A. Integer Diversity

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

You are given n integers $a_1, a_2, \ldots, a_n$. You choose any subset of the given numbers (possibly, none or all numbers) and negate these numbers (i. e. change $x \to (-x)$). What is the maximum number of different values in the array you can achieve?

Input

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

The next lines contain the description of the t test cases, two lines per a test case.

In the first line you are given one integer n ($1 \le n \le 100$): the number of integers in the array.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-100 \le a_i \le 100$).

Output

For each test case, print one integer: the maximum number of different elements in the array that you can achieve negating numbers in the array.

Example

input

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

output

```
4
3
1
```

Note

In the first example we can, for example, negate the first and the last numbers, achieving the array $[-1, 1, 2, -2]$ with four different values.

In the second example all three numbers are already different.

In the third example negation does not change anything.

B. Mirror in the String

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

You have a string $s_1 s_2 \ldots s_n$ and you stand on the left of the string looking right. You want to choose an index k ($1 \le k \le n$) and place a mirror after the k-th letter, so that what you see is $s_1 s_2 \ldots s_k s_k s_{k-1} \ldots s_1$. What is the lexicographically smallest string you can see?

A string a is lexicographically smaller than a string b if and only if one of the following holds:

- a is a prefix of b, but $a \ne b$;
- in the first position where a and b differ, the string a has a letter that appears earlier in the alphabet than the corresponding letter in b.

Input

The first line of input contains one integer t ($1 \le t \le 10\,000$): the number of test cases.

The next t lines contain the description of the test cases, two lines per a test case.

In the first line you are given one integer n ($1 \le n \le 10^5$): the length of the string.

The second line contains the string s consisting of n lowercase English characters.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case print the lexicographically smallest string you can see.

Example

input

```
4
10
codeforces
9
cbacbacba
3
aaa
4
bbaa
```

output

```
cc
cbaabc
aa
bb
```

Note

In the first test case choose $k = 1$ to obtain "cc".

In the second test case choose $k = 3$ to obtain "cbaabc".

In the third test case choose $k = 1$ to obtain "aa".

In the fourth test case choose $k = 1$ to obtain "bb".

C. Representative Edges

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

An array $a_1, a_2, \ldots, a_n$ is *good* if and only if for every subsegment $1 \leq l \leq r \leq n$, the following holds: $a_l + a_{l+1} + \ldots + a_r = \frac{1}{2}(a_l + a_r) \cdot (r - l + 1)$.

You are given an array of integers $a_1, a_2, \ldots, a_n$. In one operation, you can replace any one element of this array with any real number. Find the minimum number of operations you need to make this array good.

Input

The first line of input contains one integer t ($1 \leq t \leq 100$): the number of test cases.

Each of the next t lines contains the description of a test case.

In the first line you are given one integer n ($1 \leq n \leq 70$): the number of integers in the array.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-100 \leq a_i \leq 100$): the initial array.

Output

For each test case, print one integer: the minimum number of elements that you need to replace to make the given array good.

Example

input

```
5
4
1 2 3 4
4
1 1 2 2
2
0 -1
6
3 -2 4 -1 -4 0
1
-100
```

output

```
0
2
0
3
0
```

Note

In the first test case, the array is good already.

In the second test case, one of the possible good arrays is $[1, 1, \underline{1}, \underline{1}]$ (replaced elements are underlined).

In the third test case, the array is good already.

In the fourth test case, one of the possible good arrays is $[\underline{-2.5}, -2, \underline{-1.5}, -1, \underline{-0.5}, 0]$.

D. Keep the Average High

time limit per test: 1.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an array of integers $a_1, a_2, \ldots, a_n$ and an integer x.

You need to select the maximum number of elements in the array, such that for every subsegment $a_l, a_{l+1}, \ldots, a_r$ containing strictly more than one element $(l < r)$, either:

- At least one element on this subsegment is **not** selected, or
- $a_l + a_{l+1} + \ldots + a_r \geq x \cdot (r - l + 1)$.

Input

The first line of input contains one integer t ($1 \leq t \leq 10$): the number of test cases.

The descriptions of t test cases follow, three lines per test case.

In the first line you are given one integer n ($1 \leq n \leq 50\,000$): the number of integers in the array.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-100\,000 \leq a_i \leq 100\,000$).

The third line contains one integer x ($-100\,000 \leq x \leq 100\,000$).

Output

For each test case, print one integer: the maximum number of elements that you can select.

Example

input

```
4
5
1 2 3 4 5
2
10
2 4 2 4 2 4 2 4 2 4
3
3
-10 -5 -10
-8
3
9 9 -3
5
```

output

```
4
8
2
2
```

Note

In the first example, one valid way to select the elements is $[\underline{1}, 2, \underline{3}, \underline{4}, \underline{5}]$. All subsegments satisfy at least one of the criteria. For example, for the subsegment $l = 1$, $r = 2$ we have that the element 2 is not selected, satisfying the first criterion. For the subsegment $l = 3$, $r = 5$ we have $3 + 4 + 5 = 12 \geq 2 \cdot 3$, satisfying the second criterion.

We can't select all elements, because in this case for $l = 1$, $r = 2$ all elements are selected and we have $a_1 + a_2 = 3 < 2 \cdot 2$. Thus, the maximum number of selected elements is 4.

In the second example, one valid solution is $[\underline{2}, \underline{4}, 2, \underline{4}, \underline{2}, \underline{4}, 2, \underline{4}, \underline{2}, \underline{4}]$.

In the third example, one valid solution is $[\underline{-10}, -5, \underline{-10}]$.

In the fourth example, one valid solution is $[\underline{9}, \underline{9}, -3]$.

E. Lexicographically Small Enough

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

You are given two strings s and t of equal length n. In one move, you can swap any two adjacent characters of the string s.

You need to find the minimal number of operations you need to make string s lexicographically smaller than string t.

A string a is lexicographically smaller than a string b if and only if one of the following holds:

- a is a prefix of b, but $a \neq b$;
- in the first position where a and b differ, the string a has a letter that appears earlier in the alphabet than the corresponding letter in b.

Input

The first line of input contains one integer q ($1 \leq q \leq 10\,000$): the number of test cases.

The first line of each test case contains a single integer n ($1 \leq n \leq 10^5$).

The second line of each test case contains the string s consisting of n lowercase English letters.

The third line of each test case contains the string t consisting of n lowercase English letters.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print in a separate line the minimal number of operations you need to make string s lexicographically smaller than string t, or -1, if it's impossible.

Example

input

```
4
1
a
a
3
rll
rrr
3
caa
aca
5
ababa
aabba
```

output

```
-1
0
2
2
```

F. Tricolor Triangles

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

You are given a simple undirected graph with n vertices and m edges. Edge i is colored in the color c_i, which is either 1, 2, or 3, or left uncolored (in this case, $c_i = -1$).

You need to color all of the uncolored edges in such a way that for any three pairwise adjacent vertices $1 \leq a < b < c \leq n$, the colors of the edges $a \leftrightarrow b$, $b \leftrightarrow c$, and $a \leftrightarrow c$ are either pairwise different, or all equal. In case no such coloring exists, you need to determine that.

Input

The first line of input contains one integer t ($1 \leq t \leq 10$): the number of test cases.

The following lines contain the description of the test cases.

In the first line you are given two integers n and m ($3 \leq n \leq 64$, $0 \leq m \leq \min(256, \frac{n(n-1)}{2})$): the number of vertices and edges in the graph.

Each of the next m lines contains three integers a_i, b_i, and c_i ($1 \leq a_i, b_i \leq n$, $a_i \neq b_i$, c_i is either -1, 1, 2, or 3), denoting an edge between a_i and b_i with color c_i. It is guaranteed that no two edges share the same endpoints.

Output

For each test case, print m integers $d_1, d_2, \ldots, d_m$, where d_i is the color of the i-th edge in your final coloring. If there is no valid way to finish the coloring, print -1.

Example

input

```
4
```

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

output

```
1 2 3
1 1 1
1 2 2 3
-1
```

G. Just Add an Edge

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

You are given a directed acyclic graph with n vertices and m edges. For all edges $a \to b$ in the graph, $a < b$ holds.

You need to find the number of pairs of vertices x, y, such that $x > y$ and after adding the edge $x \to y$ to the graph, it has a Hamiltonian path.

Input

The first line of input contains one integer t ($1 \le t \le 5$): the number of test cases.

The next lines contains the descriptions of the test cases.

In the first line you are given two integers n and m ($1 \le n \le 150\,000$, $0 \le m \le \min(150\,000, \frac{n(n-1)}{2})$): the number of vertices and edges in the graph.

Each of the next m lines contains two integers a, b ($1 \le a < b \le n$), specifying an edge $a \to b$ in the graph. No edge $a \to b$ appears more than once.

Output

For each test case, print one integer: the number of pairs of vertices x, y, $x > y$, such that after adding the edge $x \to y$ to the graph, it has a Hamiltonian path.

Example

input

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

output

```
3
1
4
```

Note

In the first example, any edge $x \to y$ such that $x > y$ is valid, because there already is a path $1 \to 2 \to 3$.

In the second example only the edge $4 \to 1$ is valid. There is a path $3 \to 4 \to 1 \to 2$ if this edge is added.

In the third example you can add edges $2 \to 1$, $3 \to 1$, $4 \to 1$, $4 \to 2$.

H. Keep XOR Low

time limit per test: 1 second

memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an array $a_1, a_2, \ldots, a_n$ and an integer x.

Find the number of non-empty subsets of indices of this array $1 \le b_1 < b_2 < \ldots < b_k \le n$, such that for all pairs (i, j) where $1 \le i < j \le k$, the inequality $a_{b_i} \oplus a_{b_j} \le x$ is held. Here, $\oplus$ denotes the bitwise XOR operation. As the answer may be very large, output it modulo $998\,244\,353$.

Input

The first line of the input contains two integers n and x ($1 \le n \le 150\,000$, $0 \le x < 2^{30}$). Here, n is the size of the array.

The next line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i < 2^{30}$): the array itself.

Output

Print one integer: the number of non-empty subsets such that the bitwise XOR of every pair of elements is at most x, modulo $998\,244\,353$.

Examples

input

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

output

```
8
```

input

```
3 6
4 2 2
```

output

```
7
```

input

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

output

```
6
```