Codeforces Round #626 (Div. 1, based on Moscow Open Olympiad in Informatics)

A. Unusual Competitions

time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

A bracketed sequence is called correct (regular) if by inserting "+" and "1" you can get a well-formed mathematical expression from it. For example, sequences "(())()", "()" and "(()(()))" are correct, while ")(", "(()" and "(()))(" are not.

The teacher gave Dmitry's class a very strange task — she asked every student to come up with a sequence of arbitrary length, consisting only of opening and closing brackets. After that all the students took turns naming the sequences they had invented. When Dima's turn came, he suddenly realized that all his classmates got the correct bracketed sequence, and whether he got the correct bracketed sequence, he did not know.

Dima suspects now that he simply missed the word "correct" in the task statement, so now he wants to save the situation by modifying his sequence slightly. More precisely, he can **the arbitrary number of times** (possibly zero) perform the *reorder* operation.

The reorder operation consists of choosing an arbitrary consecutive subsegment (substring) of the sequence and then reordering all the characters in it in an arbitrary way. Such operation takes l nanoseconds, where l is the length of the subsegment being reordered. It's easy to see that reorder operation doesn't change the number of opening and closing brackets. For example for "))((" he can choose the substring ")(" and do reorder ")()(" (this operation will take 2 nanoseconds).

Since Dima will soon have to answer, he wants to make his sequence correct as fast as possible. Help him to do this, or determine that it's impossible.

Input

The first line contains a single integer n $(1 \le n \le 10^6)$ — the length of Dima's sequence.

The second line contains string of length n, consisting of characters "(" and ")" only.

Output

Print a single integer — the minimum number of nanoseconds to make the sequence correct or "-1" if it is impossible to do so.

Examples

input

```
8
))((())(
```

output

```
6
```

input

```
3
(()
```

output

```
-1
```

Note

In the first example we can firstly reorder the segment from first to the fourth character, replacing it with "()()", the whole sequence will be "()()())(". And then reorder the segment from the seventh to eighth character, replacing it with "()". In the end the sequence will be "()()()()", while the total time spent is $4 + 2 = 6$ nanoseconds.

B. Present

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Catherine received an array of integers as a gift for March 8. Eventually she grew bored with it, and she started calculated various useless characteristics for it. She succeeded to do it for each one she came up with. But when she came up with another one — xor of all pairwise sums of elements in the array, she realized that she couldn't compute it for a very large array, thus she asked for your help. Can you do it? Formally, you need to compute

$$(a_1 + a_2) \oplus (a_1 + a_3) \oplus \ldots \oplus (a_1 + a_n)$$
$$\oplus (a_2 + a_3) \oplus \ldots \oplus (a_2 + a_n)$$
$$\ldots$$
$$\oplus (a_{n-1} + a_n)$$

Here $x \oplus y$ is a bitwise XOR operation (i.e. x ^ y in many modern programming languages). You can read about it in Wikipedia: https://en.wikipedia.org/wiki/Exclusive_or#Bitwise_operation.

Input

The first line contains a single integer n ($2 \leq n \leq 400\,000$) — the number of integers in the array.

The second line contains integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^7$).

Output

Print a single integer — xor of all pairwise sums of integers in the given array.

Examples

input
```
2
1 2
```
output
```
3
```

input
```
3
1 2 3
```
output
```
2
```

Note

In the first sample case there is only one sum $1 + 2 = 3$.

In the second sample case there are three sums: $1 + 2 = 3$, $1 + 3 = 4$, $2 + 3 = 5$. In binary they are represented as $011_2 \oplus 100_2 \oplus 101_2 = 010_2$, thus the answer is 2.

$\oplus$ is the bitwise xor operation. To define $x \oplus y$, consider binary representations of integers x and y. We put the i-th bit of the result to be 1 when exactly one of the i-th bits of x and y is 1. Otherwise, the i-th bit of the result is put to be 0. For example, $0101_2 \oplus 0011_2 = 0110_2$.

C. Instant Noodles

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Wu got hungry after an intense training session, and came to a nearby store to buy his favourite instant noodles. After Wu paid for his purchase, the cashier gave him an interesting task.

You are given a bipartite graph with positive integers in all vertices of the **right** half. For a subset S of vertices of the **left** half we define $N(S)$ as the set of all vertices of the right half adjacent to at least one vertex in S, and $f(S)$ as the sum of all numbers in vertices of $N(S)$. Find the greatest common divisor of $f(S)$ for all possible non-empty subsets S (assume that GCD of empty set is 0).

Wu is too tired after his training to solve this problem. Help him!

Input

The first line contains a single integer t ($1 \leq t \leq 500\,000$) — the number of test cases in the given test set. Test case descriptions follow.

The first line of each case description contains two integers n and m ($1 \leq n, m \leq 500\,000$) — the number of vertices in either half of the graph, and the number of edges respectively.

The second line contains n integers c_i ($1 \leq c_i \leq 10^{12}$). The i-th number describes the integer in the vertex i of the right half of the graph.

Each of the following m lines contains a pair of integers u_i and v_i ($1 \leq u_i, v_i \leq n$), describing an edge between the vertex u_i of the left half and the vertex v_i of the right half. It is guaranteed that the graph does not contain multiple edges.

Test case descriptions are separated with empty lines. The total value of n across all test cases does not exceed $500\,000$, and the total value of m across all test cases does not exceed $500\,000$ as well.

Output

For each test case print a single integer — the required greatest common divisor.

Example

input

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

3 4
1 1 1
1 1
1 2
2 2
2 3

4 7
36 31 96 29
1 2
1 3
1 4
2 2
2 4
3 1
4 3
```

output

```
2
1
12
```

Note

The greatest common divisor of a set of integers is the largest integer g such that all elements of the set are divisible by g.

In the first sample case vertices of the left half and vertices of the right half are pairwise connected, and $f(S)$ for any non-empty subset is 2, thus the greatest common divisor of these values if also equal to 2.

In the second sample case the subset $\{1\}$ in the left half is connected to vertices $\{1, 2\}$ of the right half, with the sum of numbers equal to 2, and the subset $\{1, 2\}$ in the left half is connected to vertices $\{1, 2, 3\}$ of the right half, with the sum of numbers equal to 3. Thus, $f(\{1\}) = 2$, $f(\{1, 2\}) = 3$, which means that the greatest common divisor of all values of $f(S)$ is 1.

D. Reality Show

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

A popular reality show is recruiting a new cast for the third season! n candidates numbered from 1 to n have been interviewed. The candidate i has aggressiveness level l_i, and recruiting this candidate will cost the show s_i roubles.

The show host reviewes applications of all candidates from $i = 1$ to $i = n$ by increasing of their indices, and for each of them she decides whether to recruit this candidate or not. If aggressiveness level of the candidate i is strictly higher than that of any **already accepted** candidates, then the candidate i will definitely be rejected. Otherwise the host may accept or reject this candidate at her own discretion. The host wants to choose the cast so that to maximize the total *profit*.

The show makes revenue as follows. For each aggressiveness level v a corresponding profitability value c_v is specified, which can be positive as well as negative. All recruited participants enter the stage one by one by increasing of their indices. When the participant i enters the stage, events proceed as follows:

- The show makes c_{l_i} roubles, where l_i is initial aggressiveness level of the participant i.
- If there are two participants with the same aggressiveness level on stage, they immediately start a fight. The outcome of this is:
    - the defeated participant is hospitalized and leaves the show.
    - aggressiveness level of the victorious participant is increased by one, and the show makes c_t roubles, where t is the new aggressiveness level.
- The fights continue until all participants on stage have distinct aggressiveness levels.

It is allowed to select an empty set of participants (to choose neither of the candidates).

The host wants to recruit the cast so that the total profit is maximized. The profit is calculated as the total revenue from the events on stage, less the total expenses to recruit all accepted participants (that is, their total s_i). Help the host to make the show as profitable as possible.

Input

The first line contains two integers n and m ($1 \le n, m \le 2000$) — the number of candidates and an upper bound for initial aggressiveness levels.

The second line contains n integers l_i ($1 \le l_i \le m$) — initial aggressiveness levels of all candidates.

The third line contains n integers s_i ($0 \le s_i \le 5000$) — the costs (in roubles) to recruit each of the candidates.

The fourth line contains $n + m$ integers c_i ($|c_i| \le 5000$) — profitability for each aggrressiveness level.

It is guaranteed that aggressiveness level of any participant can never exceed $n + m$ under given conditions.

Output

Print a single integer — the largest profit of the show.

Examples

input

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

output

```
6
```

input

```
2 2
1 2
0 0
2 1 -100 -100
```

output

```
2
```

input

```
5 4
4 3 2 1 1
0 2 6 7 4
12 12 12 6 -3 -5 3 10 -4
```

output

```
62
```

Note

In the first sample case it is optimal to recruit candidates $1, 2, 3, 5$. Then the show will pay $1 + 2 + 1 + 1 = 5$ roubles for recruitment. The events on stage will proceed as follows:

- a participant with aggressiveness level 4 enters the stage, the show makes 4 roubles;
- a participant with aggressiveness level 3 enters the stage, the show makes 3 roubles;
- a participant with aggressiveness level 1 enters the stage, the show makes 1 rouble;
- a participant with aggressiveness level 1 enters the stage, the show makes 1 roubles, a fight starts. One of the participants leaves, the other one increases his aggressiveness level to 2. The show will make extra 2 roubles for this.

Total revenue of the show will be $4 + 3 + 1 + 1 + 2 = 11$ roubles, and the profit is $11 - 5 = 6$ roubles.

In the second sample case it is impossible to recruit both candidates since the second one has higher aggressiveness, thus it is better to recruit the candidate 1.

E. Median Mountain Range

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Berland — is a huge country with diverse geography. One of the most famous natural attractions of Berland is the "Median mountain range". This mountain range is n mountain peaks, located on one straight line and numbered in order of 1 to n. The height of the i-th mountain top is a_i.

"Median mountain range" is famous for the so called *alignment of mountain peaks* happening to it every day. At the moment of alignment simultaneously for each mountain from 2 to $n - 1$ its height becomes equal to the median height among it and two neighboring mountains. Formally, if before the alignment the heights were equal b_i, then after the alignment new heights a_i are as follows: $a_1 = b_1$, $a_n = b_n$ and for all i from 2 to $n - 1$ $a_i = \texttt{median}(b_{i-1}, b_i, b_{i+1})$. The median of three integers is the second largest number among them. For example, $\texttt{median}(5, 1, 2) = 2$, and $\texttt{median}(4, 2, 4) = 4$.

Recently, Berland scientists have proved that whatever are the current heights of the mountains, the alignment process will stabilize sooner or later, i.e. at some point the altitude of the mountains won't changing after the alignment any more. The government of Berland wants to understand how soon it will happen, i.e. to find the value of c — how many alignments will occur, which will change the height of at least one mountain. Also, the government of Berland needs to determine the heights of the mountains after c

alignments, that is, find out what heights of the mountains stay forever. Help scientists solve this important problem!

Input

The first line contains integers n ($1 \le n \le 500\,000$) — the number of mountains.

The second line contains integers $a_1, a_2, a_3, \ldots, a_n$ ($1 \le a_i \le 10^9$) — current heights of the mountains.

Output

In the first line print c — the number of alignments, which change the height of at least one mountain.

In the second line print n integers — the final heights of the mountains after c alignments.

Examples

input
```
5
1 2 1 2 1
```
output
```
2
1 1 1 1 1
```

input
```
6
1 3 2 5 4 6
```
output
```
1
1 2 3 4 5 6
```

input
```
6
1 1 2 2 1 1
```
output
```
0
1 1 2 2 1 1
```

Note

In the first example, the heights of the mountains at index 1 and 5 never change. Since the median of 1, 2, 1 is 1, the second and the fourth mountains will have height 1 after the first alignment, and since the median of 2, 1, 2 is 2, the third mountain will have height 2 after the first alignment. This way, after one alignment the heights are 1, 1, 2, 1, 1. After the second alignment the heights change into 1, 1, 1, 1, 1 and never change from now on, so there are only 2 alignments changing the mountain heights.

In the third examples the alignment doesn't change any mountain height, so the number of alignments changing any height is 0.

F. Assigning Fares

time limit per test: 6 seconds

memory limit per test: 512 megabytes

input: standard input

output: standard output

Mayor of city M. decided to launch several new metro lines during 2020. Since the city has a very limited budget, it was decided not to dig new tunnels but to use the existing underground network.

The tunnel system of the city M. consists of n metro stations. The stations are connected with $n-1$ bidirectional tunnels. Between every two stations v and u there is exactly one simple path. Each metro line the mayor wants to create is a simple path between stations a_i and b_i. Metro lines can intersects freely, that is, they can share common stations or even common tunnels. However, it's not yet decided which of two directions each line will go. More precisely, between the stations a_i and b_i the trains will go either from a_i to b_i, or from b_i to a_i, but not simultaneously.

The city M uses complicated faring rules. Each station is assigned with a positive integer c_i — the fare zone of the station. The cost of travel from v to u is defined as $c_u - c_v$ roubles. Of course, such travel only allowed in case there is a metro line, the trains on which go from v to u. Mayor doesn't want to have any travels with a negative cost, so it was decided to assign directions of metro lines and station fare zones in such a way, that fare zones are strictly increasing during any travel on any metro line.

Mayor wants firstly assign each station a fare zone and then choose a lines direction, such that all fare zones are increasing along any line. In connection with the approaching celebration of the day of the city, the mayor wants to assign fare zones so that the maximum c_i will be as low as possible. Please help mayor to design a new assignment or determine if it's impossible to do. Please note that you only need to assign the fare zones optimally, you don't need to print lines' directions. This way, you solution will be considered correct if there will be a way to assign directions of every metro line, so that the fare zones will be strictly increasing along any movement of the trains.

Input

The first line contains an integers n, m ($2 \le n, \le 500\,000,\ 1 \le m \le 500\,000$) — the number of stations in the city and the number of metro lines.

Each of the following $n - 1$ lines describes another metro tunnel. Each tunnel is described with integers v_i, u_i ($1 \le v_i,\ u_i \le n$, $v_i \ne u_i$). It's guaranteed, that there is exactly one simple path between any two stations.

Each of the following m lines describes another metro line. Each line is described with integers a_i, b_i ($1 \le a_i,\ b_i \le n$, $a_i \ne b_i$).

Output

In the first line print integer k — the maximum fare zone used.

In the next line print integers $c_1, c_2, \dots, c_n$ ($1 \le c_i \le k$) — stations' fare zones.

In case there are several possible answers, print any of them. In case it's impossible to assign fare zones, print "-1".

Examples

input

```
3 1
1 2
2 3
1 3
```

output

```
3
1 2 3
```

input

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

output

```
-1
```

input

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

output

```
-1
```

Note

In the first example, line $1 \to 3$ goes through the stations 1, 2, 3 in this order. In this order the fare zones of the stations are increasing. Since this line has 3 stations, at least three fare zones are needed. So the answer 1, 2, 3 is optimal.

In the second example, it's impossible to assign fare zones to be consistent with a metro lines.