A. Prime Minister

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Alice is the leader of the State Refactoring Party, and she is about to become the prime minister.

The elections have just taken place. There are n parties, numbered from 1 to n. The i-th party has received a_i seats in the parliament.

Alice's party has number 1. In order to become the prime minister, she needs to build a coalition, consisting of her party and possibly some other parties. There are two conditions she needs to fulfil:

- The total number of seats of all parties in the coalition must be a strict majority of all the seats, i.e. it must have **strictly more than half** of the seats. For example, if the parliament has 200 (or 201) seats, then the majority is 101 or more seats.
- Alice's party must have **at least** 2 **times more** seats than any other party in the coalition. For example, to invite a party with 50 seats, Alice's party must have at least 100 seats.

For example, if $n = 4$ and $a = [51, 25, 99, 25]$ (note that Alice'a party has 51 seats), then the following set $[a_1 = 51, a_2 = 25, a_4 = 25]$ can create a coalition since both conditions will be satisfied. However, the following sets will not create a coalition:

- $[a_2 = 25, a_3 = 99, a_4 = 25]$ since Alice's party is not there;
- $[a_1 = 51, a_2 = 25]$ since coalition should have a strict majority;
- $[a_1 = 51, a_2 = 25, a_3 = 99]$ since Alice's party should have **at least** 2 **times more** seats than any other party in the coalition.

Alice does not have to minimise the number of parties in a coalition. If she wants, she can invite as many parties as she wants (as long as the conditions are satisfied). If Alice's party has enough people to create a coalition on her own, she can invite no parties.

Note that Alice can either invite a party as a whole or not at all. It is **not possible** to invite only some of the deputies (seats) from another party. In other words, if Alice invites a party, she invites **all** its deputies.

Find and print any suitable coalition.

Input

The first line contains a single integer n ($2 \le n \le 100$) — the number of parties.

The second line contains n space separated integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 100$) — the number of seats the i-th party has.

Output

If no coalition satisfying both conditions is possible, output a single line with an integer 0.

Otherwise, suppose there are k ($1 \le k \le n$) parties in the coalition (Alice does not have to minimise the number of parties in a coalition), and their indices are $c_1, c_2, \ldots, c_k$ ($1 \le c_i \le n$). Output two lines, first containing the integer k, and the second the space-separated indices $c_1, c_2, \ldots, c_k$.

You may print the parties in any order. Alice's party (number 1) must be on that list. If there are multiple solutions, you may print any of them.

Examples

input
3 100 50 50
output
2 1 2

input
3 80 60 60
output
0

input
2 6 5
output
1 1

input
4 51 25 99 25
output
3 1 2 4

Note

In the first example, Alice picks the second party. Note that she can also pick the third party or both of them. However, she cannot become prime minister without any of them, because 100 is not a strict majority out of 200.

In the second example, there is no way of building a majority, as both other parties are too large to become a coalition partner.

In the third example, Alice already has the majority.

The fourth example is described in the problem statement.

B. WOW Factor

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Recall that string a is a subsequence of a string b if a can be obtained from b by deletion of several (possibly zero or all) characters. For example, for the string a="wowwo", the following strings are subsequences: "wowwo", "wowo", "oo", "wow", "", and others, but the following are not subsequences: "owoo", "owwwo", "ooo".

The *wow factor* of a string is the number of its subsequences equal to the word "wow". Bob wants to write a string that has a large *wow factor*. However, the "w" key on his keyboard is broken, so he types two "v"s instead.

Little did he realise that he may have introduced more "w"s than he thought. Consider for instance the string "ww". Bob would type it as "vvvv", but this string actually contains three occurrences of "w":

- "**vv**vv"
- "v**vv**v"
- "vv**vv**"

For example, the *wow factor* of the word "vvvovvv" equals to four because there are four *wow*s:

- "**vv**v**ovv**v"
- "**vv**v**o**v**vv**"
- "v**vvovv**v"
- "v**vvo**v**vv**"

Note that the subsequence "**v**v**vovv**v" does not count towards the *wow factor*, as the "v"s have to be consecutive.

For a given string s, compute and output its *wow factor*. Note that it is **not** guaranteed that it is possible to get s from another string replacing "w" with "vv". For example, s can be equal to "vov".

Input

The input contains a single non-empty string s, consisting only of characters "v" and "o". The length of s is at most 10^6.

Output

Output a single integer, the *wow factor* of s.

Examples

input
vvvovvv
output
4

input
vvovooovovvovoovoovvvvovovvvov
output
100

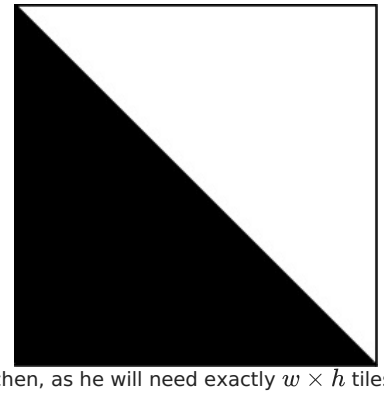
Note

The first example is explained in the legend.

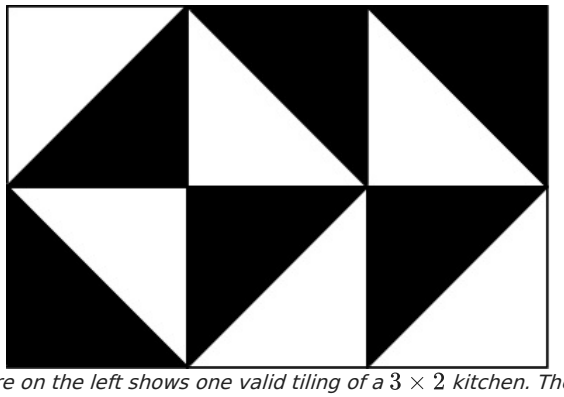
C. Tiles

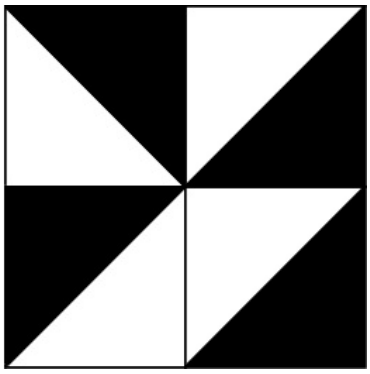
time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Bob is decorating his kitchen, more precisely, the floor. He has found a prime candidate for the tiles he will use. They come in a simple form factor — a square tile that is diagonally split into white and black part as depicted in the figure below.



The dimension of this tile is perfect for this kitchen, as he will need exactly $w \times h$ tiles without any scraps. That is, the width of the kitchen is w tiles, and the height is h tiles. As each tile can be rotated in one of four ways, he still needs to decide on how exactly he will tile the floor. There is a single aesthetic criterion that he wants to fulfil: two adjacent tiles must not share a colour on the edge — i.e. one of the tiles must have a white colour on the shared border, and the second one must be black.





The picture on the left shows one valid tiling of a 3×2 kitchen. The picture on the right shows an invalid arrangement, as the bottom two tiles touch with their white parts.

Find the number of possible tilings. As this number may be large, output its remainder when divided by 998244353 (a prime number).

Input

The only line contains two space separated integers w, h ($1 \leq w, h \leq 1\,000$) — the width and height of the kitchen, measured in tiles.

Output

Output a single integer n — the remainder of the number of tilings when divided by 998244353.

Examples

input
2 2
output
16

input
2 4
output
64

D. Prime Graph

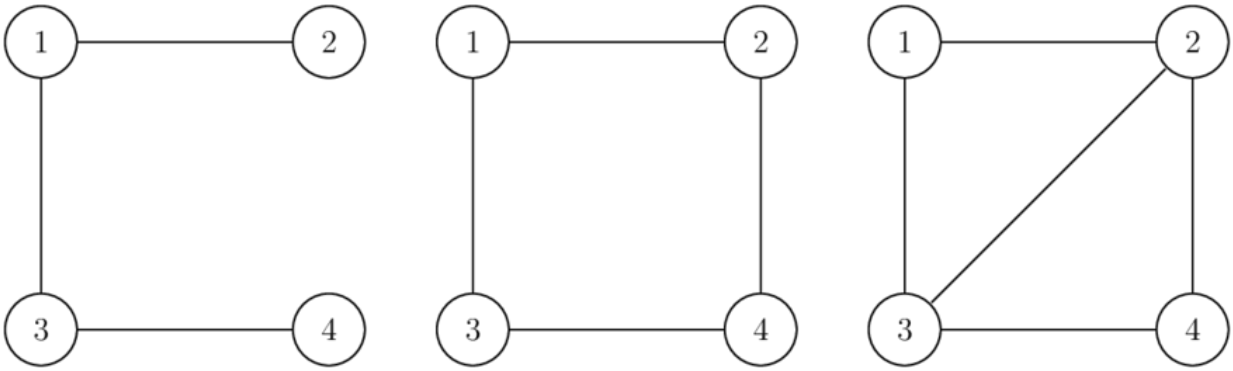
time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Every person likes prime numbers. Alice is a person, thus she also shares the love for them. Bob wanted to give her an affectionate gift but couldn't think of anything inventive. Hence, he will be giving her a graph. How original, Bob! Alice will surely be *thrilled*!

When building the graph, he needs four conditions to be satisfied:

- It must be a simple undirected graph, i.e. without multiple (parallel) edges and self-loops.
- The number of vertices must be exactly n — a number he selected. This number is not necessarily prime.
- The total number of edges must be prime.
- The degree (i.e. the number of edges connected to the vertex) of each vertex must be prime.

Below is an example for $n = 4$. The first graph (left one) is invalid as the degree of vertex 2 (and 4) equals to 1, which is not prime. The second graph (middle one) is invalid as the total number of edges is 4, which is not a prime number. The third graph (right one) is a valid answer for $n = 4$.



Note that the graph can be disconnected.

Please help Bob to find any such graph!

Input

The input consists of a single integer n ($3 \leq n \leq 1\,000$) — the number of vertices.

Output

If there is no graph satisfying the conditions, print a single line containing the integer -1.

Otherwise, first print a line containing a prime number m ($2 \leq m \leq \frac{n(n-1)}{2}$) — the number of edges in the graph. Then, print m lines, the i-th of which containing two integers u_i, v_i ($1 \leq u_i, v_i \leq n$) — meaning that there is an edge between vertices u_i and v_i. The degree of each vertex must be prime. There must be no multiple (parallel) edges or self-loops.

If there are multiple solutions, you may print any of them.

Note that the graph can be disconnected.

Examples

input
4

output
5 1 2 1 3 2 3 2 4 3 4

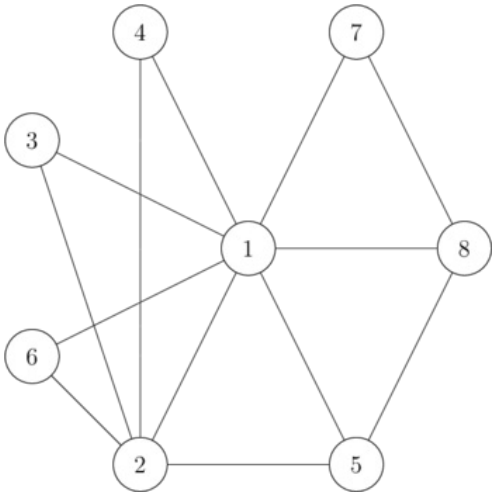
input
8

output
13 1 2 1 3 2 3 1 4 2 4 1 5 2 5 1 6 2 6 1 7 1 8 5 8 7 8

Note

The first example was described in the statement.

In the second example, the degrees of vertices are $[7, 5, 2, 2, 3, 2, 2, 3]$. Each of these numbers is prime. Additionally, the number of edges, 13, is also a prime number, hence both conditions are satisfied.



E. Archaeology

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Alice bought a Congo Prime Video subscription and was watching a documentary on the archaeological findings from Factor's Island on Loch Katrine in Scotland. The archaeologists found a book whose age and origin are unknown. Perhaps Alice can make some sense of it?

The book contains a single string of characters "a", "b" and "c". It has been pointed out that no two consecutive characters are the same. It has also been conjectured that the string contains an unusually long subsequence that reads the same from both sides.

Help Alice verify this by finding such subsequence that contains at least half of the characters of the original string, rounded down. Note that you don't have to maximise the length of it.

A string a is a subsequence of a string b if a can be obtained from b by deletion of several (possibly, zero or all) characters.

Input

The input consists of a single string s ($2 \leq |s| \leq 10^6$). The string s consists only of characters "a", "b", "c". It is guaranteed that no two consecutive characters are equal.

Output

Output a palindrome t that is a subsequence of s and $|t| \geq \lfloor \frac{|s|}{2} \rfloor$.

If there are multiple solutions, you may print any of them. You don't have to maximise the length of t.

If there are no solutions, output a string "IMPOSSIBLE" (quotes for clarity).

Examples

input
cacbac
output
aba

input
abc
output
a

input
cbacacacbcbababacbcb
output
cbaaacbcaaabc

Note

In the first example, other valid answers include "cacac", "caac", "aca" and "ccc".

F1. Short Colorful Strip

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is the first subtask of problem F. The only differences between this and the second subtask are the constraints on the value of m and the time limit. You need to solve both subtasks in order to hack this one.

There are $n+1$ distinct colours in the universe, numbered 0 through n. There is a strip of paper m centimetres long initially painted with colour 0.

Alice took a brush and painted the strip using the following process. For each i from 1 to n, **in this order**, she picks two integers $0 \le a_i < b_i \le m$, such that the segment $[a_i, b_i]$ is currently painted with a **single** colour, and repaints it with colour i.

Alice chose the segments in such a way that each centimetre is now painted in some colour other than 0. Formally, the segment $[i-1, i]$ is painted with colour c_i ($c_i \neq 0$). Every colour other than 0 is visible on the strip.

Count the number of different pairs of sequences $\{a_i\}_{i=1}^{n}$, $\{b_i\}_{i=1}^{n}$ that result in this configuration.

Since this number may be large, output it modulo 998244353.

Input

The first line contains a two integers n, m ($1 \le n \le 500$, $n = m$) — the number of colours excluding the colour 0 and the length of the paper, respectively.

The second line contains m space separated integers $c_1, c_2, \ldots, c_m$ ($1 \le c_i \le n$) — the colour visible on the segment $[i-1, i]$ after the process ends. It is guaranteed that for all j between 1 and n there is an index k such that $c_k = j$.

Note that since in this subtask $n = m$, this means that c is a permutation of integers 1 through n.

Output

Output a single integer — the number of ways Alice can perform the painting, modulo 998244353.

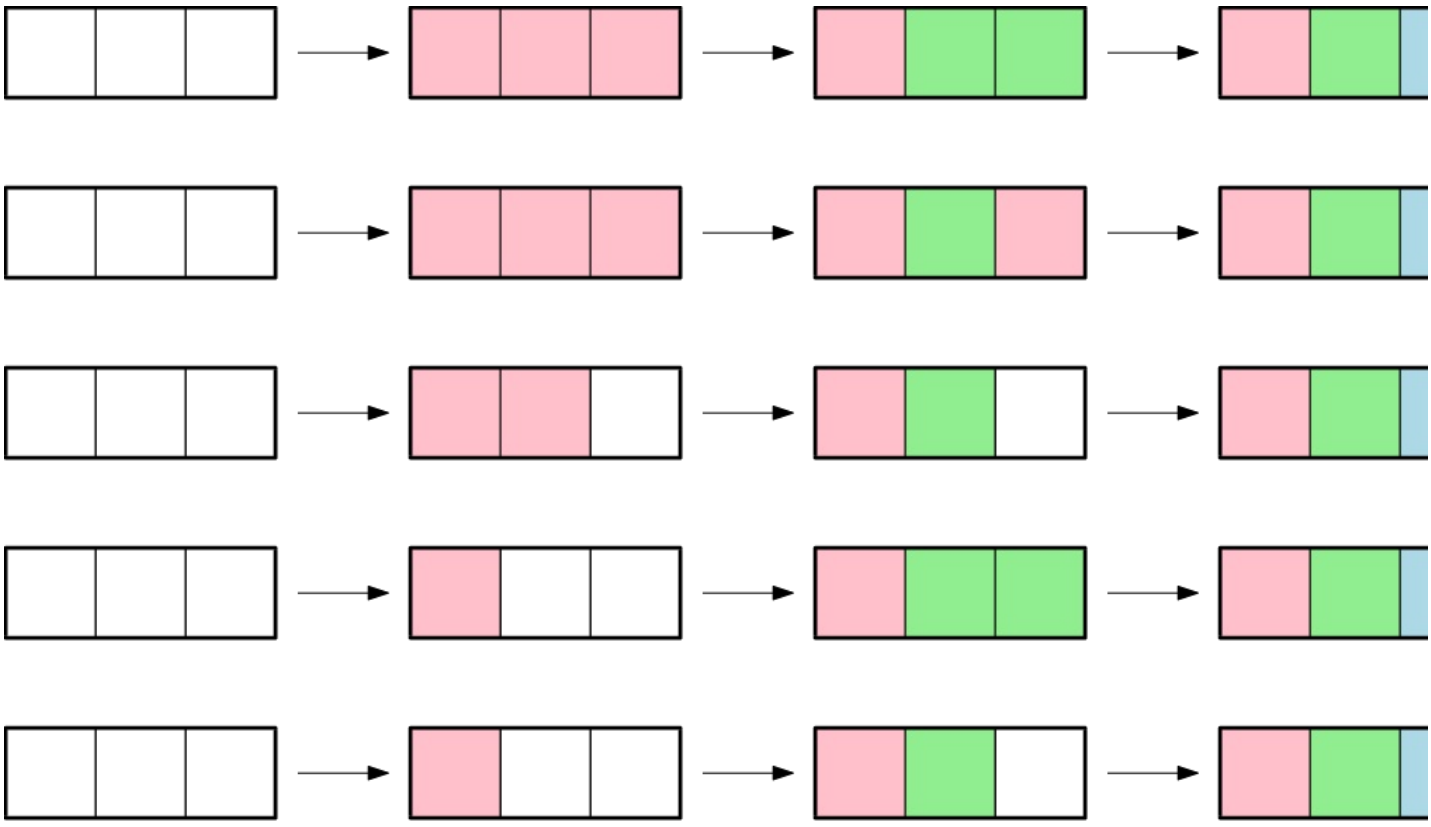
Examples

input
3 3 1 2 3
output
5

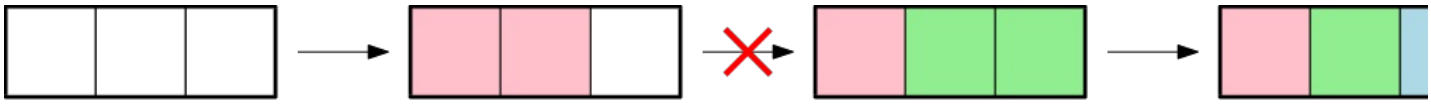
input
7 7 4 5 1 6 2 3 7
output
165

Note

In the first example, there are 5 ways, all depicted in the figure below. Here, 0 is white, 1 is red, 2 is green and 3 is blue.



Below is an example of a painting process that is not valid, as in the second step the segment $1 \;\; 3$ is not single colour, and thus may not be repainted with colour 2.



time limit per test: 6 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is the second subtask of problem F. The only differences between this and the first subtask are the constraints on the value of m and the time limit. It is sufficient to solve this subtask in order to hack it, but you need to solve both subtasks in order to hack the first one.

There are $n + 1$ distinct colours in the universe, numbered 0 through n. There is a strip of paper m centimetres long initially painted with colour 0.

Alice took a brush and painted the strip using the following process. For each i from 1 to n, **in this order**, she picks two integers $0 \leq a_i < b_i \leq m$, such that the segment $[a_i, b_i]$ is currently painted with a **single** colour, and repaints it with colour i.

Alice chose the segments in such a way that each centimetre is now painted in some colour other than 0. Formally, the segment $[i - 1, i]$ is painted with colour c_i ($c_i \neq 0$). Every colour other than 0 is visible on the strip.

Count the number of different pairs of sequences $\{a_i\}_{i=1}^{n}$, $\{b_i\}_{i=1}^{n}$ that result in this configuration.

Since this number may be large, output it modulo 998244353.

Input

The first line contains a two integers n, m ($1 \leq n \leq 500$, $n \leq m \leq 10^6$) — the number of colours excluding the colour 0 and the length of the paper, respectively.

The second line contains m space separated integers $c_1, c_2, \ldots, c_m$ ($1 \leq c_i \leq n$) — the colour visible on the segment $[i - 1, i]$ after the process ends. It is guaranteed that for all j between 1 and n there is an index k such that $c_k = j$.

Output

Output a single integer — the number of ways Alice can perform the painting, modulo 998244353.

Examples

input
3 3 1 2 3
output
5

input
2 3 1 2 1
output
1

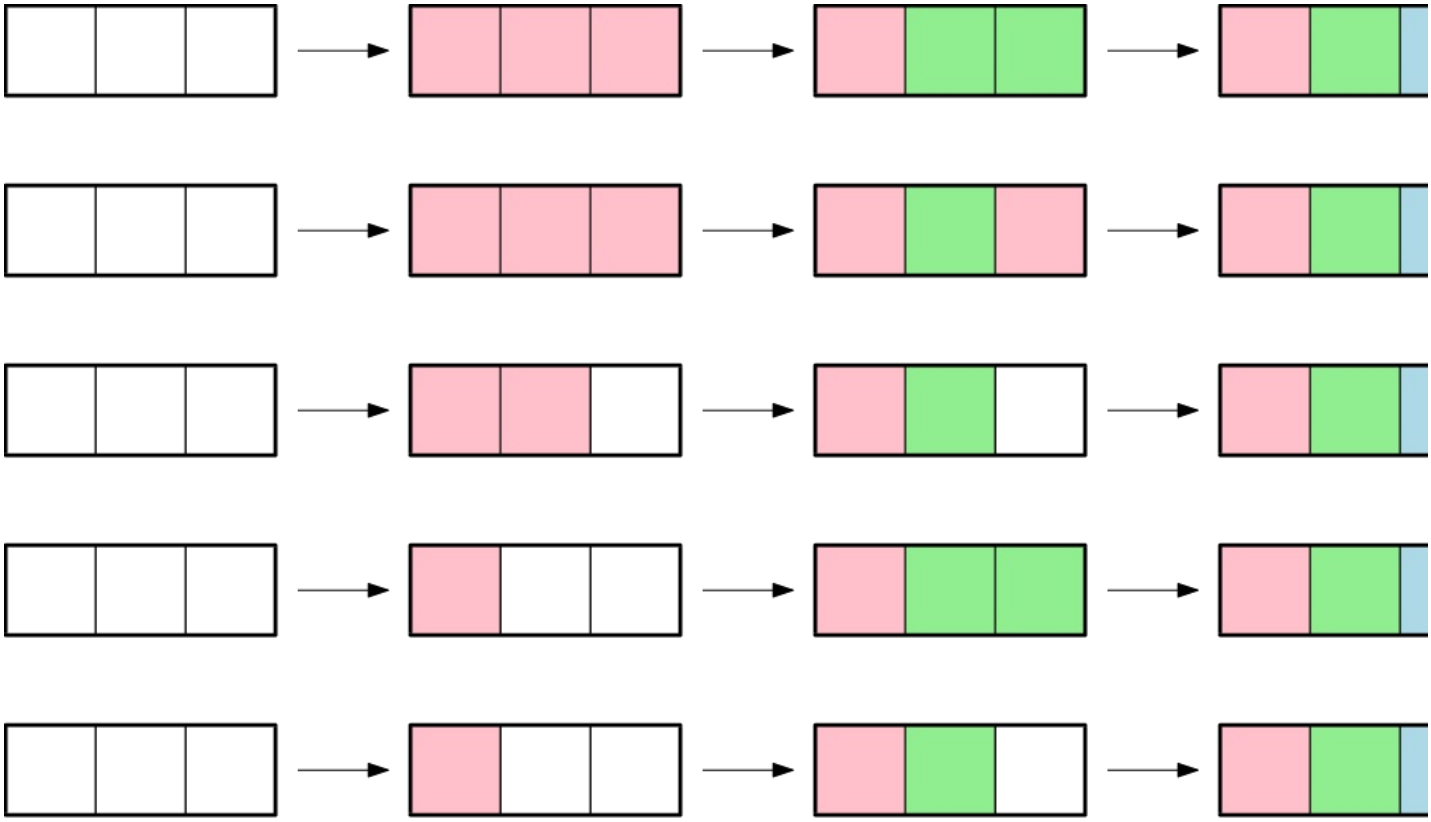
input
2 3 2 1 2
output
0

input
7 7 4 5 1 6 2 3 7
output
165

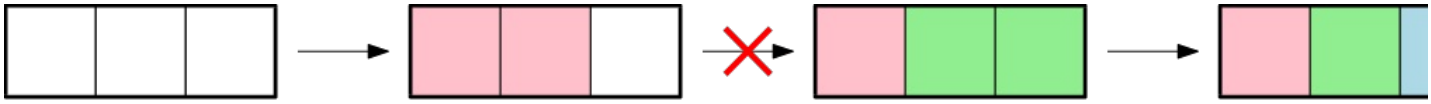
input
8 17 1 3 2 2 7 8 2 5 5 4 4 4 1 1 6 1 1
output
20

Note

In the first example, there are 5 ways, all depicted in the figure below. Here, 0 is white, 1 is red, 2 is green and 3 is blue.



Below is an example of a painting process that is not valid, as in the second step the segment 1 3 is not single colour, and thus may not be repainted with colour 2.



In the second example, Alice must first paint segment 0 3 with colour 1 and then segment 1 2 with colour 2.

G. The Awesomest Vertex

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a rooted tree on n vertices. The vertices are numbered from 1 to n; the root is the vertex number 1.

Each vertex has two integers associated with it: a_i and b_i. We denote the set of all ancestors of v (including v itself) by $R(v)$. The *awesomeness* of a vertex v is defined as

$$\left|\sum_{w \in R(v)} a_w\right| \cdot \left|\sum_{w \in R(v)} b_w\right|,$$

where $|x|$ denotes the absolute value of x.

Process q queries of one of the following forms:

- 1 v x — increase a_v by a positive integer x.
- 2 v — report the maximum *awesomeness* in the subtree of vertex v.

Input

The first line contains two integers n and q ($1 \le n \le 2 \cdot 10^5, 1 \le q \le 10^5$) — the number of vertices in the tree and the number of queries, respectively.

The second line contains $n-1$ integers $p_2, p_3, \ldots, p_n$ ($1 \le p_i < i$), where p_i means that there is an edge between vertices i and p_i.

The third line contains n integers $a_1, a_2, \ldots, a_n$ ($-5000 \le a_i \le 5000$), the initial values of a_i for each vertex.

The fourth line contains n integers $b_1, b_2, \ldots, b_n$ ($-5000 \le b_i \le 5000$), the values of b_i for each vertex.

Each of the next q lines describes a query. It has one of the following forms:

- 1 v x ($1 \le v \le n$, $1 \le x \le 5000$).
- 2 v ($1 \le v \le n$).

Output

For each query of the second type, print a single line with the maximum *awesomeness* in the respective subtree.

Example

input
5 6 1 1 2 2 10 -3 -7 -3 -10 10 3 9 3 6 2 1 2 2 1 2 6 2 1 1 2 5 2 1

output
100 91 169 240

Note

The initial *awesomeness* of the vertices is $[100, 91, 57, 64, 57]$. The most *awesome* vertex in the subtree of vertex 1 (the first query) is 1, and the most *awesome* vertex in the subtree of vertex 2 (the second query) is 2.

After the first update (the third query), the *awesomeness* changes to $[100, 169, 57, 160, 57]$ and thus the most *awesome* vertex in the whole tree (the fourth query) is now 2.

After the second update (the fifth query), the *awesomeness* becomes $[100, 234, 57, 240, 152]$, hence the most *awesome* vertex (the sixth query) is now 4.

H. Stock Exchange

time limit per test: 6 seconds
memory limit per test: 16 megabytes
input: standard input
output: standard output

Warning: This problem has an unusual memory limit!

Bob decided that he will not waste his prime years implementing GUI forms for a large corporation and instead will earn his supper on the Stock Exchange Reykjavik. The Stock Exchange Reykjavik is the only actual stock exchange in the world. The only type of transaction is to take a single share of stock x and exchange it for a single share of stock y, provided that the current price of share x is at least the current price of share y.

There are $2n$ stocks listed on the SER that are of interest to Bob, numbered from 1 to $2n$. Bob owns a single share of stocks 1 through n and would like to own a single share of each of $n + 1$ through $2n$ some time in the future.

Bob managed to forecast the price of each stock — in time $t \geq 0$, the stock i will cost $a_i \cdot \lfloor t \rfloor + b_i$. The time is currently $t = 0$. Help Bob find the earliest moment in time in which he can own a single share of each of $n + 1$ through $2n$, and the minimum number of stock exchanges he has to perform in order to do that.

You may assume that the Stock Exchange has an unlimited amount of each stock at any point in time.

Input

The first line contains a single integer n ($1 \leq n \leq 2200$) — the number stocks currently owned by Bob.

Each of the next $2n$ lines contains integers a_i and b_i ($0 \leq a_i, b_i \leq 10^9$), representing the stock price of stock i.

Output

If it is impossible for Bob to achieve his goal, output a single integer -1.

Otherwise, output two integers T and E, where T is the minimum time in which he can achieve his goal, and E is the minimum number of exchanges in which he can achieve his goal at time T.

Examples

input
1 3 10 1 16

output
3 1

input
2 3 0 2 1 1 10 1 11

output
6 3

input
1 42 42 47 47

output

```
-1
```

input

```
8
145 1729363
891 4194243
424 853731
869 1883440
843 556108
760 1538373
270 762781
986 2589382
610 99315884
591 95147193
687 99248788
65 95114537
481 99071279
293 98888998
83 99764577
769 96951171
```

output

```
434847 11
```

input

```
8
261 261639
92 123277
271 131614
320 154417
97 258799
246 17926
117 222490
110 39356
85 62864876
86 62781907
165 62761166
252 62828168
258 62794649
125 62817698
182 62651225
16 62856001
```

output

```
1010327 11
```

Note

In the first example, Bob simply waits until time $t = 3$, when both stocks cost exactly the same amount.

In the second example, the optimum strategy is to exchange stock 2 for stock 1 at time $t = 1$, then exchange one share of stock 1 for stock 3 at time $t = 5$ (where both cost 15) and then at time $t = 6$ exchange the second on for the stock number 4 (when they cost 18 and 17, respectively). Note that he can achieve his goal also with only two exchanges, but this would take total time of $t = 9$, when he would finally be able to exchange the share number 2 for the share number 3.

In the third example, Bob can never achieve his goal, as the second stock is always strictly more expensive than the first one.