Árboles y Grafos, 2025-2

Para entregar el domingo 7 de septiembre de 2025
A las 23:59 en la arena de programación

Instrucciones para la entrega

- Para esta tarea y todas las tareas futuras en la arena de programación, la entrega de soluciones es *individual*. Por favor escriba claramente su nombre, código de estudiante y sección en cada archivo de código (a modo de comentario). Adicionalmente, cite cualquier fuente de información que utilizó. Los códigos fuente que suba a la arena de programación deben der de su completa autoría.
- En cada problema debe leer los datos de entrada de la forma en la que se indica en el enunciado y debe imprimir los resultados con el formato allí indicado. No debe agregar mensajes ni agregar o eliminar datos en el proceso de lectura. La omisión de esta indicación puede generar que su programa no sea aceptado en la arena de programación.
- Puede resolver los ejercicios en C/C++ y Python. Sin embargo, deben haber por los menos soluciones a dos problemas en cada lenguaje.
- Debe enviar sus soluciones a través de la arena. Antes de subir sus soluciones asegurese de realizar pruebas con los casos de pruebas proporcionados para verificar que el programa es correcto, que finaliza y no se quede en un ciclo infinito.
- El primer criterío de evaluación será la aceptación del problema en la arena cumpliendo los requisitos indicados en los enunciados de los ejercicios y en este documento. El segundo criterio de evaluación será la complejidad computacional de la solución y el uso de los temas vistos en clase. Es necesario incluir en la cabecera del archivo comentarios que expliquen la complejidad de la solución del problema para cada caso.

  En adición a lo anterior, para efectos de la calificación se tendrán en cuenta aspectos de estilo como no usar `break` ni `continue` y que las funciones deben tener únicamente una instrucción `return` que debe estar en la última línea.

Problemas prácticos

Hay cuatro problemas prácticos cuyos enunciados aparecen a partir de la siguiente página.

A - Problem A

Source file name: `leha.cpp`
Time limit: x seconds

Leha plays a computer game, where is on each level is given a connected graph with n vertices and m edges. Graph can contain multiple edges, but can not contain self loops. Each vertex has an integer d_i, which can be equal to 0, 1 or -1. To pass the level, he needs to find a "*good*" subset of edges of the graph or say, that it doesn't exist. Subset is called "good", if by leaving only edges from this subset in the original graph, we obtain the following: for every vertex i, $d_i = -1$ or it's degree modulo 2 is equal to d_i. Leha wants to pass the game as soon as possible and ask you to help him. In case of multiple correct answers, print any of them.

Input

The input contains several test cases. The first line of each testcase contains two integers n, m ($1 \le n \le 3 \cdot 10^5$, $n - 1 \le m \le 3 \cdot 10^5$) — number of vertices and edges. The second line contains n integers $d_1, d_2, \ldots, d_n$ ($-1 \le d_i \le 1$) — numbers on the vertices. Each of the next m lines contains two integers u and v ($1 \le u, v \le n$) — edges. It's guaranteed, that graph in the input is connected.

The input must be read from standard input.

Output

For each test case print -1 in a single line, if solution doesn't exist. Otherwise in the first line k — number of edges in a subset. In the next k lines indexes of edges. Edges are numerated in order as they are given in the input, starting from 1.

The output must be written to standard output.

Sample Input	Sample Output
1 0 1 4 5 0 0 0 -1 1 2 2 3 3 4 1 4 2 4 2 1 1 1 1 2 3 3 0 -1 1 1 2 2 3 1 3	-1 0 1 1 1 2

B - Problem B

Source file name: `robot.cpp`
Time limit: x seconds

A robot has to patrol around a rectangular area which is in a form of $m \times n$ grid (m rows and n columns). The rows are labeled from 1 to m. The columns are labeled from 1 to n. A cell (i, j) denotes the cell in row i and column j in the grid. At each step, the robot can only move from one cell to an adjacent cell, i.e. from (x, y) to $(x + 1, y)$, $(x, y + 1)$, $(x - 1, y)$ or $(x, y - 1)$. Some of the cells in the grid contain obstacles. In order to move to a cell containing obstacle, the robot has to switch to turbo mode. Therefore, the robot cannot move continuously to more than k cells containing obstacles.

Your task is to write a program to find the shortest path (with the minimum number of cells) from cell $(1, 1)$ to cell (m, n). It is assumed that both these cells do not contain obstacles.

Input

The input consists of several data sets. The first line of the input file contains the number of data sets which is a positive integer. The following lines describe the data sets. For each data set, the first line contains two positive integer numbers m and n separated by space ($1 \leq m, n \leq 100$). The second line contains an integer number k ($0 \leq k \leq 100$). The i-th line of the next m lines contains n integer a_{ij} separated by space ($i = 1, 2, \ldots, m$; $j = 1, 2, \ldots, n$). The value of a_{ij} is `'1'` if there is an obstacle on the cell (i, j), and is `'0'` otherwise.

The input must be read from standard input.

Output

For each data set, if there exists a way for the robot to reach the cell (m, n), write in one line the integer number s, which is the number of moves the robot has to make; `'-1'` otherwise.

The output must be written to standard output.

Sample Input	**Sample Output**
3 2 5 0 0 1 0 0 0 0 0 0 1 0 4 6 1 0 1 1 0 0 0 0 0 1 0 1 1 0 1 1 1 1 0 0 1 1 1 0 0 2 2 0 0 1 1 0	7 10 -1

C - Problem C

Source file name: `rumor.cpp`
Time limit: x seconds

Vova promised himself that he would never play computer games . . . But recently Firestorm — a well-known game developing company — published their newest game, World of Farcraft, and it became really popular. Of course, Vova started playing it.

Now he tries to solve a quest. The task is to come to a settlement named Overcity and spread a rumor in it.

Vova knows that there are n characters in Overcity. Some characters are friends to each other, and they share information they got. Also Vova knows that he can bribe each character so he or she starts spreading the rumor; i-th character wants ci gold in exchange for spreading the rumor. When a character hears the rumor, he tells it to all his friends, and they start spreading the rumor to their friends (for free), and so on.

The quest is finished when all n characters know the rumor. What is the minimum amount of gold Vova needs to spend in order to finish the quest?

Take a look at the notes if you think you haven't understood the problem completely.

Input

The input contains several test cases. The first line of each testcase containstwo integer numbers *nandm* ($1 \le n \le 2 \cdot 10^4, 0 \le m \le 4 \cdot 10^4$) — the number of characters in Overcity and the number of pairs of friends. The second line contains n integer numbers c_i ($0 \le c_i \le 10^5$) — the amount of gold i-th character asks to start spreading the rumor. Then m lines follow, each containing a pair of numbers (x_i, y_i) which represent that characters x_i and y_i are friends ($1 \le x_i, y_i \le n, x_i \ne y_i$). It is guaranteed that each pair is listed at most once. The end of the input will be a case such that $n = m = 0$.

The input must be read from standard input.

Output

For each test case, print one number — the minimum amount of gold Vova has to spend in order to finish the quest.

The output must be written to standard output.

Sample Input	**Sample Output**
5 2 2 5 3 4 8 1 4 4 5 10 0 1 2 3 4 5 6 7 8 9 10 10 5 1 6 2 7 3 8 4 9 5 10 1 2 3 4 5 6 7 8 9 10 0 0	10 55 15

D - Problem D

Source file name: `spiders.cpp`
Time limit: x seconds

Mars is home to an unusual species of spiders — Binary spiders.

Right now, Martian scientists are observing a colony of n spiders, the i-th of which has a_i legs.

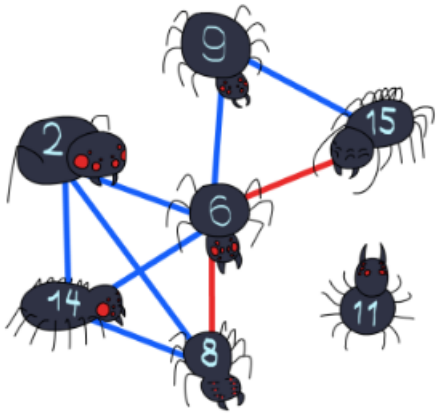
Some of the spiders are friends with each other. Namely, the i-th and j-th spiders are friends if $gcd(a_i, a_j) \neq 1$, i. e., there is some integer $k \geq 2$ such that a_i and a_j are simultaneously divided by k without a remainder. Here $gcd(x, y)$ denotes the greatest common divisor (GCD) of integers x and y.

Scientists have discovered that spiders can send messages. If two spiders are friends, then they can transmit a message directly in one second. Otherwise, the spider must pass the message to his friend, who in turn must pass the message to his friend, and so on until the message reaches the recipient.

Let's look at an example.

Suppose a spider with eight legs wants to send a message to a spider with 15 legs. He can't do it directly, because $gcd(8, 15) = 1$. But he can send a message through the spider with six legs because $gcd(8, 6) = 2$ and $gcd(6, 15) = 3$. Thus, the message will arrive in two seconds.

Right now, scientists are observing how the s-th spider wants to send a message to the t-th spider. The researchers have a hypothesis that spiders always transmit messages optimally. For this reason, scientists would need a program that could calculate the minimum time to send a message and also deduce one of the optimal routes.



Input

The first line of each testcase contains an integer n ($2 \leq n \leq 1000$) – the number of spiders in the colony. The second line of each testcase contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 50000$) — the number of legs the spiders have. The third line of input contains two integers s and t ($1 \leq s, t \leq n$) — the spiders between which the message must be sent.

The input must be read from standard input.

Output

For each test case if it is impossible to transmit a message between the given pair of spiders, print `-1`. Otherwise, in the first line of the output print the integer t ($t \geq 1$) — the number of spiders that participate in the message transmission (i. e. the minimum time of message delivery in seconds plus one). In the second line, print t different integers $b_1, b_2, \ldots, b_t$ ($1 \leq b_i \leq n$) — the ids of the spiders through which the message should follow, in order from sender to receiver.

If there are several optimal routes for the message, output the one that is lexicographically smaller.

The output must be written to standard output.

Sample Input	**Sample Output**
7 2 14 9 6 8 15 11 5 6 7 2 14 9 6 8 15 11 5 7 7 2 14 9 6 8 15 11 5 5	3 5 4 6 -1