

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 (11 de setembro de 2026).



Olimpíada Brasileira de Informática

OBI2026

Caderno de Tarefas

Modalidade Programação • Nível Júnior • Competição Feminina

11 de setembro de 2026

A PROVA TEM DURAÇÃO DE 3 HORAS

Promoção:



Sociedade Brasileira de Computação

Apoio:



Coordenação:



Instruções

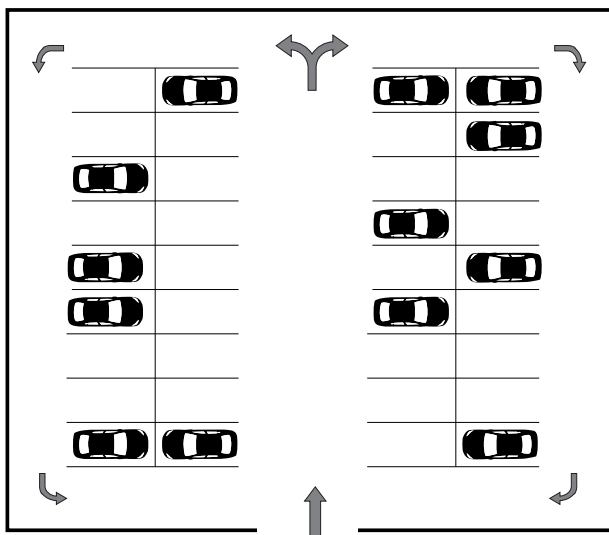
LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 14 páginas (não contando a folha de rosto), numeradas de 1 a 14. 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.

Estacionamento

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

A família de Helena está inaugurando um novo estacionamento na cidade! Helena ficou responsável por organizar a cobrança e definiu as seguintes regras: a **primeira hora** custa **R\$15,00**, e cada hora adicional custa **R\$5,00**.



Animada com a abertura, Helena percebeu que calcular o valor cobrado de cada cliente na mão poderia ser trabalhoso. Ela sabe que você sabe programar e pediu a sua ajuda para criar um programa que, dado o número de horas que um cliente usou o estacionamento, calcule automaticamente o valor a ser cobrado.

Entrada

A entrada contém um único inteiro H , o número de horas que o cliente usou o estacionamento.

Saída

A saída deve conter um único inteiro, o valor em reais a ser cobrado do cliente. Não coloque nada além disso na saída, senão será considerado errado.

Restrições

- $1 \leq H \leq 20$.

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 (100 pontos):** Sem restrições adicionais.

Exemplos**Exemplo de entrada 1**

3

Exemplo de saída 1

25

Explicação do exemplo 1: O cliente ficou $H = 3$ horas no estacionamento. A primeira hora custa R\$15,00. As 2 horas restantes custam R\$5,00 cada, totalizando R\$10,00. Portanto, o valor cobrado é $15 + 10 = 25$. A saída deve conter exatamente o número 25, nada além disso.

Exemplo de entrada 2

1

Exemplo de saída 2

15

Explicação do exemplo 2: O cliente ficou apenas $H = 1$ hora no estacionamento. Como é exatamente a primeira hora, o valor cobrado é R\$15,00. Logo, a resposta é 15.

Cabo de Guerra

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

A turma do professor Valente estava toda animada para o dia de jogos na escola. Depois de muito debate, decidiram brincar de cabo de guerra — mas havia um problema: para uma disputa justa era preciso um cabo com pelo menos X centímetros.

O professor Valente teve uma ideia: ele sabia que no depósito da escola havia N cabos de comprimentos $A_1, A_2, \dots, A_N$ centímetros. Seria possível usar alguns desses cabos?

O problema é que amarrar cabos faz com que parte do comprimento seja perdida nos nós. Quando se amarra dois cabos de comprimentos R e S , o cabo resultante tem comprimento $R + S - 10$ centímetros (pois cada nó consome 5 cm de cada cabo). Além disso, como muitos nós tornam o cabo difícil de segurar, é permitido usar no máximo 3 cabos no total (ou seja, é possível usar 1, 2 ou 3 cabos, mas não mais do que isso).

Ajude a turma a descobrir qual é a quantidade mínima de cabos necessária para conseguir um cabo de comprimento maior ou igual a X centímetros. Se não for possível, mesmo usando 3 cabos, imprima -1 .

Entrada

A primeira linha da entrada contém dois inteiros N e X , indicando respectivamente o número de cabos disponíveis e o comprimento mínimo necessário.

A segunda linha contém N inteiros $A_1, A_2, \dots, A_N$, os comprimentos dos cabos.

Saída

A saída deve conter um único inteiro: a quantidade mínima de cabos necessária (1, 2 ou 3), ou -1 caso não seja possível obter um cabo de comprimento maior ou igual a X .

Restrições

- $3 \leq N \leq 10^5$.
- $100 \leq A_i \leq 1000$, para todo $1 \leq i \leq N$.
- $100 \leq X \leq 5000$.

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):** $N = 3$.
- **Subtarefa 3 (19 pontos):** $N \leq 100$ e é garantido que existem 3 cabos que, quando amarrados, resultam em um cabo de comprimento maior ou igual a X (ou seja, a resposta nunca será -1).

- **Subtarefa 4 (31 pontos):** $N \leq 100$.
- **Subtarefa 5 (32 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

```
3 250
120 100 150
```

Exemplo de saída 1

```
2
```

Explicação do exemplo 1: Nenhum cabo sozinho tem comprimento maior ou igual a $X = 250$ cm. Ao amarrar o cabo de 120 cm com o de 150 cm, obtemos $120 + 150 - 10 = 260 \geq 250$. Portanto, são necessários apenas 2 cabos.

Exemplo de entrada 2

```
3 300
120 100 150
```

Exemplo de saída 2

```
3
```

Explicação do exemplo 2: Neste caso é necessário amarrar os 3 cabos. Ao amarrar os cabos de 120 cm e 100 cm, obtemos $120 + 100 - 10 = 210$ cm. Em seguida, amarrando com o cabo de 150 cm, obtemos $210 + 150 - 10 = 350 \geq 300$. Não é possível obter o comprimento desejado com menos de 3 cabos.

Exemplo de entrada 3

```
4 400
120 100 150 100
```

Exemplo de saída 3

```
-1
```

Explicação do exemplo 3: Observe que juntando 3 cabos o máximo que conseguimos é 350 cm, e como não podemos juntar mais cabos, então não é possível obter um cabo maior ou igual a 400 cm, por isso a resposta é -1 .

Exemplo de entrada 4

5 200
150 110 130 100 120

Exemplo de saída 4

2

Bolinha Quicante

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

Sua grande amiga Sandra acabou de te apresentar o mais novo sucesso dos jogos de celular: o *Bolinha Quicante*. O jogo consiste em uma pista reta infinita onde você deve arremessar uma bolinha mágica a partir de uma coordenada inteira X à sua escolha.

A mecânica principal do jogo é simples, mas dominar a física da bolinha é o verdadeiro desafio. A bolinha possui um pulo fixo de tamanho B . Isso significa que, se você a lançar da posição X , ela vai quicar exatamente nas posições X , $X + B$, $X + 2B$, e assim sucessivamente, quicando infinitamente pela pista.

Sandra te mostrou o mapa da fase em que está travada: existem N moedas reluzentes espalhadas pelo caminho. A i -ésima moeda está localizada na coordenada x_i e vale p_i pontos. Vale ressaltar que todas as moedas garantem pontos positivos e, em alguns casos, pode haver mais de uma moeda acumulada em uma mesma coordenada. Se a bolinha cair em uma dessas coordenadas, todas as moedas daquela posição são coletadas juntas.

Sabendo que você é craque em algoritmos, Sandra te fez um desafio: analisando o mapa da fase, qual é a maior pontuação total que é possível alcançar fazendo um **único arremesso**, escolhendo a posição inicial X de forma estratégica?

Entrada

A primeira linha de entrada contém dois inteiros N , B , representando a quantidade de moedas e o tamanho do pulo da bolinha. A segunda linha contém N inteiros $x_1, x_2, \dots, x_N$, representando a posição das moedas. A terceira linha contém N inteiros $p_1, p_2, \dots, p_N$, representando a pontuação das moedas.

Saída

A saída deve conter um único inteiro: a maior pontuação possível de se fazer com um único arremesso.

Restrições

- $1 \leq N \leq 10^5$.
- $1 \leq B \leq 10^5$.
- $0 \leq x_i \leq 10^9$, para todo $1 \leq i \leq N$.
- $1 \leq p_i \leq 10^4$, 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 (13 pontos):** $B = 1$.
- **Subtarefa 3 (14 pontos):** $N = 2$.
- **Subtarefa 4 (14 pontos):** $N \leq 1000$.
- **Subtarefa 5 (29 pontos):** $x_i \leq 10^5$, para todo $1 \leq i \leq N$.
- **Subtarefa 6 (30 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

```
4 3
10 7 2 5
4 1 3 3
```

Exemplo de saída 1

```
6
```

Explicação do exemplo 1: Neste caso, é ótimo arremessar a bolinha a partir da posição $X = 2$. Com pulos de tamanho $B = 3$, ela quicará nas posições $2, 5, 8, 11, \dots$. Com isso, ela coleta as moedas nas posições 2 (valendo 3 pontos) e 5 (valendo 3 pontos), somando 6 pontos no total. Se fosse arremessada da posição 7, ela passaria pelo 7 e pelo 10, somando apenas $1 + 4 = 5$ pontos. Portanto, a maior pontuação é 6.

Exemplo de entrada 2

```
3 2
1 3 5
10 20 30
```

Exemplo de saída 2

```
60
```

Explicação do exemplo 2: Arremessando a bolinha a partir de $X = 1$ com pulos de tamanho $B = 2$, ela passará exatamente pelas coordenadas 1, 3 e 5. Assim, ela consegue coletar todas as moedas da pista em um único arremesso, totalizando $10 + 20 + 30 = 60$ pontos.

Exemplo de entrada 3

```
5 4
2 6 10 7 7
2 2 2 30 20
```

Exemplo de saída 3

```
50
```

Explicação do exemplo 3: Neste caso, existem duas rotas principais. A primeira rota passa pelas coordenadas 2, 6 e 10, coletando três moedas que juntas somam $2 + 2 + 2 = 6$ pontos. No entanto, observe que na coordenada 7 existem duas moedas (uma valendo 30 e outra valendo 20). Ao arremessar a bolinha a partir da posição $X = 7$, ela coleta essas duas moedas de uma só vez, totalizando $30 + 20 = 50$ pontos. Portanto, vale mais a pena começar no 7, e a maior pontuação possível é 50.

Medo de Voar

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

Dona Lourdes tem muito medo de voar de avião. Sua maior preocupação é o neto, Guilherme, que adora explorar as ilhas da Nlogônia — um arquipélago com N ilhas numeradas de 1 a N .

As ilhas da Nlogônia são conectadas por M estradas. Cada estrada liga diretamente duas ilhas diferentes e pode ser percorrida nos dois sentidos. Para ir de uma ilha a outra, é possível passar por várias estradas em sequência. Porém, há ilhas que não possuem nenhuma estrada entre elas, e para viajar entre essas ilhas, Guilherme precisaria pegar um avião — o que apavora Dona Lourdes.

Guilherme planejou Q rotas de viagem. Cada rota é uma sequência de ilhas que ele pretende visitar, na ordem indicada. Para cada par de ilhas consecutivas na rota, Guilherme vai de avião caso não exista nenhum caminho por estradas entre elas. Caso contrário, ele vai de carro ou ônibus, e Dona Lourdes fica tranquila.

Ajude Dona Lourdes a descobrir, para cada rota, quantas viagens de avião Guilherme precisará fazer.

Entrada

A primeira linha da entrada contém dois inteiros N e M , indicando respectivamente o número de ilhas e o número de estradas.

As M linhas seguintes contém, cada uma, dois inteiros u e v , indicando que existe uma estrada ligando a ilha u à ilha v .

A linha seguinte contém um inteiro Q , o número de rotas planejadas.

As Q linhas seguintes descrevem as rotas. Cada linha começa com um inteiro k ($k \geq 1$), representando o número de ilhas na rota. Em seguida, na mesma linha, são dados k inteiros $a_1, a_2, \dots, a_k$, representando a sequência de ilhas da rota. É garantido que ilhas consecutivas na rota são sempre diferentes (ou seja, $a_i \neq a_{i+1}$ para todo $1 \leq i < k$).

Saída

A saída deve conter Q linhas. A i -ésima linha deve conter um único inteiro: o número de viagens de avião necessárias na i -ésima rota.

Restrições

- $2 \leq N \leq 10^5$.
- $1 \leq M \leq 2 \times 10^5$.
- $1 \leq Q \leq 10^5$.
- $1 \leq \sum k \leq 2 \times 10^5$ (soma dos tamanhos de todas as rotas).
- $1 \leq u, v \leq N$ e $u \neq v$ para toda estrada.
- $1 \leq a_i \leq N$, para todo $1 \leq i \leq k$.

Informações sobre a pontuação

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

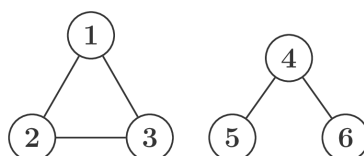
- **Subtarefa 1 (0 pontos):** Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (24 pontos):** $N \leq 1000$, $M \leq 10000$ e $\sum k \leq 1000$.
- **Subtarefa 3 (21 pontos):** Em toda estrada, uma das duas ilhas que ela liga é sempre a ilha 1.
- **Subtarefa 4 (22 pontos):** Cada grupo de ilhas interligadas forma uma rede completa, onde toda ilha está diretamente ligada a todas as outras do mesmo grupo; e $N \leq 1000$.
- **Subtarefa 5 (33 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 5 1 2 2 3 3 1 4 5 4 6 2 3 1 4 6 6 1 2 3 6 5 1	1 2

Explicação do exemplo 1: A seguir, uma visualização das ilhas neste exemplo:



A primeira rota passa pelas seguintes ilhas: 1, 4, 6. Note que:

- De 1 para 4: como não há um caminho por estradas entre as ilhas 1 e 4, é necessário um voo.
- De 4 para 6: como há uma estrada entre as ilhas 4 e 6, nenhum voo é necessário.

Portanto é necessário apenas 1 voo.

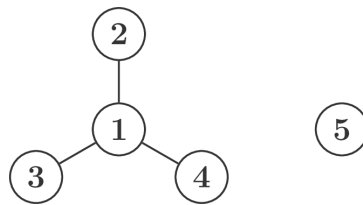
A segunda rota passa pelas seguintes ilhas: 1, 2, 3, 6, 5, 1. Note que:

- De 1 para 2: como há uma estrada entre as ilhas 1 e 2, nenhum voo é necessário.
- De 2 para 3: como há uma estrada entre as ilhas 2 e 3, nenhum voo é necessário.
- De 3 para 6: como não há um caminho por estradas entre as ilhas 3 e 6, é necessário um voo.
- De 6 para 5: como há um caminho entre as ilhas 6 e 5 (perceba que o caminho consiste em: de 6 para 4, e depois, de 4 para 5), nenhum voo é necessário.
- De 5 para 1: como não há um caminho por estradas entre as ilhas 5 e 1, é necessário um voo.

Portanto são necessários 2 voos.

Exemplo de entrada 2	Exemplo de saída 2
5 3 1 2 1 3 1 4 2 3 2 3 5 5 4 1 3 2 4	1 0

Explicação do exemplo 2: A seguir, uma visualização das ilhas neste exemplo:



Vale ressaltar que este exemplo satisfaz as restrições da Subtarefa 3.

A primeira rota passa pelas seguintes ilhas: 2, 3, 5. Note que:

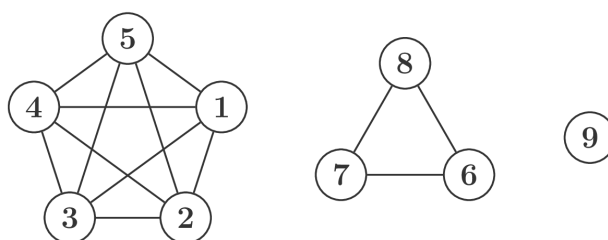
- De 2 para 3: como há um caminho por estradas entre as ilhas 2 e 3, nenhum voo é necessário.
- De 3 para 5: como não há um caminho por estradas entre as ilhas 3 e 5, é necessário um voo.

Portanto é necessário apenas 1 voo.

A segunda rota passa pelas seguintes ilhas: 4, 1, 3, 2, 4. Note que sempre há caminho entre essas ilhas, portanto, nenhum voo é necessário.

Exemplo de entrada 3	Exemplo de saída 3
<pre> 9 13 1 2 1 3 1 4 1 5 2 3 2 4 2 5 3 4 3 5 4 5 6 7 6 8 7 8 1 6 1 3 5 6 8 9 </pre>	<pre> 2 </pre>

Explicação do exemplo 3: A seguir, uma visualização das ilhas neste exemplo:



Vale ressaltar que este exemplo satisfaz as restrições da Subtarefa 4.

A única rota passa pelas ilhas (1, 3, 5, 6, 8, 9). Perceba que serão necessários dois voos, de 5 para 6, e de 8 para 9.

Exemplo de entrada 4	Exemplo de saída 4
10 9 1 10 1 7 2 7 8 7 7 4 3 5 5 6 6 9 9 3 3 2 10 4 9 1 10 8 2 7 5 9 6 3 9 1 3 10 9 8 6 7 2 5	0 1 7