Educational Codeforces Round 132 (Rated for Div. 2)

A. Three Doors

2 seconds, 256 megabytes

There are three doors in front of you, numbered from 1 to 3 from left to right. Each door has a lock on it, which can only be opened with a key with the same number on it as the number on the door.

There are three keys — one for each door. Two of them are hidden behind the doors, so that there is no more than one key behind each door. So two doors have one key behind them, one door doesn't have a key behind it. To obtain a key hidden behind a door, you should first unlock that door. The remaining key is in your hands.

Can you open all the doors?

Input

The first line contains a single integer t $(1 \le t \le 18)$ — the number of testcases.

The first line of each testcase contains a single integer x $(1 \le x \le 3)$ — the number on the key in your hands.

The second line contains three integers a, b and c $(0 \le a, b, c \le 3)$ — the number on the key behind each of the doors. If there is no key behind the door, the number is equal to 0.

Values $1, 2$ and 3 appear exactly once among x, a, b and c.

Output

For each testcase, print "$\mathtt{YES}$" if you can open all the doors. Otherwise, print "$\mathtt{NO}$".

input

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

output

```
YES
NO
YES
NO
```

B. Also Try Minecraft

2 seconds, 256 megabytes

You are beta testing the new secret Terraria update. This update will add quests to the game!

Simply, the world map can be represented as an array of length n, where the i-th column of the world has height a_i.

There are m quests you have to test. The j-th of them is represented by two integers s_j and t_j. In this quest, you have to go from the column s_j to the column t_j. At the start of the quest, you are appearing at the column s_j.

In one move, you can go from the column x to the column $x-1$ or to the column $x+1$. In this version, you have Spectre Boots, which allow you to fly. Since it is a beta version, they are bugged, so they only allow you to fly when you are going up and have infinite fly duration. When you are moving from the column with the height p to the column with the height q, then you get some amount of fall damage. If the height p is greater than the height q, you get $p-q$ fall damage, otherwise you fly up and get 0 damage.

For each of the given quests, determine the minimum amount of fall damage you can get during this quest.

Input

The first line of the input contains two integers n and m ($2 \le n \le 10^5; 1 \le m \le 10^5$) — the number of columns in the world and the number of quests you have to test, respectively.

The second line of the input contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$), where a_i is the height of the i-th column of the world.

The next m lines describe quests. The j-th of them contains two integers s_j and t_j ($1 \le s_j, t_j \le n; s_j \ne t_j$), which means you have to move from the column s_j to the column t_j during the j-th quest.

Note that s_j can be greater than t_j.

Output

Print m integers. The j-th of them should be the minimum amount of fall damage you can get during the j-th quest completion.

input

```
7 6
10 8 9 6 8 12 7
1 2
1 7
4 6
7 1
3 5
4 2
```

output

```
2
10
0
7
3
1
```

C. Recover an RBS

2 seconds, 256 megabytes

A bracket sequence is a string containing only characters "(" and ")". A regular bracket sequence (or, shortly, an RBS) is a bracket sequence that can be transformed into a correct arithmetic expression by inserting characters "1" and "+" between the original characters of the sequence. For example:

- bracket sequences "()()" and "(())" are regular (the resulting expressions are: "(1)+(1)" and "((1+1)+1)");
- bracket sequences ")(", "(" and ")" are not.

There was an RBS. Some brackets have been replaced with question marks. Is it true that there is a **unique** way to replace question marks with brackets, so that the resulting sequence is an RBS?

Input

The first line contains a single integer t ($1 \le t \le 5 \cdot 10^4$) — the number of testcases.

The only line of each testcase contains an RBS with some brackets replaced with question marks. Each character is either '(', ')' or '?'. At least one RBS can be recovered from the given sequence.

The total length of the sequences over all testcases doesn't exceed $2 \cdot 10^5$.

Output

For each testcase, print "YES" if the way to replace question marks with brackets, so that the resulting sequence is an RBS, is **unique**. If there is more than one way, then print "NO".

input

```
5
(?))
??????
()
??
?(?)()?)
```

output

```
YES
NO
YES
YES
NO
```

In the first testcase, the only possible original RBS is "(())".

In the second testcase, there are multiple ways to recover an RBS.

In the third and the fourth testcases, the only possible original RBS is "()".

In the fifth testcase, the original RBS can be either "((()()))" or "(())()()".

D. Rorororobot

2 seconds, 256 megabytes

There is a grid, consisting of n rows and m columns. The rows are numbered from 1 to n from bottom to top. The columns are numbered from 1 to m from left to right. The i-th column has the bottom a_i cells blocked (the cells in rows $1, 2, \dots, a_i$), the remaining $n - a_i$ cells are unblocked.

A robot is travelling across this grid. You can send it commands — move up, right, down or left. If a robot attempts to move into a blocked cell or outside the grid, it explodes.

However, the robot is broken — it executes each received command k times. So if you tell it to move up, for example, it will move up k times (k cells). You can't send it commands while the robot executes the current one.

You are asked q queries about the robot. Each query has a start cell, a finish cell and a value k. Can you send the robot an arbitrary number of commands (possibly, zero) so that it reaches the finish cell from the start cell, given that it executes each command k times?

The robot must stop in the finish cell. If it visits the finish cell while still executing commands, it doesn't count.

Input

The first line contains two integers n and m ($1 \le n \le 10^9$; $1 \le m \le 2 \cdot 10^5$) — the number of rows and columns of the grid.

The second line contains m integers $a_1, a_2, \dots, a_m$ ($0 \le a_i \le n$) — the number of blocked cells on the bottom of the i-th column.

The third line contains a single integer q ($1 \le q \le 2 \cdot 10^5$) — the number of queries.

Each of the next q lines contain five integers x_s, y_s, x_f, y_f and k ($a[y_s] < x_s \le n$; $1 \le y_s \le m$; $a[y_f] < x_f \le n$; $1 \le y_f \le m$; $1 \le k \le 10^9$) — the row and the column of the start cell, the row and the column of the finish cell and the number of times each your command is executed. The start and the finish cell of each query are unblocked.

Output

For each query, print "YES" if you can send the robot an arbitrary number of commands (possibly, zero) so that it reaches the finish cell from the start cell, given that it executes each command k times. Otherwise, print "NO".

input

```
11 10
9 0 0 10 3 4 8 11 10 8
6
1 2 1 3 1
1 2 1 3 2
4 3 4 5 2
5 3 11 5 3
5 3 11 5 2
11 9 9 10 1
```

output

```
YES
NO
NO
NO
YES
YES
```

E. XOR Tree

3 seconds, 256 megabytes

You are given a tree consisting of n vertices. A number is written on each vertex; the number on vertex i is equal to a_i.

Recall that a simple path is a path that visits each vertex at most once. Let the *weight* of the path be the bitwise XOR of the values written on vertices it consists of. Let's say that a tree is *good* if no simple path has weight 0.

You can apply the following operation any number of times (possibly, zero): select a vertex of the tree and replace the value written on it with an arbitrary positive integer. What is the minimum number of times you have to apply this operation in order to make the tree *good*?

Input

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

The second line contains n integers a_1, a_2, ..., a_n ($1 \le a_i < 2^{30}$) — the numbers written on vertices.

Then $n - 1$ lines follow, each containing two integers x and y ($1 \le x, y \le n; x \ne y$) denoting an edge connecting vertex x with vertex y. It is guaranteed that these edges form a tree.

Output

Print a single integer — the minimum number of times you have to apply the operation in order to make the tree *good*.

input

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

output

```
2
```

input

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

output

```
0
```

input

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

output

```
2
```

In the first example, it is enough to replace the value on the vertex 1 with 13, and the value on the vertex 4 with 42.

F. Multiset of Strings

6 seconds, 512 megabytes

You are given three integers n, k and f.

Consider all binary strings (i. e. all strings consisting of characters 0 and/or 1) of length from 1 to n. For every such string s, you need to choose an integer c_s from 0 to k.

A multiset of binary strings of length **exactly** n is considered beautiful if for every binary string s with length from 1 to n, the number of strings in the multiset such that s is their prefix is not exceeding c_s.

For example, let $n = 2$, $c_0 = 3$, $c_{00} = 1$, $c_{01} = 2$, $c_1 = 1$, $c_{10} = 2$, and $c_{11} = 3$. The multiset of strings $\{11, 01, 00, 01\}$ is beautiful, since:

- for the string 0, there are 3 strings in the multiset such that 0 is their prefix, and $3 \le c_0$;
- for the string 00, there is one string in the multiset such that 00 is its prefix, and $1 \le c_{00}$;
- for the string 01, there are 2 strings in the multiset such that 01 is their prefix, and $2 \le c_{01}$;
- for the string 1, there is one string in the multiset such that 1 is its prefix, and $1 \le c_1$;
- for the string 10, there are 0 strings in the multiset such that 10 is their prefix, and $0 \le c_{10}$;
- for the string 11, there is one string in the multiset such that 11 is its prefix, and $1 \le c_{11}$.

Now, for the problem itself. You have to calculate the number of ways to choose the integer c_s for every binary string s of length from 1 to n in such a way that the **maximum** possible size of a beautiful multiset is **exactly** f.

Input

The only line of input contains three integers n, k and f ($1 \le n \le 15$; $1 \le k, f \le 2 \cdot 10^5$).

Output

Print one integer — the number of ways to choose the integer c_s for every binary string s of length from 1 to n in such a way that the **maximum** possible size of a beautiful multiset is **exactly** f. Since it can be huge, print it modulo 998244353.

input

```
1 42 2
```

output

```
3
```

input

```
2 37 13
```

output

```
36871576
```

input

```
4 1252 325
```

output

```
861735572
```

input

```
6 153 23699
```

output

```
0
```

input

```
15 200000 198756
```

output

```
612404746
```

In the first example, the three ways to choose the integers c_s are:

- $c_0 = 0$, $c_1 = 2$, then the maximum beautiful multiset is $\{1, 1\}$;
- $c_0 = 1$, $c_1 = 1$, then the maximum beautiful multiset is $\{0, 1\}$;
- $c_0 = 2$, $c_1 = 0$, then the maximum beautiful multiset is $\{0, 0\}$.