto no intervalo. Queremos trocar todos os elementos do intervalo $[l, r)$ pelos seus resto módulo x. Como temos o máximo de cada nó, podemos navegar na árvore e encontrar o menor índice i em $[l, r)$ tal que $a_i > x$. Podemos então modificar essa entrada, e procurar o próximo. Assim, por elemento modificado iremos gastar tempo $O(\lg n)$. Pela proposição, sabemos que cada elemento vai ser modificado no máximo $O(\lg n)$ vezes. Assim, a solução é $O(m \lg^2 n)$, onde n é o tamanho do vetor e m é a quantidade de consultas.

Você consegue escrever uma prova mais certinha de que isso vai dar certo? É um exercício interessante de análise amortizada. Na mesma linha, qual seria a complexidade dessa solução se tivéssemos uma modificação de intervalo — trocar todos os elementos de $[l, r)$ para um valor x dado?

Fibonacci Words

A observação que resolve o problema é que não é necessário construir muitas das strings maiores do que o tamanho do padrão. Como as strings se repetem, pode-se fazer o *string matching* direto com os tamanhos pequenos e calcular os demais com uma programação dinâmica. Irei denotar por

$$\operatorname{occ}(P, S)$$

as ocorrências do padrão P na string S. O problema, em particular, nos pede para calcular $\operatorname{occ}(P, F_n)$ para P e n dados. Podemos contar recursivamente quantas ocorrências de P temos em F_{n-1} e F_{n-2}, e quantas novas ocorrências se formam ao concatenar ambas strings. Se assumirmos que $|P| \leq |F_{n-3}|$,

$$\begin{aligned} \operatorname{occ}(P, F_n) &= \operatorname{occ}(P, F_{n-1} + F_{n-2}) \\ &= \operatorname{occ}(P, F_{n-2} + F_{n-3} + F_{n-3} + F_{n-4}) \\ &= \operatorname{occ}(P, F_{n-1}) + \operatorname{occ}(P, F_{n-2}) \\ &\quad + \operatorname{occ}(P, F_{n-3} + F_{n-3}) - 2\operatorname{occ}(P, F_{n-3}), \end{aligned}$$

onde a última linha conta as ocorrências que começam em F_{n-1} e terminam em F_{n-2} ao contar as ocorrências de P em $F_{n-3} + F_{n-3}$ e descontar as ocorrências que já foram contabilizadas. Um argumento semelhante te permite escrever o valor de $\text{occ}(P, F_n + F_n)$ em termos de $\text{occ}(P, F_i)$ e $\text{occ}(P, F_i + F_i)$ para $i < n$.

Assim, a solução consiste em construir e contar diretamente as ocorrências de P para todo k tal que $|F_{k-3}| \leq |P|$, e então utilizar as fórmulas recursivas encontradas para $\text{occ}(P, F_n)$ e para $\text{occ}(P, F_n + F_n)$ para construir as respostas. Tome cuidado com oara separar os casos "pequenos" dos "grandes" de maneira correta.