Codeforces Round #149 (Div. 2)

A. Heads or Tails

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Petya and Vasya are tossing a coin. Their friend Valera is appointed as a judge. The game is very simple. First Vasya tosses a coin x times, then Petya tosses a coin y times. If the tossing player gets head, he scores one point. If he gets tail, nobody gets any points. The winner is the player with most points by the end of the game. If boys have the same number of points, the game finishes with a draw.

At some point, Valera lost his count, and so he can not say exactly what the score is at the end of the game. But there are things he remembers for sure. He remembers that the entire game Vasya got heads at least a times, and Petya got heads at least b times. Moreover, he knows that the winner of the game was Vasya. Valera wants to use this information to know every possible outcome of the game, which do not contradict his memories.

Input

The single line contains four integers x, y, a, b $(1 \le a \le x \le 100, 1 \le b \le y \le 100)$. The numbers on the line are separated by a space.

Output

In the first line print integer n — the number of possible outcomes of the game. Then on n lines print the outcomes. On the i-th line print a space-separated pair of integers c_i, d_i — the number of heads Vasya and Petya got in the i-th outcome of the game, correspondingly. Print pairs of integers (c_i, d_i) in the strictly increasing order.

Let us remind you that the pair of numbers (p_1, q_1) is less than the pair of numbers (p_2, q_2), if $p_1 < p_2$, or $p_1 = p_2$ and also $q_1 < q_2$.

Sample test(s)

input
3 2 1 1

output
3 2 1 3 1 3 2

input
2 4 2 2

output
0

B. Big Segment

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

A coordinate line has n segments, the i-th segment starts at the position l_i and ends at the position r_i. We will denote such a segment as $[l_i, r_i]$.

You have suggested that one of the defined segments covers all others. In other words, there is such segment in the given set, which contains all other ones. Now you want to test your assumption. Find in the given set the segment which covers all other segments, and print its number. If such a segment doesn't exist, print -1.

Formally we will assume that segment $[a, b]$ covers segment $[c, d]$, if they meet this condition $a \le c \le d \le b$.

Input

The first line contains integer n $(1 \le n \le 10^5)$ — the number of segments. Next n lines contain the descriptions of the segments. The i-th line contains two space-separated integers l_i, r_i $(1 \le l_i \le r_i \le 10^9)$ — the borders of the i-th segment.

It is guaranteed that no two segments coincide.

Output

Print a single integer — the number of the segment that covers all other segments in the set. If there's no solution, print -1.

The segments are numbered starting from 1 in the order in which they appear in the input.

Sample test(s)

input
3 1 1 2 2 3 3
output
-1

input
6 1 5 2 3 1 10 7 10 7 7 10 10
output
3

C. King's Path

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The black king is standing on a chess field consisting of 10^9 rows and 10^9 columns. We will consider the rows of the field numbered with integers from 1 to 10^9 from top to bottom. The columns are similarly numbered with integers from 1 to 10^9 from left to right. We will denote a cell of the field that is located in the i-th row and j-th column as (i, j).

You know that some squares of the given chess field are *allowed*. All allowed cells of the chess field are given as n segments. Each segment is described by three integers r_i, a_i, b_i $(a_i \le b_i)$, denoting that cells in columns from number a_i to number b_i inclusive in the r_i-th row are allowed.

Your task is to find the minimum number of moves the king needs to get from square (x_0, y_0) to square (x_1, y_1), provided that he only moves along the allowed cells. In other words, the king can be located only on allowed cells on his way.

Let us remind you that a chess king can move to any of the neighboring cells in one move. Two cells of a chess field are considered neighboring if they share at least one point.

Input

The first line contains four space-separated integers x_0, y_0, x_1, y_1 $(1 \le x_0, y_0, x_1, y_1 \le 10^9)$, denoting the initial and the final positions of the king.

The second line contains a single integer n $(1 \le n \le 10^5)$, denoting the number of segments of allowed cells. Next n lines contain the descriptions of these segments. The i-th line contains three space-separated integers r_i, a_i, b_i $(1 \le r_i, a_i, b_i \le 10^9, a_i \le b_i)$, denoting that cells in columns from number a_i to number b_i inclusive in the r_i-th row are allowed. Note that the segments of the allowed cells can intersect and embed arbitrarily.

It is guaranteed that the king's initial and final position are allowed cells. It is guaranteed that the king's initial and the final positions do not coincide. It is guaranteed that the total length of all given segments doesn't exceed 10^5.

Output

If there is no path between the initial and final position along allowed cells, print -1.

Otherwise print a single integer — the minimum number of moves the king needs to get from the initial position to the final one.

Sample test(s)

input

```
5 7 6 11
3
5 3 8
6 7 11
5 2 5
```

output

```
4
```

input

```
3 4 3 10
3
3 1 4
4 5 9
3 10 10
```

output

```
6
```

input

```
1 1 2 10
2
1 1 3
2 6 10
```

output

```
-1
```

D. Dispute

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Valera has n counters numbered from 1 to n. Some of them are connected by wires, and each of the counters has a special button.

Initially, all the counters contain number 0. When you press a button on a certain counter, the value it has increases by one. Also, the values recorded in all the counters, directly connected to it by a wire, increase by one.

Valera and Ignat started having a dispute, the dispute is as follows. Ignat thought of a sequence of n integers $a_1, a_2, ..., a_n$. Valera should choose some set of distinct counters and press buttons on each of them exactly once (on other counters the buttons won't be pressed). If after that there is a counter with the number i, which has value a_i, then Valera loses the dispute, otherwise he wins the dispute.

Help Valera to determine on which counters he needs to press a button to win the dispute.

Input

The first line contains two space-separated integers n and m $(1 \le n, m \le 10^5)$, that denote the number of counters Valera has and the number of pairs of counters connected by wires.

Each of the following m lines contains two space-separated integers u_i and v_i $(1 \le u_i, v_i \le n, u_i \ne v_i)$, that mean that counters with numbers u_i and v_i are connected by a wire. It is guaranteed that each pair of connected counters occurs exactly once in the input.

The last line contains n space-separated integers $a_1, a_2, ..., a_n$ $(0 \le a_i \le 10^5)$, where a_i is the value that Ignat choose for the i-th counter.

Output

If Valera can't win the dispute print in the first line -1.

Otherwise, print in the first line integer k $(0 \le k \le n)$. In the second line print k distinct space-separated integers — the numbers of the counters, where Valera should push buttons to win the dispute, in arbitrary order.

If there exists multiple answers, you are allowed to print any of them.

Sample test(s)

input

```
5 5
2 3
4 1
1 5
5 3
2 1
1 1 2 0 2
```

output

```
2
1 2
```

input

```
4 2
1 2
3 4
0 0 0 0
```

output

```
3
1 3 4
```

E. XOR on Segment

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You've got an array a, consisting of n integers $a_1, a_2, ..., a_n$. You are allowed to perform two operations on this array:

1. Calculate the sum of current array elements on the segment $[l, r]$, that is, count value $a_l + a_{l+1} + ... + a_r$.
2. Apply the xor operation with a given number x to each array element on the segment $[l, r]$, that is, execute $a_l = a_l \oplus x, a_{l+1} = a_{l+1} \oplus x, \ldots, a_r = a_r \oplus x$. This operation changes exactly $r - l + 1$ array elements.

Expression $x \oplus y$ means applying bitwise xor operation to numbers x and y. The given operation exists in all modern programming languages, for example in language *C++* and *Java* it is marked as "^", in *Pascal* — as "xor".

You've got a list of m operations of the indicated type. Your task is to perform all given operations, for each sum query you should print the result you get.

Input

The first line contains integer n ($1 \le n \le 10^5$) — the size of the array. The second line contains space-separated integers $a_1, a_2, ..., a_n$ ($0 \le a_i \le 10^6$) — the original array.

The third line contains integer m ($1 \le m \le 5 \cdot 10^4$) — the number of operations with the array. The i-th of the following m lines first contains an integer t_i ($1 \le t_i \le 2$) — the type of the i-th query. If $t_i = 1$, then this is the query of the sum, if $t_i = 2$, then this is the query to change array elements. If the i-th operation is of type 1, then next follow two integers l_i, r_i ($1 \le l_i \le r_i \le n$). If the i-th operation is of type 2, then next follow three integers l_i, r_i, x_i ($1 \le l_i \le r_i \le n$, $1 \le x_i \le 10^6$). The numbers on the lines are separated by single spaces.

Output

For each query of type 1 print in a single line the sum of numbers on the given segment. Print the answers to the queries in the order in which the queries go in the input.

Please, do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams, or the `%I64d` specifier.

Sample test(s)

input

```
5
4 10 3 13 7
8
1 2 4
2 1 3 3
1 2 4
1 3 3
2 2 5 5
1 1 5
2 1 2 10
1 2 3
```

output

```
26
22
0
34
11
```

input

```
6
4 7 4 0 7 3
5
2 2 3 8
1 1 5
2 3 5 1
2 4 5 6
1 2 3
```

output

```
38
28
```