Caderno de Provas

Maratona nas Estrelas

I Seletiva Interna UDESC 2024

Servidor BOCA:
https://boca.joinville.udesc.br/

Organização e Realização:

BRUTE
competitive programming

Patrocínio:

BECOMEX

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`.

Maratona nas Estrelas

I Seletiva Interna UDESC 2024

18 de Maio de 2024

Conteúdo

Problema A: Ano Estelar

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

Guizinho acabou de comprar um par de óculos novo e finalmente consegue enxergar as estrelas! Muito feliz com isso, ele começou a anotar padrões que via ao observar o céu noturno.

Após anos de observação, Guizinho anotou em seu caderno N números $a_1, a_2, ..., a_N$, representando que a estrela i aparece no céu da terra de a_i em a_i anos. Gui diz que esse número representa o período de uma estrela.

O ano de 2024 é um ano de muita sorte para ele que, ao observar o céu, percebeu que todas as N estrelas das quais anotou o período estão visíveis ao mesmo tempo! Guizinho batizou os anos em que todas essas estrelas aparecem no céu ao mesmo tempo de "anos estelares".

Depois de admirar o céu por tempo o bastante, ele pensou sobre o futuro da observação de estrelas: Sabendo que o ano atual é um ano estelar, quais estrelas estarão no céu da Terra se mais X anos se passarem?

Entrada

A primeira linha da entrada consiste de dois inteiros N ($1 \leq N \leq 3 \cdot 10^5$) e X ($0 \leq X \leq 10^9$), a quantidade de estrelas que Guizinho anotou em seu caderno e quantos anos no futuro ele quer considerar, respectivamente.

A próxima linha consiste de N inteiros, $a_1, a_2, ..., a_N$ ($1 \leq a_i \leq 10^9$), o período de cada estrela.

Saída

A primeira linha da saída deve conter um inteiro K ($0 \leq K \leq N$), a quantidade de estrelas que estarão no céu depois de X anos.

A próxima linha deve conter K inteiros $i_1, i_2, ..., i_K$, as estrelas que estarão no céu depois de X anos. As estrelas podem ser mostradas em qualquer ordem.

Exemplos

entrada padrão	saída padrão
1 4 2	1 1
5 3 3 4 1 9 81	2 1 3
3 2 11 1 3	1 2
4 0 4 3 92 7	4 1 2 3 4

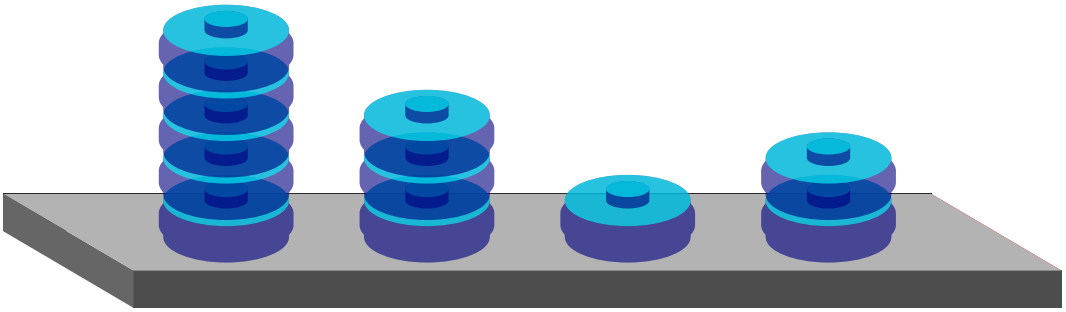
Problema B: Bit Tennis 2

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

Giovana e Julia, após serem as campeãs do torneio de Bit Tennis em duplas na Terra, foram viajar para outro planeta em busca de mais competição. Durante a viagem, lembraram que têm um tabuleiro de peças holográficas que as permite jogar vários jogos e, como já estavam enjoadas de todos os jogos (incluindo Bit Tennis), elas resolveram inventar um jogo novo: Bit Tennis 2.

As regras do jogo são as seguintes:

- O jogo começa com N pilhas de peças holográficas, sendo que a i-ésima pilha tem a_i peças;
- Julia começa o jogo e, após isso, as jogadas se alternam entre Giovana e ela;
- A cada jogada, quem está na vez deve escolher uma pilha e tirar qualquer quantidade de peças que seja uma potência de 2;
- Quem não puder fazer nenhuma jogada, perde o jogo.



Para o jogo com tamanhos de pilhas `[5, 1, 3, 2]`, Julia, ao começar, tem uma estratégia que impossibilita que Giovana ganhe o jogo.

As duas decidiram que o jogo estava muito difícil para quem joga por segundo, por isso, adicionaram uma nova regra; Giovana, antes de começar o jogo, deve realizar a seguinte operação exatamente X vezes:

- Escolher uma pilha e dobrar a quantidade de peças nela. Ou seja, se Giovana escolheu a pilha i, que tem a_i peças, ela passa a ter $2 \cdot a_i$ peças.

As duas, assim como quando inventaram o Bit Tennis, rapidamente aprenderam a jogar de forma ótima e observaram que o resultado do jogo parece estar determinado antes de a primeira jogada sequer ser feita. Curiosas sobre esse fato, elas pediram a sua ajuda. Dados N, X e o tamanho de cada uma das pilhas, elas pedem que você determine quem será a vencedora.

Entrada

A primeira linha da entrada contém dois números N ($1 \leq N \leq 10^5$) e X ($0 \leq X \leq 10^9$), a quantidade de pilhas e a quantidade de operações que a Giovana deve fazer, respectivamente. A segunda linha contém N valores a_i ($1 \leq a_i \leq 10^9$), a quantidade de peças holográficas em cada pilha.

Saída

Imprima "`Giovana`" ou "`Julia`", o nome da vencedora do jogo.

Exemplos

entrada padrão	saída padrão
4 2 5 3 1 2	Julia
1 10 32	Julia
2 1 4 2	Giovana

Problema C: Canção

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

Lívia é a capitã de uma espaçonave que foi atacada por piratas espaciais, e, como último recurso de sobrevivência, ela precisou se ejetar da nave.

Agora, Lívia se encontra flutuando pelo espaço sem rumo e as únicas coisas que possui consigo são seu celular e um fone de ouvido. Por sorte, ela conseguiu mandar um SOS antes de sair da nave e sabe que, apesar de poder demorar alguns dias, ajuda está a caminho. Para passar o tempo, ela decidiu escutar algumas canções de sua artista favorita, a Saylor Twift.

Lívia criou sua própria notação musical para as melodias que escuta. Ela considera que a canção contém no máximo 26 notas musicais distintas e, ao escutar uma melodia, ela representa cada nota musical por uma das letras do alfabeto latino (de "`a`" até "`z`"). Portanto, uma melodia é representada por uma sequência de letras.

Além disso, Lívia anotou N pares de notas (u, v) para representar que a nota v soa bem após a nota u. Agora, ela quer montar uma playlist com todas as canções bonitas de Saylor Twift, então ela pediu a sua ajuda. Lívia considera bonitas somente as melodias em que, para toda nota tocada, exceto a primeira, ela soa bem após a nota que veio antes dela.

Lívia insiste no fato de que uma determinada nota v soar bem quando toca depois de uma nota u não significa que a nota u também soaria bem depois da nota v. Dada uma melodia de Saylor Twift, diga se Lívia achará ela bonita ou não.

Entrada

A primeira linha da entrada contém uma string S ($1 \leq |S| \leq 10^5$), representando a canção da Saylor Twift.

A segunda linha contém um inteiro N ($0 \leq N \leq 676$) , a quantidade de pares de notas musicais que combinam.

Cada uma das próximas N linhas contém duas letras do alfabeto latino, u e v, representando que a nota u combina com a nota v.

Saída

A saída deve conter "`SIM`" se a Lívia achará a melodia bonita ou "`NAO`" se Lívia não achar.

Exemplos

entrada padrão	saída padrão
abac 3 a b b a a c	SIM
abac 3 a b c a b a	NAO
taylorswift 10 l o a y y l i f o r r s s w w i t a f t	SIM
cde 2 c d c e	NAO

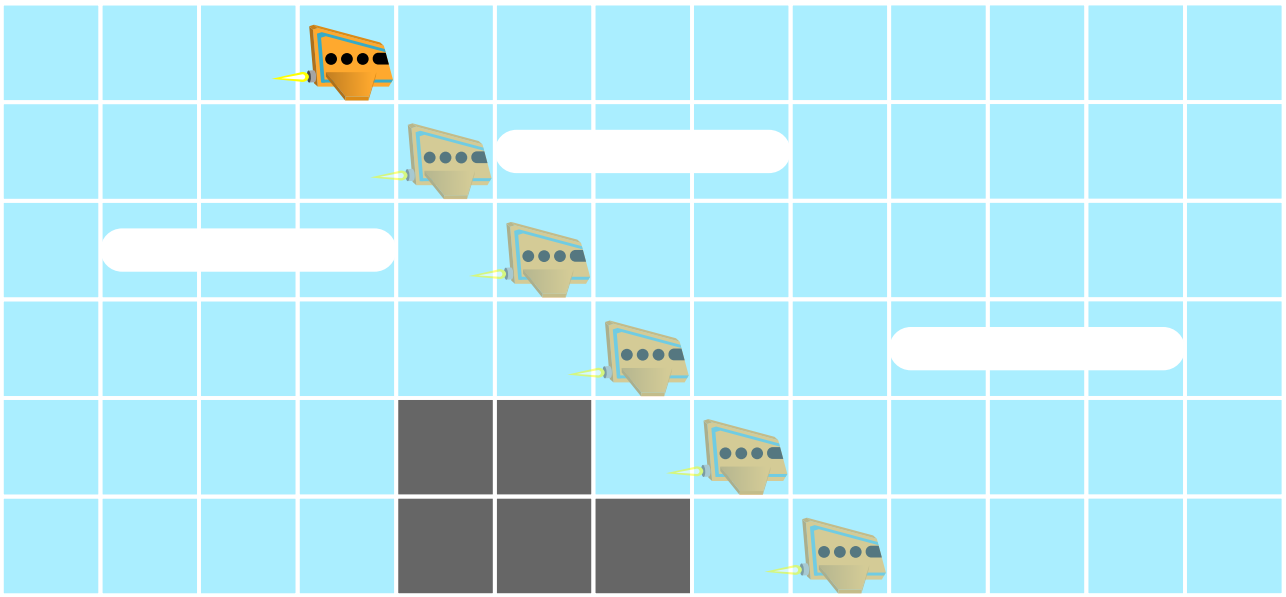
Problema D: Desvio de Rota

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

O Comandante Rosso, renomado por seus pousos precisos e seguros, encontra-se em uma situação crítica em meio ao vazio do espaço durante mais uma de suas missões. Uma tempestade solar inesperada interrompe sua missão de exploração, danificando os sistemas de navegação de sua nave, a Geometric Voyager, e forçando um pouso de emergência no remoto planeta JOI-47.

O terreno de JOI-47 é notoriamente irregular: há uma cadeia de montanhas quilométricas e apenas uma pequena faixa de terra onde se pode pousar, localizada depois das montanhas.

Ao entrar na densa atmosfera do planeta, Rosso precisa ajustar meticulosamente a altura inicial de descida da Geometric Voyager, pois o sistema de controle de altitude automatizado da nave faz com que ela desça a uma taxa constante de um quilômetro por segundo. Cada segundo de descida corresponde a um quilômetro de avanço em direção à pista de pouso. Como a pista de pouso disponível é curta, Rosso quer escolher a **menor altura possível que evite todas as montanhas**. A nave evita uma montanha se sua altura é estritamente maior do que a altura da mesma quando passa por ela.



No exemplo acima a menor altura inicial necessária para passar pelas montanhas é 5; se a altura inicial fosse menor, a nave bateria na primeira ou na segunda montanha. As montanhas tem altura $[2, 2, 1]$ e a nave terá altura $[4, 3, 2]$ ao passar pelas montanhas. Note que a altura da Geometric Voyager é medida pela quantidade de quadrados que estão abaixo dela e quando a nave está pousada, a sua altura é zero.

Você é o copiloto de Rosso e ele quer testar se você está apto a pilotar sozinho a Geometric Voyager. Erros não são uma opção, pois uma altura inicial mal calculada poderia levar a nave a colidir com as montanhas ou não conseguir pousar a tempo. Dadas as alturas das N montanhas, informe a altura ideal para entrar em rota de pouso.

Entrada

A primeira linha da entrada consiste de um inteiro N ($1 \leq N \leq 10^5$), o número de montanhas entre a nave de Rosso e a faixa de terra para pouso.

A segunda linha contém N inteiros $h_1, h_2, \cdots, h_N$ ($1 \leq h_i \leq 10^9$), sendo que h_i representa a altura da i-ésima montanha.

Saída

A saída deve conter um inteiro, representando a menor altura (em quilômetros) necessária para entrar na rota de pouso sem colidir com nenhuma montanha.

Exemplos

entrada padrão	saída padrão
3 2 2 1	5
1 10	12
5 3 4 2 3 1	8

Problema E: El Café

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

Max e seu irmão mais velho Min tinham uma amizade inabalável a vida toda. Porém, há alguns anos, Min se mudou para a Galáxia Sombrero, onde abriu uma cafeteria, a El Café, e obteve muito sucesso.

Seguindo a paixão de sua família por café, Max se candidatou a uma vaga de emprego na cafeteria do irmão. A primeira etapa do processo seletivo é uma simulação de um dia normal de trabalho, o que seria tranquilo se não fosse pela maneira nada convencional que a cafeteria serve seus clientes.

Na El Café existe apenas um tipo de bebida, a "*vainilla helada*", que consiste em uma combinação de TODOS (todos mesmo) os ingredientes disponíveis na loja naquele momento. Além disso, cada ingrediente deve ser distribuído igualmente entre as bebidas pedidas, de acordo com a quantidade de cada um. Dessa forma, quando um cliente faz um pedido na El Café, ele informa apenas a quantidade de *vainillas heladas* que deseja e, caso seja possível, o funcionário deve preparar o pedido. Case não seja possível, deve informar ao cliente que não há como fazer aquela quantidade de bebidas no momento.

Se, por exemplo, a lista da quantidade de cada ingrediente no momento fosse $[24, 12, 3]$ e um cliente pedisse 3 *vainillas heladas*, Max deve informá-lo que consegue fazer o pedido, já que pode fazer três *vainillas heladas*, cada uma com a lista de quantidade de ingredientes $[8, 4, 1]$. Por outro lado, se o cliente tivesse pedido 2 bebidas, Max deveria ter lhe informado que não é possível no momento.

No meio do expediente, novos ingredientes podem chegar e alguns dos ingredientes mais novos podem vencer, obrigando Max a jogá-los fora (a validade funciona diferente na Galáxia Sombrero).

Para aumentar suas chances de conseguir a vaga, Max resolve ligar para você (pois sabe de suas habilidades em programação) e pedir para que faça um programa que o ajude a treinar para o processo seletivo.

O programa deverá atender 3 tipos de pedidos:

- Tipo 1: informa que um novo ingrediente chegou em quantidade G. Note que cada ingrediente que chega é diferente dos demais.
- Tipo 2: informa que os últimos K ingredientes, ou seja, os K ingredientes mais novos, venceram e devem ser jogados fora.
- Tipo 3: simula um possível pedido de X *vainillas heladas* de um cliente, o programa deve somente informar se é possível ou não atender a tal pedido.

Dado o número N de diferentes ingredientes no começo do expediente, a quantidade a_i de cada ingrediente e os Q pedidos, escreva um programa que ajude Max a passar no teste.

Entrada

A primeira linha consiste de dois inteiros N e Q ($1 \leq N, Q \leq 10^5$), representando o número de ingredientes no começo do expediente e a quantidade de pedidos respectivamente.

A segunda linha contém N inteiros $a_1, a_2, \cdots, a_N$ ($1 \leq a_i \leq 10^9$), onde a_i representa o i-ésimo ingrediente na ordem de mais velho para mais novo.

Depois, seguem Q linhas que representam os pedidos.

- 1 G: pedido do tipo 1 indicando que um ingrediente novo chegou em quantidade G ($1 \leq G \leq 10^9$).
- 2 K: pedido do tipo 2 indicando que os K ($1 \leq K \leq 10^5$) ingredientes mais novos venceram e devem ser desconsiderados. É garantido que existem pelo menos $K + 1$ ingredientes diferentes para esse tipo de pedido.
- 3 X: pedido do tipo 3 perguntando se é possível servir X ($1 \leq X \leq 10^9$) *vanilla heladas* naquele momento.

Saída

Para cada pedido do tipo 3 imprima uma linha. Se for possível atender ao pedido do cliente imprima "`SIM`", do contrário imprima "`NAO`".

Exemplos

entrada padrão	saída padrão
5 5 12 12 6 4 2 2 3 3 4 1 6 1 3 3 2	SIM NAO
3 3 20 20 4 3 5 1 10 3 2	NAO SIM

Problema F: Frogs ou Toads?

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

Eric acabou de comprar um jogo novo para o seu NES, chamado Battletoads. O objetivo do jogo é derrotar a Dark Queen e seu exército de mutantes espaciais.

Depois de muito tempo tentando completar o jogo, ele percebeu que era muito difícil matar a Dark Queen, pois o seu personagem não tinha nenhum tipo de arma e precisava derrotá-la apenas com socos.

Frustrado por não conseguir derrotar a chefe do jogo, Eric decidiu desenvolver um jogo novo, que batizou de Battlefrogs. Nesse jogo, a chefe final se chama Light Queen e os personagens jogáveis estão sempre equipados com uma arma laser. A arma funciona da seguinte maneira:

Seja E a energia da arma.

- Se $E > 0$, Eric consegue disparar um raio laser e E diminui em uma unidade. Se Eric disparar o raio em um mutante espacial, o mesmo irá morrer, e, se Eric disparar o raio na Light Queen, a vida dela irá diminuir em uma unidade.
- Se $E \leq 0$, Eric consegue disparar um raio laser e E diminui em uma unidade. Se Eric disparar o raio em um mutante espacial, o mesmo irá morrer, e, se Eric disparar o raio na Light Queen, a vida dela **não será afetada**.

Antes de chegar no nível da Light Queen, Eric precisa passar por N níveis enumerados de 1 até N, começando pelo 1. Cada nível possui s_i mutantes espaciais no caminho e um banco de energia com potência e_i que carrega a energia de sua arma em e_i unidades. Além disso, logo no início de cada nível há uma rota alternativa com um buraco de minhoca protegido por X mutantes espaciais que faz com que Eric pule k níveis à frente.

Se Eric usar o buraco de minhoca no nível i, ele imediatamente pula para o nível $i + k$, ou, caso o $(i + k)$-ésimo nível não exista, para o nível da Light Queen. Ao usar o buraco de minhoca, Eric não precisa matar nenhum dos s_i mutantes, mas não poderá carregar a sua arma em e_i unidades de energia e é obrigado a matar os X mutantes que protegem o buraco de minhoca. Se ele decidir não usar o buraco de minhoca no i-ésimo nível, ele é obrigado a derrotar os s_i mutantes espaciais, poderá aumentar a energia de sua arma em e_i unidades e depois avançará para o $(i + 1)$-ésimo nível ou, caso o $(i + 1)$-ésimo nível não exista, para o nível da Light Queen.

Sabendo que a Light Queen morre quando sua vida chega em 0 e que a arma de Eric começa com 0 de energia, ele agora está pensando no balanceamento do jogo e precisa decidir o quanto de vida vai atribuir à chefe final. Ele perguntou para você: qual é a maior vida que a Light Queen pode ter, de modo que ainda seja possível derrotá-la?

Entrada

A primeira linha da entrada consiste de 3 inteiros N ($1 \leq N \leq 2 \cdot 10^5$), a quantidade de níveis em Battlefrogs, k ($1 \leq k \leq N$), quantos níveis à frente cada buraco de minhoca leva Eric, e X ($0 \leq X \leq 10^9$), a quantidade de mutantes espaciais que protegem cada buraco de minhoca.

Depois disso há N linhas, onde a i-ésima linha contém dois inteiros s_i e e_i, representando a quantidade de mutantes espaciais e a potência do banco de energia no nível i, respectivamente.

Saída

A saída deve conter um único inteiro, a maior vida que a Light Queen pode ter tal que é possível derrotá-la.

Exemplos

entrada padrão	saída padrão
4 2 1 1 2 4 6 5 2 3 11	8
5 3 20 3 4 2 2 5 6 3 1 8 5	0

Notas

Se a arma de Eric não conseguir chegar na Light Queen com energia maior que zero, a vida da Light Queen deve ser zero.

Problema G: Gargantua

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

Léo e Ema são dois astronautas que vivem na Terra. Um dia, Ema foi convocado para uma missão no planeta Miller, que orbita um buraco negro chamado Gargantua.

Por orbitar um buraco negro, o tempo em Miller passa X vezes mais devagar do que na Terra.

Sabendo que Léo tinha A anos de idade antes de Ema viajar e que Ema ficou Y anos (no tempo de Miller) em Miller, qual será a idade de Léo quando Ema voltar para a Terra?

Entrada

A entrada consiste em três linhas, cada uma com um dos inteiros A, X, Y ($0 \leq A, X, Y \leq 100$), a idade de Léo antes de Ema ir para Miller, a quantidade de vezes que o tempo passa mais devagar em Miller do que na Terra e o número de anos que Ema ficou em Miller, respectivamente.

Saída

A saída deve conter um inteiro, a idade de Léo quando Ema voltar para a Terra.

Exemplos

entrada padrão	saída padrão
20 3 4	32
31 17 3	82
1 2 3	7

Problema H: Higgs

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

Risa é a maior ladra da Galáxia GianParme, e ela está em busca do seu maior roubo até hoje. O furto vai acontecer no planeta Amorim, batizado com esse nome por causa de seu destemido líder Matheus Amorim, que liderou o planeta por muitos anos até seu trágico falecimento.

O objetivo de Risa é roubar a maior relíquia desse planeta, a coroa que Matheus usava durante seu reinado. Esse artefato está guardado em um museu no planeta Amorim. O primeiro passo do plano é resolver como chegar nesse planeta.

A galáxia de GianParme pode ser modelada por N planetas enumerados de 1 até N e M conexões entre os mesmos. Cada conexão é representada por dois planetas (u, v) e um nível de "Higgs" h, e as conexões são construídas de tal maneira que de qualquer planeta é possível viajar para qualquer outro planeta usando as conexões na Galáxia.

O nível de Higgs é uma forma de segurança que existe na galáxia; todas as pessoas que moram em GianParme possuem o seu próprio nível de Higgs, e, para ela usar uma conexão entre dois planetas, o seu nível de Higgs precisa ser **maior ou igual** ao nível de Higgs da conexão.

O planeta em que Risa está agora é o planeta com número 1 e o planeta Amorim, com número N. Como Risa é uma ladra conhecida, o nível de Higgs dela é 0 e, para a sua viagem até esse planeta, ela pode fazer duas coisas:

- Subornar o guarda de uma conexão e passar pela mesma sem se preocupar com o seu nível de Higgs.
- Falsificar a sua identidade para ter nível de Higgs igual ao valor que ela quiser.

Porém, falsificar uma identidade é uma tarefa muito difícil, então, ela precisa saber: Qual é o menor nível de Higgs que ela precisa falsificar para que ela consiga concluir a sua viagem até o planeta Amorim, sabendo que ela tem dinheiro suficiente para subornar K guardas?

Entrada

A primeira linha da entrada consiste de três inteiros N ($2 \leq N \leq 3{\cdot}10^5$), M ($1 \leq M \leq min(3{\cdot}10^5, \frac{N\cdot(N-1)}{2})$) e K ($0 \leq K \leq M$), o número de planetas, conexões e guardas que Risa pode subornar, respectivamente.

As próximas M linhas contêm três inteiros u, v ($1 \leq u, v \leq N$) e h ($1 \leq h \leq 10^9$), representando que existe uma conexão entre u e v com nível de Higgs h.

Saída

A saída deve conter um inteiro, o menor nível de Higgs que Risa precisa para concluir a viagem.

Exemplos

entrada padrão	saída padrão
4 4 1 1 2 3 2 4 4 1 3 3 3 4 1	1
6 6 2 1 2 1 2 3 3 3 4 10 4 6 4 3 5 2 5 6 4	2
5 6 2 1 2 3 1 3 2 2 3 1 3 4 5 4 5 3 2 4 4	2

Problema I: Itwise Bor

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

João está visitando um planeta alienígena muito longe da Terra, o planeta PEI-X40, governado por Nei, um líder pacífico que foi muito receptivo com a vinda de João.

Os habitantes desse planeta usam a base binária para representar os números: por exemplo, o número 6 na Terra é representado como "110" por lá. Nessa representação, eles chamam cada dígito de "it", sobre os quais eles definem as operações chamadas "itwise". Destas, as principais são o "band" e o "bor".

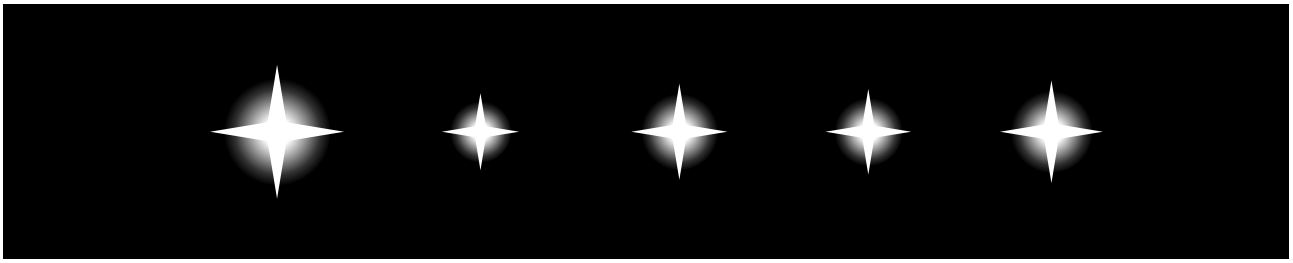
A operação bor é particularmente interessante para um observador de estrelas como João, pois foi determinado por Nei que a beleza de uma constelação é o itwise bor dos brilhos de todas as estrelas que estão nela.

Curioso, João estudou um pouco sobre o bor e aprendeu que ele é representado pelo símbolo "|" e funciona da seguinte maneira:

Para cada posição i na representação binária de X e de Y, se X tem o i-ésimo it igual a 1 e/ou Y tem o i-ésimo it igual a 1, então $X \mid Y$ tem o i-ésimo it igual a 1. Se não, $X \mid Y$ tem o i-ésimo it igual a 0.

Por exemplo, no sistema binário, $101 \mid 010 = 111$. Isso equivale a $5 \mid 2 = 7$ no sistema decimal terrestre.

Ao olhar para o céu de PEI-X40, João percebeu uma constelação muito peculiar: havia N estrelas no céu, uma do lado da outra. Anotou então o brilho de cada estrela $b_1, b_2, \cdots, b_N$, na ordem em que estavam.



Após rabiscar algumas expressões matemáticas, notou que essa constelação pode ser dividida em várias constelações contínuas de tal forma que a estrutura de possuir as estrelas uma do lado da outra seria mantida.

Agora, João quer separar a constelação em várias outras constelações de tal maneira que a soma das belezas de todas essas constelações seja máxima, e, dentro de todas essas maneiras que dão soma máxima, ele quer a separação que tem o menor número de constelações. Como os habitantes de PEI-X40 não treinaram ele muito bem para esse tipo de coisa, ele pediu ajuda a você.

Entrada

A entrada consiste em duas linhas.

A primeira linha da entrada consiste de um inteiro N ($1 \leq N \leq 3 \cdot 10^5$), a quantidade de estrelas na constelação original.

A próxima linha contém N inteiros $b_1, b_2, \cdots, b_N$ ($0 \leq b_i < 2^{30}$). O brilho das estrelas na ordem que elas aparecem no céu.

Saída

A saída deve conter dois inteiros S e K, a maior soma de beleza possível e a menor quantidade de grupos que consegue atingir essa soma, respectivamente.

Exemplos

entrada padrão	saída padrão
5 4 1 2 1 3	11 3
3 1 2 3	6 2
4 7 7 7 7	28 4
5 1 3 4 8 2	18 3

Notas

No primeiro exemplo, João pode dividir em três grupos: $[4, 1]$, $[2, 1]$ e $[3]$ para obter a soma máxima de $5 + 3 + 3 = 11$.

No segundo exemplo, João pode dividir em dois grupos: $[1, 2]$ e $[3]$ para obter a soma máxima de $3 + 3 = 6$.

Problema J: Jugando Fuerte

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

O bilionário explorador Granza, fundador da grande empresa de viagem espacial Space-Y, gosta de explorar o Universo a bordo de sua nave, a StarPuma-GTE.

Em uma de suas viagens pela Galáxia Sombrero, Granza descobriu um planeta chamado Guadalajara. Este planeta era famoso não apenas por suas paisagens deslumbrantes, mas também por seus imensos cassinos e um jogo de cartas peculiar conhecido como At-Poker.

At-Poker era considerado o passatempo mundial de Guadalajara, envolvendo habilidade, sorte e estratégia. Nesse planeta, os cassinos são templos luminosos onde fortunas são feitas e perdidas em questão de rodadas e Granza, sempre um homem de desafios, não pôde resistir à tentação de participar. Como diriam em Guadalajara, Granza "jugó fuerte" — apostou alto — e, infelizmente, suas habilidades de poker terrestre pouco serviram contra as complexas estratégias de At-Poker e ele logo encontrou-se sem grande parte de sua fortuna.

No At-Poker, N jogadores numerados de 1 a N se arranjam em uma mesa circular na ordem $1, 2, 3...N$, de modo que para todo jogador i de 1 até $N-1$, ele está à esquerda do jogador $i+1$, e o jogador N está à esquerda do jogador 1.

No jogo de At-Poker, cada jogador recebe um deck de cartas representado por uma string de caracteres minúsculos e um número G_i. O livro de regras oficial diz o seguinte:

Regras do At-Poker

1. **Formação da Mão:** Cada jogador i forma sua mão a partir da concatenação dos decks dos G_i jogadores à sua esquerda, além do próprio deck. Isso é feito de maneira circular, onde N é o total de jogadores. Por exemplo, se $N = 5$ e $G_i = 3$ para o jogador $i = 3$, então a mão do jogador 3 é formada pela concatenação dos decks dos jogadores 5, 1, 2 e 3, nessa ordem.

2. **Coringas e Pontuações:** Existem algumas strings que são colocadas na mesa, e essas são chamadas coringas. Cada string coringa j tem um valor X_j associado.

3. **Cálculo da Pontuação:** A pontuação de um jogador é determinada pelo valor mais alto entre os coringas que ocorrem em sua mão como uma subsequência contígua, onde a ocorrência de um coringa deve terminar dentro do deck original do jogador, ou seja, não pode estar totalmente no deck dos jogadores à esquerda dele. Se nenhum coringa ocorrer na mão do jogador, a pontuação do mesmo será zero.

Determinado a não deixar Guadalajara como um perdedor, Granza se propôs a aprender as estratégias do At-Poker. Ele sabia que a única maneira de retornar à Terra com suas perdas recuperadas seria se ele pudesse garantir uma vitória. Para isso, ele necessitava de um sistema que pudesse prever, com base nas mãos iniciais de cada jogador, quantos pontos cada um faria ao final de uma rodada.

Entrada

A primeira linha da entrada contém um inteiro N ($1 \leq N \leq 10^5$), que representa o número de jogadores na mesa.

As próximas N linhas descrevem cada jogador. A i-ésima linha contém uma string s_i ($1 \leq |s_i| \leq 10^5$) de caracteres minúsculos (de "`a`" até "`z`") e um inteiro G_i ($0 \leq G_i \leq N-1$). A string s_i representa o deck de cartas do jogador i e o inteiro G_i indica quantos decks dos jogadores à sua esquerda são usados para formar sua mão no jogo.

A linha seguinte contém um inteiro M ($1 \leq M \leq 10^5$), que indica o número de coringas disponíveis na mesa.

As próximas M linhas listam os coringas. A i-ésima linha contém uma string t_i ($1 \leq |t_i| \leq 10^5$) e um inteiro X_i ($1 \leq X_i \leq 10^9$). A string t_i é o padrão a ser procurado nas mãos dos jogadores, e o inteiro X_i é a pontuação atribuída a esse padrão se ele for encontrado na mão de um jogador, terminando dentro de seu deck original.

É garantido que tanto o somatório dos comprimentos das strings s_i quanto o somatório dos comprimentos das strings t_i não ultrapassam 10^5.

Saída

A saída deve consistir de uma linha contendo N inteiros separados por espaço, o i-ésimo valor deve ser a pontuação do jogador i no jogo dado.

Exemplos

entrada padrão	saída padrão
3 abd 0 dcbca 0 dbbca 0 3 ab 3 dc 7 c 5	3 7 5
5 adui 0 aba 0 gbcaffdf 0 abfh 0 abbfcafd 0 8 bfc 5 fd 4 ab 7 daduia 16 bagbc 12 bca 1 i 2 fh 10	2 7 4 10 7
5 adui 2 aba 2 gbcaffdf 1 abfh 0 abbfcafd 3 8 bfc 5 fd 4 ab 7 daduia 16 bagbc 12 bca 1 i 2 fh 10	2 16 12 10 7

Problema K: Kosmos

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

Machado, um renomado pesquisador da Agência Espacial do BRUTE, inventou um dispositivo que ele acredita ser capaz de revelar padrões importantes para a trajetória da órbita dos planetas. Esse dispositivo, batizado de "Kosmos", lhe apresentou uma sequência numérica revolucionária, porém, como ele foi programado em C puro, houve um vazamento de memória que causou a autodestruição do dispositivo. Machado, depois de descobrir isso, imediatamente começou a programar tudo o que ele faz em Rust. Agora, abalado por ter perdido a sua invenção, a única coisa que ele lembra é que a sequência apresentada pelo dispositivo era definida pela seguinte fórmula recursiva:

$$F(0) = 1$$

$$F(1) = 2$$

$$F(n) = F(n-1) \cdot F(n-2), \text{ se } n \geq 2$$

Kosmos tinha poder computacional o bastante para calcular qualquer termo dessa sequência de 1 até 10^{18}. Como valor desse termo pode ser imenso, Machado só está interessado no resto da divisão desse número por 998244353. Porém, sem Kosmos, Machado não faz ideia de como computar o N-ésimo termo dessa sequência se o N for tão grande. Por isso, ele pediu a sua ajuda.

Entrada

A entrada consiste de uma linha com um único número inteiro, $N (0 \leq N \leq 10^{18})$.

Saída

Imprima o N-ésimo termo da sequência gerada por Kosmos. Como esse valor pode ser muito grande, imprima o resto da divisão dele por 998244353.

Exemplos

entrada padrão	saída padrão
0	1
1	2
5	32
123456789123456789	433257388
998244353	470934745

Problema L: Lango Mocos

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

Dudu é um grande astronauta da empresa BRUTE — Brazilian Universal Technological Expeditions, e ele recebeu a tarefa difícil de viajar para o planeta "LK-4D4" e coletar algumas rochas preciosas classificadas como "lango mocos".

Depois de anos em sono criogênico, Dudu finalmente chegou a esse planeta e começou a estudar os lango mocos que existem lá. Após muito tempo estudando, ele descobriu que os lango mocos possuem componentes químicos nunca vistos antes pelos humanos e que determinados tipos dessa rocha são tóxicos quando em contato com alguns outros tipos.

Ele separou os lango mocos em N tipos, e escreveu em seu caderno M pares. Um par (u, v) escrito no caderno de Dudu significa que o lango moco do tipo u é tóxico quando em contato com o lango moco do tipo v (e vice versa).

Fascinado com os lango mocos, Dudu decidiu que quer levar todos os tipos dessa rocha preciosa de volta para a Terra. Como a viagem de volta é muito longa, Dudu tem que guardar os lango mocos dentro de sacolas especiais que os protegem dos perigos do espaço, e obviamente ele não quer botar dois lango mocos que são tóxicos entre si em uma mesma sacola. Dentro de cada sacola cabe uma quantidade infinita de lango mocos, pois as mesmas são feitas de um material expansível desenvolvido pelo BRUTE.

Mais formalmente, se uma sacola tem os tipos $a_1, a_2, ..., a_k$, não pode existir nenhum par (i, j) $(1 \leq i, j \leq k)$ tal que a_i é tóxico com a_j.

Infelizmente, Dudu só tem duas sacolas especiais, e agora ele não sabe como vai separar os lango mocos.

Para resolver esse problema, Dudu mandou uma mensagem para você, um dos melhores programadores do BRUTE, e pediu para que separe todos os lango mocos em 2 sacolas, ou dizer que essa missão é impossível. Se existir mais de uma maneira de fazer essa separação, você pode escolher qualquer uma dessas maneiras.

Entrada

A primeira linha da entrada possui dois inteiros $N(1 \leq N \leq 10^5)$ e $M(0 \leq M \leq min(10^5, \frac{N \cdot (N-1)}{2}))$, a quantidade de tipos de lango mocos e a quantidade de pares que Dudu escreveu em seu caderno.

As próximas M linhas contêm dois inteiros cada, u e v, representando que o lango moco do tipo u é tóxico com o lango moco do tipo v (e vice versa).

Saída

A primeira linha da saída deve consistir de uma palavra "`POSSIVEL`" (sem aspas) caso seja possível fazer a separação ou "`IMPOSSIVEL`" (sem aspas) caso seja impossível.

Se a resposta for possível, você deve imprimir mais 3 linhas.

Na primeira dessas linhas deve haver dois inteiros $K(K \geq 0)$ e $H(H \geq 0)$ sendo $K + H = N$, a quantidade de lango mocos na primeira e na segunda sacola, respectivamente.

A segunda linha deve conter K inteiros $a_1, ..., a_K$, os tipos de lango mocos na primeira sacola.

A terceira linha deve conter H inteiros $b_1, ..., b_H$, os tipos de lango mocos na segunda sacola.

Os lango mocos de cada sacola podem ser impressos em qualquer ordem.

Exemplos

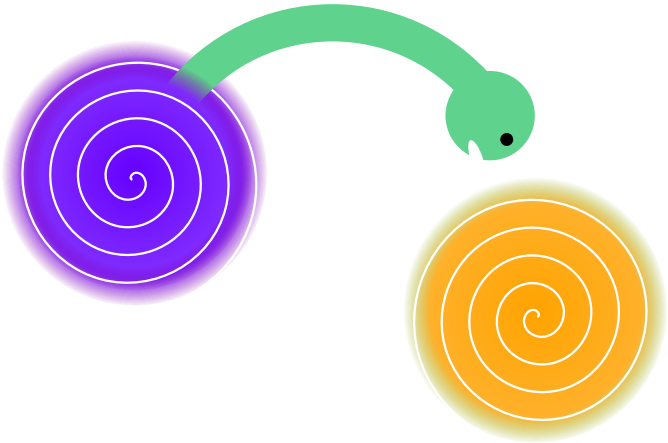
entrada padrão	saída padrão
2 1 1 2	POSSIVEL 1 1 1 2
1 0	POSSIVEL 1 0 1
6 5 1 2 2 3 3 1 4 5 5 6	IMPOSSIVEL
5 4 1 2 1 3 1 4 3 5	POSSIVEL 2 3 1 5 2 3 4

Problema M: Minhocas Gigantes

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

Joãozinho é um astrônomo que acabou de chegar na UFMG para começar o seu mestrado sobre as "minhocas" do multiverso.

O multiverso é o conjunto de todos os universos que existem, e Joãozinho está estudando uma maneira hierárquica de representar o multiverso. No mestrado dele, cada universo no multiverso recebe um inteiro u que o representa, e nesse universo, existem várias minhocas gigantes que possibilitam viagens para outros universos.



A primeira coisa que Joãozinho fez ao começar a sua pesquisa foi anotar todos os fatos que ele sabe sobre o multiverso, aqui está tudo que ele anotou.

- Se um universo u possui uma minhoca gigante que o permite viajar para um universo v, então é fisicamente impossível que o universo v tenha uma minhoca gigante que o permite viajar para o universo u.
- O multiverso pode ser modelado por uma árvore direcionada, onde os nodos dessa árvore são os universos e uma aresta entre o universo u e v representa que existe uma minhoca gigante que permite viajar do universo u para o universo v.
- Existe exatamente um universo do qual a partir dele, é possível visitar todos os outros universos. Joãozinho deu a esse universo o número 1.
- Se o universo u possui uma minhoca que o permite viajar para o universo v, então a quantidade de estrelas no universo u é **estritamente maior** que a quantidade de estrelas do universo v.

A orientadora de mestrado de Joãozinho, Karina, lhe deu como primeira tarefa a escrita de um programa que simula um multiverso qualquer que obedece as regras do multiverso real, e ela precisa que esse programa seja capaz de responder perguntas muito específicas sobre qualquer multiverso.

Após semanas de trabalho duro, Joãozinho está quase acabando essa tarefa, falta apenas uma "sub-tarefa" que ele não consegue resolver, e ela é definida da seguinte maneira:

Dado K universos $a_1, a_2, \cdots, a_K$, calcule a seguinte soma:

$$\sum_{i=1}^{K} \sum_{j=i}^{K} f(i,j)$$

Onde $f(i,j)$ é o número que representa o universo com a menor quantidade de estrelas e que também é possível, a partir dele, viajar para os universos $a_i, a_{i+1}, \cdots, a_{j-1}, a_j$.

Joãozinho está sem ideias de como resolver esse problema, então ele pediu para você escrever um programa que recebe Q perguntas desse tipo, e as responde.

Entrada

A primeira linha da entrada possui dois inteiros N ($1 \leq N \leq 10^5$) e Q ($1 \leq Q \leq 10^5$), o número de universos no multiverso simulado, e a quantidade de perguntas, respectivamente.

As próximas $N-1$ linhas possuem dois inteiros u e v ($1 \leq u, v \leq N$), representando que há uma minhoca gigante no universo u que permite viajar para o universo v.

É garantido que o multiverso dado obedece todos os fatos e convenções que Joãozinho anotou.

As próximas Q linhas possuem um inteiro K ($1 \leq K \leq N$), seguido de K inteiros $a_1, \cdots, a_K$, os universos para a pergunta. É garantido que os universos para a pergunta são todos distintos, também **é garantido que a soma de K sobre todas as perguntas não passa de** $3 \cdot 10^5$.

Saída

A saída deve consistir de Q linhas, para cada linha você deve imprimir um único inteiro, a resposta para a pergunta.

Exemplos

entrada padrão	saída padrão
5 2 1 2 1 3 3 4 3 5 1 2 2 4 5	2 12
4 1 1 2 1 3 1 4 3 2 3 4	12
1 1 1 1	1

Notas

Explicação para o primeiro exemplo:

Na primeira pergunta, apenas um universo é dado, então o universo com menor quantidade de estrelas que consegue chegar no universo 2 é o próprio universo 2. Na segunda pergunta, a resposta é $f(1,1) + f(1,2) + f(2,2) = 4 + 3 + 5 = 12$.

Problema N: Navegação entre Escudos

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

Enzo é um grande fã do jogo Lemmings, e a suas partes favoritas do jogo são proteger os Lemmings dos perigos e quando eles constroem rampas para ir de um lugar a outro em segurança.

Enzo, como acha o jogo Lemmings muito datado, resolveu criar uma versão futurística do jogo. Batizou seu jogo de "Almeidings 3077", e fez ele em 2D visto de cima.

Em Almeidings 3077, há várias criaturas – os Almeidings – que devem sair de um lugar inicial (x_i, y_i) e ir até um lugar final (x_f, y_f) de forma segura. Para haver um paralelo com a construção de rampas do jogo original, Enzo criou a mecânica de Escudos Protetores. Um Escudo Protetor é uma estrutura feita pelos Almeidings que permite que eles passem em segurança por áreas perigosas.

Enzo definiu as seguintes regras para seu jogo:

- O jogo é em 2D e o seu espaço pode ser representado por uma matriz $N \times M$.
- Um Escudo Protetor tem formato de cruz e centro em (x, y). Um Escudo Protetor com centro em (x, y) protege todas as células que estão na linha x e todas as células que estão na coluna y.
- Um Almeiding pode estar na célula (x, y) se e somente se essa célula estiver protegida por um Escudo Protetor.
- Os Almeidings podem se mover em 4 direções (direita, esquerda, cima, baixo):
  - De (x, y) para $(x + 1, y)$.
  - De (x, y) para $(x - 1, y)$.
  - De (x, y) para $(x, y + 1)$.
  - De (x, y) para $(x, y - 1)$.

Enzo, depois de decidir que o objetivo do jogo é fazer os Almeidings conquistar os espaços, decidiu simulá-lo com você. Ele te fará Q consultas, e cada consulta pode ser de um dos dois tipos:

- Tipo 1: Fazer com que um Almeiding construa um Escudo Protetor com centro em (x, y).
- Tipo 2: Se houver um Almeiding em (x_i, y_i), há como ele chegar em (x_f, y_f)?

A sua tarefa é responder todas as perguntas do tipo 2 corretamente.

Entrada

A primeira linha de entrada contém três inteiros N, M ($1 \leq N \cdot M \leq 10^6$), as dimensões da matriz que representa o jogo, seguidos por Q ($1 \leq Q \leq 2 \cdot 10^5$), a quantidade de consultas que Enzo vai fazer.

As próximas Q linhas representam as consultas, que podem ser de dois tipos:

- 1 x y: Tipo 1, construir um Escudo Protetor com centro em (x, y) $(1 \leq x \leq N, 1 \leq y \leq M)$.
- 2 x_i y_i x_f y_f: Tipo 2, se há como um Almeiding ir de (x_i, y_i) a (x_f, y_f) $(1 \leq x_i, x_f \leq N)$, $(1 \leq y_i, y_f \leq M)$.

Saída

Para cada pergunta do tipo 2, imprima uma linha contendo "`SIM`" ou "`NAO`", a resposta para a pergunta.

Exemplos

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

Problema O: Osmos V

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

Malu está em uma viagem ao distante planeta Osmos V a bordo de sua nave, a HB-20000. Viajando a uma velocidade próxima à da luz, Malu não consegue determinar sua posição exata no universo. No entanto, ela sabe que a duração total da viagem é de exatamente X anos e que, até agora, já se passaram Y anos.

Em quantos anos Malu vai chegar ao seu destino?

Entrada

A entrada consiste em duas linhas.

A primeira linha contém o inteiro X ($1 \leq X \leq 10^9$), que representa quantos anos a viagem inteira terá.

A segunda linha contém um inteiro Y ($0 \leq Y \leq X$), que representa quantos anos já se passaram.

Saída

A saída deve conter um inteiro, que representa a quantidade de anos que falta para Malu chegar ao seu destino.

Exemplos

entrada padrão	saída padrão
10 5	5
1 1	0
100000 4312	95688