A. Insectología

Límite de tiempo: 1 segundos

Límite de memoria: 256 megabytes

Paz es una investigadora especializada en el estudio de insectos. Durante una expedición en el bosque, identificó un grupo de ejemplares que necesita capturar para analizarlos en el laboratorio.

Debido a la naturaleza de su investigación, todos los insectos deben ser capturados exactamente al mismo tiempo. Si alguno queda fuera de la captura inicial, podría desplazarse y alterar los resultados del estudio. Por esta razón, Paz ha decidido utilizar una única red que cubra simultáneamente a todos los insectos observados.

La red tiene forma rectangular y sus lados miden a y b. Para fabricar la red se requiere una cantidad de material proporcional a su área, por lo que una red más grande implica un mayor costo de producción. Como el presupuesto disponible para la expedición es limitado, Paz desea construir la red más pequeña posible que siga siendo capaz de cubrir a todos los insectos.

Luego de registrar cuidadosamente la posición de cada insecto sobre un plano cartesiano, tu tarea consiste en determinar el área mínima que debe tener una red rectangular para cubrirlos a todos.

Se considera que un insecto está cubierto si se encuentra en el interior de la red o sobre alguno de sus bordes.

La red debe tener sus lados paralelos a los ejes coordenados. Entre todas las redes posibles que cubren a todos los insectos, se debe encontrar aquella de área mínima.

Entrada

La primera línea de la entrada contiene un entero n ($1 \leq n \leq 10^6$), que representa la cantidad de insectos observados.

Las siguientes n líneas contienen dos enteros x e y, correspondientes a las coordenadas de cada insecto en el plano cartesiano.

Se garantiza que $-10^9 \leq x, y \leq 10^9$.

Salida

La salida debe contener tres enteros separados por espacios: A, a y b.

- A corresponde al área mínima de una red rectangular capaz de cubrir a todos los insectos.
- a corresponde al ancho de dicha red.
- b corresponde a la altura de dicha red.

Se considera que un insecto está cubierto si se encuentra en el interior de la red o sobre alguno de sus bordes.

Ejemplos

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

B. Baúles de *Luz de Plata*

Límite de tiempo: 1 segundos

Límite de memoria: 512 megabytes

En el pueblo de *Luz de Plata* tienen una costumbre para dar el paso a la mayoría de edad. Ya que hace cientos de años sus ancestros seguían un estilo de vida nómada, para honrarlos cada individuo debe irse de viaje por un año al alcanzar la mayoría de edad. Para iniciar el viaje, lo primero es conseguir un baúl para transportar sus pertenencias durante el viaje. Donde hay una necesidad, hay un negocio, y esto Javiera lo sabe, por ello quiere iniciarse en el negocio de la fabricación de baúles.

Sin embargo, la gente de *Luz de Plata* se ha vuelto tremendamente supersticiosa desde la llegada de oráculos al pueblo. Cómo desean fuertemente que sus viajes sean exitosos y puedan regresar a salvo, solo aceptarán baúles que sean de **buena suerte**.

Los oráculos utilizan un sistema conformado de un número de la suerte personal k y de 3 listas de números de la suerte de tamaño n denominadas a, b y c, para facilitar la implementación de estos números, las listas siempre se entregan en orden, de menor a mayor.

Un baúl es un paralelepípedo definido por sus dimensiones x, y y z, para ser de **buena suerte** estos deben coincidir con un número de la lista a, b y c, respectivamente. Javiera pudo observar que puede representar un baúl de de **buena suerte** como una 3-tupla (i, j, k), donde sus dimensiones corresponden a $x = a_i$, $y = b_j$ y $z = c_k$.

Además la gente de *Luz de Plata* es muy humilde, elegir el baúl más grande sería mal visto y prefieren elegir un baúl más chico, para esto consideran todos los baúles de **buena suerte** posibles y se quedan con el k-ésimo baúl con mayor superficie.

Fabricar un baúl toma mucho tiempo, por lo que Javiera necesita predecir qué baúl querrán comprar. Sin embargo, mientras Javiera decidia pedirte ayuda, ¡se dió cuenta que múltiples baúles de la suerte podrían tener la misma superficie! Como Javiera odia el no-determinismo, decidió que si 2 baúles de **buena suerte** tienen la misma superficie, entonces el lexicográficamente menor lo considerará más chico.

Entrada

La primera línea de la entrada contiene 2 enteros n y k, $(1 \leq n \leq 2 \cdot 10^5, 1 \leq k \leq \text{mín}(n^3, 2 \cdot 10^5))$, la cantidad de números de la suerte de cada lista y el número de la suerte de un habitante particular de *Luz de Plata*.

Las siguientes tres líneas contienen n enteros cada una, los números de suerte de las listas a, b y c. $(1 \leq a_i, b_i, c_i \leq 2 \cdot 10^9)$

Las listas de números de la suerte se entregan ordenadas, es decir, siempre se cumple que: $a_i \leq a_{i+1}$, $b_i \leq b_{i+1}$ y $c_i \leq c_{i+1}$.

Salida

La salida debe contener 3 enteros i, j y k $(1 \leq i, j, k \leq n)$, los índices de las listas a, b y c, respectivamente. De forma que represente el baúl de **buena suerte** con k-ésima mayor superficie, o mayor orden lexicográfico en caso de empates.

Ejemplos

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

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

Entrada 3	Salida 3
2 4 1 5 5 10 1 5	1 2 2

Nota

En el primer y segundo caso de ejemplo hay 3^3 tuplas (i, j, k) que representan todos los baúles de **buena suerte** posibles. Las 4 primeras con mayor superficie son:
1. $(3, 3, 3)$ con superficie $2a_3b_3 + 2b_3c_3 + 2c_3a_3 = 412$
2. $(3, 3, 1)$ con superficie $2a_3b_3 + 2b_3c_1 + 2c_1a_3 = 340$
3. $(3, 2, 3)$ con superficie $2a_3b_2 + 2b_2c_3 + 2c_3a_3 = 276$
4. $(3, 3, 1)$ con superficie $2a_3b_3 + 2b_3c_1 + 2c_1a_3 = 268$

Por lo tanto el segundo baúl de **buena suerte** con major superficie es $(3, 3, 1)$ y el tercero es $(3, 2, 3)$

En el tercer caso de ejemplo hay 2^2 tuplas (i, j, k), en orden estas son:
1. $(2, 2, 2)$ con superficie 250
2. $(2, 1, 2)$ con superficie 150
3. $(2, 2, 0)$ con superficie 130
4. $(1, 2, 2)$ con superficie 130

Notar que como $(1, 2, 2) < (2, 2, 0)$ según orden lexicográfico, entonces se considera $(1, 2, 2)$ como un baúl de **buena suerte** más chico. Por esto el cuarto baúl de **buena suerte** más grande es $(1, 2, 2)$.

C. Competencia con Compresión

Límite de tiempo: 2 segundos

Límite de memoria: 512 megabytes

Sofía trabaja en Servicios de Compresión en Todo (SCT), que, debido a las alzas en los precios de memoria y discos, se ha vuelto cada vez más demandado. En particular, muchos clientes están desesperados por comprimir aún más sus imágenes, por lo que Sofía ha decidido mejorar el estado del arte y crear su propio sistema de compresión de imágenes sin pérdida, pero no lo quiere publicar porque tiene miedo a que le roben el algoritmo.

Ella necesita su ayuda para evaluar su algoritmo. Ha preparado este problema, que permite comparar soluciones con su implementación. ¿Puedes ganarle?

Interacción

Para cada caso de Sofía, tu solución es ejecutada dos veces.

Compresión (1)

La primera línea contiene la palabra `compress`.

La segunda línea contiene dos enteros positivos n y m $(1 \leq n \cdot m \leq 2^{20})$, las dimensiones de la imagen.

Las siguientes n líneas contienen m píxeles. Cada píxel está escrito como ocho dígitos hexadecimales `RRGGBBAA`, donde cada par representa los canales rojo, verde, azul y alfa, respectivamente.

La salida debe contener una línea con un entero k, la cantidad de valores usados por la representación comprimida. La siguiente línea debe contener k enteros a_i en representación decimal $(0 \leq a_i < 2^{32})$. Estos valores serán entregados, en el mismo orden, a la fase de descompresión.

Descompresión (2)

La primera línea contiene la palabra `decompress`.

La segunda línea contiene dos enteros positivos n y m $(1 \leq n \cdot m \leq 2^{20})$, las dimensiones de la imagen original.

La tercera línea contiene un entero k, generado por la fase de compresión de la solución.

La cuarta línea contiene los k enteros a_i en representación decimal $(0 \leq a_i < 2^{32})$, generados por la fase de compresión y en el mismo orden.

La salida debe contener una matriz de n por m píxeles, idéntica a la imagen original entregada durante la fase de compresión.

Puntaje

Para cada caso se evalúa tu algoritmo con base en su eficiencia en comparación con el de Sofía. En caso de que cualquiera de los píxeles sea distinto al píxel de la imagen original, se asignarán 0 puntos en ese caso de prueba.

La eficiencia de compresión se define como la razón entre la cantidad de valores utilizados y la cantidad de píxeles:

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

En cada caso de prueba se utiliza la eficiencia que alcanza el algoritmo secreto de Sofía t y la alcanzada por la solución x para calcular un puntaje S asociado al caso:

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

El puntaje total es el promedio de los puntajes de cada caso. Sofía probará tu solución con múltiples imágenes. Para ayudarte, ha decidido publicar algunas de este conjunto y otras que fueron generadas de manera similar, las imagenes se encuentran en este link: repositorio.imagenes.

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

Es posible recibir el veredicto `Idleness Limit Exceeded`. Este veredicto aparece cuando tu programa deja de producir salida por demasiado tiempo, por ejemplo si queda esperando más entrada de la que recibirá o si tarda demasiado calculando antes de responder. Dependiendo de la causa, puede corresponder en la práctica a un `Time Limit Exceeded` o a una respuesta con formato incorrecto.

D. Desembocadura

Límite de tiempo: 1 segundos

Límite de memoria: 256 megabytes

Cuenta la leyenda que el **Gran Río** nace donde el aire es delgado y la nieve nunca se derrite del todo, en las cumbres compartidas por **Perú** y **Colombia**. Allí brotan, de docenas de manantiales escondidos, los primeros hilos de agua que más tarde se convertirán en el caudal más poderoso del continente.

Durante siglos, exploradoras soñaron con remontar sus aguas y, sobre todo, con responder la pregunta que a todos obsesionaba: si uno se dejara llevar por la corriente, ¿por dónde saldría finalmente al mar?

La travesía no es para cobardes. Cada afluente se une a otro en un *cruce*, y desde cada cruce el agua se reparte y se reúne formando un laberinto de canales que se extiende de un extremo a otro de Sudamérica. Las expediciones tardaban meses en cartografiar un solo tramo: anotaban qué cruce desaguaba en cuál y, cuando dos partidas que habían partido de nacientes distintas se reencontraban, dejaban constancia del punto exacto donde sus aguas por fin corrían juntas. De aquellas hazañas sólo sobreviven mapas incompletos y bitácoras amarillentas.

Hoy, la cartógrafa **Marcela** asegura haber resuelto el viejo enigma. Con esos retazos de información —sin haber visto jamás el océano— afirma poder señalar con el dedo el cruce por donde el Gran Río abandona el continente. Tu papel es el de la escéptica: construir una cuenca en la que su método, por más ingenioso que parezca, lo deje dudando entre dos salidas distintas.

Formalmente, la cuenca es un conjunto de n *cruces* unidos por $n-1$ *tramos* navegables, de modo que entre cada par de cruces existe **un único camino fluvial**.

El agua de toda la cuenca fluye hacia un único cruce de salida: la **desembocadura**. Para cualquier cruce x, su agua recorre el único trayecto que lo lleva hasta la desembocadura; decimos que todo cruce de ese trayecto está *aguas abajo* de x.

Para dos cruces a y b, su **confluencia** es el cruce donde por primera vez se juntan las aguas que bajan desde a y desde b camino al mar. Formalmente,

$$\text{confluencia}(a, b)$$

es el cruce más lejano de la desembocadura que está aguas abajo tanto de a como de b.

Marcela dispone de **registros hidrográficos** —cada uno anota, para un par de *extremos* de la cuenca (las nacientes y demás cruces a los que llega un solo tramo), en qué cruce confluían sus aguas— y afirma que, conociendo sólo qué tramos existen (pero **no** cuál cruce es la desembocadura), le bastan para descubrir la desembocadura. Los registros con que cuenta son exactamente los que produce su propio procedimiento:

```
leer entero n                              # cantidad de cruces de la cuenca
para i de 1 a n-1:
    leer u, v                              # existe un tramo navegable entre u y v
    grado[u] += 1 ;  grado[v] += 1
    agregar v a Tramos[u] ;  agregar u a Tramos[v]

extremos = []                              # cruces de grado 1, en orden creciente
para i de 1 a n:
    si grado[i] == 1:
        extremos.append(i)

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

si |extremos| es impar:
    a = extremos[0]
    b = extremos[|extremos|-1]
    registros.append((a, b, confluencia(a, b)))

candidatos = []
para r de 1 a n:
    si compatible(r, registros):
        candidatos.append(r)

assert |candidatos| == 1   # si esto falla, Marcela NO halla la desembocadura
responder(candidatos[0])
```

Cada registro es una **tripleta** (a, b, c) que guarda una *consulta y su respuesta*: a y b son los dos extremos consultados y c es la confluencia calculada con la desembocadura **real** (la que tú declaras). Se guardan los tres valores —y no sólo c— porque la verificación necesita saber *qué par* produjo cada respuesta para poder recomputarlo suponiendo la desembocadura en otro cruce.

La función

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

prueba si el cruce r podría ser la desembocadura: devuelve verdadero si y sólo si, suponiendo que toda la cuenca desaguara por r, **cada** registro daría exactamente la misma confluencia que se observó.

Como la desembocadura real siempre es compatible consigo misma, el `assert` falla exactamente cuando **existe otro cruce r distinto del verdadero** que produce las mismas confluencias en los pares consultados: en ese caso Marcela **no logra distinguir** por dónde sale el río al mar, y su método falla.

Tu tarea: diseñar a mano una cuenca (los cruces y sus tramos) y declarar el cruce real de la desembocadura de modo que el método de Marcela **NO** consiga identificarlo (es decir, que la lista `candidatos` quede con más de un cruce). Se otorga mayor puntaje mientras menos cruces tenga tu cuenca.

Entrada

Este problema no tiene input.

Salida

En la primera línea, dos enteros n y r ($1 \le r \le n \le 1000$): la cantidad de cruces y el cruce real de la desembocadura que declaras.

En cada una de las siguientes $n-1$ líneas, dos enteros u y v ($1 \leq u, v \leq n$, $u \neq v$) que describen un tramo. El conjunto de tramos debe dejar la cuenca conectada y sin circuitos (un único camino fluvial entre cada par de cruces), sin tramos repetidos.

Nota

A modo de ilustración, considera la siguiente cuenca con desembocadura real en el cruce 1 (**no** es una respuesta válida: aquí el método de Marcela *sí* acierta; sirve sólo para mostrar cómo funciona el procedimiento):

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

Los extremos (cruces de grado 1) son 3, 5, 6, 8, 9. Marcela consulta pares de extremos y las confluencias que observa resultan compatibles únicamente con la desembocadura en el cruce 1, de modo que `candidatos = {1}`, el `assert` se cumple y el método identifica correctamente la desembocadura.

Tu objetivo es justamente lo contrario: una cuenca en la que `candidatos` quede con más de un cruce.

E. Experimentos Seriales

Límite de tiempo: 2 segundos

Límite de memoria: 256 megabytes

Valeria está preparando su tesis para finalmente poder titularse y escapar de la universidad. Parte del modelamiento de los experimentos que está realizando, consiste en un arreglo $A = [a_1, ..., a_n]$ de largo n con valores enteros fijos entre 0 y n, y otros valores marcados con -1 que deben ser completados con información experimental.

Una vez llenados los valores faltantes, para llevar a cabo sus experimentos, necesita ejecutar un algoritmo sobre el arreglo que hace

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

operaciones. Ella tiene un presentimiento de que esto puede ser demasiado, pero aún no encuentra un caso en el que llenar los valores faltantes maximice este valor.

Tu tarea consiste en rellenar los valores faltantes de modo que $f(A)$ tenga el mayor valor posible.

Entrada

La primera línea de la entrada contiene un entero n, $(1 \leq n \leq 10^4)$, el número de elementos del arreglo de Valeria.

Salida

La salida debe contener n enteros $a_1, ..., a_n (0 \leq a_i \leq n)$, los valores del arreglo generado.

Scoring

Para cada caso se evalúa tu solución según el valor de $f(B)$, donde B es el arreglo obtenido luego de completar los valores faltantes. En caso de que la solución no complete correctamente el arreglo, se considerará que el caso tiene 0 puntos.

Una solución es válida si reemplaza todos los valores -1 por enteros no negativos y mantiene sin cambios todas las posiciones que ya tenían un valor fijo.

Sea $x = f(B)$ el valor alcanzado por la solución. Para cada caso se define un valor de referencia r y un valor objetivo u, con $r \leq u$.

El puntaje S asociado al caso se calcula como:

$$S = \begin{cases} 0 & \text{si la salida no es válida} \\ 100 \cdot \frac{x}{r} & \text{si } u = r \\ 50 \cdot \frac{x}{r} & \text{si } u > r \text{ y } x \leq r \\ \min\left(100, 50 + 50 \cdot \sqrt{\frac{x-r}{u-r}}\right) & \text{si } u > r \text{ y } x > r \end{cases}$$

Finalmente, el puntaje total es el promedio de los puntajes de cada caso.

Ejemplos

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

[1] El mex de un conjunto de valores es el menor entero no negativo que no aparece en él. Por ejemplo, $\text{mex}(0, 1, 3) = 2$, $\text{mex}(1, 2, 3) = 0$ y $\text{mex}(0, 1, 2) = 3$.

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

Nota

En los casos de ejemplo se muestran posibles soluciones, no necesariamente óptimas.