Competidor(a): _____

Número de inscrição: _____–_____ *(opcional)*

OBI2019

Caderno de Tarefas

Modalidade **Programação** • **Nível 2** • Fase **Nacional**

21 de setembro de 2019

A PROVA TEM DURAÇÃO DE **5** HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

Instruções

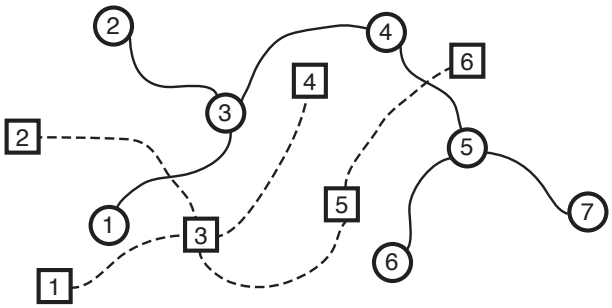
LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 9 páginas (não contando a folha de rosto), numeradas de 1 a 9. 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 tarefas 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 2 devem ser arquivos com sufixo *_py2.py*; soluções na linguagem Python 3 devem ser arquivos com sufixo *_py3.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.

Metrô da Nlogônia

Nome do arquivo: `metro.c`, `metro.cpp`, `metro.pas`, `metro.java`, `metro.js`, `metro_py2.py` *ou* `metro_py3.py`

Há dois sistemas de metrô na capital da Nlogônia, operados por duas empresas diferentes. O dois sistemas, denominados Círculo e Quadrado, são independentes e não conectados entre si, ou seja, não há nenhuma estação em comum e nenhum trilho em comum. Em cada sistema há exatamente um caminho possível entre duas estações quaisquer, possivelmente passando por outras estações do sistema. A figura abaixo mostra uma representação de dois sistemas de metrô independentes, similares ao metrô da capital da Nlogônia. Apropriadamente, no sistema Círculo as estações são representadas por círculos, e no sistema Quadrado as estações são representadas por quadrados.



Vamos chamar de *diâmetro do sistema* de metrô o maior número de estações no trajeto entre qualquer par de estações do sistema. Assim, o diâmetro do sistema Círculo na figura acima é cinco (trajeto `2-3-4-5-7` por exemplo) e o diâmetro do sistema Quadrado é quatro (trajeto `4-3-5-6` por exemplo).

O rei da Nlogônia decidiu que os dois sistemas existentes devem ser integrados, para facilitar a vida dos usuários. A integração vai ser implementada através da construção de um único novo trecho de metrô ligando exatamente um par de estações existentes (uma estação do sistema Círculo e uma estação do sistema Quadrado). O rei determinou ainda que o diâmetro do sistema integrado seja o menor possível.

Você pode ajudar a planejar a integração dos sistemas? Dadas as descrições dos dois sistemas, sua tarefa é determinar qual par de estações deve ser ligado para realizar a integração como desejada pelo rei.

Entrada

A primeira linha contém dois inteiros N e M, indicando respectivamente o número de estações do sistema Círculo e do sistema Quadrado. No sistema Círculo as estações são identificadas por números de 1 a N e no sistema Quadrado as estações são identificadas por números de 1 a M. Cada uma das $N-1$ linhas seguintes descreve as ligações entre estações do sistema Círculo e contém dois inteiros A e B indicando que existe uma ligação entre as estações A e B. Cada uma das $M-1$ linhas seguintes descreve as ligações entre estações do sistema Quadrado e contém dois inteiros X e Y indicando que existe uma ligação entre as estações X e Y.

Saída

Seu programa deve produzir dois inteiros, o primeiro representando uma estação do sistema Círculo e o segundo representando uma estação do sistema Quadrado. Se houver mais de um par possível, indique qualquer um entre os possíveis.

Restrições

- $2 \leq N, M \leq 10^5$
- $1 \leq A, B \leq N$

- $1 \leq X, Y \leq M$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 10 pontos, $N, M \leq 100$.
- Para um conjunto de casos de testes valendo outros 20 pontos, $N, M \leq 1000$.

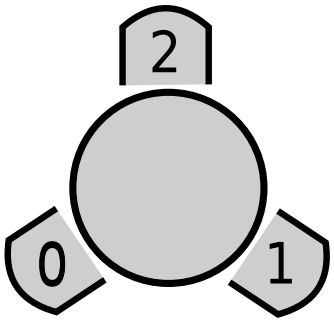
Exemplo de entrada 1	**Exemplo de saída 1**
7 6 1 3 3 2 3 4 4 5 5 7 6 5 1 3 2 3 3 4 3 5 5 6	4 3

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

Mesa redonda

Nome do arquivo: `mesa.c`, `mesa.cpp`, `mesa.pas`, `mesa.java`, `mesa.js`, `mesa_py2.py` *ou* `mesa_py3.py`

Ana, Beatriz e Carolina sempre saem juntas para tomar café numa padaria onde as mesas são circulares e têm três cadeiras numeradas 0, 1 e 2, no sentido anti-horário, como ilustrado na figura ao lado.



Elas gostam de decidir quem vai sentar em qual cadeira com uma brincadeira gerando números aleatórios nos seus celulares. Primeiro Ana sorteia um número inteiro A e, começando da cadeira 1, seguindo no sentido anti-horário, conta A cadeiras e senta na cadeira em que a contagem terminar. Depois Beatriz sorteia um número B e faz a mesma coisa: começando da cadeira 1, no sentido anti-horário, conta B cadeiras. Se a cadeira final estiver livre, Beatriz senta nela. Caso seja a cadeira onde Ana está sentada, então Beatriz senta na próxima cadeira no sentido anti-horário. Claro, ao final, Carolina senta na cadeira que estiver livre.

Por exemplo, se Ana sortear 8, ela vai contar $[1, 2, 0, 1, 2, 0, 1, 2]$ e sentar na cadeira 2. Depois, se Beatriz sortear 6, ela vai contar $[1, 2, 0, 1, 2, 0]$ e sentar na cadeira 0. Assim, Carolina senta na cadeira 1. Num outro exemplo, se Ana sortear 3, ela vai contar $[1, 2, 0]$ e sentar na cadeira 0. Depois, se Beatriz sortear 9, ela vai contar $[1, 2, 0, 1, 2, 0, 1, 2, 0]$ e, como Ana já está sentada na cadeira 0, Beatriz senta na cadeira 1. Dessa forma, Carolina senta na cadeira 2.

Neste problema, dados os números sorteados por Ana e Beatriz, seu programa deve imprimir o número da cadeira onde Carolina vai sentar.

Entrada

A primeira linha da entrada contém um inteiro A representando o número sorteado por Ana. A segunda linha da entrada contém um inteiro B representando o número sorteado por Beatriz.

Saída

Seu programa deve imprimir uma linha contendo um número inteiro indicando a cadeira onde Carolina vai sentar.

Restrições

- $1 \leq A \leq 1000$
- $1 \leq B \leq 1000$

Exemplo de entrada 1	**Exemplo de saída 1**
8 6	1

Exemplo de entrada 2	**Exemplo de saída 2**
3 9	2

Exploração do Capitão Levi

Nome do arquivo: `exploracao.c`, `exploracao.cpp`, `exploracao.pas`, `exploracao.java`, `exploracao.js`, `exploracao_py2.py` *ou* `exploracao_py3.py`

O Capitão Levi está indo para mais uma expedição pela tropa de exploração e, como sempre, ele resolveu olhar o mapa do local que ele e sua equipe estavam a caminho para que pudessem criar a melhor estratégia possível. Como todos sabem, a tropa de exploração é responsável por enfrentar titãs e deixar os habitantes da cidade mais protegidos.

O mapa do local pode ser resumido a um plano cartesiano e os titãs podem ser representados como pontos nesse plano. No entanto, seu dispositivo de manobra bidimensional(DMB) está defeituoso e agora Levi só consegue se locomover de um titã para outro titã durante o combate se eles estão em uma determinada direção, um em relação ao outro.

Se existe um titã no ponto $A = (X_a, Y_a)$ e um outro titã no ponto $B = (X_b, Y_b)$ ele consegue ir de A pra B se o coeficiente angular da reta que passa pelos pontos A e B for maior ou igual a $\frac{P}{Q}$. Observe que os pontos A e B devem ser distintos e que não existem pontos com a mesma coordenada X. Levi quer contar quantos pares de pontos distintos A e B existem, tais que há um titã em A e em B e ele consegue ir de A para B, ou seja $\frac{Y_a - Y_b}{X_a - X_b} \geq \frac{P}{Q}$.

No entanto, existem muitos titãs no mapa e por isso Levi pediu sua ajuda para contabilizar os pares, lembrando que o par (A,B) e (B,A) são o mesmo par, ou seja, a ordem dos pontos não faz diferença.

Entrada

A primeira linha da entrada contém três números inteiros N, P e Q, indicando respectivamente a quantidade de titãs, e os dois inteiros descritos no enunciado. Cada uma das N linhas seguintes contém dois inteiros X e Y, indicando as coordenadas de um titã.

Saída

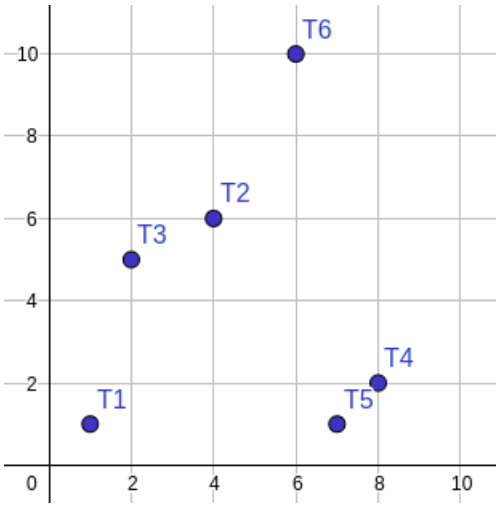
A saída consiste em um único número inteiro, representando a quantidade de pares de titãs entre os quais Levi pode se locomover respeitando as condições do enunciado.

Restrições

- $2 \leq N \leq 5 * 10^5$
- $-10^9 \leq P, Q \leq 10^9$
- $P \neq 0$ e $Q \neq 0$
- $1 \leq X, Y \leq 10^7$
- Não existem dois titãs com a mesma coordenada X

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 15 pontos, $2 \leq N \leq 10^3$, $P = 1$ e $Q = 1$.
- Para um conjunto de casos de testes valendo 20 pontos, $2 \leq N \leq 6 * 10^4$, $P = 1$ e $Q = 1$.
- Para um conjunto de casos de testes valendo 15 pontos, todos os titãs estão sobre uma mesma reta.
- Para um conjunto de casos de testes valendo 20 pontos, $P > 0$ e $Q > 0$.
- Para um conjunto de casos de testes valendo 30 pontos, não há restrições adicionais.



Exemplo de entrada 1	**Exemplo de saída 1**
6 1 1 1 1 4 6 2 5 8 2 7 1 6 10	6

Exemplo de entrada 2	**Exemplo de saída 2**
6 1 1 7 7 1 1 2 2 16 16 11 11 200 200	15

Exemplo de entrada 3	**Exemplo de saída 3**
8 418732 641936 60693 28595 15649 57089 77335 92158 57291 25242 21420 56599 62278 58106 52009 12362 41982 64916	11

Grand Prix da Nlogônia

Nome do arquivo: `prix.c`, `prix.cpp`, `prix.pas`, `prix.java`, `prix.js`, `prix_py2.py` *ou* `prix_py3.py`

A Nlogônia irá realizar o Grand Prix de corrida de carros. Foram dados planos de construção de um circuito para a realização do evento e você ficou responsável pela avaliação do plano. Um grafo direcionado de N vertices e M arestas é considerado um Grand Prix se existe algum ciclo direcionado, ou seja, existe um vertice P e um caminho direcionado saindo de P que chega novamente em P. A Nlogônia pode ser representada como um grafo direcionado que contêm N esquinas, numeradas de 1 a N. Foram dados para você M planos de construção, cada um contendo três inteiros U, L e R, que significa o seguinte: caso esse plano seja aceito, será construída uma estrada direcionada da esquina U para a esquina i, para todo $L \leq i \leq R$. Sua tarefa é computar o menor inteiro X tal que aceitando todos os planos de 1 até X, teremos um Grand Prix em Nlogônia.

Entrada

A primeira linha da entrada contém dois inteiros N e M, representando, respectivamente, o número de esquinas e o número de planos. As M linhas seguintes contêm, cada uma, três inteiros U, L e R, descrevendo um plano de construção.

Saída

Imprima um inteiro X, o menor inteiro tal que aceitando todos os planos de 1 até X, inclusive, conseguiremos um Grand Prix. Caso Nlogônia não consiga realizar o Grand Prix, imprima -1.

Restrições

- $2 \leq N \leq 200000$
- $1 \leq M \leq 200000$
- $1 \leq L \leq R \leq N$
- $1 \leq U \leq N$
- É garantido que não existe uma aresta de um vertice indo para ele mesmo.

Informações sobre a pontuação

- Em um conjunto de casos de teste valendo 10 pontos, $N \leq 200000$, $M \leq 200000$ e L = R para todo plano.
- Em um conjunto de casos de teste valendo 10 pontos, $N \leq 1000$, $M \leq 500$.
- Em um conjunto de casos de teste valendo 10 pontos, $N \leq 500$, $M \leq 20000$.
- Em um conjunto de casos de teste valendo 25 pontos, $N \leq 200000$, $M \leq 200000$ e é garantido que L = 1 para todo plano.
- Em um conjunto de casos de teste valendo 45 pontos, nenhuma restrição adicional.

Exemplo de entrada 1	**Exemplo de saída 1**
8 6 5 6 8 3 1 2 6 2 4 1 4 5 8 4 7 2 3 6	4

Exemplo de entrada 2	**Exemplo de saída 2**
2 2 1 2 2 2 1 1	2

Exemplo de entrada 3	**Exemplo de saída 3**
5 6 1 2 5 3 4 5 5 2 2 1 2 4 3 4 5 4 2 2	-1

Etiquetas

Nome do arquivo: `etiquetas.c`, `etiquetas.cpp`, `etiquetas.pas`, `etiquetas.java`, `etiquetas.js`, `etiquetas_py2.py` *ou* `etiquetas_py3.py`

Uma fita esticada horizontalmente é composta de N quadrados de dimensão 1×1, cada um deles contendo um número inteiro anotado. Temos também K etiquetas retangulares idênticas, de dimensão $1 \times C$, onde C é um inteiro. Nosso objetivo é colar todas as etiquetas sobre a fita de modo que a soma dos inteiros que não estiverem cobertos por nenhuma etiqueta ao final seja a máxima possível. Cada etiqueta deve ser colada na horizontal, ao longo da fita. Duas etiquetas não podem estar sobrepostas e cada quadrado da fita deve estar ou totalmente coberto por uma etiqueta, ou totalmente descoberto.

Entrada

A primeira linha da entrada contém três inteiros N, K e C, representando, respectivamente, o comprimento da fita, o número de etiquetas e o comprimento das etiquetas. A segunda linha da entrada contém N inteiros A indicando a sequência de números anotados nos quadrados da fita.

Saída

Seu programa deve imprimir uma linha contendo um inteiro indicando a soma máxima possível de inteiros descobertos na fita depois que todas as etiquetas sejam coladas seguindo as condições do enunciado.

Restrições

- $1 \leq N \leq 10000$
- $-10000 \leq A \leq 10000$
- $1 \leq K \leq 10000$
- $1 \leq C \leq 10000$
- $K \times C \leq N$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 10 pontos, os números no vetor estão ordenados crescentemente.
- Para um conjunto de casos de testes valendo outros 10 pontos, $C = 1$.
- Para um conjunto de casos de testes valendo outros 20 pontos, $K = 1$.
- Para um conjunto de casos de testes valendo outros 20 pontos, $K, N \leq 100$.

Exemplo de entrada 1	**Exemplo de saída 1**
`12 2 3` `1 22 4 -8 9 2 10 -1 5 5 32 -11`	`58`

Exemplo de entrada 2	**Exemplo de saída 2**
`1 1 1` `10000`	`0`