A. Insetologia

Límite de tiempo: 1 segundos

Límite de memoria: 256 megabytes

Paz é uma pesquisadora especializada no estudo de insetos. Durante uma expedição na floresta, ela identificou um grupo de espécimes que precisa capturar para analisá-los em laboratório.

Devido à natureza de sua pesquisa, todos os insetos devem ser capturados exatamente ao mesmo tempo. Se algum deles ficar fora da captura inicial, poderá se deslocar e alterar os resultados do estudo. Por esse motivo, Paz decidiu utilizar uma única rede que cubra simultaneamente todos os insetos observados.

A rede tem formato retangular e seus lados medem a e b. Para fabricar a rede é necessária uma quantidade de material proporcional à sua área, portanto uma rede maior implica um custo de produção mais elevado. Como o orçamento disponível para a expedição é limitado, Paz deseja construir a menor rede possível que ainda seja capaz de cobrir todos os insetos.

Após registrar cuidadosamente a posição de cada inseto em um plano cartesiano, sua tarefa consiste em determinar a área mínima que uma rede retangular deve ter para cobri-los todos.

Considera-se que um inseto está coberto se estiver no interior da rede ou sobre qualquer uma de suas bordas.

A rede deve ter seus lados paralelos aos eixos coordenados. Entre todas as redes possíveis que cobrem todos os insetos, deve-se encontrar aquela com área mínima.

Entrada

A primeira linha da entrada contém um inteiro n ($1 \leq n \leq 10^6$), que representa a quantidade de insetos observados.

As próximas n linhas contêm dois inteiros x e y, correspondentes às coordenadas de cada inseto no plano cartesiano.

É garantido que $-10^9 \leq x, y \leq 10^9$.

Salida

A saída deve conter três inteiros separados por espaços: A, a e b.

- A corresponde à área mínima de uma rede retangular capaz de cobrir todos os insetos.
- a corresponde à largura dessa rede.
- b corresponde à altura dessa rede.

Considera-se que um inseto está coberto se estiver no interior da rede ou sobre qualquer uma de suas bordas.

Ejemplos

Entrada 1	Salida 1
3 1 1 4 1 2 3	6 3 2

B. Baús de *Luz de Prata*

Limite de tempo: 1 segundos

Limite de memória: 512 megabytes

Na vila de *Luz de Prata* existe um costume para dar o passo para a maioridade. Já que há centenas de anos os seus antepassados seguiam um estilo de vida nómada, para honrá-los cada indivíduo deve partir numa viagem de um ano ao atingir a maioridade. Para iniciar a viagem, a primeira coisa é conseguir um baú para transportar os seus pertences durante a viagem. Onde há uma necessidade, há um negócio, e a Javiera sabe disso, por isso quer iniciar-se no negócio da fabricação de baús.

Na vila de *Luz de Prata* existe um costume para dar o passo para a maioridade. Já que há centenas de anos os seus antepassados seguiam um estilo de vida nómada, para honrá-los cada indivíduo deve partir numa viagem de um ano ao atingir a maioridade. Para iniciar a viagem, a primeira coisa é conseguir um baú para transportar os seus pertences durante a viagem. Onde há uma necessidade, há um negócio, e a Javiera sabe disso, por isso quer iniciar-se no negócio da fabricação de baús.

No entanto, as pessoas de *Luz de Prata* tornaram-se tremendamente supersticiosas desde a chegada de oráculos à vila. Como desejam fortemente que as suas viagens sejam bem-sucedidas e que possam regressar a salvo, apenas aceitarão baús que sejam de **boa sorte**.

Os oráculos utilizam um sistema composto por um número da sorte pessoal k e por 3 listas de números da sorte de tamanho n denominadas a, b e c. Para facilitar a implementação destes números, as listas são sempre entregues em ordem, do menor para o maior.

Um baú é um paralelepípedo definido pelas suas dimensões x, y e z; para ser de **boa sorte**, estas devem coincidir com um número da lista a, b e c, respetivamente. Javiera percebeu que pode representar um baú de **boa sorte** como uma 3-tupla (i, j, k), onde as suas dimensões correspondem a $x = a_i$, $y = b_j$ e $z = c_k$.

Além disso, as pessoas de *Luz de Prata* são muito humildes, escolher o maior baú seria malvisto e preferem escolher um baú mais pequeno. Para isso, consideram todos os baús de **boa sorte** possíveis e ficam com o k-ésimo baú com maior superfície.

Fabricar um baú leva muito tempo, pelo que Javiera precisa de prever que baú quererão comprar. No entanto, enquanto a Javiera decidia pedir-te ajuda, percebeu que vários baús da sorte poderiam ter a mesma superfície! Como Javiera odeia o não-determinismo, decidiu que se 2 baús de **boa sorte** tiverem a mesma superfície, então considerará o lexicograficamente menor como o mais pequeno.

Entrada

A primeira linha da entrada contém 2 inteiros n e k, $(1 \leq n \leq 2 \cdot 10^5, 1 \leq k \leq \text{mín}(n^3, 2 \cdot 10^5))$, a quantidade de números da sorte de cada lista e o número da sorte de um habitante particular de *Luz de Prata*.

As seguintes três linhas contêm n inteiros cada uma, os números da sorte das listas a, b e c. $(1 \leq a_i, b_i, c_i \leq 2 \cdot 10^9)$

As listas de números da sorte são entregues ordenadas, ou seja, cumpre-se sempre que: $a_i \leq a_{i+1}$, $b_i \leq b_{i+1}$ e $c_i \leq c_{i+1}$.

Saída

A saída deve conter 3 inteiros i, j e k $(1 \leq i, j, k \leq n)$, os índices das listas a, b e c, respetivamente. De forma que represente o baú de **boa sorte** com a k-ésima maior superfície, ou maior ordem lexicográfica em caso de empates.

Exemplos

Entrada 1	Saída 1
3 2 1 3 10 2 4 8 3 5 7	3 3 2

Entrada 2	Saída 2
3 3 1 3 10 2 4 8 3 5 7	3 2 3

Nota

No primeiro e no segundo caso de exemplo existem 3^3 tuplas (i, j, k) que representam todos os baús de **boa sorte** possíveis. As 4 primeiras com maior superfície são:

1. $(3, 3, 3)$ com superfície $2a_3b_3 + 2b_3c_3 + 2c_3a_3 = 412$
2. $(3, 3, 1)$ com superfície $2a_3b_3 + 2b_3c_1 + 2c_1a_3 = 340$
3. $(3, 2, 3)$ com superfície $2a_3b_2 + 2b_2c_3 + 2c_3a_3 = 276$
4. $(3, 3, 1)$ com superfície $2a_3b_3 + 2b_3c_1 + 2c_1a_3 = 268$

Portanto, o segundo baú de **boa sorte** com maior superfície é $(3, 3, 1)$ e o terceiro é $(3, 2, 3)$

No terceiro caso de exemplo existem 2^2 tuplas (i, j, k), por ordem estas são:

1. $(2, 2, 2)$ com superfície 250
2. $(2, 1, 2)$ com superfície 150
3. $(2, 2, 0)$ com superfície 130
4. $(1, 2, 2)$ com superfície 130

Notar que como $(1, 2, 2) < (2, 2, 0)$ segundo a ordem lexicográfica, então considera-se $(1, 2, 2)$ como um baú de **boa sorte** mais pequeno. Por isso, o quarto maior baú de **boa sorte** é $(1, 2, 2)$.

C. Competição com Compressão

Límite de tiempo: 2 segundos

Límite de memoria: 512 megabytes

Sofia trabalha na empresa Serviços de Compressão em Tudo (SCT), que, devido às altas nos preços de memória e discos, tem se tornado cada vez mais procurada. Em particular, muitos clientes estão desesperados para comprimir ainda mais suas imagens, por isso Sofia decidiu melhorar o estado da arte e criar seu próprio sistema de compressão de imagens sem perdas, mas não quer publicá-lo porque tem medo de que roubem seu algoritmo.

Ela precisa da sua ajuda para avaliar seu algoritmo. Preparou este problema, que permite comparar soluções com sua implementação. Você consegue vencê-la?

Interação

Para cada caso de Sofia, sua solução é executada duas vezes.

Compressão (1)

A primeira linha contém a palavra `compress`.

A segunda linha contém dois inteiros positivos n e m $(1 \leq n \cdot m \leq 2^{20})$, as dimensões da imagem.

As próximas n linhas contêm m pixels. Cada pixel está escrito como oito dígitos hexadecimais `RRGGBBAA`, em que cada par representa os canais vermelho, verde, azul e alfa, respectivamente.

A saída deve conter uma linha com um inteiro k, a quantidade de valores usados pela representação comprimida. A linha seguinte deve conter k inteiros a_i em representação decimal $(0 \leq a_i < 2^{32})$. Esses valores serão entregues, na mesma ordem, à fase de descompressão.

Descompressão (2)

A primeira linha contém a palavra `decompress`.

A segunda linha contém dois inteiros positivos n e m $(1 \leq n \cdot m \leq 2^{20})$, as dimensões da imagem original.

A terceira linha contém um inteiro k, gerado pela fase de compressão da solução.

A quarta linha contém os k inteiros a_i em representação decimal $(0 \leq a_i < 2^{32})$, gerados pela fase de compressão e na mesma ordem.

A saída deve conter uma matriz de n por m pixels, idêntica à imagem original entregue durante a fase de compressão.

Pontuação

Para cada caso, seu algoritmo é avaliado com base em sua eficiência em comparação com o de Sofia. Caso qualquer um dos pixels seja diferente do pixel da imagem original, serão atribuídos 0 pontos nesse caso de teste.

A eficiência de compressão é definida como a razão entre a quantidade de valores utilizados e a quantidade de pixels:

$$\frac{k}{n \cdot m}$$

Em cada caso de teste, utiliza-se a eficiência alcançada pelo algoritmo secreto de Sofia t e a alcançada pela solução x para calcular uma pontuação S associada ao caso:

$$S = \begin{cases} 100 & \text{se } x \leq t \\ 100 \cdot \max\left(\frac{1-x}{1-t}, 0\right)^2 & \text{se } x > t \end{cases}$$

A pontuação total é a média das pontuações de cada caso. Sofia testará sua solução com múltiplas imagens. Para ajudá-lo, ele decidiu publicar algumas imagens deste conjunto e outras que foram geradas de maneira semelhante. As imagens estão disponíveis neste link: repositorio.imagens.

Ejemplos

Entrada 1	Salida 1
compress 2 2 FF0000FF 00FF00FF 0000FFFF 00000000	4 4278190335 16711935 65535 0

Entrada 2	Salida 2
decompress 2 2 4 4278190335 16711935 65535 0	FF0000FF 00FF00FF 0000FFFF 00000000

Nota

É possível receber o veredito `Idleness Limit Exceeded`. Esse veredito aparece quando seu programa deixa de produzir saída por tempo demais, por exemplo, se fica esperando mais entrada do que receberá ou se demora demais calculando antes de responder. Dependendo da causa, isso pode corresponder, na prática, a um `Time Limit Exceeded` ou a uma resposta com formato incorreto.

D. Desembocadura

Limite de tempo: 1 segundos

Limite de memória: 256 megabytes

Conta a lenda que o **Grande Rio** nasce onde o ar é rarefeito e a neve nunca derrete por completo, nos picos compartilhados entre **Peru** e **Colômbia**. Ali brotam, de dezenas de nascentes escondidas, os primeiros fios d'água que mais tarde se transformarão na maior vazão do continente.

Durante séculos, exploradoras sonharam em subir suas águas e, acima de tudo, em responder à pergunta que a todos fascinava: se alguém se deixasse levar pela correnteza, por onde acabaria chegando ao mar?

A travessia não é para covardes. Cada afluente encontra outro em uma *confluência*, e, a partir de cada encontro, a água se divide e se reúne, formando um labirinto de canais que se estende de um extremo ao outro da América do Sul. As expedições levavam meses para cartografar um único trecho: anotavam para qual confluência cada cruzamento escoava e, quando duas equipes que haviam partido de nascentes diferentes voltavam a se encontrar, registravam o ponto exato onde suas águas finalmente passavam a correr juntas. Dessas façanhas restaram apenas mapas incompletos e diários amarelados.

Hoje, a cartógrafa **Marcela** afirma ter resolvido o antigo enigma. Com esses fragmentos de informação — sem jamais ter visto o oceano — ela garante ser capaz de apontar o cruzamento por onde o Grande Rio deixa o continente. Seu papel é o da cética: construir uma bacia hidrográfica em que seu método, por mais engenhoso que pareça, a faça hesitar entre duas saídas diferentes.

Formalmente, a bacia é um conjunto de n *cruzamentos* ligados por $n-1$ *trechos* navegáveis, de modo que entre qualquer par de cruzamentos existe **um único caminho fluvial**.

Toda a água da bacia escoa para um único cruzamento de saída: a **foz**. Para qualquer cruzamento x, sua água percorre o único caminho que o leva até a foz; dizemos que todo cruzamento pertencente a esse caminho está *rio abaixo* de x.

Para dois cruzamentos a e b, sua **confluência** é o cruzamento onde, pela primeira vez, as águas que descem de a e de b se encontram a caminho do mar. Formalmente,

$$\text{confluência}(a, b)$$

é o cruzamento mais distante da foz que está rio abaixo tanto de a quanto de b.

Marcela dispõe de **registros hidrográficos** — cada um anota, para um par de *extremos* da bacia (as nascentes e quaisquer outros cruzamentos de grau 1), em qual cruzamento suas águas se encontravam — e afirma que, conhecendo apenas quais trechos existem (mas **não** qual cruzamento é a foz), isso basta para descobrir a desembocadura. Os registros de que dispõe são exatamente os produzidos por seu próprio procedimento:

```
ler inteiro n                                # quantidade de cruzamentos da bacia
para i de 1 até n-1:
    ler u, v                                 # existe um trecho navegável entre u e v
    grau[u] += 1 ; grau[v] += 1
    adicionar v a Trechos[u] ; adicionar u a Trechos[v]

extremos = []                                # cruzamentos de grau 1, em ordem crescente
para i de 1 até n:
    se grau[i] == 1:
        extremos.append(i)

registros = []
i = 0
enquanto i + 1 < |extremos|:
    a = extremos[i]
    b = extremos[i+1]
    registros.append((a, b, confluência(a, b)))
    i += 2

se |extremos| é ímpar:
    a = extremos[0]
    b = extremos[|extremos|-1]
    registros.append((a, b, confluência(a, b)))

candidatos = []
para r de 1 até n:
    se compatível(r, registros):
        candidatos.append(r)

assert |candidatos| == 1    # se isso falhar, Marcela NÃO encontra a foz
responder(candidatos[0])
```

Cada registro é uma **tríplice** (a, b, c) que armazena uma *consulta e sua resposta*: a e b são os dois extremos consultados e c é a confluência calculada usando a foz **real** (aquela que você declara). Os três valores são armazenados — e não apenas c — porque a verificação precisa saber *qual par* produziu aquela resposta para poder recalculá-la supondo que outro cruzamento seja a foz.

A função

$$\text{compatível}(r, \text{registros})$$

verifica se o cruzamento r poderia ser a foz: ela retorna verdadeiro se, supondo que toda a bacia escoe por r, **cada** registro produzir exatamente a mesma confluência que foi observada.

Como a foz real é sempre compatível consigo mesma, o `assert` falha exatamente quando **existe outro cruzamento r, diferente do verdadeiro, que produz as mesmas confluências para os pares consultados**: nesse caso Marcela **não consegue distinguir** por onde o rio deságua no mar, e seu método falha.

Sua tarefa: projete manualmente uma bacia hidrográfica (os cruzamentos e seus trechos) e declare qual é o cruzamento correspondente à foz real, de modo que o método de Marcela **NÃO** consiga identificá-la (isto é, que a lista `candidatos` contenha mais de um cruzamento). A pontuação será maior quanto menor for o número de cruzamentos da bacia.

Entrada

Este problema não possui entrada.

Saída

Na primeira linha, dois inteiros n e r ($1 l e r l e n l e 1000$): a quantidade de cruzamentos e o cruzamento correspondente à foz real que você declara.

Em cada uma das $n-1$ linhas seguintes, dois inteiros u e v ($1 l e u, v l e n$, $u = v$) que descrevem um trecho navegável. O conjunto de trechos deve formar uma bacia conectada e sem ciclos (um único caminho fluvial entre cada par de cruzamentos), sem trechos repetidos.

Nota

Como ilustração, considere a seguinte bacia hidrográfica cuja foz real é o cruzamento 1 (**não** é uma resposta válida: neste caso o método de Marcela *acerta*; serve apenas para mostrar como o procedimento funciona):

```
9 1
1 2
2 3
2 4
4 5
4 6
1 7
7 8
7 9
```

Os extremos (cruzamentos de grau 1) são 3, 5, 6, 8 e 9. Marcela consulta pares de extremos e as confluências observadas são compatíveis apenas com a foz no cruzamento 1, de modo que `candidatos = {1}`, o `assert` é satisfeito e o método identifica corretamente a foz.

Seu objetivo é justamente o contrário: construir uma bacia em que `candidatos` contenha mais de um cruzamento.

E. Experimentos Seriais

Límite de tiempo: 8 segundos

Límite de memoria: 256 megabytes

Valeria está preparando sua tese para finalmente se formar e escapar da universidade. Parte da modelagem dos experimentos que ela está realizando consiste em um arranjo $A = [a_1, ..., a_n]$ de tamanho N com valores inteiros fixos entre 0 e n, e outros valores marcados com -1 que devem ser preenchidos com informação experimental.

Depois de preencher os valores faltantes, para realizar seus experimentos, ela precisa executar um algoritmo sobre o arranjo que faz

$$f(A) = \sum_{l=1}^{n} \sum_{r=l}^{n} \text{mex}(a_l, ..., a_r)^1$$

operações. Ela tem um pressentimento de que isso pode ser demais, mas ainda não encontrou um caso em que preencher os valores faltantes maximize esse valor.

Sua tarefa consiste em preencher os valores faltantes de modo que $f(A)$ tenha o maior valor possível.

Entrada

A primeira linha da entrada contém um inteiro n, $(1 \leq n \leq 10^4)$, o número de elementos do arranjo de Valeria.

Salida

A saída deve conter n inteiros $a_1, ..., a_n (0 \leq a_i \leq n)$, os valores do arranjo gerado.

Pontuação

Para cada caso, sua solução é avaliada segundo o valor de $f(B)$, onde B é o arranjo obtido depois de completar os valores faltantes. Caso a solução não complete corretamente o arranjo, considera-se que o caso tem 0 pontos.

Uma solução é válida se substitui todos os valores -1 por inteiros não negativos e mantém inalteradas todas as posições que já tinham um valor fixo.

Seja $x = f(B)$ o valor alcançado pela solução. Para cada caso, define-se um valor de referência r e um valor objetivo u, com $r \leq u$.

A pontuação S associada ao caso é calculada como:

$$S = \begin{cases} 0 & \text{se a saída não é válida} \\ 100 \cdot \frac{x}{r} & \text{se } u = r \\ 50 \cdot \frac{x}{r} & \text{se } u > r \text{ e } x \leq r \\ \min\left(100, 50 + 50 \cdot \sqrt{\frac{x-r}{u-r}}\right) & \text{se } u > r \text{ e } x > r \end{cases}$$

Finalmente, a pontuação total é a média das pontuações de cada caso.

Ejemplos

Entrada 1	Salida 1
4 -1 3 -1 2	0 3 1 2

[1] O mex de um conjunto de valores é o menor inteiro não negativo que não aparece nele. Por exemplo, $\text{mex}(0, 1, 3) = 2$, $\text{mex}(1, 2, 3) = 0$ e $\text{mex}(0, 1, 2) = 3$.

Entrada 2	Salida 2
3 -1 0 -1	1 0 2

Nota

Nos casos de exemplo são mostradas possíveis soluções, não necessariamente ótimas.