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 2 • Competição Feminina

11 de setembro de 2026

A PROVA TEM DURAÇÃO DE 3 HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

Coordenação:

INSTITUTO DE COMPUTAÇÃO UNICAMP

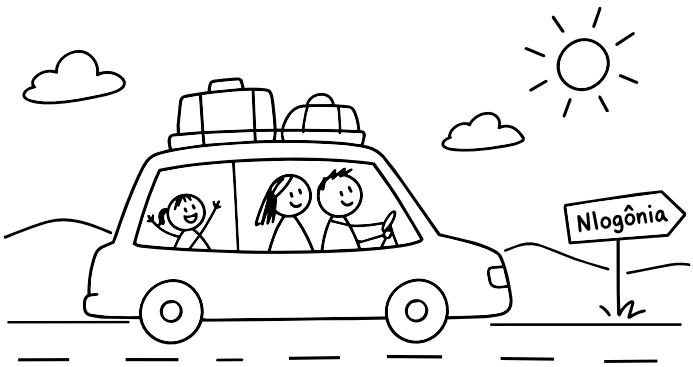
Instruções

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.

Viagem em Família

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



Liz e sua família estão planejando uma viagem. Eles vão de carro, e o pai de Liz sempre gosta de encher o tanque antes de viajar. Sabendo que a capacidade do tanque é C litros, que o carro deles faz 12 km por litro, e que a distância entre a casa da família e o destino é D km, determine se eles conseguem chegar ao destino sem ter que parar para abastecer o carro.

Entrada

A primeira linha da entrada contém um único inteiro C, a capacidade do tanque em litros.

A segunda linha da entrada contém um único inteiro D, a distância até o destino em km.

Saída

A saída deve conter uma única letra: `S` se a família consegue chegar sem parar para abastecer, ou `N` caso contrário. Não coloque nada além disso na saída, senão será considerado errado.

Restrições

- $30 \leq C \leq 80$.
- $100 \leq D \leq 1000$.

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

```
40
480
```

Exemplo de saída 1

```
S
```

Explicação do exemplo 1: Nesse caso, $C \times 12 = 40 \times 12 = 480$ km, que é exatamente igual à distância $D = 480$ km. Portanto, a família consegue chegar ao destino sem precisar parar para abastecer. A saída deve ser exatamente `S`, nada além disso.

Exemplo de entrada 2

```
40
500
```

Exemplo de saída 2

```
N
```

Explicação do exemplo 2: Nesse caso, $C \times 12 = 40 \times 12 = 480$ km, que é menor do que a distância $D = 500$ km. Portanto, a família não consegue chegar ao destino sem precisar parar para abastecer. A saída deve ser exatamente `N`, nada além disso.

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 contêm, 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 $a_1, a_2, \ldots, a_k$, 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.

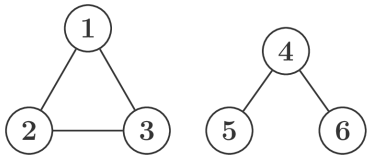
- **Subtarefa 1 (0 pontos)**: Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (24 pontos):** $N \leq 1000$, $M \leq 10000$ e $\sum k \leq 1000$.
- **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.

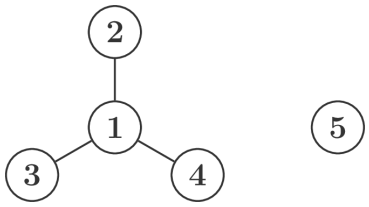
A segunda rota passa pelas seguintes ilhas: 1, 2, 3, 6, 5, 1. Note que:

- 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:

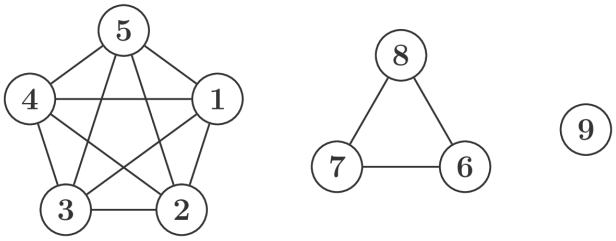
- De 2 para 3: como há um caminho por estradas entre as ilhas 2 e 3, nenhum voo é necessário.
- De 3 para 5: como não há um caminho por estradas entre as ilhas 3 e 5, é necessário um voo.

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**
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	2

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

Estoque de Moedas

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

Erika é uma pessoa que adora viajar e, como recordação dos lugares que visita, ela traz para casa diversas moedas de outros países. No entanto, recentemente sua coleção ficou tão grande que armazená-la se tornou um desafio.

Assim, a garota encomendou uma caixa personalizada que permite estocar suas moedas em N pilhas, uma ao lado da outra. Cada pilha possui um limite de altura, de forma que a i-ésima pilha pode conter no máximo H_i moedas ($1 \leq i \leq N$).

Erika estava muito animada para organizar sua coleção, mas percebeu que, dependendo de como ela a distribuía, as moedas deslizavam para os lados, e as pilhas desmoronavam. Após alguns testes, ela determinou que uma configuração é instável se existem duas pilhas adjacentes tal que a diferença da quantidade de moedas entre elas é maior que 1.

Por exemplo, para $N = 3, H_1 = 2, H_2 = 5$ e $H_3 = 4$:

- Caso Erika coloque 2, 5 e 3 moedas nas pilhas, respectivamente, essa configuração seria instável, pois a diferença da quantidade de moedas entre a primeira e segunda pilhas é maior que 1.
- No entanto, caso ela coloque 2, 3 e 4 moedas nas pilhas, respectivamente, essa configuração seria estável, pois a diferença da quantidade de moedas entre pilhas adjacentes é menor ou igual a 1.
- Note também que Erika não pode colocar 3, 4 e 4 moedas nas pilhas, respectivamente, pois essa configuração ultrapassa o limite de moedas da primeira pilha.

Assim, dadas as especificações da caixa de Erika, ajude-a a calcular a quantidade **máxima** de moedas que ela consegue guardar de maneira estável.

Entrada

A primeira linha da entrada contém um inteiro N, a quantidade de pilhas que cabem no estoque.

A segunda linha da entrada contém N inteiros $H_1, H_2, \ldots, H_N$, a quantidade máxima de moedas de cada pilha.

Saída

A saída deve conter um único inteiro, a quantidade máxima de moedas que Erika consegue guardar em seu estoque.

Restrições

- $2 \leq N \leq 10^5$
- $1 \leq H_i \leq 10^5$, 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 (15 pontos):** $H_i \leq 3$, para todo $1 \leq i \leq N$.
- **Subtarefa 3 (11 pontos):** $|H_i - H_{i+1}| = 1$, para todo $1 \leq i < N$.
- **Subtarefa 4 (16 pontos):** $H_i \leq H_{i+1}$, para todo $1 \leq i < N$.
- **Subtarefa 5 (26 pontos):** $N \leq 1000$.
- **Subtarefa 6 (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
2 5 4
```

Exemplo de saída 1

```
9
```

Explicação do exemplo 1: Este é o exemplo mostrado no enunciado. Note que não é possível guardar mais de 9 moedas na caixa.

Exemplo de entrada 2

```
5
2 3 2 1 3
```

Exemplo de saída 2

```
10
```

Explicação do exemplo 2: Note que este exemplo satisfaz as restrições da Subtarefa 2.

Exemplo de entrada 3

```
4
6 7 6 5
```

Exemplo de saída 3

```
24
```

Explicação do exemplo 3: Note que este exemplo satisfaz as restrições da Subtarefa 3.

Exemplo de entrada 4

```
8
3 3 4 7 7 7 8 10
```

Exemplo de saída 4

```
45
```

Explicação do exemplo 4: Note que este exemplo satisfaz as restrições da Subtarefa 4.

Amizade

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

A Olimpíada Brasileira de Informática chegou ao seu momento mais aguardado: a cerimônia de encerramento! Os N competidores estão alinhados em fila, numerados de 1 a N, para a foto oficial. O fotógrafo tira diversas fotos, cada uma cobrindo um intervalo contíguo de competidores: uma foto do intervalo $[l, r]$ inclui os competidores $l, l+1, \ldots, r$.

Cada competidor tem suas próprias preferências sobre a foto. O competidor i possui a_i amigos entre os outros competidores. Se $a_i > 0$, então o competidor i só ficará satisfeito com a foto se pelo menos um de seus a_i amigos também estiver nela — o amigo não precisa estar ao lado de i, basta estar presente no intervalo. Se $a_i = 0$, o competidor i não tem preferência e sempre ficará satisfeito.

Uma foto do intervalo $[l, r]$ é **válida** se todos os competidores do intervalo estiverem satisfeitos. Lembre-se de que a amizade não é necessariamente simétrica: se o competidor x deseja sair na foto com o competidor y, não significa que y deseja sair na foto com x.

Dado o número de amigos de cada competidor e a lista de amigos, determine quantas fotos válidas o fotógrafo pode tirar. Uma foto é determinada pelo intervalo $[l, r]$; portanto, fotos com intervalos distintos são consideradas fotos diferentes.

Entrada

A primeira linha da entrada contém um inteiro N, o número de competidores.

Cada uma das N linhas seguintes descreve um competidor. A linha $i+1$ (para $1 \leq i \leq N$) começa com um inteiro a_i, o número de amigos do competidor i, seguido de a_i inteiros distintos entre si e diferentes de i, que são os índices dos amigos do competidor i.

Saída

A saída deve conter um único inteiro: o número de fotos válidas.

Restrições

- $1 \leq N \leq 10^5$.
- $0 \leq S \leq 5 \times 10^5$, onde $S = a_1 + a_2 + \cdots + a_N$.
- $0 \leq a_i < N$, para todo $1 \leq i \leq N$.
- Os índices dos amigos estão entre 1 e 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. Para cada subtarefa, sua pontuação será a máxima dentre todas as submissões.

- **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 (12 pontos):** $N \leq 100$.
- **Subtarefa 3 (13 pontos):** $N \leq 500$.
- **Subtarefa 4 (17 pontos):** $N \leq 5000$.
- **Subtarefa 5 (24 pontos):** Para todo competidor i com $a_i > 0$, todos os seus amigos têm índice maior do que i.
- **Subtarefa 6 (34 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

```
5
1 4
1 5
2 5 2
0
1 2
```

Exemplo de saída 1

```
3
```

Explicação do exemplo 1: Note que a foto $\{3, 5\}$ não é válida pois:

- como o competidor 3 está incluído, o competidor 5 ou 2 devem estar. Como 5 está, então 3 está satisfeito.
- o competidor 4 não possui amigos, então está automaticamente satisfeito.
- como o competidor 5 está incluído, o competidor 2 precisa estar. Como 2 não está na foto, o competidor 5 não está satisfeito. Logo essa foto não é válida.

As únicas fotos válidas são: $\{1, 5\}$, $\{2, 5\}$, $\{4, 4\}$.

Exemplo de entrada 2

```
8
2 3 5
1 4
0
1 6
0
1 8
0
0
```

Exemplo de saída 2

```
11
```

Explicação do exemplo 2: Este exemplo satisfaz a Subtarefa 5.

Exemplo de entrada 3

```
7
1 3
2 1 4
1 5
0
2 3 7
1 4
1 5
```

Exemplo de saída 3

```
11
```