A. Enlarge GCD

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

Mr. F has n positive integers, $a_1, a_2, \ldots, a_n$.

He thinks the greatest common divisor of these integers is too small. So he wants to enlarge it by removing some of the integers.

But this problem is too simple for him, so he does not want to do it by himself. If you help him, he will give you some scores in reward.

Your task is to calculate the minimum number of integers you need to remove so that the greatest common divisor of the remaining integers is bigger than that of all integers.

Input

The first line contains an integer n ($2 \leq n \leq 3 \cdot 10^5$) — the number of integers Mr. F has.

The second line contains n integers, $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 1.5 \cdot 10^7$).

Output

Print an integer — the minimum number of integers you need to remove so that the greatest common divisor of the remaining integers is bigger than that of all integers.

You should not remove all of the integers.

If there is no solution, print «-1» (without quotes).

Examples

input

```
3
1 2 4
```

output

```
1
```

input

```
4
6 9 15 30
```

output

```
2
```

input

```
3
1 1 1
```

output

```
-1
```

Note

In the first example, the greatest common divisor is 1 in the beginning. You can remove 1 so that the greatest common divisor is enlarged to 2. The answer is 1.

In the second example, the greatest common divisor is 3 in the beginning. You can remove 6 and 9 so that the greatest common divisor is enlarged to 15. There is no solution which removes only one integer. So the answer is 2.

In the third example, there is no solution to enlarge the greatest common divisor. So the answer is -1.

B. Little C Loves 3 II

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

Little C loves number «3» very much. He loves all things about it.

Now he is playing a game on a chessboard of size $n \times m$. The cell in the x-th row and in the y-th column is called (x, y). Initially, The chessboard is empty. Each time, he places two chessmen on two different empty cells, the Manhattan distance between which is exactly 3. The Manhattan distance between two cells (x_i, y_i) and (x_j, y_j) is defined as $|x_i - x_j| + |y_i - y_j|$.

He want to place as many chessmen as possible on the chessboard. Please help him find the maximum number of chessmen he can place.

Input

A single line contains two integers n and m ($1 \le n, m \le 10^9$) — the number of rows and the number of columns of the chessboard.

Output

Print one integer — the maximum number of chessmen Little C can place.

Examples

input

```
2 2
```

output

```
0
```

input

```
3 3
```

output

```
8
```

Note

In the first example, the Manhattan distance between any two cells is smaller than 3, so the answer is 0.

In the second example, a possible solution is $(1, 1)(3, 2)$, $(1, 2)(3, 3)$, $(2, 1)(1, 3)$, $(3, 1)(2, 3)$.

C. Region Separation

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

There are n cities in the Kingdom of Autumn, numbered from 1 to n. People can travel between any two cities using $n - 1$ two-directional roads.

This year, the government decides to separate the kingdom. There will be regions of different levels. The whole kingdom will be the region of level 1. Each region of i-th level should be separated into several (at least two) regions of $i + 1$-th level, unless i-th level is the last level. Each city should belong to exactly one region of each level and for any two cities in the same region, it should be possible to travel between them passing the cities in the same region only.

According to research, for each city i, there is a value a_i, which describes the importance of this city. All regions of the same level should have an equal sum of city importances.

Your task is to find how many plans there are to determine the separation of the regions that all the conditions are satisfied. Two plans are considered different if and only if their numbers of levels are different or there exist two cities in the same region of one level in one plan but in different regions of this level in the other plan. Since the answer may be very large, output it modulo $10^9 + 7$.

Input

The first line contains one integer n ($1 \le n \le 10^6$) — the number of the cities.

The second line contains n integers, the i-th of which is a_i ($1 \le a_i \le 10^9$) — the value of each city.

The third line contains $n - 1$ integers, $p_1, p_2, \ldots, p_{n-1}$; p_i ($p_i \le i$) describes a road between cities p_i and $i + 1$.

Output

Print one integer — the number of different plans modulo $10^9 + 7$.

Examples

input

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

output

```
4
```

input

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

output

```
2
```

input

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

output

```
3
```

Note

For the first example, there are 4 different plans:

Plan 1: Level-1: $\{1, 2, 3, 4\}$.

Plan 2: Level-1: $\{1, 2, 3, 4\}$, Level-2: $\{1, 2\}$,$\{3, 4\}$.

Plan 3: Level-1: $\{1, 2, 3, 4\}$, Level-2: $\{1\}$,$\{2\}$,$\{3\}$,$\{4\}$.

Plan 4: Level-1: $\{1, 2, 3, 4\}$, Level-2: $\{1, 2\}$,$\{3, 4\}$, Level-3: $\{1\}$,$\{2\}$,$\{3\}$,$\{4\}$.

For the second example, there are 2 different plans:

Plan 1: Level-1: $\{1, 2, 3, 4\}$.

Plan 2: Level-1: $\{1, 2, 3, 4\}$, Level-2: $\{1\}$,$\{2\}$,$\{3\}$,$\{4\}$.

For the third example, there are 3 different plans:

Plan 1: Level-1: $\{1, 2, 3, 4\}$.

Plan 2: Level-1: $\{1, 2, 3, 4\}$, Level-2: $\{1, 2\}$,$\{3, 4\}$.

Plan 3: Level-1: $\{1, 2, 3, 4\}$, Level-2: $\{1, 3\}$,$\{2\}$,$\{4\}$.

D. Intervals of Intervals

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

Little D is a friend of Little C who loves intervals very much instead of number "3".

Now he has n intervals on the number axis, the i-th of which is $[a_i, b_i]$.

Only the n intervals can not satisfy him. He defines the value of an *interval of intervals* $[l, r]$ ($1 \leq l \leq r \leq n$, l and r are both integers) as the total length of the union of intervals from the l-th to the r-th.

He wants to select exactly k different *intervals of intervals* such that the sum of their values is maximal. Please help him calculate the maximal sum.

Input

First line contains two integers n and k ($1 \leq n \leq 3 \cdot 10^5$, $1 \leq k \leq \min\{\frac{n(n+1)}{2}, 10^9\}$) — the number of intervals Little D has and the number of *intervals of intervals* he will select.

Each of the next n lines contains two integers a_i and b_i, the i-th line of the n lines describing the i-th interval ($1 \leq a_i < b_i \leq 10^9$).

Output

Print one integer — the maximal sum of values Little D can get.

Examples

input

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

output

```
3
```

input

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

output

```
15
```

Note

For the first example, Little D will select $[1, 2]$, the union of the first interval and the second interval is $[1, 4]$, whose length is 3.

For the second example, Little D will select $[1, 2]$, $[2, 3]$ and $[1, 3]$, the answer is $5 + 6 + 4 = 15$.

E. Little C Loves 3 III

time limit per test: 1 second
memory limit per test: 64 megabytes
input: standard input
output: standard output

Little C loves number «3» very much. He loves all things about it.

Now he is interested in the following problem:

There are two arrays of 2^n intergers $a_0, a_1, \ldots, a_{2^n-1}$ and $b_0, b_1, \ldots, b_{2^n-1}$.

The task is for each $i(0 \leq i \leq 2^n - 1)$, to calculate $c_i = \sum a_j \cdot b_k$ ($j|k = i$ and $j\&k = 0$, where "$|$" denotes bitwise or operation and "$\&$" denotes bitwise and operation).

It's amazing that it can be proved that there are exactly 3^n triples (i, j, k), such that $j|k = i$, $j\&k = 0$ and $0 \leq i, j, k \leq 2^n - 1$. So Little C wants to solve this excellent problem (because it's well related to 3) excellently.

Help him calculate all c_i. Little C loves 3 very much, so he only want to know each $c_i\&3$.

Input

The first line contains one integer $n(0 \leq n \leq 21)$.

The second line contains 2^n integers in $[0, 3]$ without spaces — the i-th of them is a_{i-1}.

The third line contains 2^n integers in $[0, 3]$ without spaces — the i-th of them is b_{i-1}.

Output

Print one line contains 2^n integers in $[0, 3]$ without spaces — the i-th of them is $c_{i-1}\&3$. (It's obvious that $c_i\&3$ is in $[0, 3]$).

Examples

input

```
1
11
11
```

output

```
12
```

input

```
2
0123
3210
```

output

```
0322
```