

---

# I Simulado para Ingressantes 2015

## Comentários dos Problemas

Universidade de São Paulo

e

Universidade Federal do ABC

---

## Problema A: Bactérias

Autor do problema: André Hahn Pereira

Análise: André Hahn Pereira

---

A ideia desse problema é notar que os valores não cabem em inteiros normais, sendo necessário utilizar alguma outra técnica. A princípio, duas técnicas possíveis para obter a resposta correta são a utilização de logaritmos para comparar a magnitude ou utilização de *BigIntegers* do Java.

Apesar de ser prática, a classe *BigIntegers* do Java é bastante lenta. Utilizá-la para este problema resultaria no chamado TLE (*Time limit exceeded*).

A técnica do logaritmo consiste basicamente de utilizar propriedades básicas para trabalhar com números menores.

$$a_1^{b_1} > a_2^{b_2} \iff \log a_1^{b_1} > \log a_2^{b_2} \iff b_1 \log a_1 > b_2 \log a_2.$$

## Problema B: Cavaleiros da Távola Redonda

Autor do problema: Renzo Gonzalo Gómez Diaz

Análise: Renzo Gonzalo Gómez Diaz

---

Em essência, este problema pede para decidir se um dado número  $N$  é divisível por 11. Como o número  $N \leq 10^{10000}$ , claramente, não podemos guardar esse valor em um tipo de dado inteiro (`int`, `long`, `long long`, etc). Portanto, podemos usar um vetor de caracteres para armazenar o número  $N$ . Uma forma de achar a resposta é usando o critério de divisibilidade por 11. *Um número é divisível por 11, caso a soma dos algarismos das posições pares subtraídos da soma dos algarismos das posições ímpares, resultar em um número divisível por 11.*

Outra forma de resolver o problema é usar propriedades de aritmética modular. Mais especificamente,

$$\begin{aligned}(a + b)\%c &= (a\%c + b\%c)\%c, \\(a \times b)\%c &= (a\%c \times b\%c)\%c.\end{aligned}$$

Além disso, notamos que  $N = M \times 10 + d_1$ , onde  $M = \lfloor N/10 \rfloor$  e  $d_1$  é o algarismo das unidades de  $N$ . Usando essas informações podemos achar o resultado mediante um laço `for`, onde construímos o número  $N\%11$  adicionando um novo dígito a cada iteração.

## Problema C: Imparmiopia

Autor do problema: Monael Pinheiro Ribeiro

Análise: Giovana Delfino

---

Esse é bastante simples, basta passar por cada dígito e ver se o resto da divisão por 2 é diferente de 0, se sim o algarismo é ímpar. Talvez a parte mais “complicada” do problema seja saber como ler um número e conseguir passar por cada dígito dele, para fazer isso você pode ler como uma string e passar por cada caracter dela tirando ‘0’ (perceba que cada caracter da *string* lida está entre ‘0’ e ‘9’), sabemos que cada caracter equivale a um número da tabela ASCII e que os caracteres ‘0’ até ‘9’ estão em sequência. Sendo assim, se eu tenho o caracter ‘0’ e subtraio ‘0’ eu obtenho o número 0, se eu tenho o caracter ‘1’ e tiro ‘0’ eu obtenho o número 1, e assim vai.

Outra forma seria ler como um inteiro e acessar cada algarismo pegando o resto da divisão do número por 10 e depois dividindo o número por 10, repetindo isso até que o mesmo seja 0. Por exemplo, para 1234, o resto da divisão por 10 é 4, divido por 10, pego somente a parte inteira e obtenho 123, e o processo se repete até o número ser 0. Para esta abordagem é importante notar que  $N$  pode valer até  $2^{32} - 1$ . Variáveis do tipo `int` utilizam 32 bits de armazenamento, sendo um

dos bits para representar o sinal, logo, o maior inteiro que elas podem representar é  $2^{31} - 1$ . Uma solução é utilizar variáveis `unsigned int`, que não possuem sinal, e representam inteiros até  $2^{32} - 1$ .

## Problema D: *Free shop*

---

Autor do problema: Victor Colombo

Análise: Victor Colombo

---

Esse problema é uma variação do problema da sublista contínua de soma máxima, onde deseja-se sublista contínua de soma mínima.

Pode-se inverter os valores dados e aplicar o Algoritmo de Kadane<sup>1</sup>.

Complexidade:  $O(N)$

## Problema E: Competição Pirata

---

Autor do problema: Victor Colombo

Análise: Victor Colombo

---

Quando existem  $i$  ( $i \leq K$ ) canecos, o primeiro pirata ganha, pois pode tomar os  $i$  canecos restantes. Caso existam  $K + 1$  canecos, o primeiro pirata perde, pois, qualquer quantidade  $j$  ( $j \leq K$ ) de canecos que ele tomar, deixará  $K + 1 - j \leq K$  canecos restantes, possibilitando a vitória do oponente.

Como tanto com 0 canecos quanto  $K + 1$  canecos o jogador atual perde, podemos adotar  $K + 1$  como o estado inicial do jogo e repetir o raciocínio.

Assim, verifica-se que o primeiro jogador perde apenas se  $N$  é múltiplo de  $K + 1$ .

Complexidade:  $O(1)$

## Problema F: IntegraMIX

---

Autor do problema: Stefano Tommasini

Análise: Victor Colombo

---

Procuramos  $s$  tal que exista um  $t \in \mathbb{Z}$  para o qual  $s^t = S$  e  $|s|$  seja mínimo. Note que tal  $s$  sempre existe, pois, no pior dos casos,  $t = 1 \Rightarrow s = S$ . Por definição, um dado  $s$  é uma solução válida se, para cada  $0 \leq t < \frac{|S|}{|s|}$ , temos que  $s[i] = S[i + t|s|]$  para todo  $0 \leq i < |s|$ . Isso pode ser verificado em  $O(|S|)$ .

Agora precisamos encontrar tal  $s$ . Uma forma de fazer isso é testar todas as possibilidades, isto é, todos os  $s = S[0 \dots x]$ , para  $0 \leq x < |S|$ . A complexidade disso é  $O(|S|)$ , o que resulta numa complexidade total de  $O(|S|^2)$ , insatisfatória para os limites.

Podemos considerar apenas  $x$  que sejam divisores de  $|S|$ , uma vez que são os únicos candidatos. É possível (e muito complicado) provar o limite superior<sup>2</sup> da função número de divisores de  $n$ ,  $\sigma(n)$ .

Era necessário conjecturar que os divisores de um número são poucos, sendo que para isso era necessário conhecimento em Teoria dos Números e, de preferência, realizar algumas simulações. Isso reduziria a complexidade total para  $O(|S|\sigma(|S|))$ .

Complexidade:  $O(|S|^{\frac{1}{\log \log |S|} + 1})$

---

<sup>1</sup>[http://en.wikipedia.org/wiki/Maximum\\_subarray\\_problem](http://en.wikipedia.org/wiki/Maximum_subarray_problem)

<sup>2</sup><https://terrytao.wordpress.com/2008/09/23/the-divisor-bound/>

## Problema G: Velha

---

Autor do problema: Stefano Tommasini

Análise: Victor Colombo

---

Nesse problema, basta verificar se há três  $x$  ou três  $o$  alinhados em alguma das linhas, colunas ou diagonais. Apesar de bem simples, era preciso tomar cuidado com a leitura da entrada e prestar atenção para não se confundir os índices de matriz que definem diagonais.

## Problema H: Ideais principais

---

Autor do problema: Renzo Gonzalo Gómez Diaz

Análise: Bruno Pasqualotto Cavalari

---

Este problema é bem simples. Como explica o enunciado, um ideal principal  $a\mathbb{Z}$  é o conjunto dos inteiros múltiplos de  $a$ . Logo, basta verificar se algum dos números divide o outro.

## Problema I: Lista de problemas ótima

---

Autor do problema: Marcio T. I. Oshiro

Análise: Victor Colombo

---

Nosso objetivo é calcular a diferença  $d(x)$  de duas funções quadráticas  $n_s(x)$  e  $n_d(x)$ . A função  $d(x)$ , a princípio, poderia ser uma parábola ( $a_s \neq a_d$ ), uma reta ( $a_s = a_d$  e  $b_s \neq b_d$ ) ou uma constante ( $a_s = a_d$  e  $b_s = b_d$ ), mas, pelas restrições apresentadas no enunciado, é garantido que sempre será uma parábola.

Por  $d(x)$  ser uma parábola, precisamos verificar sua concavidade para ver se ela apresenta ponto de mínimo ou de máximo. Quando  $a_d > a_s$ , a concavidade é para baixo, logo a função apresenta ponto de mínimo.

Revisando as condições estabelecidas, verifica-se que  $a_d > a_s$  é condição suficiente para existência de mínimo. Calculando a derivada  $d'(x)$  da função  $d(x)$ , temos que

$$d'(x) = 2(a_d - a_s)x + (b_d - b_s) = 0 \Rightarrow x = \frac{-(b_d - b_s)}{2(a_d - a_s)}$$

Complexidade:  $O(1)$

## Problema J: Miojo

---

Autor do problema: Fábio Moreira & Daniel Fleischman

Análise: Bruno Pasqualotto Cavalari

---

Recebemos do *input*  $A$  e  $B$ , o tempo de cada uma das ampuhetas, e  $T$ , o tempo necessário para o preparo do miojo. Queremos encontrar uma forma de medir o tempo  $T$  usando apenas as ampuhetas  $A$  e  $B$ . Isso é, na verdade, equivalente a encontrar soluções para a equação

$$Ax + By = T.$$

Vejamos o exemplo do enunciado. Nele, temos  $A = 5$ ,  $B = 7$ ,  $T = 3$ . Segue que  $x = -2$  e  $y = -1$  é uma solução, pois  $2 \cdot 5 + (-1) \cdot 7 = 3$ . Isso indica que se passarão duas unidades de tempo  $A$  e uma unidade de tempo  $B$ . Portanto o processo todo demora 10 minutos.

No entanto, note que  $x = -5$ ,  $y = 4$  também é uma solução. Neste caso temos que o processo demora  $4 \cdot 7 = 28 > 25 = 5 \cdot 5$ . O nosso problema, portanto, não se reduz apenas a encontrar uma solução, mas uma solução “minimal”.

Antes, alguns fatos básicos sobre Teoria dos Números. O tipo de equação que estamos tentando resolver é chamada equação diofantina linear. Este tipo de equação admite solução, se, e somente se,  $\text{mdc}(A, B)$  divide  $T$ . Neste caso, se  $(x_0, y_0)$  é uma solução específica da equação, para  $t \in \mathbb{Z}$ , a solução geral é dada por:

$$x = x_0 + t \frac{B}{\text{mdc}(A, B)} \quad \text{e} \quad y = y_0 - t \frac{A}{\text{mdc}(A, B)}.$$

Nós queremos encontrar a solução que minimiza  $\max\{|Ax|, |Ay|\}$ . Mas note que, no nosso caso,  $A$  e  $B$  são positivos. Isso mostra que, à medida que  $t$  aumenta,  $x$  aumenta e  $Ax$  fica cada vez maior em módulo, se afastando de nossa solução. De modo análogo, à medida que  $t$  diminui,  $y$  diminui e fica mais distante de 0, de modo que o módulo de  $Ay$  também aumenta.

Não é difícil perceber que a solução que procuramos deve estar entre as soluções que tem o menor valor em módulo.

Sendo assim, sejam  $x(t)$  e  $y(t)$  as soluções da equação em função de  $t \in \mathbb{Z}$ . Pense um pouco e você perceberá que as únicas soluções que nos interessam são aquelas que têm módulo tão pequeno que apenas uma variação em  $t$  muda o seu sinal. Ou seja, estamos interessados nos pares de solução  $(x(t), y(t))$  e  $(x(t+1), y(t+1))$  que satisfazem  $x(t) < 0$  e  $x(t+1) \geq 0$ . Substituindo  $t$  nessas desigualdades, é fácil descobrir o seu valor. Verifique que vale:

$$-\frac{x_0 \cdot \text{mdc}(A, B)}{B} - 1 \leq t < -\frac{x_0 \cdot \text{mdc}(A, B)}{B}$$

(Atenção: tome cuidado, pois nada garante que  $\frac{x_0 \cdot \text{mdc}(A, B)}{B}$  é inteiro).

Basta, portanto, encontrar uma solução  $(x_0, y_0)$  usando o Algoritmo de Euclides estendido<sup>3</sup>, e encontrar o valor de  $t \in \mathbb{Z}$  que satisfaz as condições acima. Olhamos então para as duas soluções (com  $t$  e  $t+1$ , isto é, a que tem  $x$  negativo e  $x$  positivo) e comparamos os valores de  $\max\{|Ax|, |By|\}$ . O menor valor é a nossa resposta.

Além disso, você também poderia simular o processo no computador. Basta que sua simulação seja rápida o suficiente.

## Problema K: Jenyffer, Rute, Raquel, Cláudia e Narcisa

---

Autor do problema: Marcio T. I. Oshiro

Análise: Victor Sena Molero

---

O problema das máquinas de café consistia em saber qual era o mínimo de máquinas, entre as  $N$  disponíveis, que se devia usar para comprar  $M$  cafés, dado que cada máquina  $i$  tinha uma quantidade  $q_i$  de café.

A forma mais fácil de resolver esse problema é usando um algoritmo guloso. Sempre vale a pena começar a comprar pela máquina que tem mais café e, então, esgotar todos os cafés disponíveis nela para, depois, passar pra próxima (lógico, se for necessário comprar mais café).

Se a quantidade de cafés que se precisa comprar for nula ou negativa, imprime-se o contador da quantidade de máquinas visitadas. Se a quantidade de máquinas ainda não usadas for nula, imprime-se N000!. Agora, é importante determinar que, para implementar isso com um vetor normal em C, a operação de retirar algo do vetor deve ser substituída por algo mais inteligente, como mudar seu valor para 0 (o que substituiria a operação de checar se  $q$  não é vazio por checar se o máximo de  $q$  é 0).

---

<sup>3</sup>[http://pt.wikipedia.org/wiki/Algoritmo\\_de\\_Euclides\\_estendido](http://pt.wikipedia.org/wiki/Algoritmo_de_Euclides_estendido)

Existem algumas formas de implementar isso. Se você usar um vetor desordenado e aplicar o algoritmo acima, a busca pelo máximo custará tempo linear  $O(N)$ , logo, o algoritmo todo será quadrático  $O(N^2)$ . Isso, no limite proposto de  $N \leq 10^3$  máquinas, é suficiente.

Se os valores do vetor  $q$  estiverem ordenados, achar o máximo custará tempo constante  $O(1)$ , deixando assim o algoritmo acima linear, para este caso. Então, pode-se primeiro preprocessar o vetor  $q$ , ordenando os seus valores. Os algoritmos mais simples de ordenação (*bubble sort*, *insertion sort*, *selection sort*) custam  $O(N^2)$  no pior caso, logo não ajudam a melhorar a solução anterior. Algoritmos mais sofisticados de ordenação (*heap sort*, *merge sort*, *quick sort*) já ajudam bastante, baixando o tempo da solução para  $O(N \log N)$ .

Já que as cotas vão até 50 cafés por máquina. Também se poderia fazer ordenação por contagem (*counting sort*), deixando a solução toda linear. O que acredito ser a melhor solução possível.

Um erro comum para este problema durante o simulado foi não levar em consideração o caso no qual  $M = 0$ . Note que esse caso está dentro das restrições e a resposta para ele deve ser 0. Muitos já iniciavam o contador da quantidade de máquinas com 1.