Caderno de Provas

Rinha de Calouros

Rinha de Calouros UDESC 2024/1

Servidor BOCA:
http://200.19.107.68/boca

Organização e Realização:

BRUTE
competitive programming

Informações do ambiente de testes

- Em todas as questões, o limite de memória é de 1 gigabyte.
- Há um limite no tamanho na pilha de 100 megabytes.
- O código-fonte submetido pode ter no máximo 100 kilobytes.

Comandos de compilação utilizados

- C: `gcc -static -O2 <nome_do_arquivo>.c -lm -o <nome_do_programa>`
- C++: `g++ -std=c++20 -static -O2 -lm <nome_do_arquivo>.cpp -o <nome_do_programa>`
- Java: `javac <nomeDoArquivo>.java`
- Kotlin: `kotlinc -J-Xms1024m -J-Xmx1024m -J-Xss100m -include-runtime <nome_do_arquivo>.kt`

Comandos de entrada e saída de dados

- C: `scanf, getchar, printf, putchar`
- C++: `as mesmas de C ou os objetos cout e cin`
- Java: `Scanner, BufferedReader, BufferedWriter e System.out.println`
- Python: `read, readline, readlines, input, print, write`

Template exemplo

A correção é automatizada, portanto siga atentamente as exigências do problema quanto ao formato da entrada e saída de seu programa. Fornecemos um exemplo de problema e um template esperado para a sua resolução:

A entrada consiste em uma linha contendo um inteiro A.

A saída consiste de uma linha contendo o inteiro B, que é igual ao inteiro A multiplicado por 2.

- C:

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    int A;
    scanf("%d", &A);
    int B = A * 2;
    printf("%d\n", B);
    return 0;
}
```

- C++:

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    int A;
    cin >> A;
    int B = A * 2;
    cout << B << "\n";
    return 0;
}
```

- Java:

```
import java.io.IOException;
import java.util.Scanner;

public class C {
    public static void main(String[] args) throws IOException {
        Scanner scan = new Scanner(System.in);
        int A;
        A = scan.nextInt();
        int B = A * 2;
        System.out.printf("%d\n", B);
    }
}
```

Obs: o nome do arquivo deve ser LETRA_DO_PROBLEMA.java e a classe deve ser LETRA_DO_PROBLEMA, como no exemplo acima, se o problema solucionado fosse o problema C.

- Python:

```
A = int(input())
B = A * 2;
print(B)
```

Obs: Caso a entrada contenha vários elementos em uma única linha, como por exemplo "uma linha contendo três inteiros A, B e C"usamos o seguinte comando:

```
A, B, C = map(int, input().split())
```

C/C++

- Seu programa deve retornar zero, executando, como último comando, `return 0` ou `exit(0)`.
- É sabido que em alguns casos de problemas com entradas muito grandes, os objetos `cin` podem ser lentos, por conta da sincronização de buffers da biblioteca `stdio`. Caso deseje utilizar `cin` e `cout`, um jeito de se contornar este problema é executando o comando `ios_base::sync_with_stdio(0)`, no início de sua função `main`. Note que, neste caso, o uso de `scanf` e `printf` no mesmo programa é contra-indicado, uma vez que, com buffers separados, comportamentos inadequados podem ocorrer.
- Comando para executar uma solução C/C++: `./<nome_do_programa>`.

Java

- Não declare "`package`" no seu programa Java.
- Note que a convenção para o nome do arquivo fonte deve ser obedecida, o que significa que o nome de sua classe pública deve ser uma letra maiúscula (A, B, C, ...).
- Comando para executar uma solução Java: `java -Xms1024m -Xmx1024m -Xss20m <nomeDoArquivo>`.

Kotlin

- Não declare "`package`" no seu programa Kotlin.
- Comando para executar uma solução Kotlin: `kotlin -J-Xms1024m -J-Xmx1024m -J-Xss100m <nomeDoArquivoKt>`.
- Não é garantido que soluções em Kotlin conseguirão executar dentro do tempo limite alocado.

Python

- O Python está na versão 3.10 e em erro de sintaxe, será retornado `Runtime Error`.
- Comando para executar uma solução Python: `python3 <nome_do_arquivoPy>.py`.
- Não é garantido que soluções em Python conseguirão executar dentro do tempo limite alocado.

Instruções para uso do sistema de submissão BOCA

Tudo no BOCA é feito pela interface web, então antes de fazer qualquer ação certifique-se que:

- O navegador está aberto e na URL do BOCA que se encontra na capa.
- O BOCA está logado com o login do time (com o nome de usuário e senha fornecidos).

Submissão de soluções

Para enviar uma solução para um problema, acesse a aba `Runs`, escolha o problema apropriado, a linguagem utilizada e envie o arquivo fonte.

Resultado da submissão

Para ver o resultado de uma submissão, verifique a aba `Runs`. Os vereditos que você pode receber dos juízes são:

1. YES
2. NO - Compilation error
3. NO - Runtime error
4. NO - Time limit exceeded
5. NO - Wrong answer
6. NO - Contact staff
7. NO - Class name mismatch
8. NO - Wrong language
9. NO - Problem mismatch

Os significados de 1, 2, 3 e 4 são auto-explicativos.

- Sobre 1 e 5:
  - Se a saída da solução do time é exatamente igual à saída dos juízes, a resposta é "YES";
  - Caso contrário, o veredito é "Wrong Answer".
- Sobre 6: Usado em circunstâncias inesperadas. Neste caso, utilize o menu "Clarifications" e forneça o número da run para maiores esclarecimentos.
- Sobre 7: Apenas para submissões Java, veredito quando o time submete uma solução com nome da classe principal diferente do especificado, de forma que a execução falha. Não é usado no caso de submissões C/C++ ou Python.
- Sobre 8: Principalmente para Python3, mas pode ser usado também para outras linguagens.
- Sobre 9: Identificação errada de problema ao submeter a solução. Note que nem sempre é possível distinguir entre os vereditos 3, 7, 8 e 9. Por exemplo, quando um time submete um arquivo `B.java` no qual uma classe pública A é definida como solução para o problema A, o veredito pode ser "Class name mismatch" ou "Problem mismatch".

Esclarecimentos

Para solicitar esclarecimentos sobre o enunciado de um problema, acesse a aba `Clarifications`, escolha o problema apropriado, escreva sua dúvida e submeta.

Placar

Para ver o placar local você deve acessar a aba `Score`.

Rinha de Calouros

Rinha de Calouros UDESC 2024/1

7 de Maio de 2024

Conteúdo

Problema A: Aluguel de Filmes

Arquivo:	`A.[c|cpp|java|kt|py]`
Limite de tempo:	1 segundo

Eric é um grande entusiasta de cinema e, por motivos nostálgicos, prefere alugar seus filmes em locadoras em vez de assinar serviços de streaming.

Recentemente, Eric foi agraciado com uma bolsa do projeto de vôlei de sua universidade, o BRUTE (BRUTE Rumo à União e Triunfo Esportivo) e, embora esteja entusiasmado, ele sabe que seu orçamento não é infinito. Determinado a aproveitar ao máximo sua experiência na locadora, Eric decidiu que vai tentar alugar o máximo possível de filmes sem exceder seu orçamento limitado de X reais.

No entanto, surge um obstáculo: entre os N filmes disponíveis na locadora, alguns deles são lançamentos recentes. A locadora estabelece uma regra que permite que cada pessoa alugue apenas um lançamento por vez. Isso significa que Eric precisa levar em consideração essa restrição ao escolher os filmes que irá alugar. Sua tarefa é determinar a quantidade máxima de filmes que Eric pode levar para casa, respeitando seu orçamento e a restrição da locadora em relação aos lançamentos.

Entrada

A primeira linha da entrada consiste em dois inteiros N e X ($1 \leq N \leq 10^5, 1 \leq X \leq 10^8$), o número de filmes disponíveis na locadora e o orçamento de Eric.

A segunda linha da entrada consiste em N inteiros $a_1, a_2, ..., a_N$ ($1 \leq a_i \leq 1000$), representando o preço de cada filme na locadora.

A terceira linha da entrada consiste em N inteiros $b_1, b_2, ..., b_N$ ($0 \leq b_i \leq 1$), onde $b_i = 1$ indica que o filme i é um lançamento recente e $b_i = 0$ indica que o filme i não é um lançamento recente.

Saída

Imprima uma única linha contendo a quantidade máxima de filmes que Eric pode alugar, respeitando o orçamento e a restrição da locadora em relação aos lançamentos.

Exemplos

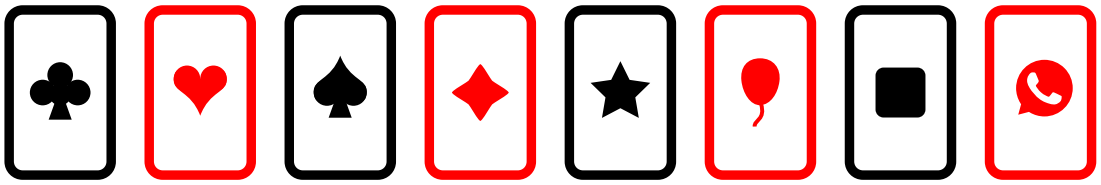
entrada padrão	saída padrão
5 12 2 1 4 3 1 0 0 1 0 1	4
8 23 2 2 4 1 7 10 3 6 1 1 1 0 1 0 0 0	5

Problema B: Baralho

Arquivo:	`B.[c|cpp|java|kt|py]`
Limite de tempo:	1 segundo

Em uma noite de jogos do BRUTE, Gaspa decidiu inventar um novo jogo de cartas!

Para o jogo, Gaspa elaborou um novo baralho que tem N cartas. Cada uma delas tem um naipe, identificado com um número, pois foram criados muitos naipes novos. Nesse baralho, cada carta tem somente um naipe, pois seu valor não importa. Também, a quantidade de cartas de um determinado naipe pode não ser igual à quantidade de cartas de outro naipe, diferentemente de baralhos de jogo convencionais.



Gaspa decidiu que o naipe de paus é identificado pelo número 1, copas é identificado pelo número 2 e assim por diante, até algum dos M naipes que ele criou.

A principal regra do jogo é a seguinte:

- Só se pode jogar uma carta de naipe igual ao naipe da última carta jogada. Ou seja, se o ultimo jogador jogou uma carta de naipe X, o próximo jogador só pode jogar uma carta do naipe X.

Confuso com essa regra, Machado perguntou: "Ué, Gaspa, mas daí a gente só vai poder usar cartas com o naipe da primeira carta jogada?"

Gaspa disse que sim, que era esse o propósito do jogo: dar uma vantagem para o jogador que começa jogando.

Machado não viu sentido nessa regra, então propôs mudar o baralho para que ele tenha apenas um naipe. Gaspa até aceitou a proposta de Machado, mas com uma condição: o número de cartas que serão jogadas fora tem que ser o menor possível.

Para fazer isso, Gaspa pediu a sua ajuda, Dado as N cartas com o seus naipes, diga para ele a menor quantidade de cartas que ele precisa jogar fora para que só exista um naipe no baralho final.

Entrada

A entrada consiste em duas linhas.

A primeira linha da entrada contém dois inteiros N ($1 \leq N \leq 10^5$) e M ($1 \leq M \leq N$), a quantidade de cartas no baralho e a quantidade de naipes no jogo, respectivamente.

A segunda linha contem N números $a_1, a_2, \cdots, a_N$ ($1 \leq a_i \leq M$), onde a_i representa o naipe da i-ésima carta.

Saída

A saída deve conter um único inteiro, a menor quantidade de cartas que Gaspa precisa jogar fora para que só exista um naipe no baralho final.

Exemplos

entrada padrão	saída padrão
5 3 1 2 3 3 1	3
5 5 5 5 5 5 5	0
7 4 1 4 2 4 3 1 1	4

Problema C: Corte de Cria

Arquivo:	`C.[c|cpp|java|kt|py]`
Limite de tempo:	2 segundos

Enzo cansou do jeito que o seu cabelo está, ele disse que não parece "de cria". Para Enzo, um corte de cria é aquele que é completamente reto ("na régua", como alguns dizem).

Um corte de cabelo é representado por uma lista de inteiros c. Cada posição i dessa lista representa uma parte do cabelo, e c_i representa o quanto essa parte do cabelo está a cima da "linha imaginária do cria", mais especificamente, em um "corte de cria", $c_i = 0$ para todo i.

A distância de um corte de cabelo ao "corte de cria" é definida pelo maior valor na lista c, por exemplo:

- Se $c = [4, 2, 10, 0]$, a distância é 10.
- Se $c = [1, 1, 2]$, a distância é 2.
- Se $c = [0, 0, 0, 0, 0]$, a distância é 0 (um corte de cria!).

Para chegar o mais próximo possível desse tal "corte de cria", Enzo quer que a distância do seu cabelo ao corte de cria seja a menor possível, para isso, ele vai ao seu barbeiro que tanto confia.

O barbeiro que corta o cabelo de Enzo tem uma política de preços peculiar: o preço de um corte de cabelo é igual ao número de minutos que levou para fazer o corte. Como o barbeiro quer maximizar seu lucro, a cada minuto, ele abaixa o nível de alguma parte do cabelo em somente uma unidade.

O seu papel nessa história toda é responder para Enzo a seguinte pergunta: qual é a menor distância ao corte de cria que o Barbeiro poderá atingir se Enzo tiver X reais?

Entrada

A primeira linha da entrada consiste de dois inteiros $N(1 \leq N \leq 10^5)$ e $X(1 \leq X \leq 10^9)$, a quantidade de partes em que o cabelo de Enzo está separado e a quantidade de reais que Enzo possui, respectivamente.

A segunda linha consiste de N inteiros $c_1, c_2, ..., c_N(1 \leq c_i \leq 10^9)$, o nível da i-ésima parte do cabelo de Enzo.

Saída

A saída deve conter um inteiro, a menor distância possível ao corte de cria que o barbeiro de Enzo pode atingir.

Exemplos

entrada padrão	saída padrão
4 3 1 4 3 0	2
5 10 4 3 2 1 0	0
7 8 1 10 3 5 7 8 2	6

Problema D: D-bonacci

Arquivo:	`D.[c|cpp|java|kt|py]`
Limite de tempo:	1 segundo

Malu está explorando uma Floresta Encantada para aprender sobre a matemática das criaturas mágicas. A primeira aula era sobre a sequência de D-bonacci.

Dado um número D, as criaturas mágicas definem o D-bonacci como:

$$F(n) = \begin{cases} 1 & \text{se } n \leq D \\ \sum_{i=1}^{D} F(n-i) & \text{se } n > D \end{cases}$$

Ou seja, os D primeiros valores são 1 e os próximos serão todos o somatório dos D anteriores. A sequência de Fibonacci que conhecemos é o D-bonacci com $D = 2$. Porém, nas profundezas da Floresta Encantada, eles não acham que o caso de $D = 2$ seja tão especial, por isso eles sempre consideram a sequência como tendo um valor D que a caracteriza.

Por exemplo, a sequência com $D = 3$ (3-bonacci) seria $1, 1, 1, 3, 5, 9, 17...$

A sequência com $D = 4$ (4-bonacci) seria $1, 1, 1, 1, 4, 7, 13, 25...$

Agora, a criatura sábia do bosque, Zellner, fez uma pergunta para Malu: "Dados N e D, me diga a quantidade de números ímpares no D-bonacci até N."

Como Malu está atualmente em uma reunião de planejamento de exploração da floresta e não consegue responder, ela pediu a sua ajuda. Qual a quantidade de números ímpares nos N primeiros termos de D-bonacci?

Entrada

A entrada consiste de uma linha com os inteiros N ($1 \leq N \leq 10^{15}$) e D ($1 \leq D \leq 10^{15}$).

Saída

Imprima um único número inteiro: a resposta para a pergunta do professor.

Exemplos

entrada padrão	saída padrão
`8 2`	`6`
`10 3`	`10`
`11234567891011 6`	`9629629620867`

Problema E: Escolhendo Outfits

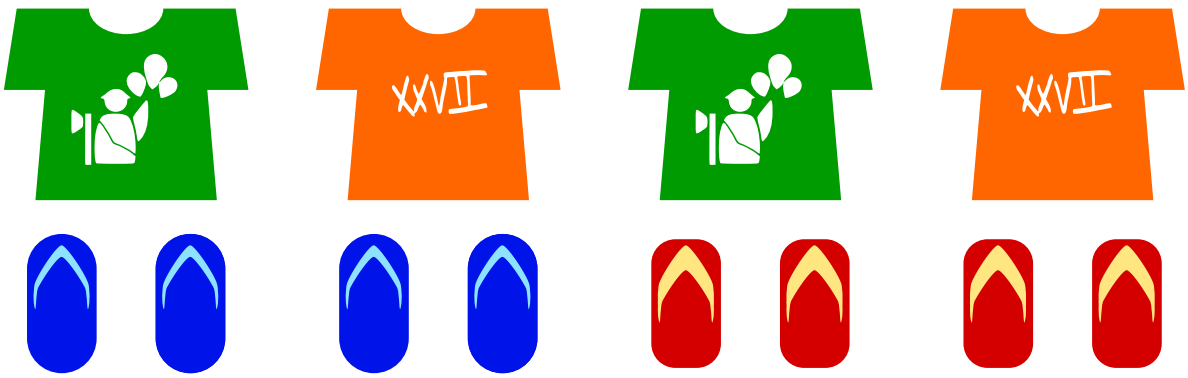
Arquivo:	`E.[c|cpp|java|kt|py]`
Limite de tempo:	1 segundo

João é um programador e maratonista há muitos anos e, por consequência disso, ele acumulou muitas peças de roupa que ganhou de presente por participar de eventos maratonísticos, principalmente camisetas e chinelos.

A mãe de João abriu o guarda-roupa dele e percebeu que ele tinha N camisas e N chinelos (uma camisa e um chinelo para cada evento que João já participou), o que é muita roupa! Chocada, ela decidiu que ia doar algumas camisetas e chinelos até a João tenha N dessas peças de roupa no total, ou seja, que soma da quantidade de camisas e chinelos seja N no total.

João, sabendo do plano de sua mãe, pediu para ela apenas um favor, que ela doasse as camisas e chinelos de tal forma que a quantidade de *outfits* distintos que ele pode fazer seja a maior possivel.

Um *outfit* é uma combinação de camisa e chinelo. Um *outfit* é diferente de outro se a camisa ou o chinelo é diferente.



Por exemplo, com 2 camisetas e 2 chinelos, João consegue montar 4 *outfits* distintos.

Você consegue ajudar a mãe de João e dizer para ela a maior quantidade de *outfits* que João pode ter depois de doar as camisas e chinelos até que João tenha N dessas peças de roupa?

Entrada

A entrada consiste de um inteiro N ($2 \leq N \leq 2 \cdot 10^5$), a quantidade de camisas e chinelos que João tem.

Saída

A saída deve conter um inteiro, a maior quantidade de *outfits* que João pode ter se a sua mãe doar as camisas e chinelos de forma ótima.

Exemplos

entrada padrão	saída padrão
5	6
8	16
200000	10000000000

Problema F: Festa dos (números) Primos

Arquivo:	`F.[c|cpp|java|kt|py]`
Limite de tempo:	1 segundo

Dudu foi encarregado de fazer uma festa para comemorar a classificação do BRUTE para a final mundial da ICPC e decidiu convidar todos os membros do BRUTE para a sua casa.

Dudu sabe que o BRUTE tem exatamente P membros, onde P é um número primo (ninguém sabe o porquê dessa coincidência), e ele vai preparar várias caipirinhas antes da festa começar.

Antes de Dudu fazer as caipirinhas, foram impostas pela administração do BRUTE as seguinte regras:

- A quantidade de caiprinhas para cada membro do BRUTE deve ser a mesma.
- A quantidade de capirinhas para cada membro do BRUTE deve ser maior que 1.
- A quantidade de capirinhas para cada membro do BRUTE não deve ser um múltiplo de X, pois o BRUTE odeia o número X.

Por motivos óbvios, o BRUTE não odeia o número 1 nem nenhum número maior ou igual a P.

Dudu pretende economizar o máximo possível, para isso, ele perguntou para você: qual é a menor quantidade de caiprinhas que ele pode fazer tal que todas as regras são satisfeitas?

Entrada

A entrada consiste em duas linhas.

A primeira linha da entrada contém um inteiro P ($3 \leq P \leq 999983$), a quantidade de membros do BRUTE. É garantido que P é um número primo.

A segunda linha da entrada contém um inteiro X ($2 \leq X < P$), o número que o BRUTE odeia.

Saída

A saída deve conter um único inteiro, a menor quantidade de caiprinhas que Dudu pode fazer que satisfaz todas as regras.

Exemplos

entrada padrão	saída padrão
11 2	33
13 3	26
862067 4	1724134

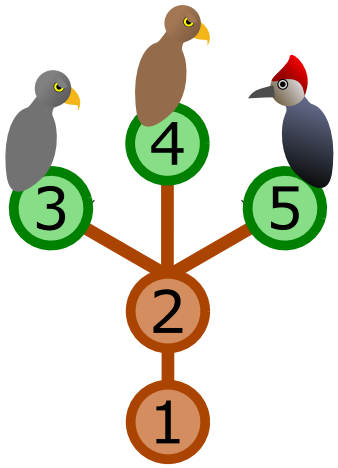
Problema G: Gaviões

Arquivo:	`G.[c|cpp|java|kt|py]`
Limite de tempo:	1 segundo

Léo e Ema são gaviões que moram em uma árvore chamada Joinlipa.

A árvore onde eles moram pode ser representada por N "casas" (que na verdade são cantinhos aconchegantes no tronco da árvore) enumeradas de 1 a N e $N-1$ galhos que conectam as casas entre si, e é claro que partindo de qualquer casa é possível viajar para qualquer outra casa usando os galhos da árvore.

Duas dessas casas são de Léo e Ema e, como são grandes amigos, gostam muito de visitar a casa um do outro. Apesar de poderem voar, eles sempre se movem andando pelos galhos da árvore. Muitas vezes Ema vai até a casa de Léo e outras vezes é Léo quem vai na casa de Ema. Porém, os dois têm um inimigo em comum, o tal do "Sid", um pica-pau, e não gostam quando precisam passar pela casa de dele para visitar um ao outro.



Se a árvore Joinlipa for como na imagem, a casa de Léo for a 3, a de Ema, a 4, e a de Sid, a 5, eles conseguem ir na casa um do outro sem passar pela casa de Sid. Note que, se a casas de Sid fosse a de número 2, eles não conseguiriam.

Como os habitantes de Joinlipa mudam de casa o tempo todo, Léo e Ema vão te fazer Q perguntas: Dados os números da casa de Léo, de Ema e de Sid, diga a eles se é possível Ema visitar Léo ou Léo visitar Ema sem que eles tenham que passar pela casa de Sid.

Entrada

A entrada consiste em várias linhas.

A primeira linha da entrada contém dois inteiros N ($3 \leq N \leq 1000$) e Q ($1 \leq Q \leq 1000$), o número de casas em Joilipa e a quantidade de perguntas que Léo e Ema vão fazer para você, respectivamente.

As próximas $N-1$ linhas contêm dois inteiros (u, v), representando que existe um galho que conecta a casa u à casa v, e vice-versa.

As próximas Q linhas contêm três inteiros L, E e S, o número da casa de Léo, Ema, e Sid, respectivamente.

Saída

A saída deve conter Q linhas, onde a i-ésima linha de conter a resposta da i-ésima pergunta:

"`SIM`" (sem aspas) se é possivel Ema visitar Léo ou Léo visitar Ema sem que eles tenham que passar pela casa de Sid ou "`NAO`" (sem aspas) se não for possível.

Exemplos

entrada padrão	saída padrão
5 2 1 2 2 3 2 4 2 5 3 4 5 3 5 2	SIM NAO
3 4 1 2 2 3 3 2 1 3 1 2 2 1 3 3 2 1	SIM NAO SIM SIM