Problem A. A Calibration Lab

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

A calibration laboratory stores signal modules in pairs.

There are n cases. The i-th case contains a pair of signal modules: one of type a_i and one of type b_i. Types a_i and b_i may be equal.

If you select two different cases, you will obtain four signal modules in total. The laboratory has two identical test benches, and you must place exactly two modules on each bench. A calibration run is valid only if the multiset of module types placed on each bench is identical.

Count the number of ways to choose two different cases such that it is possible to distribute the modules between the two test benches and obtain a valid calibration run.

Input

The input is given in the following format:

$$
\begin{array}{l}
n \\
a_1 \ \ b_1 \\
a_2 \ \ b_2 \\
\vdots \\
a_n \ \ b_n
\end{array}
$$

- $2 \le n \le 3 \cdot 10^5$
- $1 \le a_i, b_i \le n$
- All input values are integers.

Output

Print the answer in a single line.

Example

standard input	*standard output*
4 2 3 3 2 1 1 2 2	2

Problem B. Box

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

A UPC Box is an ordinary rectangular box that is currently popular around the world. There are n UPC Boxes. For each $i = 1, 2, \ldots, n$, the i-th box has an integer weight a_i.

You will build a vertical stack by repeatedly choosing one remaining box and inserting it at the very bottom of the current stack. When a box of weight w is inserted at the bottom of the existing stack whose total weight is x, that box receives a load equal to $\left\lfloor \frac{x}{w} \right\rfloor$.

Determine the minimum possible total load over all boxes.

Input

The input is given in the following format:

n
$a_1\ a_2\ \ldots\ a_n$

- $2 \le n \le 2 \cdot 10^5$
- $1 \le a_i \le 10^9$
- All input values are integers.

Output

Output the answer in a single line.

Example

standard input	*standard output*
5 3 1 4 1 5	3

Problem C. Chairs and Columns

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

There are $h \times w$ chairs arranged in h rows and w columns. We denote the chair in the i-th row from the top and the j-th column from the left by (i, j).

Some chairs may have luggage placed on them. The situation of the chairs is represented by h strings $s_1, s_2, \ldots, s_h$, each of length w. If the j-th character of s_i is '`#`', then there is luggage on (i, j). If it is '`.`', then there is no luggage on (i, j). It is guaranteed that there is at least one chair on which there is no luggage.

We want to seat people on these chairs. At most one person can sit on each chair, and a person cannot sit on a chair that has luggage on it. Moreover, two people cannot sit on chairs that are adjacent to each other vertically or horizontally. Under these conditions, we want to seat as many people as possible. Let m be the maximum number of people we can seat observing these rules. Now, suppose one person arrives. For each chair, determine whether we may seat this person there. Specifically, determine whether it is possible to seat this person on that chair, and in addition, still be able to seat $m - 1$ more people under the rules.

Input

The input is given in the following format:

h w
s_1
s_2
$\vdots$
s_h

- $1 \le h \le 400$
- $1 \le w \le 400$
- Both h and w are integers.
- s_i is a string of length w consisting of '`#`' and '`.`' ($1 \le i \le h$).
- There exists a position (i, j) such that the j-th character of s_i is '`.`'.

Output

Print h lines. On the i-th line ($1 \le i \le h$), print a string of length w. For each (i, j), if we can seat the newly arrived person on (i, j), then the j-th character of the string on the i-th line must be `1`. Otherwise, it must be `0`.

Example

standard input	*standard output*
3 4 ##.. .... #.##	0011 1011 0100

Problem D. Do Count The Inversions

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

You are given positive integers n and k, and also a string s of length n consisting of lowercase English letters. Let t be the string obtained by concatenating k copies of s.

Find the number of inversions in the suffix array of t, modulo 998 244 353.

For a string s of length n, the suffix array of s is a permutation of integers from 1 to n that represents the starting positions of all non-empty suffixes of s, sorted in lexicographical order.

For a permutation $p_1, p_2, \ldots, p_n$ of integers from 1 to n, the number of inversions is the number of pairs (i, j) such that $i < j$, $1 \le i, j \le n$ and $p_i > p_j$.

Input

The input is given in the following format:

n k
s

- $1 \le n \le 2 \cdot 10^5$
- $1 \le k \le 10^{12}$
- Both n and k are integers.
- s is a string of length n consisting of lowercase English letters.

Output

Output the answer in a single line.

Examples

standard input	*standard output*
3 3 upc	27
23 52026 ukrainianprogrammingcup	181493084

Problem E. Earth and Circles

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

On the sphere centered at $(0, 0, 0)$ with radius r, there are n circles. The i-th circle is defined as the set of intersection points between the sphere and the following plane:

- The plane passes through the point (x_i, y_i, z_i).
- The plane is orthogonal to $\vec{v} = (x_i, y_i, z_i)$.

It is guaranteed that $\vec{v} \neq \vec{0}$.

It is also guaranteed that no two circles share any common point.

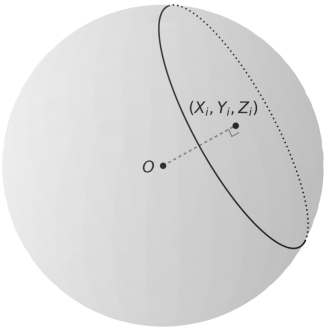

Figure E-1: The circle defined by (X_i, Y_i, Z_i)

We define the distance between two points on the sphere as follows:

For a path on the sphere connecting these two points, count how many of the circles the path intersects. The distance is the minimum possible value of this count over all such paths.

You may choose one point on the sphere and designate it as the Pole.

Find the minimum possible value of

$$\max_{p \in \text{sphere}} \text{distance}(\text{Pole}, p).$$

Input

The input is given in the following format:

$$\begin{array}{l} n \;\; r \\ x_1 \;\; y_1 \;\; z_1 \\ x_2 \;\; y_2 \;\; z_2 \\ \vdots \\ x_n \;\; y_n \;\; z_n \end{array}$$

- $1 \leq n \leq 2000$

- $2 \leq r \leq 10^6$
- $0 < ||(x_i, y_i, z_i)|| < r$
- No two circles share any common point.
- All input values are integers.

Output

Output the answer in a single line.

Example

standard input	*standard output*
5 100 14 -11 -2 -54 46 8 54 -57 -12 -34 39 7 64 -74 -4	3

Note

Illustration to the Sample Input:

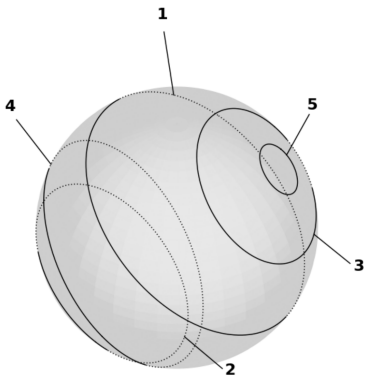

Figure E-2: Illustration of Sample Input

Problem F. Find the Minimal Number of Moves

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

You are given an infinite complete binary tree with vertices labeled by positive integers. The root is vertex 1, and for every vertex x ($x \geq 2$), its parent is $\left\lfloor \frac{x}{2} \right\rfloor$.

You are also given a string $s = s_0 s_1 \dots s_{n-1}$ of length n. Each character of s is either '`L`' or '`R`'.

Consider the following process. You are currently at some vertex u, and you want to reach vertex v by repeatedly moving through the tree.

On the i-th move (1-indexed), suppose you are at vertex x. You may choose one of the following moves:

- **Downward move:** If $s_{(i-1) \bmod n}$ is '`L`', move to vertex $2x$. Otherwise, move to vertex $2x + 1$.
- **Upward move:** Move to vertex $\left\lfloor \frac{x}{2} \right\rfloor$. You can choose this move only if $x \geq 2$.

Note that you cannot stay at the same vertex.

You are given q independent queries. In the i-th query, you start at vertex u_i and want to reach vertex v_i.

For each query, determine whether it is possible to reach v_i from u_i. If it is possible, find the minimum number of moves required.

Input

The input is given in the following format:

n
s
q
u_1 v_1
u_2 v_2
$\vdots$
u_q v_q

- $1 \leq n \leq 10^6$
- $1 \leq q \leq 2 \cdot 10^5$
- $1 \leq u_i, v_i \leq 10^{18}$
- All numbers in the input are integers.
- s is a string of length n consisting of '`L`' and '`R`'.

Output

Output q lines. On the i-th line, print the minimum number of moves required to reach v_i from u_i if it is possible; otherwise, print "-1".

Examples

standard input	*standard output*
5 LLRLR 3 1 12 9 2 913 2025	7 2 23
1 L 1 1 3	-1

Problem G. Grid And Its Beauty

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

For a grid with each cell colored either white or black, let us define the *beauty* of the grid as follows.

Consider performing the following operation any number of times:

- Choose a path from the upper-left corner to the lower-right corner, consisting only of downward and rightward moves.
- Invert the colors of all cells on the chosen path.

The *beauty* of the grid is defined as the maximum possible number of cells colored black.

You have a grid with h rows and w columns. Initially, all cells are colored white.

You need to process q queries in order. The i-th query is given in the following format:

- You are given two integers t_i and x_i.
- If $t_i = 1$, invert the colors of all cells in the x_i-th row from the top.
- If $t_i = 2$, invert the colors of all cells in the x_i-th column from the left.

After each query, find the beauty of the current grid.

Input

The input is given in the following format:

$$\begin{array}{l} h\ \ w\ \ q \\ t_1\ \ x_1 \\ t_2\ \ x_2 \\ \vdots \\ t_q\ \ x_q \end{array}$$

- $1 \le h, w \le 2 \cdot 10^5$
- $1 \le q \le 2 \cdot 10^5$
- $t_i \in \{1, 2\}$
- $t_i = 1 \Longrightarrow 1 \le x_i \le h$
- $t_i = 2 \Longrightarrow 1 \le x_i \le w$
- All input values are integers.

Output

Print q lines. On the i-th line ($1 \le i \le q$), print the answer for the i-th query.

Example

standard input	*standard output*
3 4 5 2 2 2 3 1 1 1 2 2 3	9 12 10 10 9

Problem H. Handle The Queries On Tree

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

You are given a tree with n vertices, numbered 1 through n. Edge i connects vertices a_i and b_i. Each vertex i is assigned a color c_i.

You are asked to process q queries. In each query, four integers u_1, v_1, u_2, v_2 are given.

For each query, determine the maximum integer k ($0 \le k \le n$) such that the following condition holds:

For every $j = 1, 2, \ldots, k$, the number of vertices of color j on the path from u_1 to v_1 is equal to the number of vertices of color j on the path from u_2 to v_2.

Input

The input is given in the following format:

n
$a_1\ b_1$
$a_2\ b_2$
$\vdots$
$a_{n-1}\ b_{n-1}$
$c_1\ c_2\ \ldots\ c_n$
q
$\texttt{query}_1$
$\texttt{query}_2$
$\vdots$
$\texttt{query}_q$

Each query is given in the following format:

$u_1\ v_1\ u_2\ v_2$

- $2 \le n \le 10^5$
- $1 \le a_i, b_i \le n$ for $1 \le i \le n-1$
- $1 \le c_i \le n$ for $1 \le i \le n$
- $1 \le q \le 10^5$
- $1 \le u_1, v_1, u_2, v_2 \le n$ for $1 \le i \le q$
- The given graph is a tree.
- All input values are integers.

Output

Print q lines. On the i-th line ($1 \le i \le q$), print the answer for the i-th query.

Example

standard input	*standard output*
6 2 3 4 3 6 2 3 5 2 1 1 2 2 3 1 1 4 1 6 5 4 6 5 1 5 1 1 6 6 1 5 4 2	0 6 6 0

Problem I. Inside Yamanote

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 seconds
Memory limit:	1024 mebibytes

There are n cities numbered from 0 to $n-1$. A railway goes around all n cities, and city i and city $(i+1) \bmod n$ can be traveled between in l_i time units.

You will implement the following policy:

- You may construct any number of railways that allow travel between any two cities in any non-negative time.
- You then select one city among the n cities to be the capital. The minimum travel time from the capital to city i using the railways is defined as that city's undevelopment index d_i.

The reputation of this policy will be determined by word of mouth from the m residents who will move this year. Resident j will move from city x_j to city y_j after the policy is implemented. The reputation will be the sum of $d_{x_j} - d_{y_j}$.

Your goal is to maximize the reputation of the policy. However, renovation work on the existing railway is also in progress, and you must revise the policy accordingly.

The travel times of existing railways will change q times. At the k-th change, the travel time between city t_k and city $(t_k+1) \bmod n$ changes to s_k. These changes are persistent. After each change, output the maximum possible reputation of the policy under the new conditions.

Input

The input is given in the following format:

$$
\begin{array}{l}
n\ \ m\ \ q \\
l_0\ \ l_1\ \ \ldots\ \ l_{n-1} \\
x_1\ \ y_1 \\
x_2\ \ y_2 \\
\vdots \\
x_m\ \ y_m \\
t_1\ \ s_1 \\
t_2\ \ s_2 \\
\vdots \\
t_q\ \ s_q
\end{array}
$$

- $3 \le n \le 2 \cdot 10^5$
- $1 \le m, q \le 2 \cdot 10^5$
- $0 \le l_i \le 10^6$ for $1 \le i \le n$
- $0 \le x_j, y_j \le n-1$ for $1 \le j \le m$
- $x_j \ne y_j$ for $1 \le j \le m$
- $0 \le t_k \le n-1$ for $1 \le k \le q$
- $0 \le s_k \le 10^6$ for $1 \le k \le q$

- All input values are integers.

Output

Print q lines. On the k-th line ($1 \le k \le q$), output the answer for the query k. It can be proven that the answer is an integer.

Example

standard input	*standard output*
5 3 4 1 5 2 1 3 0 2 1 3 4 2 4 0 1 3 4 2 3 9	8 7 8 15

Problem J. Juggle With Tree And Queries

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 seconds
Memory limit:	1024 mebibytes

This is an interactive problem.

The interaction consists of two phases.

You must first choose an integer n ($2 \le n \le 65$). Then you must output the chosen n and a tree T on n vertices satisfying the following conditions:

- The vertices are numbered $1, 2, \ldots, n$.
- Each edge of T has an integer weight between 0 and 10^9, inclusive.

Then you are given an integer q, followed by q integers $x_1, x_2, \ldots, x_q$. For each i ($1 \le i \le q$), you must express x_i as the sum of the weights of at most 5 paths in T.

Formally, for each i, you must output:

- an integer k ($1 \le k \le 5$), and
- k pairs of vertices $(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)$

such that

$$\sum_{j=1}^{k} w(u_j, v_j) = x_i,$$

where $w(u, v)$ denotes the weight of the path between vertices u and v in T.

The weight of a path is defined as the sum of the weights of the edges contained in the path.

Interaction Protocol

The interaction proceeds as follows.

In the beginning, you choose an integer n and a weighted tree T, and output them in the following format:

n
a_1 b_1 c_1
a_2 b_2 c_2
$\vdots$
a_{n-1} b_{n-1} c_{n-1}

Each triple (a_i, b_i, c_i) represents an edge between vertices a_i and b_i with weight c_i.

The following conditions must hold: $2 \le n \le 65$, $1 \le a_i, b_i \le n$, $0 \le c_i \le 10^9$. All values are integers.

After that, you receive a line with an integer q ($1 \le q \le 10^4$). Then, q lines with integers $x_1, x_2, \ldots, x_q$ ($1 \le x_i \le 10^9$) are given one by one. For each integer x_i, you must output your answer in the following format:

$k\ u_1\ v_1\ u_2\ v_2\ \ldots\ u_k\ v_k$

Here, k ($1 \le k \le 5$) is the number of paths, and each pair (u_j, v_j) represents a path between vertices u_j and v_j. All values are integers.

Note that the value x_i ($i \ge 2$) is provided only after the answer for x_{i-1} has been printed.

Example

standard input	*standard output*
3 110 210 20	31 2 10 3 1 100 12 3 32 1 1 3 1 3 21 2 1 2

Problem K. Knotwork Machine

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

A knotwork machine has m hooks numbered from 0 to $m-1$ and performs exactly m stages. For every stage r and every pair of hooks h and k, the integer $coef[r][h][k]$ is the weight of a fastening made from hook h to hook k at that stage.

Before the first stage, the machine is in the following state:

```
integer position = 0;
integer value = 1;
stack history = {};
```

Here, `position` is the active hook, `value` is the weight of the current knotwork, and `history` stores the hooks of open loops.

At each stage $r = 0, 1, \ldots, m-1$, choose an integer z_r such that $-1 \le z_r < m$ and perform one of the following operations.

- If $0 \le z_r$, open a new loop toward hook z_r:

  ```
  value *= coef[r][position][next];
  history.push(position);
  position = next;
  ```

  In this code, `next` denotes z_r.

- If $z_r = -1$, close the most recently opened loop. This operation is allowed only when `history` is not empty:

  ```
  assert (!history.empty());
  value *= coef[r][position][history.top()];
  position = (position + history.top()) % m;
  history.pop();
  ```

The stack uses the last-in, first-out rule:

- `push(position)` adds `position` to the top of the stack;
- `pop()` removes the top element;
- `top()` returns the top element without removing it.

A sequence $z_0, z_1, \ldots, z_{m-1}$ is valid if it never tries to close a loop while the stack is empty. For every $d = 0, 1, \ldots, m-1$, consider all valid sequences for which the final value of `position` is d. Compute the sum of the corresponding final values of `value` and output it modulo 998 244 353.

Input

The input is given in the following format:

$$
\begin{array}{l}
m \\
coef_{0,0,0} \ \cdots \ coef_{0,0,m-1} \\
\vdots \\
coef_{0,m-1,0} \ \cdots \ coef_{0,m-1,m-1} \\
\vdots \\
coef_{m-1,0,0} \ \cdots \ coef_{m-1,0,m-1} \\
\vdots \\
coef_{m-1,m-1,0} \ \cdots \ coef_{m-1,m-1,m-1}
\end{array}
$$

- $coef_{r,h,k}$ denotes the value of $coef[r][h][k]$;
- $1 \le m \le 30$;
- $0 \le coef_{r,h,k} \le 10^9$ for $0 \le r, h, k < m$;
- all input values are integers.

Output

Print m lines. For every $d = 0, 1, \ldots, m-1$, print on line $d+1$ the required sum for sequences whose final value of `position` is d.

Examples

standard input	*standard output*
2 2 14 98 686 3 15 75 375	1062 6330
3 4 3 4 3 4 3 3 3 3 4 3 3 4 4 3 4 4 4 4 3 3 3 4 3 3 4 4	703 736 680
4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	1920 1920 1920 1920

Problem L. Let's Find The Sum

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

You are given a positive integer n.

Find the value

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\left\lfloor\frac{n}{ij}\right\rfloor.$$

You are given t test cases, so find the answer for each.

Input

The input is given in the following format:

t
$\texttt{case}_1$
$\texttt{case}_2$
$\vdots$
$\texttt{case}_t$

- Each $\texttt{case}_i$ represents the i-th test case containing one integer n.
- $1 \le t \le 100$
- $1 \le n \le 10^9$
- All input values are integers.

Output

Print t lines. On the i-th line, output the answer to the i-th test case.

Example

standard input	*standard output*
10 3 30 300 3000 30000 300000 3000000 30000000 300000000 987654321	7 276 6333 115307 1835386 26772576 367844322 4838715247 61580715264 227268023762