

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 Sênior • Fase 3

27 de setembro de 2025

A PROVA TEM DURAÇÃO DE 5 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.

Fotos de Relíquias

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

Jonas, um famoso arqueólogo, quer viralizar nas redes sociais publicando fotos de relíquias encontradas em suas jornadas. Para tirar fotos agradáveis, Jonas escolheu uma paisagem como plano de fundo para suas fotos.

A paisagem escolhida por Jonas é dividida em N seções, de forma que algumas são seções de destaque e outras não. Ele escreveu uma lista A de N inteiros que descrevem as N seções da paisagem, da esquerda para a direita, de forma que, se $A_i = 1$ então a i -ésima seção é de destaque, e caso contrário $A_i = 0$.

Jonas escolheu uma relíquia e vai tirar uma foto dela usando uma região contínua da paisagem como plano de fundo. Para que a relíquia seja o foco principal, a foto deve conter uma única seção de destaque, onde a relíquia será posicionada. Portanto, ele deve escolher um par de inteiros (l, r) que satisfaça as seguintes condições:

- $1 \leq l \leq r \leq N$.
- As seções $l, l+1, \dots, r$ farão parte da foto.
- Dentre as seções presentes na foto, exatamente uma é de destaque, ou seja, existe um único índice k tal que $l \leq k \leq r$ e $A_k = 1$.

Jonas quer saber de quantas maneiras diferentes ele pode tirar a foto. Ajude ele a determinar a quantidade de pares de inteiros (l, r) que satisfazem todas as condições desejadas.

Entrada

A primeira linha da entrada contém um único inteiro N , indicando o número de seções na paisagem.

A segunda linha da entrada possui N inteiros A_1, A_2, \dots, A_N , onde $A_i = 1$ se a i -ésima seção é uma seção de destaque, ou $A_i = 0$ caso contrário.

Saída

Seu programa deve imprimir uma única linha contendo um único inteiro, o número de modos que Jonas pode tirar a foto.

Restrições

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

- $1 \leq N \leq 100\,000$.
- $A_i = 0$ ou $A_i = 1$ 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 (20 pontos):** $N \leq 100$.
- **Subtarefa 3 (22 pontos):** $N \leq 3000$.
- **Subtarefa 4 (26 pontos):** Existe exatamente uma seção de destaque na paisagem, ou seja, há um único índice i na lista tal que $A_i = 1$.
- **Subtarefa 6 (32 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	Exemplo de saída 1
5 1 0 0 1 0	9

Explicação do exemplo 1: Os pares (l, r) que seguem as condições para a foto são: $(1, 1)$, $(1, 2)$, $(1, 3)$, $(2, 4)$, $(2, 5)$, $(3, 4)$, $(3, 5)$, $(4, 4)$ e $(4, 5)$.

Exemplo de entrada 2	Exemplo de saída 2
12 0 0 0 1 0 0 0 0 0 0 0 0	36

Explicação do exemplo 2: Este exemplo satisfaz as restrições da subtarefa 4.

Escadaria

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

O arqueólogo Jonas encontrou uma antiga escadaria nas ruínas do arquipélago da Nlogônia.

Jonas sabe que originalmente cada degrau tinha uma altura inteira e também que a escadaria seguia uma regra especial: a diferença entre as alturas de dois degraus consecutivos nunca poderia ser maior que 1. Por exemplo, se um degrau possuía altura 5, o próximo poderia ter altura 4, 5 ou 6, mas não poderia ter altura 2, 3 ou 7, pois a diferença seria maior do que 1.

O problema é que parte da escadaria foi destruída. Em algumas posições a altura original ainda é conhecida, mas em outras ela é completamente desconhecida. No total existem N degraus, e Jonas escreveu um número A_i associado ao i -ésimo degrau:

- Se $A_i = -1$, então a altura do degrau i é desconhecido.
- Se $A_i > 0$, então a altura do degrau i é A_i .

Sua tarefa é ajudar Jonas a terminar sua pesquisa sobre as ruínas de Nlogônia. Para isso, ele precisa saber, para cada um dos N degraus, a **maior** altura possível que o degrau pode ter em alguma reconstrução válida.

Entrada

A primeira linha da entrada possui o inteiro N , indicando a quantidade de degraus da escadaria.

A segunda linha possui N inteiros A_1, A_2, \dots, A_N , onde o i -ésimo inteiro indica o valor anotado por Jonas para o degrau i . Se $A_i = -1$, a altura do degrau i é desconhecida; caso contrário, a altura do degrau i é A_i .

Saída

Seu programa deve imprimir uma única linha contendo N inteiros (separados por espaços em branco), onde o i -ésimo deles representa a maior altura possível do degrau i em alguma reconstrução válida da escadaria.

Restrições

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

- $2 \leq N \leq 200\,000$.
- $-1 \leq A_i \leq 1\,000\,000$.
- $A_i \neq 0$.
- É garantido que existe ao menos uma maneira válida de completar a escadaria.
- Existe pelo menos um degrau com altura conhecida inicialmente (ou seja, algum i tal que $A_i \neq -1$).

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 (7 pontos):**
 - $N = 3$.
 - $A_1 \neq -1$, $A_2 = -1$ e $A_3 \neq -1$. Ou seja, somente A_2 é desconhecido.
- **Subtarefa 3 (22 pontos):** $N \leq 1000$ e só existe um i tal que $A_i \neq -1$. Ou seja, só um degrau tem sua altura conhecida.
- **Subtarefa 4 (15 pontos):** $N \leq 1000$.
- **Subtarefa 5 (25 pontos):** Apenas A_1 e A_N são diferentes de -1 . Ou seja, os únicos degraus com altura conhecida são 1 e N .
- **Subtarefa 6 (31 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	Exemplo de saída 1
5 5 -1 -1 -1 6	5 6 7 7 6

Explicação do exemplo 1: 5, 6, 7, 7, 6 são os maiores valores possíveis para as alturas de cada um dos degraus de 1 até 5, respectivamente.

Por exemplo, o maior valor possível para o degrau 1 é 5, pois a altura deste degrau já é conhecida; e o maior valor possível para o degrau 3 é 7, pois, de todas sequências válidas de alturas, esse é o maior valor que o degrau 3 pode ter.

Note que a sequência 5, 4, 5, 6, 6 é válida, mas não corresponde aos valores máximos dos índices de 2 a 4.

Exemplo de entrada 2	Exemplo de saída 2
12 -1 -1 4 -1 -1 -1 2 -1 -1 2 -1 -1	6 5 4 5 4 3 2 3 3 2 3 4

Exemplo de entrada 3	Exemplo de saída 3
10 -1 -1 -1 10 -1 -1 -1 -1 -1 -1	13 12 11 10 11 12 13 14 15 16

Hidrovias e Rodovias

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

O arqueólogo Jonas continuou sua pesquisa no arquipélago da Nlogônia e subiu a escadaria, evitando os degraus destruídos. No topo das ruínas, ele encontrou uma grande sala, a qual continha diversas relíquias feitas de um metal desconhecido, bem como um mapa com um X marcado. O arqueólogo ficou muito animado com a descoberta, mas achou o sistema de transportes da região muito complicado.

O arquipélago da Nlogônia possui N ilhas, as quais são interligadas por M conexões, cada uma conectando duas ilhas distintas. Cada conexão pode ser percorrida nas duas direções e pode ser uma rodovia ou uma hidrovia. Jonas sabe que um caminho entre duas ilhas A e B é uma sequência $A = i_1, i_2, \dots, i_k = B$ de ilhas distintas em que existe alguma conexão, de qualquer tipo, entre todo par de ilhas consecutivas. Observe que é possível que o sistema de transportes **não** seja conectado, ou seja, é possível que não exista qualquer caminho entre algum par de ilhas.

Além disso, o arqueólogo estudou muito a História da região e sabe que, nos últimos K anos:

- Nenhuma hidrovia foi construída.
- No máximo uma rodovia foi construída em cada ano. Ou seja, **no máximo K rodovias** foram construídas ao todo.

Sabendo que o arquipélago era um local muito próspero no passado, Jonas se perguntou como as pessoas não se perdiam nos diversos caminhos, e imaginou que o sistema de transportes deveria ser simples há K anos atrás. Ele define que uma rede de transportes é simples se **existe no máximo um caminho entre qualquer par de ilhas no mapa**.

Portanto, dada a atual rede de transportes do arquipélago, ajude Jonas a verificar se é possível que ela fosse simples há K anos atrás. Isto é, determine se existe um sistema em que há no máximo um caminho entre qualquer par de ilhas e que condiz com a História da região descrita pelo arqueólogo.

Entrada

A primeira linha da entrada possui três inteiros N , M e K indicando, respectivamente, o número de ilhas no arquipélago da Nlogônia, o número de conexões entre pares de ilhas, e a quantidade de anos considerada por Jonas.

As próximas M linhas possuem três inteiros cada e descrevem as conexões. A i -ésima destas linhas contém três inteiros a_i , b_i e t_i indicando as ilhas a_i e b_i que são ligadas pela i -ésima conexão. t_i representa o tipo da conexão: $t_i = 1$ indica que essa conexão é uma hidrovia, e $t_i = 2$ indica que essa conexão é uma rodovia.

É garantido que não existem duas conexões que ligam o mesmo par de ilhas. Além disso, nenhuma conexão liga uma ilha a ela mesma.

Saída

Seu programa deverá imprimir uma única linha contendo um único caractere. Caso exista um sistema de transportes simples que seja condizente com a História do arquipélago, imprima o caractere **S** (a letra S maiúscula). Caso não exista, imprima o caractere **N** (a letra N maiúscula).

Restrições

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

- $1 \leq N \leq 100\,000$.
- $1 \leq M \leq 100\,000$.
- $0 \leq K \leq 100\,000$.
- $1 \leq a_i, b_i \leq N$ e $a_i \neq b_i$ para todo $1 \leq i \leq M$.
- $1 \leq t_i \leq 2$ para todo $1 \leq i \leq M$.
- Não existem duas conexões que ligam o mesmo par de ilhas.

Informações sobre a pontuação

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

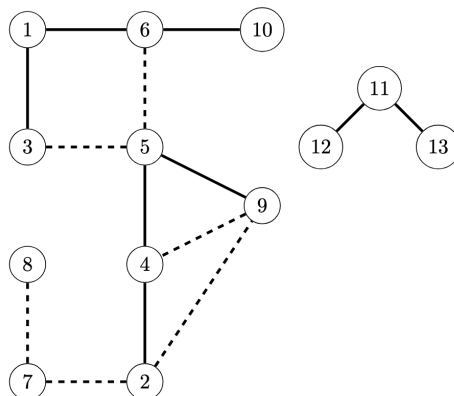
- **Subtarefa 1 (0 pontos):** Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (11 pontos):** Todas as conexões são hidrovias.
- **Subtarefa 3 (11 pontos):** Todas as conexões são rodovias. Além disso, para quaisquer duas ilhas u e v , existe pelo menos um caminho de u para v .
- **Subtarefa 4 (14 pontos):** Todas as conexões são rodovias.
- **Subtarefa 5 (18 pontos):** $K = 1$.
- **Subtarefa 6 (19 pontos):** $N \leq 1\,000$ e $M \leq 1\,000$.
- **Subtarefa 7 (27 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	Exemplo de saída 1
13 14 3 2 9 2 4 2 1 7 8 2 4 5 1 1 3 1 5 3 2 9 4 2 7 2 2 6 1 1 5 9 1 5 6 2 10 6 1 11 12 1 11 13 1	S

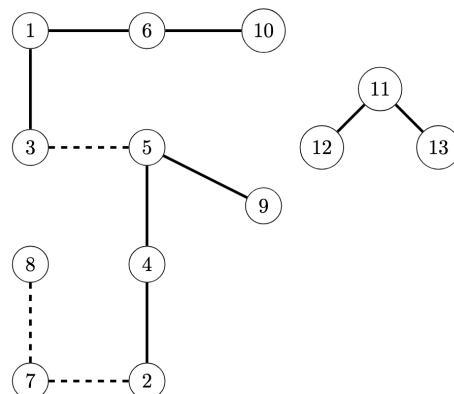
Explicação do exemplo 1:

Atualmente, o sistema de transportes da Nlogônia pode ser visto da seguinte forma:



Note que as linhas contínuas indicam as hidrovias e as linhas tracejadas indicam rodovias. No entanto, nos últimos 3 anos, é possível que as rodovias (2,9), (9,4) e (5,6) tenham sido construídas.

Portanto, a seguinte rede de transportes é condizente com os conhecimentos do arqueólogo:



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

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

Note que o atual arquipélago da Nlogônia não possui rodovias, logo, o sistema local de transportes manteve-se inalterado nos últimos 2 anos.

No entanto, existem dois caminhos entre as ilhas 5 e 6: $5 - 6$ e $5 - 2 - 6$.

Logo, é impossível que a rede de transportes tenha sido simples há 2 anos.

Exemplo de entrada 3	Exemplo de saída 3
6 9 3 1 2 2 5 3 2 5 2 2 2 3 2 1 6 2 1 4 2 6 2 2 5 1 2 4 5 2	N

Explicação do exemplo 3: Este exemplo satisfaz as restrições das subtarefas 3 e 4.

Harmonia Nasal

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

O arqueólogo Jonas deseja chegar no local marcado com um X em seu mapa, mas, para isso, precisa saber ler as legendas escritas nele. Jonas descobriu que o texto está escrito no idioma do povo Guarani¹, e por isso precisa aprender a língua.

Em particular, ele deve se tornar proficiente no fenômeno da **harmonia nasal**, em que um som nasal (por exemplo ‘ã’) pode afetar como outras partes da palavra são pronunciadas.

Para isso, ele anotou N sílabas presentes na língua Guarani, sendo que a i -ésima sílaba tem **nasalidade** a_i e **resistência** b_i . Ele quer formar uma única palavra de maior nasalidade possível usando o seguinte processo:

1. Primeiro, Jonas pode zerar a resistência de no máximo K sílabas, ou seja, ele pode escolher até K sílabas e transformar b_i em 0.
2. Depois, Jonas escolhe uma única sílaba i para começar a palavra. Sendo assim, a palavra começará com nasalidade $A = a_i$ e resistência $B = b_i$. Vale ressaltar que durante todo o processo existirá uma única palavra, sua nasalidade será denominada A e sua resistência B . Observe que a sílaba escolhida i nunca mais pode ser usada.
3. Por fim, ele pode realizar a seguinte operação inúmeras vezes (inclusive nenhuma vez), desde que cumpra todas as condições:
 - Jonas pode escolher uma sílaba não escolhida ainda i tal que $a_i + A \geq B$ e anexá-la ao fim da palavra.
 - A palavra passa a ter nasalidade $a_i + A - B$ e resistência b_i .
 - Vale ressaltar que a sílaba escolhida i nunca mais pode ser usada.

Dada a lista de sílabas e o fator K , ajude Jonas dizendo qual é a maior nasalidade de uma palavra que ele consegue obter usando o processo descrito acima.

Entrada

A primeira linha da entrada contém dois inteiros, N e K , que indicam o número de sílabas que Jonas está trabalhando e o número de vezes que ele pode fazer a primeira operação.

A segunda linha da entrada contém N inteiros, a_1, a_2, \dots, a_N , que representam o fator de nasalidade de cada uma das sílabas.

A terceira linha da entrada contém N inteiros, b_1, b_2, \dots, b_N , que representam o fator de resistência de cada uma das sílabas.

Saída

A saída deve conter uma única linha contendo um inteiro, a maior nasalidade que Jonas consegue obter.

Restrições

- $1 \leq N \leq 100\,000$.
- $0 \leq K \leq N$.

¹Em particular, o povo Guarani-Mbyá, que vive no sul e sudeste do Brasil.

- $1 \leq a_i, b_i \leq 1\,000\,000\,000$, 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 sub tarefa 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):** $K = N - 1$.
- **Subtarefa 3 (11 pontos):** $b_i = b_{i+1}$ para todo $1 \leq i < N$. Em outras palavras, todas as resistências são iguais.
- **Subtarefa 4 (21 pontos):** $a_i \geq b_i$ para todo $1 \leq i \leq N$.
- **Subtarefa 5 (25 pontos):** $K = 0$.
- **Subtarefa 6 (12 pontos):** $K \leq 10$.
- **Subtarefa 7 (19 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	Exemplo de saída 1
5 1 3 5 2 1 7 9 2 3 1 5	13

Explicação do exemplo 1: Nesse caso, Jonas pode obter uma palavra de nasalidade 13 da seguinte forma:

- Jonas primeiro zera b_1 . Agora a sílaba 1 tem resistência 0.
- Jonas começa sua palavra com a sílaba 1. Agora ela tem nasalidade $A = 3$ e resistência $B = 0$.
- Jonas adiciona a sílaba 2, com $a_2 = 5$ e $b_2 = 2$. Perceba que ele pode realizar essa operação pois a sílaba 2 ainda não foi escolhida e $a_2 + A \geq B$. Agora a palavra passa a ter nasalidade $a_2 + A - B = 5 + 3 - 0 = 8$ e resistência $b_2 = 2$.
- Jonas adiciona a sílaba 5, com $a_5 = 7$ e $b_5 = 5$. Perceba que ele pode realizar essa operação pois a sílaba 5 ainda não foi escolhida e $a_5 + A \geq B$ (nesse momento a palavra tem nasalidade $A = 8$ e resistência $B = 2$). Então a palavra passa a ter nasalidade $a_5 + A - B = 7 + 8 - 2 = 13$ e resistência $b_5 = 5$.

Note que Jonas poderia adicionar mais sílabas, mas para maximizar a nasalidade ele escolheu não fazer isso, nesse caso.

Exemplo de entrada 2	Exemplo de saída 2
5 2 1 2 3 4 5 2 2 2 2 2	12

Explicação do exemplo 2: Esse exemplo satisfaz a subtarefa 3.

Exemplo de entrada 3	Exemplo de saída 3
1 0 10 10	10

Explicação do exemplo 3: Nesse caso é apenas possível formar uma palavra de nasalidade 10 e resistência 10. Esse exemplo satisfaz a subtarefa 2.

Exemplo de entrada 4	Exemplo de saída 4
1 1 10 10	10

Explicação do exemplo 4: Note que apesar de ser possível mudar a resistência da sílaba, a resposta não muda em relação ao exemplo 3.

Energia

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

Quando o arqueólogo Jonas terminou de forjar a barra de ORicalco, uma antiga maldição do vulcão foi imediatamente ativada. O chão tremeu, e diante dele surgiu novamente uma imensa escadaria, formada por N degraus, onde cada degrau i ($1 \leq i \leq N$) tinha uma altura representada por A_i .

Nesse momento, Jonas lembrou de um conceito importante das suas aulas de física vulcânica: a energia gasta para ir de um degrau i até um degrau j depende apenas das posições e das alturas dos dois degraus, assim como uma diferença de potencial entre dois pontos. A fórmula usada para calcular a energia gasta para ir de um degrau i para o j é:

$$E(i, j) = |i - j| + |A_i - A_j|$$

Onde $|x|$ representa o valor absoluto de x .

Sendo um mestre dos vulcões, Jonas sabe que um par (i, j) , com $1 \leq i < j \leq N$, é chamado de vulcônico se $E(i, j) = K$. Ele também decidiu ler a parte da lenda que explicava que, para se livrar da maldição, Jonas precisaria falar a todo momento quantos pares (i, j) são vulcônicos.

Para a surpresa do pesquisador, a partir do minuto 1, os degraus vão pouco a pouco desmoronando. No minuto t , o degrau D_t desmorona, fazendo com que seja impossível ir de qualquer degrau a esquerda de D_t para algum a direita de D_t .

No total, Q degraus desmoronaram nos minutos $1, 2, \dots, Q$. Então Jonas precisa falar, para cada minuto t de 0 (o minuto inicial, quando nenhum degrau havia desmoronado) até Q , quantos pares (i, j) existem tal que $E(i, j) = K$ e nenhum degrau desmorou entre (i, j) ainda. Ajude Jonas a escapar dessa terrível maldição!

Entrada

A primeira linha da entrada contém três inteiros N , Q e K , indicando respectivamente o número de degraus, o número de desmoronamentos e o valor alvo da energia.

A segunda linha contém N inteiros A_1, A_2, \dots, A_N , descrevendo a altura de cada degrau.

Em seguida, seguem Q linhas, onde a t -ésima delas contém um inteiro D_t , representando o degrau que desmoronou no minuto t .

Saída

O programa deve imprimir $Q + 1$ linhas. A primeira linha deve conter o número de pares vulcônicos existentes antes de qualquer desmoronamento. Cada uma das Q linhas seguintes deve conter o número de pares vulcônicos após cada um dos desmoronamentos.

Restrições

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

- $1 \leq N \leq 200\,000$.
- $0 \leq Q < N$.
- $1 \leq A_i, K \leq 1\,000\,000\,000$.

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

- **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 (10 pontos):** $Q = 0$, $N \leq 3\,000$.
- **Subtarefa 3 (19 pontos):** $N \leq 3\,000$.
- **Subtarefa 4 (31 pontos):** $Q = 0$.
- **Subtarefa 5 (35 pontos):** $N \leq 60\,000$.
- **Subtarefa 6 (5 pontos):** Sem restrições adicionais.

Exemplos

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

Explicação do exemplo 1: Inicialmente, os pares (i, j) com energia igual a 5 são: $(1, 5), (2, 6), (2, 7), (3, 5), (4, 7), (4, 8)$. $E(3, 5) = |3 - 5| + |4 - 1| = 5$, por exemplo. Depois de bloquear a posição 6, os pares são: $(1, 5), (3, 5)$. Depois de bloquear a posição 2, o único par é: $(3, 5)$. Depois de bloquear a posição 1, o único par é: $(3, 5)$.

Exemplo de entrada 2	Exemplo de saída 2
7 0 10 10 23 8 15 16 13 14	6

Explicação do exemplo 2: Os pares (i, j) com energia igual a 10 são: $(1, 5), (1, 7), (3, 5), (3, 7), (2, 4), (2, 5)$. $E(1, 7) = |1 - 7| + |10 - 14| = 10$, por exemplo.