Educational Codeforces Round 46 (Rated for Div. 2)

A. Codehorses T-shirts

2 seconds, 256 megabytes

Codehorses has just hosted the second Codehorses Cup. This year, the same as the previous one, organizers are giving T-shirts for the winners.

The valid sizes of T-shirts are either "M" or from 0 to 3 "X" followed by "S" or "L". For example, sizes "M", "XXS", "L", "XXXL" are valid and "XM", "Z", "XXXXL" are not.

There are n winners to the cup for both the previous year and the current year. Ksenia has a list with the T-shirt sizes printed for the last year cup and is yet to send the new list to the printing office.

Organizers want to distribute the prizes as soon as possible, so now Ksenia is required not to write the whole list from the scratch but just make some changes to the list of the previous year. In one second she can choose arbitrary position in any word and replace its character with some uppercase Latin letter. Ksenia can't remove or add letters in any of the words.

What is the minimal number of seconds Ksenia is required to spend to change the last year list to the current one?

The lists are unordered. That means, two lists are considered equal if and only if the number of occurrences of any string is the same in both lists.

Input

The first line contains one integer n ($1 \le n \le 100$) — the number of T-shirts.

The i-th of the next n lines contains a_i — the size of the i-th T-shirt of the list for the previous year.

The i-th of the next n lines contains b_i — the size of the i-th T-shirt of the list for the current year.

It is guaranteed that all the sizes in the input are valid. It is also guaranteed that Ksenia can produce list b from the list a.

Output

Print the minimal number of seconds Ksenia is required to spend to change the last year list to the current one. If the lists are already equal, print 0.

input

```
3
XS
XS
M
XL
S
XS
```

output

```
2
```

input

```
2
XXXL
XXL
XXL
XXXS
```

output

```
1
```

input

```
2
M
XS
XS
M
```

output

```
0
```

In the first example Ksenia can replace "M" with "S" and "S" in one of the occurrences of "XS" with "L".

In the second example Ksenia should replace "L" in "XXXL" with "S".

In the third example lists are equal.

B. Light It Up

1 second, 256 megabytes

Recently, you bought a brand new smart lamp with programming features. At first, you set up a schedule to the lamp. Every day it will turn power on at moment 0 and turn power off at moment M. Moreover, the lamp allows you to set a program of switching its state (states are "lights on" and "lights off"). Unfortunately, some program is already installed into the lamp.

The lamp allows only *good* programs. Good program can be represented as a non-empty array a, where $0 < a_1 < a_2 < \cdots < a_{|a|} < M$. All a_i must be integers. Of course, preinstalled program is a good program.

The lamp follows program a in next manner: at moment 0 turns power and light on. Then at moment a_i the lamp flips its state to opposite (if it was lit, it turns off, and vice versa). The state of the lamp flips instantly: for example, if you turn the light off at moment 1 and then do nothing, the total time when the lamp is lit will be 1. Finally, at moment M the lamp is turning its power off regardless of its state.

Since you are not among those people who read instructions, and you don't understand the language it's written in, you realize (after some testing) the only possible way to alter the preinstalled program. You can **insert at most one** element into the program a, so it still should be a *good* program after alteration. Insertion can be done between any pair of consecutive elements of a, or even at the begining or at the end of a.

Find such a way to alter the program that the total time when the lamp is lit is maximum possible. Maybe you should leave program untouched. If the lamp is lit from x till moment y, then its lit for $y - x$ units of time. Segments of time when the lamp is lit are summed up.

Input

First line contains two space separated integers n and M ($1 \le n \le 10^5$, $2 \le M \le 10^9$) — the length of program a and the moment when power turns off.

Second line contains n space separated integers $a_1, a_2, \ldots, a_n$ ($0 < a_1 < a_2 < \cdots < a_n < M$) — initially installed program a.

Output

Print the only integer — maximum possible total time when the lamp is lit.

input

```
3 10
4 6 7
```

output

```
8
```

input

```
2 12
1 10
```

output

```
9
```

input

```
2 7
3 4
```

output

```
6
```

In the first example, one of possible optimal solutions is to insert value $x = 3$ before a_1, so program will be $[3, 4, 6, 7]$ and time of lamp being lit equals $(3 - 0) + (6 - 4) + (10 - 7) = 8$. Other possible solution is to insert $x = 5$ in appropriate place.

In the second example, there is only one optimal solution: to insert $x = 2$ between a_1 and a_2. Program will become $[1, 2, 10]$, and answer will be $(1 - 0) + (10 - 2) = 9$.

In the third example, optimal answer is to leave program untouched, so answer will be $(3 - 0) + (7 - 4) = 6$.

C. Covered Points Count

3 seconds, 256 megabytes

You are given n segments on a coordinate line; each endpoint of every segment has integer coordinates. Some segments can degenerate to points. Segments can intersect with each other, be nested in each other or even coincide.

Your task is the following: for every $k \in [1..n]$, calculate the number of points with integer coordinates such that the number of segments that cover these points equals k. A segment with endpoints l_i and r_i covers point x if and only if $l_i \le x \le r_i$.

Input

The first line of the input contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of segments.

The next n lines contain segments. The i-th line contains a pair of integers l_i, r_i ($0 \le l_i \le r_i \le 10^{18}$) — the endpoints of the i-th segment.

Output

Print n space separated integers $cnt_1, cnt_2, \ldots, cnt_n$, where cnt_i is equal to the number of points such that the number of segments that cover these points equals to i.

input

```
3
0 3
1 3
3 8
```

output

```
6 2 1
```

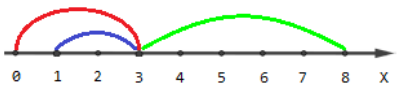
input

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

output

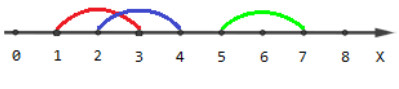
```
5 2 0
```

The picture describing the first example:



Points with coordinates $[0, 4, 5, 6, 7, 8]$ are covered by one segment, points $[1, 2]$ are covered by two segments and point $[3]$ is covered by three segments.

The picture describing the second example:



Points $[1, 4, 5, 6, 7]$ are covered by one segment, points $[2, 3]$ are covered by two segments and there are no points covered by three segments.

D. Yet Another Problem On a Subsequence

2 seconds, 256 megabytes

The sequence of integers $a_1, a_2, \ldots, a_k$ is called a good array if $a_1 = k - 1$ and $a_1 > 0$. For example, the sequences $[3, -1, 44, 0]$, $[1, -99]$ are good arrays, and the sequences $[3, 7, 8]$, $[2, 5, 4, 1]$, $[0]$ — are not.

A sequence of integers is called good if it can be divided into a positive number of good arrays. Each good array should be a subsegment of sequence and each element of the sequence should belong to exactly one array. For example, the sequences $[2, -3, 0, 1, 4]$, $[1, 2, 3, -3, -9, 4]$ are good, and the sequences $[2, -3, 0, 1]$, $[1, 2, 3, -3 - 9, 4, 1]$ — are not.

For a given sequence of numbers, count the number of its **subsequences** that are good sequences modulo **998244353**.

Input

The first line contains the number n ($1 \le n \le 10^3$) — the length of the initial sequence. The following line contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$) — the sequence itself.

Output

In the single line output one integer — the number of subsequences of the original sequence that are good sequences modulo **998244353**.

input

```
3
2 1 1
```

output

```
2
```

input

```
4
1 1 1 1
```

output

```
7
```

In the first test case, two good subsequences — $[a_1, a_2, a_3]$ and $[a_2, a_3]$.

In the second test case, seven good subsequences — $[a_1, a_2, a_3, a_4]$, $[a_1, a_2]$, $[a_1, a_3]$, $[a_1, a_4]$, $[a_2, a_3]$, $[a_2, a_4]$ and $[a_3, a_4]$.

E. We Need More Bosses

2 seconds, 256 megabytes

Your friend is developing a computer game. He has already decided how the game world should look like — it should consist of n locations connected by m **two-way** passages. The passages are designed in such a way that it should be possible to get from any location to any other location.

Of course, some passages should be guarded by the monsters (if you just can go everywhere without any difficulties, then it's not fun, right?). Some crucial passages will be guarded by really fearsome monsters, requiring the hero to prepare for battle and designing his own tactics of defeating them (commonly these kinds of monsters are called **bosses**). And your friend wants you to help him place these bosses.

The game will start in location s and end in location t, but these locations are not chosen yet. After choosing these locations, your friend will place a boss in each passage such that it is impossible to get from s to t without using this passage. Your friend wants to place as much bosses as possible (because more challenges means more fun, right?), so he asks you to help him determine the maximum possible number of bosses, considering that any location can be chosen as s or as t.

Input

The first line contains two integers n and m ($2 \le n \le 3 \cdot 10^5$, $n - 1 \le m \le 3 \cdot 10^5$) — the number of locations and passages, respectively.

Then m lines follow, each containing two integers x and y ($1 \le x, y \le n$, $x \ne y$) describing the endpoints of one of the passages.

It is guaranteed that there is no pair of locations directly connected by two or more passages, and that any location is reachable from any other location.

Output

Print one integer — the maximum number of bosses your friend can place, considering all possible choices for s and t.

input

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

output

```
2
```

input

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

output

```
3
```

F. One Occurrence

3 seconds, 768 megabytes

You are given an array a consisting of n integers, and q queries to it. i-th query is denoted by two integers l_i and r_i. For each query, you have to find **any** integer that occurs **exactly once** in the subarray of a from index l_i to index r_i (a subarray is a contiguous subsegment of an array). For example, if $a = [1, 1, 2, 3, 2, 4]$, then for query $(l_i = 2, r_i = 6)$ the subarray we are interested in is $[1, 2, 3, 2, 4]$, and possible answers are 1, 3 and 4; for query $(l_i = 1, r_i = 2)$ the subarray we are interested in is $[1, 1]$, and there is no such element that occurs exactly once.

Can you answer all of the queries?

Input

The first line contains one integer n $(1 \le n \le 5 \cdot 10^5)$.

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 5 \cdot 10^5)$.

The third line contains one integer q $(1 \le q \le 5 \cdot 10^5)$.

Then q lines follow, i-th line containing two integers l_i and r_i representing i-th query $(1 \le l_i \le r_i \le n)$.

Output

Answer the queries as follows:

If there is no integer such that it occurs in the subarray from index l_i to index r_i exactly once, print 0. Otherwise print any such integer.

input

```
6
1 1 2 3 2 4
2
2 6
1 2
```

output

```
4
0
```

G. Two-Paths

3.5 seconds, 256 megabytes

You are given a weighted tree (undirected connected graph with no cycles, loops or multiple edges) with n vertices. The edge $\{u_j, v_j\}$ has weight w_j. Also each vertex i has its own value a_i assigned to it.

Let's call a path starting in vertex u and ending in vertex v, where each edge can appear no more than twice (regardless of direction), a *2-path*. Vertices can appear in the 2-path multiple times (even start and end vertices).

For some 2-path p profit $\Pr(p) = \sum_{v \in \text{distinct vertices in } p} a_v - \sum_{e \in \text{distinct edges in } p} k_e \cdot w_e$, where k_e is the number of times edge e appears in p. That is, vertices are counted once, but edges are counted the number of times they appear in p.

You are about to answer m queries. Each query is a pair of vertices (qu, qv). For each query find 2-path p from qu to qv with maximal profit $\Pr(p)$.

Input

The first line contains two integers n and q $(2 \le n \le 3 \cdot 10^5, 1 \le q \le 4 \cdot 10^5)$ — the number of vertices in the tree and the number of queries.

The second line contains n space-separated integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 10^9)$ — the values of the vertices.

Next $n - 1$ lines contain descriptions of edges: each line contains three space separated integers u_i, v_i and w_i $(1 \le u_i, v_i \le n, u_i \ne v_i, 1 \le w_i \le 10^9)$ — there is edge $\{u_i, v_i\}$ with weight w_i in the tree.

Next q lines contain queries (one per line). Each query contains two integers qu_i and qv_i $(1 \le qu_i, qv_i \le n)$ — endpoints of the 2-path you need to find.

Output

For each query print one integer per line — maximal profit $\Pr(p)$ of the some 2-path p with the corresponding endpoints.

input

```
7 6
6 5 5 3 2 1 2
1 2 2
2 3 2
2 4 1
4 5 1
6 4 2
7 3 25
1 1
4 4
5 6
6 4
3 4
3 7
```

output

```
9
9
9
8
12
-14
```

Explanation of queries:

1. $(1, 1)$ — one of the optimal 2-paths is the following: $1 \to 2 \to 4 \to 5 \to 4 \to 2 \to 3 \to 2 \to 1$. $\Pr(p) = (a_1 + a_2 + a_3 + a_4 + a_5) - (2 \cdot w(1, 2) + 2 \cdot w(2, 3) + 2 \cdot w(2, 4) + 2 \cdot w(4, 5)) = 21$.
2. $(4, 4)$: $4 \to 2 \to 1 \to 2 \to 3 \to 2 \to 4$. $\Pr(p) = (a_1 + a_2 + a_3 + a_4) - 2 \cdot (w(1, 2) + w(2, 3) + w(2, 4)) = 19 - 2 \cdot 10 = 9$.
3. $(5, 6)$: $5 \to 4 \to 2 \to 3 \to 2 \to 1 \to 2 \to 4 \to 6$.
4. $(6, 4)$: $6 \to 4 \to 2 \to 1 \to 2 \to 3 \to 2 \to 4$.
5. $(3, 4)$: $3 \to 2 \to 1 \to 2 \to 4$.
6. $(3, 7)$: $3 \to 2 \to 1 \to 2 \to 4 \to 5 \to 4 \to 2 \to 3 \to 7$.