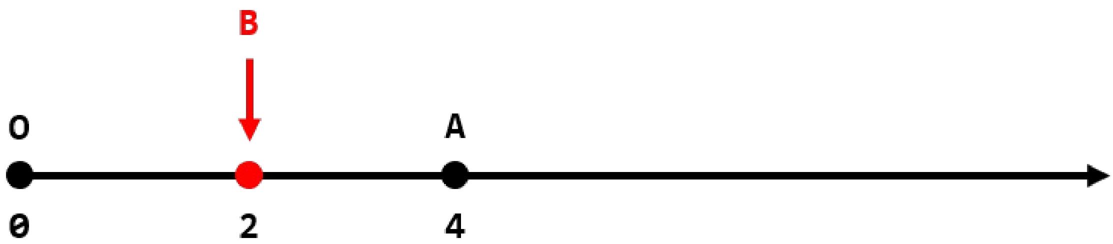
A. Distance and Axis

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

We have a point A with coordinate $x = n$ on OX-axis. We'd like to find an *integer point* B (also on OX-axis), such that the *absolute difference* between the distance from O to B and the distance from A to B is equal to k.



The description of the first test case.

Since sometimes it's impossible to find such point B, we can, in one step, increase or decrease the coordinate of A by 1. What is the minimum number of steps we should do to make such point B exist?

Input

The first line contains one integer t ($1 \le t \le 6000$) — the number of test cases.

The only line of each test case contains two integers n and k ($0 \le n, k \le 10^6$) — the initial position of point A and desirable absolute difference.

Output

For each test case, print the minimum number of steps to make point B exist.

Example

input

```
6
4 0
5 8
0 1000000
0 0
1 0
1000000 1000000
```

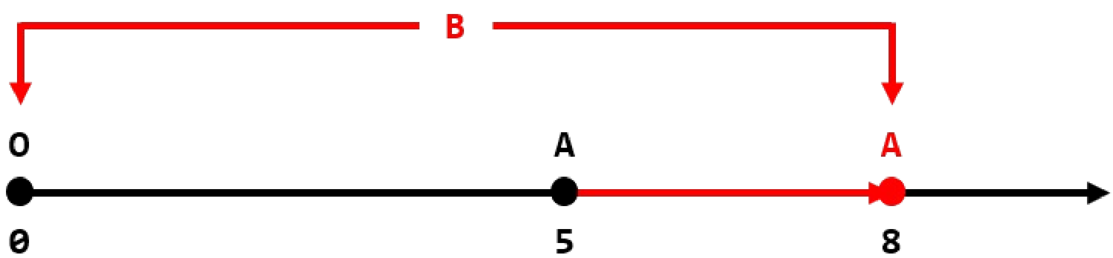
output

```
0
3
1000000
0
1
0
```

Note

In the first test case (picture above), if we set the coordinate of B as 2 then the absolute difference will be equal to $|(2-0)-(4-2)| = 0$ and we don't have to move A. So the answer is 0.

In the second test case, we can increase the coordinate of A by 3 and set the coordinate of B as 0 or 8. The absolute difference will be equal to $|8-0| = 8$, so the answer is 3.



B. Ternary Sequence

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

You are given two sequences $a_1, a_2, \dots, a_n$ and $b_1, b_2, \dots, b_n$. Each element of both sequences is either 0, 1 or 2. The number of elements 0, 1, 2 in the sequence a is x_1, y_1, z_1 respectively, and the number of elements 0, 1, 2 in the sequence b is x_2, y_2, z_2 respectively.

You can rearrange the elements in both sequences a and b however you like. After that, let's define a sequence c as follows:

$$c_i = \begin{cases} a_i b_i & \text{if } a_i > b_i \\ 0 & \text{if } a_i = b_i \\ -a_i b_i & \text{if } a_i < b_i \end{cases}$$

You'd like to make $\sum_{i=1}^n c_i$ (the sum of all elements of the sequence c) as large as possible. What is the maximum possible sum?

Input

The first line contains one integer t ($1 \le t \le 10^4$) — the number of test cases.

Each test case consists of two lines. The first line of each test case contains three integers x_1, y_1, z_1 ($0 \le x_1, y_1, z_1 \le 10^8$) — the number of 0-s, 1-s and 2-s in the sequence a.

The second line of each test case also contains three integers x_2, y_2, z_2 ($0 \le x_2, y_2, z_2 \le 10^8$; $x_1 + y_1 + z_1 = x_2 + y_2 + z_2 > 0$) — the number of 0-s, 1-s and 2-s in the sequence b.

Output

For each test case, print the maximum possible sum of the sequence c.

Example

input

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

output

```
4
2
0
```

Note

In the first sample, one of the optimal solutions is:

$a = \{2, 0, 1, 1, 0, 2, 1\}$

$b = \{1, 0, 1, 0, 2, 1, 0\}$

$c = \{2, 0, 0, 0, 0, 2, 0\}$

In the second sample, one of the optimal solutions is:

$a = \{0, 2, 0, 0, 0\}$

$b = \{1, 1, 0, 1, 0\}$

$c = \{0, 2, 0, 0, 0\}$

In the third sample, the only possible solution is:

$a = \{2\}$

$b = \{2\}$

$c = \{0\}$

C. Mere Array

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

You are given an array $a_1, a_2, \dots, a_n$ where all a_i are integers and greater than 0.

In one operation, you can choose two different indices i and j ($1 \le i, j \le n$). If $gcd(a_i, a_j)$ is equal to the minimum element of the **whole array** a, you can swap a_i and a_j. $gcd(x, y)$ denotes the greatest common divisor (GCD) of integers x and y.

Now you'd like to make a non-decreasing using the operation any number of times (possibly zero). Determine if you can do this.

An array a is non-decreasing if and only if $a_1 \le a_2 \le \ldots \le a_n$.

Input

The first line contains one integer t ($1 \le t \le 10^4$) — the number of test cases.

The first line of each test case contains one integer n ($1 \le n \le 10^5$) — the length of array a.

The second line of each test case contains n positive integers $a_1, a_2, \dots a_n$ ($1 \le a_i \le 10^9$) — the array itself.

It is guaranteed that the sum of n over all test cases doesn't exceed 10^5.

Output

For each test case, output "YES" if it is possible to make the array a non-decreasing using the described operation, or "NO" if it is impossible to do so.

Example

input

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

output

```
YES
YES
YES
NO
```

Note

In the first and third sample, the array is already non-decreasing.

In the second sample, we can swap a_1 and a_3 first, and swap a_1 and a_5 second to make the array non-decreasing.

In the forth sample, we cannot the array non-decreasing using the operation.

D. Maximum Distributed Tree

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

You are given a tree that consists of n nodes. You should label each of its $n-1$ edges with an integer in such way that satisfies the following conditions:

- each integer must be greater than 0;
- the product of all $n-1$ numbers should be equal to k;
- the number of 1-s among all $n-1$ integers must be minimum possible.

Let's define $f(u, v)$ as the sum of the numbers on the simple path from node u to node v. Also, let $\sum\limits_{i=1}^{n-1} \sum\limits_{j=i+1}^{n} f(i, j)$ be a *distribution index* of the tree.

Find the maximum possible distribution index you can get. Since answer can be too large, print it modulo $10^9 + 7$.

In this problem, since the number k can be large, the result of the prime factorization of k is given instead.

Input

The first line contains one integer t ($1 \le t \le 100$) — the number of test cases.

The first line of each test case contains a single integer n ($2 \le n \le 10^5$) — the number of nodes in the tree.

Each of the next $n-1$ lines describes an edge: the i-th line contains two integers u_i and v_i ($1 \le u_i, v_i \le n$; $u_i \ne v_i$) — indices of vertices connected by the i-th edge.

Next line contains a single integer m ($1 \le m \le 6 \cdot 10^4$) — the number of prime factors of k.

Next line contains m prime numbers $p_1, p_2, \ldots, p_m$ ($2 \le p_i < 6 \cdot 10^4$) such that $k = p_1 \cdot p_2 \cdot \ldots \cdot p_m$.

It is guaranteed that the sum of n over all test cases doesn't exceed 10^5, the sum of m over all test cases doesn't exceed $6 \cdot 10^4$, and the given edges for each test cases form a tree.

Output

Print the maximum distribution index you can get. Since answer can be too large, print it modulo $10^9 + 7$.

Example

input

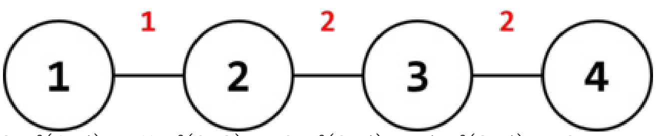
```
3
4
1 2
2 3
3 4
2
2 2
4
3 4
1 3
3 2
2
3 2
7
6 1
2 3
4 6
7 3
5 1
3 6
4
7 5 13 3
```

output

```
17
18
286
```

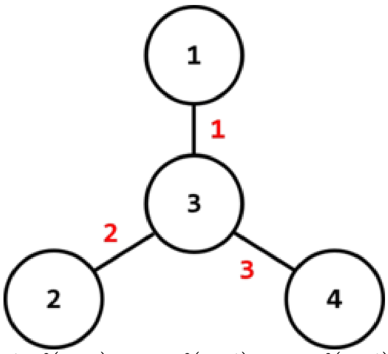
Note

In the first test case, one of the optimal ways is on the following image:



In this case, $f(1,2) = 1$, $f(1,3) = 3$, $f(1,4) = 5$, $f(2,3) = 2$, $f(2,4) = 4$, $f(3,4) = 2$, so the sum of these 6 numbers is 17.

In the second test case, one of the optimal ways is on the following image:



In this case, $f(1,2) = 3$, $f(1,3) = 1$, $f(1,4) = 4$, $f(2,3) = 2$, $f(2,4) = 5$, $f(3,4) = 3$, so the sum of these 6 numbers is 18.

E. Divide Square

time limit per test: 2 seconds

memory limit per test: 384 megabytes

input: standard input

output: standard output

There is a square of size $10^6 \times 10^6$ on the coordinate plane with four points $(0,0)$, $(0,10^6)$, $(10^6,0)$, and $(10^6,10^6)$ as its vertices.

You are going to draw segments on the plane. All segments are either horizontal or vertical and *intersect with at least one side of the square*.

Now you are wondering how many pieces this square divides into after drawing all segments. Write a program calculating the number of pieces of the square.

Input

The first line contains two integers n and m ($0 \le n, m \le 10^5$) — the number of horizontal segments and the number of vertical segments.

The next n lines contain descriptions of the horizontal segments. The i-th line contains three integers y_i, lx_i and rx_i ($0 < y_i < 10^6$; $0 \le lx_i < rx_i \le 10^6$), which means the segment connects (lx_i, y_i) and (rx_i, y_i).

The next m lines contain descriptions of the vertical segments. The i-th line contains three integers x_i, ly_i and ry_i ($0 < x_i < 10^6$; $0 \le ly_i < ry_i \le 10^6$), which means the segment connects (x_i, ly_i) and (x_i, ry_i).

It's guaranteed that **there are no two segments on the same line**, and each segment intersects with at least one of square's sides.

Output

Print the number of pieces the square is divided into after drawing all the segments.

Example

input

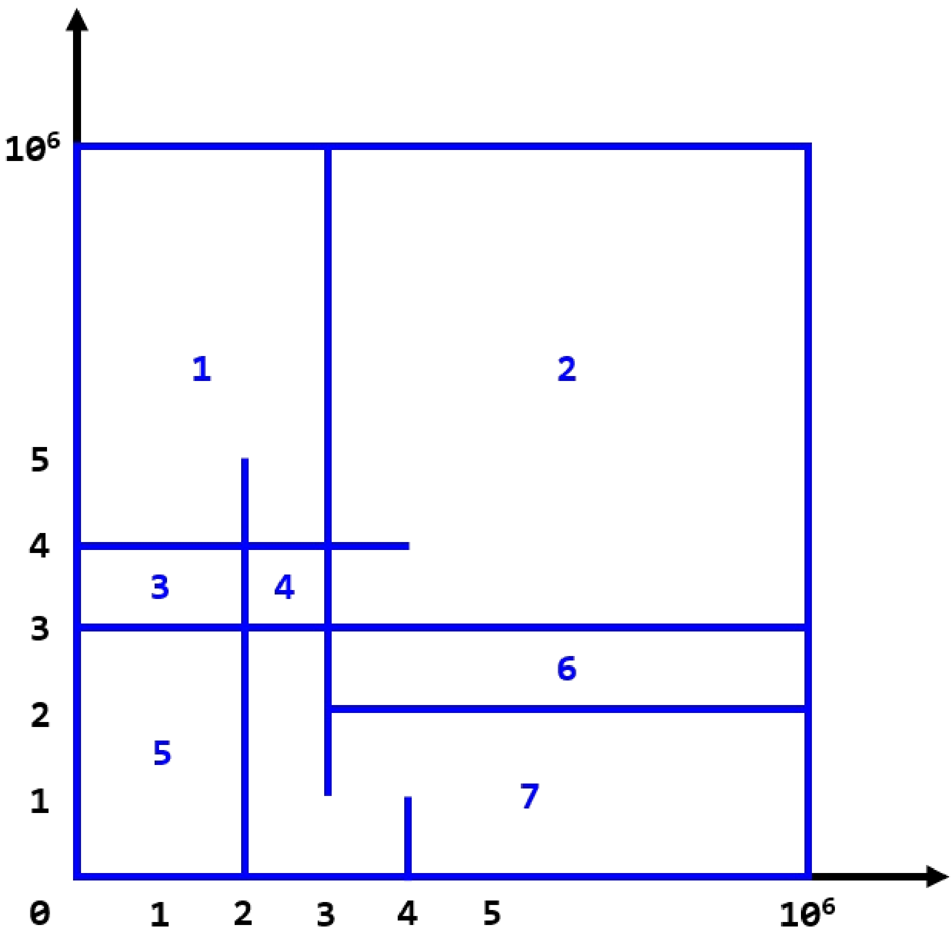
```
3 3
2 3 1000000
4 0 4
3 0 1000000
4 0 1
2 0 5
3 1 1000000
```

output

```
7
```

Note

The sample is like this:



F. Reverse and Swap

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

You are given an array a of length 2^n. You should process q queries on it. Each query has one of the following 4 types:

1. $Replace(x, k)$ — change a_x to k;

2. $Reverse(k)$ — reverse each subarray $[(i-1) \cdot 2^k + 1, i \cdot 2^k]$ for all i ($i \geq 1$);
3. $Swap(k)$ — swap subarrays $[(2i-2) \cdot 2^k + 1, (2i-1) \cdot 2^k]$ and $[(2i-1) \cdot 2^k + 1, 2i \cdot 2^k]$ for all i ($i \geq 1$);
4. $Sum(l, r)$ — print the sum of the elements of subarray $[l, r]$.

Write a program that can quickly process given queries.

Input

The first line contains two integers n, q ($0 \leq n \leq 18$; $1 \leq q \leq 10^5$) — the length of array a and the number of queries.

The second line contains 2^n integers $a_1, a_2, \ldots, a_{2^n}$ ($0 \leq a_i \leq 10^9$).

Next q lines contains queries — one per line. Each query has one of 4 types:

- "1 x k" ($1 \leq x \leq 2^n$; $0 \leq k \leq 10^9$) — $Replace(x, k)$;
- "2 k" ($0 \leq k \leq n$) — $Reverse(k)$;
- "3 k" ($0 \leq k < n$) — $Swap(k)$;
- "4 l r" ($1 \leq l \leq r \leq 2^n$) — $Sum(l, r)$.

It is guaranteed that there is at least one Sum query.

Output

Print the answer for each Sum query.

Examples

input

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

output

```
24
```

input

```
3 8
7 0 8 8 7 1 5 2
4 3 7
2 1
3 2
4 1 6
2 3
1 5 16
4 8 8
3 0
```

output

```
29
22
1
```

Note

In the first sample, initially, the array a is equal to $\{7, 4, 9, 9\}$.

After processing the first query. the array a becomes $\{7, 8, 9, 9\}$.

After processing the second query, the array a_i becomes $\{9, 9, 7, 8\}$

Therefore, the answer to the third query is $9 + 7 + 8 = 24$.

In the second sample, initially, the array a is equal to $\{7, 0, 8, 8, 7, 1, 5, 2\}$. What happens next is:

1. $Sum(3, 7) \to 8 + 8 + 7 + 1 + 5 = 29$;
2. $Reverse(1) \to \{0, 7, 8, 8, 1, 7, 2, 5\}$;
3. $Swap(2) \to \{1, 7, 2, 5, 0, 7, 8, 8\}$;
4. $Sum(1, 6) \to 1 + 7 + 2 + 5 + 0 + 7 = 22$;
5. $Reverse(3) \to \{8, 8, 7, 0, 5, 2, 7, 1\}$;
6. $Replace(5, 16) \to \{8, 8, 7, 0, 16, 2, 7, 1\}$;
7. $Sum(8, 8) \to 1$;
8. $Swap(0) \to \{8, 8, 0, 7, 2, 16, 1, 7\}$.