

Competidor(a): _____

Número de inscrição: _____ – _____ (opcional)

Este Caderno de Tarefas não pode ser levado para casa após a prova. Após a prova entregue este Caderno de Tarefas para seu professor guardar. Os professores poderão devolver os Cadernos de Tarefas aos competidores após o término do período de aplicação das provas (3 de outubro de 2026).



Olimpíada Brasileira de Informática

OBI2026

Caderno de Tarefas

Modalidade Programação • Nível 1 • Fase 3

3 de outubro de 2026

A PROVA TEM DURAÇÃO DE 4 HORAS

Promoção:



Sociedade Brasileira de Computação

Apoio:



Coordenação:



Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 11 páginas (não contando a folha de rosto), numeradas de 1 a 11. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar a Internet, livros, anotações ou qualquer outro material durante a prova. É permitida a consulta ao *help* do ambiente de programação se este estiver disponível.
- As tarefas têm o mesmo valor na correção.
- A correção é automatizada, portanto siga atentamente as exigências da tarefa quanto ao formato da entrada e saída de seu programa; em particular, seu programa não deve escrever frases como “Digite o dado de entrada.” ou similares.
- Não implemente nenhum recurso gráfico nas suas soluções (janelas, menus, etc.), nem utilize qualquer rotina para limpar a tela ou posicionar o cursor.
- As tarefas **não** estão necessariamente ordenadas, neste caderno, por ordem de dificuldade; procure resolver primeiro as questões mais fáceis.
- Preste muita atenção no nome dos arquivos fonte indicados nas tarefas. Soluções na linguagem C devem ser arquivos com sufixo *.c*; soluções na linguagem C++ devem ser arquivos com sufixo *.cc* ou *.cpp*; soluções na linguagem Pascal devem ser arquivos com sufixo *.pas*; soluções na linguagem Java devem ser arquivos com sufixo *.java* e a classe principal deve ter o mesmo nome do arquivo fonte; soluções na linguagem Python 3 devem ser arquivos com sufixo *.py*; e soluções na linguagem Javascript devem ter arquivos com sufixo *.js*.
- Na linguagem Java, **não** use o comando *package*, e note que o nome de sua classe principal deve usar somente letras minúsculas (o mesmo nome do arquivo indicado nas tarefas).
- Para tarefas diferentes você pode escolher trabalhar com linguagens diferentes, mas apenas uma solução, em uma única linguagem, deve ser submetida para cada tarefa.
- Ao final da prova, para cada solução que você queira submeter para correção, copie o arquivo fonte para o seu diretório de trabalho ou pen-drive, conforme especificado pelo seu professor.
- Não utilize arquivos para entrada ou saída. Todos os dados devem ser lidos da entrada padrão (normalmente é o teclado) e escritos na saída padrão (normalmente é a tela). Utilize as funções padrão para entrada e saída de dados:
 - em Pascal: *readln*, *read*, *writeln*, *write*;
 - em C: *scanf*, *getchar*, *printf*, *putchar*;
 - em C++: as mesmas de C ou os objetos *cout* e *cin*.
 - em Java: qualquer classe ou função padrão, como por exemplo *Scanner*, *BufferedReader*, *BufferedWriter* e *System.out.println*
 - em Python: *read*, *readline*, *readlines*, *input*, *print*, *write*
 - em Javascript: *scanf*, *printf*
- Procure resolver a tarefa de maneira eficiente. Na correção, eficiência também será levada em conta. As soluções serão testadas com outras entradas além das apresentadas como exemplo nas tarefas.

Arroz

Nome do arquivo: `arroz.c`, `arroz.cpp`, `arroz.pas`, `arroz.java`, `arroz.js` ou `arroz.py`

Pedro, o rei da N-logônia, vai se casar. O grande casamento está marcado e o banquete promete ser o mais grandioso que o reino já viu. Por isso, o cozinheiro real, responsável pela festa, determinou que serão suficientes N quilos de arroz para garantir que todos os convidados sejam bem servidos.

A guilda de comerciantes do reino vende arroz em duas formas:

- Pacote de 1 kg por A reais cada unidade.
- Pacote de 5 kg por B reais cada unidade.

O cozinheiro pode comprar tantos pacotes de cada tipo quanto quiser, mas deve obter **pelo menos** N quilos de arroz para a festa. Ou seja, é permitido sobrar arroz, mas não faltar.

Como o orçamento do reino é limitado, Pedro, o rei, decretou que deve ser gasta a menor quantidade de dinheiro possível para preparar o banquete. Assim, dados os valores de N , A e B , ajude o cozinheiro real a calcular o menor custo possível para obter pelo menos N quilos de arroz.

Entrada

A primeira linha da entrada contém um inteiro N , a quantidade de quilos de arroz necessária.

A segunda linha da entrada contém um inteiro A , o preço do pacote de 1 kg.

A terceira linha da entrada contém um inteiro B , o preço do pacote de 5 kg.

Saída

A saída deve conter um único inteiro: o menor valor em reais para obter pelo menos N quilos de arroz.

Restrições

- $1 \leq N \leq 10^{10}$.
- $1 \leq A \leq 10$.
- $1 \leq B \leq 50$.

Para competidores que utilizam C++ ou Java: Observe que o valor de N na entrada pode ser muito grande para caber em um inteiro de 32 bits. É recomendado o uso de inteiros de 64 bits (`long long` em C++; `long` em Java). (*Competidores usando Python podem ignorar este aviso.*)

Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos):** Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.

- **Subtarefa 2 (30 pontos):** $N \leq 10^3$.
- **Subtarefa 3 (30 pontos):** $N \leq 10^5$.
- **Subtarefa 4 (30 pontos):** $N \leq 2 \times 10^8$.
- **Subtarefa 5 (10 pontos):** Sem restrições adicionais.

Dica de pontuação: se o seu programa funcionar corretamente apenas para as restrições de uma subtarefa, você receberá os pontos daquela subtarefa. Pode valer a pena entregar uma solução parcial!

Exemplos

Exemplo de entrada 1	Exemplo de saída 1
14 6 20	60

Explicação do exemplo 1: Temos $N = 14$, $A = 6$ e $B = 20$. Neste caso é ótimo comprar 3 pacotes de 5 kg, totalizando 15 kg (perceba que sobra 1 kg), o custo total é $= 3 \times 20 = 60$ reais.

Exemplo de entrada 2	Exemplo de saída 2
100 10 2	40

Exemplo de entrada 3	Exemplo de saída 3
16 6 20	66

Exemplo de entrada 4	Exemplo de saída 4
10000000000 1 3	6000000000

Explicação do exemplo 4: Neste exemplo, $N = 10^{10}$, $A = 1$ e $B = 3$. A resposta é $2 \times 10^9 \times 3 = 6\,000\,000\,000$.

Atenção: a resposta deste exemplo não cabe em um inteiro de 32 bits. Em C++, utilize `long long int`; em Java, utilize `long`.

Campeonato de Robótica

Nome do arquivo: `campeonato.c`, `campeonato.cpp`, `campeonato.pas`, `campeonato.java`,
`campeonato.js` ou `campeonato.py`

Na prova prática do campeonato regional de robótica, as equipes precisam cumprir uma tarefa de organização na arena. Tudo começa quando um robô da organização alinha N pequenas caixas numeradas em uma linha reta sobre o tapete. A ordem em que esse primeiro robô deixa as caixas forma a sequência inicial A .

O seu robô precisa entrar na arena logo em seguida e reorganizar essas caixas até que elas fiquem exatamente na ordem da sequência alvo B , que é definida pelos juízes. Você é o membro da equipe responsável por programar o robô que fará todo esse processo de organização.

Para realizar a tarefa, o seu robô foi construído com um sistema de garra dupla. O detalhe é que o chassi é rígido e a distância entre as duas garras não pode ser ajustada. Por causa desse design estrutural, o seu robô só consegue agarrar e trocar de posição duas caixas se a distância entre elas na linha for exatamente igual a d posições.

O robô pode repetir essa manobra de troca quantas vezes o seu código comandar, respeitando sempre a distância mecânica d . Como o tempo da prova é curto, você precisa garantir que a estratégia vai funcionar antes de ligar a máquina.

Sua tarefa é escrever um programa que leia a sequência inicial A , a sequência alvo B e a medida d da garra do robô, e então calcule rapidamente a viabilidade da prova, informando apenas se é possível ou não organizar as caixas na ordem exigida pelo campeonato.

Entrada

A primeira linha da entrada contém dois inteiros N, d , onde N é o número de caixas e d é a medida da garra do robô.

A segunda linha da entrada contém N inteiros A_i , onde A_i é o número da i -ésima caixa da sequência A .

A terceira, e última, linha da entrada contém N inteiros B_i , onde B_i é o número da i -ésima caixa da sequência B .

Saída

A saída deve conter uma única linha contendo um único caractere: **S** caso seja possível, ou **N** caso contrário.

Restrições

- $1 \leq N \leq 10^5$.
- $1 \leq d < N$.
- $1 \leq A_i \leq 10^9$, para todo $1 \leq i \leq N$.
- $1 \leq B_i \leq 10^9$, para todo $1 \leq i \leq N$.

Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos):** Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (18 pontos):** $N \leq 100$ e $A_i, B_i \leq 10^5$ para todo $1 \leq i \leq N$.
- **Subtarefa 3 (10 pontos):** $d = 1$ e $A_i, B_i \leq 10^5$ para todo $1 \leq i \leq N$.
- **Subtarefa 4 (21 pontos):** $d = 2$.
- **Subtarefa 5 (23 pontos):** $B_i = i$.
- **Subtarefa 6 (28 pontos):** Sem restrições adicionais.

Dica de pontuação: se o seu programa funcionar corretamente apenas para as restrições de uma subtarefa, você receberá os pontos daquela subtarefa. Pode valer a pena entregar uma solução parcial!

Exemplos

Exemplo de entrada 1	Exemplo de saída 1
6 2 9 7 3 5 3 3 3 3 9 5 3 7	S

Explicação do exemplo 1: As trocas podem ser feitas da seguinte forma:

- [9, 7, 3, 5, 3, 3] - início
- [9, 7, 3, 3, 3, 5] - após trocar A_4 e A_6 (note que a distância é exatamente $d = 2$)
- [3, 7, 9, 3, 3, 5] - após trocar A_1 e A_3
- [3, 3, 9, 7, 3, 5] - após trocar A_2 e A_4
- [3, 3, 9, 5, 3, 7] - após trocar A_4 e A_6

Exemplo de entrada 2	Exemplo de saída 2
4 2 2 4 1 3 1 2 3 4	N

Explicação do exemplo 2: Este exemplo satisfaz as restrições da subtarefa 5. Perceba que neste exemplo só é possível chegar nas seguintes ordens: [2, 4, 1, 3], [1, 4, 2, 3], [2, 3, 1, 4], [1, 3, 2, 4].

Exemplo de entrada 3	Exemplo de saída 3
5 1 1000 2000 3000 4000 5000 1000 2000 3000 4000 5	N

Explicação do exemplo 3: Este exemplo satisfaz as restrições da subtarefa 3.

Bicicleta

Nome do arquivo: `bicicleta.c`, `bicicleta.cpp`, `bicicleta.pas`, `bicicleta.java`,
`bicicleta.js` ou `bicicleta.py`

Henrique é um ciclista apaixonado por trilhas de *mountain bike*. Ele encontrou um novo mapa de uma montanha que possui N pontos de descanso, interligados por M trilhas de mão dupla. O mapa informa a altitude H_i de cada ponto de descanso, que são numerados de 1 a N .

Percorrer essas trilhas exige muita energia. Se Henrique atravessar uma trilha, indo de um ponto A para um ponto B , e a altitude de B for maior que a de A , ele terá que realizar um grande esforço para subir a diferença de $H_B - H_A$ metros de altitude. Por outro lado, se a altitude de B for menor que a de A , ele precisará controlar os freios na descida, enfrentando um esforço correspondente à diferença $H_A - H_B$.

Vale lembrar que os esforços se acumulam separadamente ao longo do trajeto. Por exemplo: se o garoto subir 7 metros, depois descer 5 metros, descer mais 4 metros e então subir 8 metros, ele terá subido no total $7 + 8 = 15$ metros e descido $5 + 4 = 9$ metros.

Henrique começará seu trajeto no ponto 1 e deseja chegar ao ponto N . No entanto, conhecendo seu preparo físico atual, ele sabe que aguenta subir no máximo X metros, e descer no máximo Y metros ao longo de todo o percurso.

Assim, dados o mapa da montanha e os limites X e Y , ajude o garoto a descobrir se é possível completar essa trilha com seu preparo físico atual.

Entrada

A primeira linha contém quatro inteiros N , M , X e Y , representando o número de pontos de descanso, o número de trilhas, o limite máximo de subida e o limite máximo de descida, respectivamente.

A segunda linha contém N inteiros $H_1, H_2, \dots, H_N$, em que H_i é a altitude do ponto de descanso i .

As próximas M linhas descrevem as trilhas. Cada linha contém dois inteiros U e V , indicando que existe uma trilha de mão dupla conectando os pontos U e V .

Saída

A saída deve conter uma única linha contendo um único caractere: **S** caso seja possível ir do ponto 1 ao ponto N respeitando os limites, ou **N** caso contrário.

Restrições

- $2 \leq N \leq 10^5$
- $1 \leq M \leq 2 \times 10^5$
- $0 \leq X, Y \leq 2 \times 10^9$
- $0 \leq H_i \leq 10^4$, para todo $1 \leq i \leq N$

Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos):** Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (8 pontos):** $X = 2 \times 10^9$ e $Y = 2 \times 10^9$.
- **Subtarefa 3 (9 pontos):** $Y = 0$.
- **Subtarefa 4 (15 pontos):** $N, X, Y \leq 100$ e $M \leq 200$.
- **Subtarefa 5 (24 pontos):** $N, X \leq 1000$ e $M \leq 2000$.
- **Subtarefa 6 (26 pontos):** $H_1 = H_N$ e $X = Y$.
- **Subtarefa 7 (18 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	Exemplo de saída 1
4 3 15 15 10 20 10 20 1 2 2 3 3 4	N

Explicação do exemplo 1: Neste exemplo, Henrique precisa ir do ponto 1 ao 4 e o único caminho possível é $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$. As variações de altitude são: sobe 10 metros (do 1 para o 2), desce 10 metros (do 2 para o 3) e sobe 10 metros (do 3 para o 4). O total subido é 20, mas o limite X é 15, portanto é impossível.

Exemplo de entrada 2	Exemplo de saída 2
4 3 25 25 10 20 10 20 1 2 2 3 3 4	S

Explicação do exemplo 2: Neste exemplo o mapa é idêntico ao anterior, mas o preparo físico de Henrique permite subir até 25 metros e descer até 25 metros, o que é suficiente para realizar o percurso que exige subir 20 metros e descer 10 metros.

Exemplo de entrada 3	Exemplo de saída 3
5 2 2000000000 2000000000 10 20 30 40 50 1 2 4 5	N

Explicação do exemplo 3: Este exemplo satisfaz as restrições da subtarefa 2.

Exemplo de entrada 4	Exemplo de saída 4
4 3 100 0 10 20 30 40 1 2 2 3 3 4	S

Explicação do exemplo 4: Este exemplo satisfaz as restrições da subtarefa 3.

Exemplo de entrada 5	Exemplo de saída 5
12 15 7 6 6 19 20 1 0 5 9 18 17 12 15 7 6 10 12 5 6 1 4 11 3 5 8 5 5 6 2 7 10 2 11 3 5 7 5 9 2 8 11 10 7 12	S

Exemplo de entrada 6	Exemplo de saída 6
12 15 6 6 6 19 20 1 0 5 9 18 17 12 15 7 6 10 12 5 6 1 4 11 3 5 8 5 5 6 2 7 10 2 11 3 5 7 5 9 2 8 11 10 7 12	N

Terreno

Nome do arquivo: `terreno.c`, `terreno.cpp`, `terreno.pas`, `terreno.java`, `terreno.js` ou `terreno.py`

A grande corporação de tecnologia RetangulCorp está planejando construir sua nova super-sede no Condado de Byte, que é famoso por sua geografia peculiar, onde o terreno pode ser perfeitamente mapeado como uma grade (matriz) de N linhas por M colunas.

Atualmente, o condado é dividido entre vários proprietários diferentes. Cada proprietário é dono de uma **região conexa** na grade, ou seja, é possível ir de qualquer quadrado do terreno de um dono para qualquer outro quadrado do mesmo dono andando apenas para cima, para baixo, para a esquerda ou para a direita, sem sair da propriedade deste dono.

A RetangulCorp está avaliando Q opções diferentes de local para a sede. Cada opção é um sub-retângulo do Condado de Byte, definido por suas coordenadas. A empresa sabe que a parte mais difícil da compra não é o dinheiro, mas sim as negociações. Eles querem saber, para cada opção de sub-retângulo, com quantos proprietários **diferentes** eles terão que negociar (ou seja, quantas regiões distintas possuem pelo menos um quadrado dentro da opção escolhida).

Como você é o engenheiro de dados da RetangulCorp, sua tarefa é escrever um programa que responda rapidamente a todas essas consultas!

Entrada

A primeira linha da entrada contém dois inteiros N e M , representando o número de linhas e colunas do Condado de Byte.

As N linhas seguintes contêm M inteiros cada. O j -ésimo inteiro da i -ésima linha representa o identificador único do proprietário daquele quadrado. Vale lembrar que diferentes quadrados com o mesmo identificador pertencem à mesma região conexa do mesmo dono.

A linha seguinte contém um inteiro Q , o número de opções de sub-retângulos sendo consideradas pela empresa.

As Q linhas seguintes contêm quatro inteiros cada: l_1, c_1, l_2 e c_2 , indicando que a empresa quer saber o número de proprietários distintos na região que vai da linha l_1 até a linha l_2 , e da coluna c_1 até a coluna c_2 . Vale destacar que as linhas são identificadas por índices de 1 a N , de cima para baixo no mapa, e as colunas são identificadas por índices de 1 a M , da esquerda para a direita no mapa.

Saída

Para cada consulta, imprima uma única linha contendo um número inteiro: a quantidade de proprietários distintos que possuem pelo menos um quadrado dentro do sub-retângulo da consulta.

Restrições

- $1 \leq N, M \leq 10^3$
- $1 \leq Q \leq 10^5$
- O identificador de cada região é um inteiro entre 1 e 10^6 .
- O número total de regiões (donos diferentes) no Condado é no máximo 10^6 .
- $1 \leq l_1 \leq l_2 \leq N$ e $1 \leq c_1 \leq c_2 \leq M$.

Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos):** Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (11 pontos):** $N, M, Q \leq 500$.
- **Subtarefa 3 (16 pontos):** Cada região pertence inteiramente a uma única linha da matriz.
- **Subtarefa 4 (20 pontos):** O identificador de cada região é um inteiro entre 1 e 50.
- **Subtarefa 5 (22 pontos):** Todas as opções estudadas pela empresa começam no canto superior esquerdo do Condado ($l_1 = 1$ e $c_1 = 1$).
- **Subtarefa 6 (31 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	Exemplo de saída 1
3 4 1 3 2 1 1 2 2 1 1 1 1 1 2 1 1 3 2 2 1 2 4	3 2

Explicação do exemplo 1: O sub-retângulo da primeira consulta vai da linha 1 até a 3 e da coluna 1 até a 2. Corresponde a:

1	3
1	2
1	1

Nele, podemos ver quadrados pertencentes a 3 donos distintos (1, 2 e 3).

Já o sub-retângulo da segunda consulta vai da linha 2 até a 2 e da coluna 1 até a 4. Corresponde a:

1	2	2	1
---	---	---	---

Nele, há quadrados pertencentes a apenas 2 donos distintos (1 e 2).

Exemplo de entrada 2	Exemplo de saída 2
2 5 1 1 2 2 2 3 3 3 4 5 2 1 2 2 4 2 3 2 5	4 3

Explicação do exemplo 2: Este exemplo satisfaz as restrições da Subtarefa 3, pois todos os quadrados de um mesmo dono estão numa mesma linha.

Exemplo de entrada 3	Exemplo de saída 3
3 3 1 1 2 1 3 2 4 4 2 3 1 1 2 2 1 1 3 3 1 1 1 3	2 4 2

Explicação do exemplo 3: Este exemplo satisfaz as restrições da Subtarefa 5, pois todas as opções avaliadas pela empresa começam exatamente no canto superior esquerdo (1, 1).

Exemplo de entrada 4	Exemplo de saída 4
4 4 1 1 1 2 3 2 1 2 3 2 2 2 4 4 4 2 2 1 1 3 3 2 2 4 4	3 3