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 (27 de setembro de 2025).

Olimpíada Brasileira de Informática

OBI2025

Caderno de Tarefas

Modalidade Programação • Nível 1 • Fase 3

27 de setembro de 2025

A PROVA TEM DURAÇÃO DE 4 HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

Coordenação:

v1.0

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.

Redes de Descanso

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

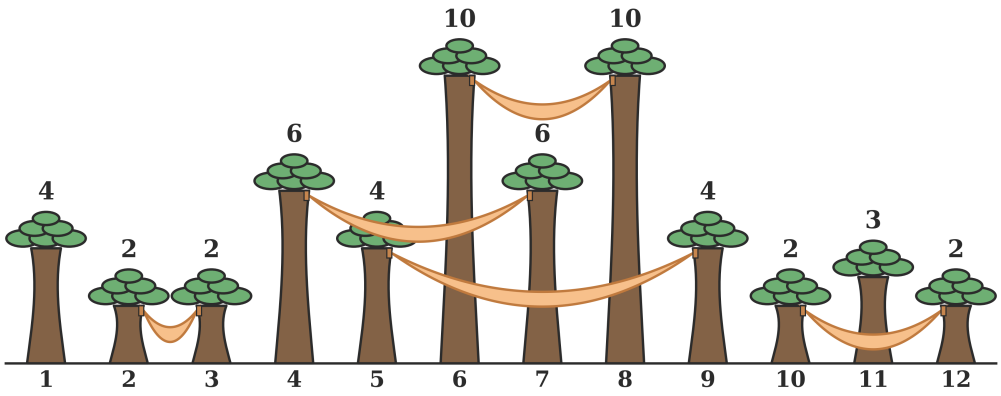
O departamento de Antropologia da Unicamp tem estudado os guaranis, o maior povo indígena do Brasil. Como parte de uma viagem do departamento, você está visitando uma tribo que possui o hábito de dormir em redes de descanso penduradas em árvores.

Na região da floresta perto da aldeia, existem N árvores alinhadas, numeradas de 1 a N da esquerda para a direita, com alturas variadas. Por motivos de segurança, o cacique da aldeia determinou algumas regras para pendurar redes:

- Cada rede deve estar pendurada em duas árvores distintas que possuam a mesma altura.
- Em cada árvore, no máximo uma rede pode ser pendurada.
- Duas redes penduradas em árvores de mesma altura não devem se intersectar – ou seja, se duas redes estão penduradas na mesma altura, então uma delas precisa estar completamente à esquerda da outra.

Observe que sempre é possível pendurar uma rede em duas árvores de mesma altura que não tenham nenhuma rede pendurada ainda, mesmo que entre as duas existam árvores mais altas (a rede pode passar pela frente das árvores altas).

A figura abaixo ilustra um exemplo com 12 árvores, com a altura de cada árvore indicada acima dela, e um modo válido de pendurar 5 redes:



A aldeia está se preparando para uma grande festa. Ao fim desta festa, todos os convidados dormirão em redes. O cacique precisa saber quantas pessoas ele pode convidar para festa, sabendo que somente uma pessoa pode dormir em cada rede. Deste modo, sua tarefa é ajudar o cacique a descobrir qual o número **máximo** de redes que ele consegue pendurar nas árvores, respeitando todas as regras que ele estabeleceu.

Entrada

A primeira linha da entrada contém um único inteiro N, o número de árvores alinhadas na floresta.

A segunda linha da entrada contém N inteiros $a_1, a_2, \ldots, a_N$, separados por espaços em branco, indicando as alturas das N árvores da esquerda para a direita.

Saída

Seu programa deverá imprimir uma única linha contendo um único inteiro, o número máximo de redes que podem ser penduradas nas árvores.

Restrições

É garantido que todo caso de teste satisfaz as restrições abaixo.

- $1 \leq N \leq 100\,000$.
- $1 \leq a_i \leq 100\,000$ 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 (20 pontos):** $a_i = 1$ para todo $1 \leq i \leq N$.
- **Subtarefa 3 (20 pontos):** $a_i \leq 3$ para todo $1 \leq i \leq N$.
- **Subtarefa 4 (20 pontos):**
  - $N \leq 100$.
  - $a_i \leq 100$ para todo $1 \leq i \leq N$.
  - Não existem três árvores com a mesma altura.
- **Subtarefa 5 (20 pontos):**
  - $N \leq 100$.
  - $a_i \leq 100$ para todo $1 \leq i \leq N$.
- **Subtarefa 6 (20 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
12 4 2 2 6 4 10 6 10 4 2 3 2	5

Explicação do exemplo 1: Este é o exemplo mostrado no enunciado. A figura do enunciado ilustra um modo de pendurar 5 redes. É possível verificar que é impossível pendurar 6 redes.

Exemplo de entrada 2	**Exemplo de saída 2**
7 1 1 1 1 1 1 1	3

Exemplo de entrada 3	**Exemplo de saída 3**
10 7 2 1 3 2 3 5 5 7 6	4

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

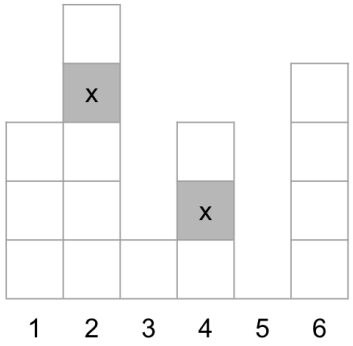
Diagonal

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

Joana é dona de uma construtora muito famosa. Em seu novo empreendimento, ela está construindo um condomínio de N prédios, onde atualmente o i-ésimo prédio possui A_i apartamentos, um por andar. Como o condomínio ainda está em construção, é possível que a quantidade de apartamentos em um prédio ainda seja 0. Mas nos prédios onde há pelo menos um apartamento, eles são numerados a partir da base, ou seja, o apartamento no primeiro andar recebe o número 1, o apartamento no segundo andar recebe o número 2, e assim por diante, até o apartamento no último andar que recebe o número A_i.

Joana é uma entusiasta da construção civil, por isso está sempre usando as técnicas mais avançadas neste setor. Dessa vez, ela construiu o condomínio utilizando um material especial, que prometia ser mais resistente. Porém, foram descobertos problemas em relação à propagação de som neste material. Um funcionário identificou que, quando alguém em um determinado apartamento emite um som, este som é ouvido em todos os apartamentos que estão em algum prédio à esquerda, em um andar mais alto e que estejam na mesma diagonal que o apartamento inicial.

Mais precisamente, se alguém no apartamento i do prédio j emite um som, ele é ouvido no apartamento a do prédio b se e somente se $b < j$ e $a - i = j - b$. Por exemplo, na figura a seguir, se um som é emitido no apartamento 2 do prédio 4, ele será ouvido no apartamento 4 do prédio 2, mas não será ouvido no apartamento 1 do prédio 3 (pois o apartamento não está acima do andar inicial) nem no apartamento 4 do prédio 6 (pois o prédio não está à esquerda do prédio inicial). A figura abaixo ilustra esse cenário.



Joana ficou muito decepcionada ao saber desse problema. Porém, quantos apartamentos ouvem um som depende de onde ele é inicialmente emitido. Assim, para avaliar os possíveis danos, Joana deseja que você calcule o **maior número de apartamentos** (incluindo o apartamento inicial) que podem ouvir um som emitido em algum apartamento do condomínio.

Entrada

A primeira linha da entrada contém um único inteiro N, representando a quantidade de prédios do condomínio.

A segunda linha da entrada contém N inteiros, A_1, A_2, ..., A_N, que representam a quantidade de apartamentos de cada prédio, da esquerda para a direita.

Saída

Seu programa deverá produzir uma única linha contendo somente um inteiro, o maior número de apartamentos que podem ouvir um som emitido em algum apartamento do condomínio, se o apartamento inicial for escolhido de maneira a maximizar esta quantidade.

Restrições

- $1 \leq N \leq 300\,000$.
- $0 \leq A_i \leq 1\,000\,000\,000$ para todo $1 \leq i \leq N$.

Informações sobre a pontuação

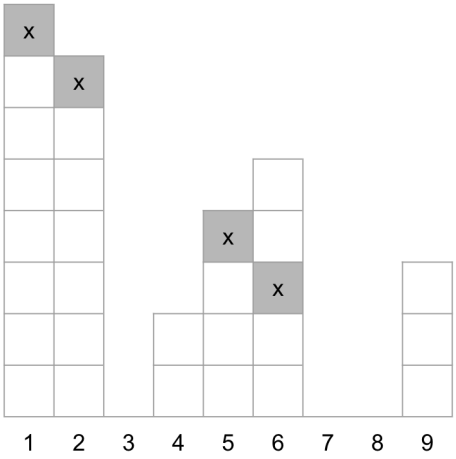
A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **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 (29 pontos):** $N \leq 1\,000$ e $A_i \leq 1\,000$ para todo $1 \leq i \leq N$.
- **Subtarefa 3 (41 pontos):** $N \leq 1\,000$.
- **Subtarefa 4 (30 pontos):** Sem restrições adicionais.

Exemplos

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

Explicação do exemplo 1: O número máximo de apartamentos é 4, que é atingido se escolhermos como apartamento inicial o apartamento 3 do prédio 6, conforme ilustrado na figura abaixo.



É possível verificar que não existe nenhuma escolha de apartamento inicial que faria com que 5 ou mais apartamentos ouvissem o som.

Exemplo de entrada 2	**Exemplo de saída 2**
10 1 16 27 5 4 27 381 29 57 1000	7

Exemplo de entrada 3	**Exemplo de saída 3**
1 0	0

Empregos de Júlio

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

Júlio trabalha muito para sustentar sua família. No entanto, ele percebeu que um único emprego não seria suficiente para pagar as contas e, por isso, decidiu arranjar dois! Assim, ele foi contratado para ser entregador de jornais e segurança de shopping. A princípio, ele pensou que conseguiria trabalhar em ambos ao mesmo tempo, mas logo descobriu que isso não seria tão fácil.

Como os dois serviços demandam muito tempo e esforço, Júlio desistiu de pegar turnos em ambos ao mesmo tempo. Dessa forma, ele pretende escolher em qual deles ele vai trabalhar em cada um dos próximos N dias, sendo que, para cada um desses, Júlio sabe o quanto ele ganharia se trabalhasse como entregador ou segurança. Mais especificamente, no i-ésimo dia, ele ganhará a_i reais para entregar jornais e b_i reais para vigiar o shopping.

O chefe de Júlio no emprego de segurança ficou sabendo da decisão de seu empregado e, a fim de segurá-lo no trabalho, ofereceu um incentivo a ele. Assim, ele prometeu a Júlio que, após trabalhar K dias na segurança do shopping, ele ganhará em dobro em todo e qualquer turno que ele escolher, ou seja, ganhará $2b_i$ reais se vigiar o shopping no i-ésimo dia. Note que os primeiros K dias trabalhados nesse emprego não precisam ser consecutivos, e ele ganhará o valor normal do dia.

Júlio está muito interessado na proposta do chefe, mas, como a quantidades N e K de dias são muito grandes, ele não sabe em qual emprego ele deve trabalhar em cada dia. Sua tarefa é: dados os valores N e K, bem como as listas $a_1 \dots a_N$ e $b_1 \dots b_N$, ajude Júlio a saber qual o **máximo** de dinheiro que ele pode conseguir, considerando a proposta feita por um de seus chefes.

Entrada

A primeira linha da entrada contém dois inteiros, N e K, que indicam o número de dias que Júlio irá trabalhar e a quantidade de vezes que ele deve trabalhar de segurança para ganhar em dobro.

A segunda linha da entrada contém N inteiros, $a_1, a_2, \dots, a_N$, que representam o quanto ele receberia em cada dia caso trabalhasse entregando jornais.

A terceira linha da entrada contém N inteiros, $b_1, b_2, \dots, b_N$, que representam o quanto ele receberia em cada dia caso trabalhasse vigiando o shopping.

Saída

A saída deve conter uma única linha contendo um inteiro, o máximo que Júlio consegue receber nos próximos N dias.

Restrições

- $1 \leq N \leq 100\,000$.
- $0 \leq K \leq N$.
- $1 \leq a_i, b_i \leq 1\,000\,000\,000$, para todo $1 \leq i \leq N$.

Para competidores que utilizam C++ ou Java: Observe que alguns valores na saída podem ser muito grandes para caberem 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 ou JavaScript 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 (13 pontos):** $K = 0$ e $N \leq 1\,000$.
- **Subtarefa 3 (21 pontos):** $K = 1$ e $N \leq 1\,000$.
- **Subtarefa 4 (31 pontos):** $N \leq 1\,000$.
- **Subtarefa 5 (14 pontos):** $b_i = b_j$, para todo $1 \leq i \leq j \leq N$.
- **Subtarefa 6 (21 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
5 0 30 40 30 50 70 10 30 20 20 40	260

Explicação do exemplo 1: Este exemplo satisfaz as restrições da subtarefa 2. Note que, como $K = 0$, Júlio ganha em dobro no segundo emprego desde o início. Assim, ele pode trabalhar como entregador nos dias 1 e 4, e como segurança nos dias 2, 3 e 5. Dessa forma, ele ganharia $30 + 2 \cdot 30 + 2 \cdot 20 + 50 + 2 \cdot 40 = 260$ reais.

Exemplo de entrada 2	**Exemplo de saída 2**
5 1 30 40 30 50 70 10 30 20 20 40	240

Explicação do exemplo 2: Este exemplo satisfaz as restrições da subtarefa 3. Note que, como $K = 1$, Júlio ganha em dobro no segundo emprego após o primeiro dia de trabalho como segurança. Assim, ele pode trabalhar como entregador no dia 4 e como segurança nos dias 1, 2, 3 e 5. Dessa forma, ele ganharia $10 + 2 \cdot 30 + 2 \cdot 20 + 50 + 2 \cdot 40 = 240$ reais.

Exemplo de entrada 3	**Exemplo de saída 3**
5 2 30 40 25 15 20 10 10 10 10 10	130

Explicação do exemplo 3: Este exemplo satisfaz as restrições da subtarefa 5.

Recarga

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

Jonas quer fazer uma viagem de Pelém para Pique-Pique, duas grandes cidades do Prasil, seu país de origem, mas seu carro quebrou! Sendo assim, ele decide comprar um carro elétrico.

Na hora de escolher qual carro comprar, ele percebe que cada um tem uma bateria de **capacidade** diferente. Essa decisão pode ter um grande impacto, uma vez que nem todas as cidades tem uma **estação de recarga** para a bateria, e portanto uma capacidade baixa pode tornar a viagem inviável.

Sabemos que existem N cidades no Prasil, sendo Pelém a cidade 1 e Pique-Pique a cidade N. Ademais, existem M rodovias de mão dupla, onde a i-ésima rodovia liga a cidade a_i com a cidade b_i. Ao percorrer a i-ésima rodovia, Jonas sabe que um carro elétrico consome c_i unidades de energia da bateria. Note que a bateria não pode ter energia negativa em nenhum momento, e portanto, Jonas só pode percorrer a rodovia i se a bateria tiver ao menos c_i unidades de energia.

Também existem K cidades com estações de recarga para carros elétricos. Ao chegar em uma dessas cidades, Jonas pode recuperar a energia da bateria por completo, ou seja, a energia da bateria volta a ser a capacidade máxima. É garantido que Pelém (a cidade 1) tem uma estação de recarga, e Pique-Pique (a cidade N) não tem.

Dadas essas informações, ajude Jonas respondendo qual é a menor capacidade para a bateria de seu carro elétrico de forma que seja possível fazer uma viagem de Pelém até Pique-Pique. Perceba que não é necessário minimizar a distância total percorrida, nem a quantidade de cidades intermediárias, Jonas está interessado apenas na menor capacidade para a bateria com a qual é possível realizar a viagem (isso porque ele notou que, quanto maior a capacidade da bateria, o carro é mais caro).

Entrada

A primeira linha da entrada contém dois inteiros, N e M, que indica o número total de cidades e o número total de rodovias, respectivamente.

As próximas M linhas possuem três inteiros cada e descrevem as rodovias. A i-ésima destas linhas contém três inteiros a_i, b_i e c_i indicando que as cidades a_i e b_i são ligadas pela i-ésima rodovia, e a quantidade de energia c_i consumida para percorrê-la.

A próxima linha tem um inteiro K, a quantidade de cidades com estações de recarga.

A última linha contém K inteiros $x_1, x_2, \ldots, x_K$, indicando as cidades que contém uma estação de recarga.

Saída

A saída deve conter uma única linha contendo um inteiro, a capacidade mínima que a bateria do carro de Jonas precisa ter para ele conseguir fazer sua viagem de Pelém para Pique-Pique.

Restrições

- $2 \leq N \leq 50\,000$.
- $1 \leq M \leq 100\,000$.
- $1 \leq a_i, b_i \leq N$ e $a_i \neq b_i$, para todo $1 \leq i \leq M$.
- É garantido que não há duas rodovias diferentes que liguem o mesmo par de cidades.

- $1 \leq c_i \leq 10^9$, para todo $1 \leq i \leq M$.
- $1 \leq K < N$
- $1 \leq x_j < N$, para todo $1 \leq j \leq K$.
- A cidade 1 contém uma estação de recarga e a cidade N não contém.
- É garantido que com uma capacidade de bateria suficientemente alta é possível fazer uma viagem entre qualquer par de cidades.

Para competidores que utilizam C++ ou Java: Observe que alguns valores na saída podem ser muito grandes para caberem 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 ou JavaScript podem ignorar este aviso.)*

Informações sobre a pontuação

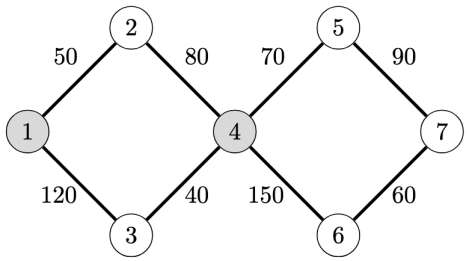
A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **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 (22 pontos):** $N, M \leq 1000$ e $K = 1$, ou seja, apenas a cidade 1 possui estação de recarga.
- **Subtarefa 3 (23 pontos):** $K = N - 1$, ou seja, todas as cidades exceto N possuem estação de recarga.
- **Subtarefa 4 (29 pontos):** $N, M \leq 1000$ e $K \leq 100$.
- **Subtarefa 5 (26 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
7 8 1 2 50 1 3 120 2 4 80 3 4 40 4 5 70 4 6 150 5 7 90 6 7 60 2 1 4	160

Explicação do exemplo 1: Neste exemplo temos 7 cidades e 8 rodovias conforme a figura a seguir:



Perceba que, com uma capacidade de 160 unidades de energia, Jonas consegue realizar a viagem da cidade 1 para a cidade 7 da seguinte forma:

- Jonas começa sua viagem na cidade 1 com 160 unidades de energia.
- Jonas viaja da cidade 1 para a cidade 3, o que consome 120 unidades de energia, restando em sua bateria $160 - 120 = 40$.
- Jonas viaja da cidade 3 para a cidade 4, o que consome 40 unidades de energia, restando em sua bateria $40 - 40 = 0$.
- Na cidade 4 há uma estação de recarga, portanto, Jonas recarrega sua bateria, que volta a ter 160 unidades de energia.
- Jonas viaja da cidade 4 para a cidade 5, o que consome 70 unidades de energia, restando em sua bateria $160 - 70 = 90$.
- Jonas viaja da cidade 5 para a cidade 7, o que consome 90 unidades de energia, restando em sua bateria $90 - 90 = 0$.
- Jonas finalmente chega na cidade 7, conforme desejado. Perceba que, neste exemplo de viagem ele chega com 0 unidade de energia em sua bateria, mas isso não importa para Jonas.

É possível verificar que com uma capacidade menor que 160 é impossível viajar da cidade 1 para a cidade 7.

Exemplo de entrada 2	**Exemplo de saída 2**
5 6 1 2 10 1 3 60 2 4 30 3 5 70 4 5 90 2 5 100 3 1 3 4	70