Topic Stream Mashup: Bitwise Operations

A. Mocha and Math

1 second, 256 megabytes

Mocha is a young girl from high school. She has learned so much interesting knowledge from her teachers, especially her math teacher. Recently, Mocha is learning about binary system and very interested in bitwise operation.

This day, Mocha got a sequence a of length n. In each operation, she can select an arbitrary interval $[l, r]$ and for all values i ($0 \le i \le r - l$), replace a_{l+i} with $a_{l+i} \,\&\, a_{r-i}$ at the same time, where $\&$ denotes the bitwise AND operation. This operation can be performed **any number of times**.

For example, if $n = 5$, the array is $[a_1, a_2, a_3, a_4, a_5]$, and Mocha selects the interval $[2, 5]$, then the new array is $[a_1, a_2 \,\&\, a_5, a_3 \,\&\, a_4, a_4 \,\&\, a_3, a_5 \,\&\, a_2]$.

Now Mocha wants to minimize the maximum value in the sequence. As her best friend, can you help her to get the answer?

Input

Each test contains multiple test cases.

The first line contains a single integer t ($1 \le t \le 100$) — the number of test cases. Each test case consists of two lines.

The first line of each test case contains a single integer n ($1 \le n \le 100$) — the length of the sequence.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$).

Output

For each test case, print one integer — the minimal value of the maximum value in the sequence.

input
4 2 1 2 3 1 1 3 4 3 11 3 7 5 11 7 15 3 7
output
0 1 3 3

In the first test case, Mocha can choose the interval $[1, 2]$, then the sequence becomes $[0, 0]$, where the first element is $1 \,\&\, 2$, and the second element is $2 \,\&\, 1$.

In the second test case, Mocha can choose the interval $[1, 3]$, then the sequence becomes $[1, 1, 1]$, where the first element is $1 \,\&\, 3$, the second element is $1 \,\&\, 1$, and the third element is $3 \,\&\, 1$.

B. AND 0, Sum Big

2 seconds, 256 megabytes

Baby Badawy's first words were "AND 0 SUM BIG", so he decided to solve the following problem. Given two integers n and k, count the number of arrays of length n such that:

- all its elements are integers between 0 and $2^k - 1$ (inclusive);
- the bitwise AND of all its elements is 0;
- the sum of its elements is as large as possible.

Since the answer can be very large, print its remainder when divided by $10^9 + 7$.

Input

The first line contains an integer t ($1 \le t \le 10$) — the number of test cases you need to solve.

Each test case consists of a line containing two integers n and k ($1 \le n \le 10^5$, $1 \le k \le 20$).

Output

For each test case, print the number of arrays satisfying the conditions. Since the answer can be very large, print its remainder when divided by $10^9 + 7$.

input
2 2 2 100000 20
output
4 226732710

In the first example, the 4 arrays are:

- $[3, 0]$,
- $[0, 3]$,
- $[1, 2]$,
- $[2, 1]$.

C. Rock and Lever

1 second, 256 megabytes

> "You must lift the dam. With a lever. I will give it to you. You must block the canal. With a rock. I will not give the rock to you."

Danik urgently needs rock and lever! Obviously, the easiest way to get these things is to ask Hermit Lizard for them.

Hermit Lizard agreed to give Danik the lever. But to get a stone, Danik needs to solve the following task.

You are given a positive integer n, and an array a of positive integers. The task is to calculate the number of such pairs (i, j) that $i < j$ and $a_i \,\&\, a_j \ge a_i \oplus a_j$, where $\&$ denotes the bitwise AND operation, and $\oplus$ denotes the bitwise XOR operation.

Danik has solved this task. But can you solve it?

Input

Each test contains multiple test cases.

The first line contains one positive integer t ($1 \le t \le 10$) denoting the number of test cases. Description of the test cases follows.

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

The second line contains n positive integers a_i ($1 \le a_i \le 10^9$) — elements of the array.

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

Output

For every test case print one non-negative integer — the answer to the problem.

input
5 5 1 4 3 7 10 3 1 1 1 4 6 2 5 3 2 2 4 1 1

output

```
1
3
2
0
0
```

In the first test case there is only one pair: $(4, 7)$: for it $4 \ \& \ 7 = 4$, and $4 \oplus 7 = 3$.

In the second test case all pairs are good.

In the third test case there are two pairs: $(6, 5)$ and $(2, 3)$.

In the fourth test case there are no good pairs.

D. Palindrome Pairs

4.0 s, 256 megabytes

After learning a lot about space exploration, a little girl named Ana wants to change the subject.

Ana is a girl who loves palindromes (string that can be read the same backwards as forward). She has learned how to check for a given string whether it's a palindrome or not, but soon she grew tired of this problem, so she came up with a more interesting one and she needs your help to solve it:

You are given an array of strings which consist of only small letters of the alphabet. Your task is to find **how many** palindrome pairs are there in the array. A palindrome pair is a pair of strings such that the following condition holds: **at least one** permutation of the concatenation of the two strings is a palindrome. In other words, if you have two strings, let's say "`aab`" and "`abcac`", and you concatenate them into "`aababcac`", we have to check if there exists a permutation of this new string such that it is a palindrome (in this case there exists the permutation "`aabccbaa`").

Two pairs are considered different if the strings are located on **different indices**. The pair of strings with indices (i, j) is considered **the same** as the pair (j, i).

Input

The first line contains a positive integer N ($1 \leq N \leq 100\,000$), representing the length of the input array.

Eacg of the next N lines contains a string (consisting of lowercase English letters from '`a`' to '`z`') — an element of the input array.

The total number of characters in the input array will be less than $1\,000\,000$.

Output

Output one number, representing **how many palindrome pairs** there are in the array.

input

```
3
aa
bb
cd
```

output

```
1
```

input

```
6
aab
abcac
dffe
ed
aa
aade
```

output

```
6
```

The first example:

1. `aa` + `bb` → `abba`.

The second example:

1. `aab` + `abcac` = `aababcac` → `aabccbaa`
2. `aab` + `aa` = `aabaa`
3. `abcac` + `aa` = `abcacaa` → `aacbcaa`
4. `dffe` + `ed` = `dffeed` → `fdeedf`
5. `dffe` + `aade` = `dffeaade` → `adfaafde`
6. `ed` + `aade` = `edaade` → `aeddea`

E. Little Girl and Maximum XOR

1 second, 256 megabytes

A little girl loves problems on bitwise operations very much. Here's one of them.

You are given two integers l and r. Let's consider the values of $a \oplus b$ for all pairs of integers a and b ($l \leq a \leq b \leq r$). Your task is to find the maximum value among all considered ones.

Expression $x \oplus y$ means applying bitwise excluding or operation to integers x and y. The given operation exists in all modern programming languages, for example, in languages C++ and $Java$ it is represented as "^", in $Pascal$ — as "`xor`".

Input

The single line contains space-separated integers l and r ($1 \leq l \leq r \leq 10^{18}$).

Please, do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Output

In a single line print a single integer — the maximum value of $a \oplus b$ for all pairs of integers a, b ($l \leq a \leq b \leq r$).

input

```
1 2
```

output

```
3
```

input

```
8 16
```

output

```
31
```

input

```
1 1
```

output

```
0
```

F. Apollo versus Pan

2 seconds, 256 megabytes

Only a few know that Pan and Apollo weren't only battling for the title of the GOAT musician. A few millenniums later, they also challenged each other in math (or rather in fast calculations). The task they got to solve is the following:

Let $x_1, x_2, \ldots, x_n$ be the sequence of n non-negative integers. Find this value:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} (x_i \ \& \ x_j) \cdot (x_j \mid x_k)$$

Here $\&$ denotes the bitwise and, and $\mid$ denotes the bitwise or.

Pan and Apollo could solve this in a few seconds. Can you do it too? For convenience, find the answer modulo $10^9 + 7$.

Input

The first line of the input contains a single integer t ($1 \leq t \leq 1\,000$) denoting the number of test cases, then t test cases follow.

The first line of each test case consists of a single integer n ($1 \leq n \leq 5 \cdot 10^5$), the length of the sequence. The second one contains n non-negative integers $x_1, x_2, \ldots, x_n$ ($0 \leq x_i < 2^{60}$), elements of the sequence.

The sum of n over all test cases will not exceed $5 \cdot 10^5$.

Output

Print t lines. The i-th line should contain the answer to the i-th text case.

input

```
8
2
1 7
3
1 2 4
4
5 5 5 5
5
6 2 2 1 0
1
0
1
1
6
1 12 123 1234 12345 123456
5
536870912 536870911 1152921504606846975 1152921504606846974
1152921504606846973
```

output

```
128
91
1600
505
0
1
502811676
264880351
```

G. Bitwise Formula

3 seconds, 512 megabytes

Bob recently read about bitwise operations used in computers: `AND`, `OR` and `XOR`. He have studied their properties and invented a new game.

Initially, Bob chooses integer m, bit depth of the game, which means that all numbers in the game will consist of m bits. Then he asks Peter to choose some m-bit number. After that, Bob computes the values of n variables. Each variable is assigned either a constant m-bit number or result of bitwise operation. Operands of the operation may be either variables defined before, or the number, chosen by Peter. After that, Peter's score equals to the sum of all variable values.

Bob wants to know, what number Peter needs to choose to get the minimum possible score, and what number he needs to choose to get the maximum possible score. In both cases, if there are several ways to get the same score, find the minimum number, which he can choose.

Input

The first line contains two integers n and m, the number of variables and bit depth, respectively ($1 \le n \le 5000$; $1 \le m \le 1000$).

The following n lines contain descriptions of the variables. Each line describes exactly one variable. Description has the following format: name of a new variable, space, sign "`:=`", space, followed by one of:

1. Binary number of exactly m bits.
2. The first operand, space, bitwise operation ("`AND`", "`OR`" or "`XOR`"), space, the second operand. Each operand is either the name of variable defined before or symbol '`?`', indicating the number chosen by Peter.

Variable names are strings consisting of lowercase Latin letters with length at most 10. All variable names are different.

Output

In the first line output the minimum number that should be chosen by Peter, to make the sum of all variable values minimum possible, in the second line output the minimum number that should be chosen by Peter, to make the sum of all variable values maximum possible. Both numbers should be printed as m-bit binary numbers.

input

```
3 3
a := 101
b := 011
c := ? XOR b
```

output

```
011
100
```

input

```
5 1
a := 1
bb := 0
cx := ? OR a
d := ? XOR ?
e := d AND bb
```

output

```
0
0
```

In the first sample if Peter chooses a number 011_2, then $a = 101_2$, $b = 011_2$, $c = 000_2$, the sum of their values is 8. If he chooses the number 100_2, then $a = 101_2$, $b = 011_2$, $c = 111_2$, the sum of their values is 15.

For the second test, the minimum and maximum sum of variables a, bb, cx, d and e is 2, and this sum doesn't depend on the number chosen by Peter, so the minimum Peter can choose is 0.

H. Kefa and Dishes

2 seconds, 256 megabytes

When Kefa came to the restaurant and sat at a table, the waiter immediately brought him the menu. There were n dishes. Kefa knows that he needs exactly m dishes. But at that, he doesn't want to order the same dish twice to taste as many dishes as possible.

Kefa knows that the i-th dish gives him a_i units of satisfaction. But some dishes do not go well together and some dishes go very well together. Kefa set to himself k rules of eating food of the following type — if he eats dish x exactly before dish y (there should be no other dishes between x and y), then his satisfaction level raises by c.

Of course, our parrot wants to get some maximal possible satisfaction from going to the restaurant. Help him in this hard task!

Input

The first line of the input contains three space-separated numbers, n, m and k ($1 \le m \le n \le 18$, $0 \le k \le n * (n - 1)$) — the number of dishes on the menu, the number of portions Kefa needs to eat to get full and the number of eating rules.

The second line contains n space-separated numbers a_i, ($0 \le a_i \le 10^9$) — the satisfaction he gets from the i-th dish.

Next k lines contain the rules. The i-th rule is described by the three numbers x_i, y_i and c_i ($1 \le x_i, y_i \le n$, $0 \le c_i \le 10^9$). That means that if you eat dish x_i right before dish y_i, then the Kefa's satisfaction increases by c_i. It is guaranteed that there are no such pairs of indexes i and j ($1 \le i < j \le k$), that $x_i = x_j$ and $y_i = y_j$.

Output

In the single line of the output print the maximum satisfaction that Kefa can get from going to the restaurant.

input

```
2 2 1
1 1
2 1 1
```

output

```
3
```

input

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

output

```
12
```

In the first sample it is best to first eat the second dish, then the first one. Then we get one unit of satisfaction for each dish and plus one more for the rule.

In the second test the fitting sequences of choice are 4 2 1 or 2 1 4. In both cases we get satisfaction 7 for dishes and also, if we fulfill rule 1, we get an additional satisfaction 5.

I. 505

1 second, 256 megabytes

A binary matrix is called **good** if every **even** length square sub-matrix has an **odd** number of ones.

Given a binary matrix a consisting of n rows and m columns, determine the minimum number of cells you need to change to make it good, or report that there is no way to make it good at all.

All the terms above have their usual meanings — refer to the Notes section for their formal definitions.

Input

The first line of input contains two integers n and m ($1 \leq n \leq m \leq 10^6$ and $n \cdot m \leq 10^6$) — the number of rows and columns in a, respectively.

The following n lines each contain m characters, each of which is one of `0` and `1`. If the j-th character on the i-th line is `1`, then $a_{i,j} = 1$. Similarly, if the j-th character on the i-th line is `0`, then $a_{i,j} = 0$.

Output

Output the minimum number of cells you need to change to make a good, or output -1 if it's not possible at all.

input

```
3 3
101
001
110
```

output

```
2
```

input

```
7 15
000100001010010
100111010110001
101101111100100
010000111111010
111010010100001
000011001111101
111111011010011
```

output

```
-1
```

In the first case, changing $a_{1,1}$ to 0 and $a_{2,2}$ to 1 is enough.

You can verify that there is no way to make the matrix in the second case good.

Some definitions —

- A binary matrix is one in which every element is either 1 or 0.
- A sub-matrix is described by 4 parameters — r_1, r_2, c_1, and c_2; here, $1 \leq r_1 \leq r_2 \leq n$ and $1 \leq c_1 \leq c_2 \leq m$.
- This sub-matrix contains all elements $a_{i,j}$ that satisfy both $r_1 \leq i \leq r_2$ and $c_1 \leq j \leq c_2$.
- A sub-matrix is, further, called an even length square if $r_2 - r_1 = c_2 - c_1$ and $r_2 - r_1 + 1$ is divisible by 2.

J. Xor Tree

2 seconds, 256 megabytes

For a given sequence of **distinct** non-negative integers $(b_1, b_2, \ldots, b_k)$ we determine if it is **good** in the following way:

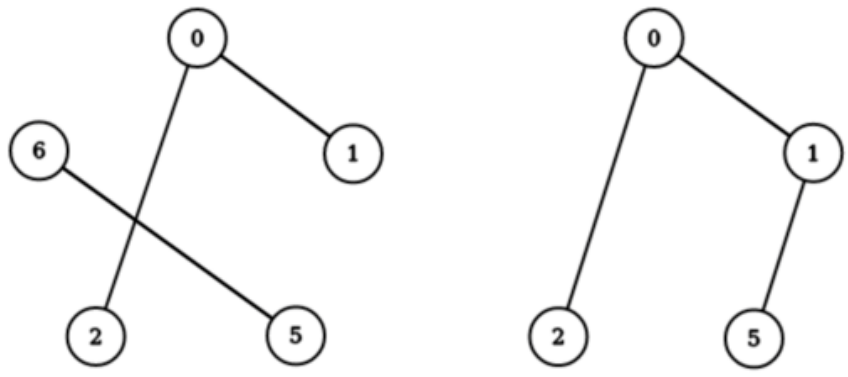
- Consider a graph on k nodes, with numbers from b_1 to b_k written on them.
- For every i from 1 to k: find such j ($1 \leq j \leq k$, $j \neq i$), for which $(b_i \oplus b_j)$ **is the smallest** among all such j, where $\oplus$ denotes the operation of bitwise XOR (https://en.wikipedia.org/wiki/Bitwise_operation#XOR). Next, draw an **undirected** edge between vertices with numbers b_i and b_j in this graph.
- We say that the sequence is **good** if and only if the resulting graph forms a **tree** (is connected and doesn't have any simple cycles).

It is possible that for some numbers b_i and b_j, you will try to add the edge between them twice. Nevertheless, you will add this edge only once.

You can find an example below (the picture corresponding to the first test case).

Sequence $(0, 1, 5, 2, 6)$ **is not** good as we **cannot** reach 1 from 5.

However, sequence $(0, 1, 5, 2)$ **is** good.



You are given a sequence $(a_1, a_2, \ldots, a_n)$ of **distinct** non-negative integers. You would like to remove some of the elements (possibly none) to make the **remaining** sequence good. What is the minimum possible number of removals required to achieve this goal?

It can be shown that for any sequence, we can remove some number of elements, leaving at least 2, so that the remaining sequence is good.

Input

The first line contains a single integer n ($2 \leq n \leq 200,000$) — length of the sequence.

The second line contains n **distinct** non-negative integers $a_1, a_2, \ldots, a_n$ ($0 \leq a_i \leq 10^9$) — the elements of the sequence.

Output

You should output exactly one integer — the minimum possible number of elements to remove in order to make the remaining sequence good.

input

```
5
0 1 5 2 6
```

output

```
1
```

input

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

output

```
2
```

Note that numbers which you remove **don't** impact the procedure of telling whether the resulting sequence is good.

It is possible that for some numbers b_i and b_j, you will try to add the edge between them twice. Nevertheless, you will add this edge only once.

K. Bitwise Queries (Hard Version)

4 seconds, 256 megabytes

The only difference between the easy and hard versions is the constraints on the number of queries.

This is an interactive problem.

Ridbit has a hidden array a of n integers which he wants Ashish to guess. Note that n is a **power of two**. Ashish is allowed to ask three different types of queries. They are of the form

- `AND` i j: ask for the bitwise AND of elements a_i and a_j $(1 \leq i, j \leq n, i \neq j)$
- `OR` i j: ask for the bitwise OR of elements a_i and a_j $(1 \leq i, j \leq n, i \neq j)$
- `XOR` i j: ask for the bitwise XOR of elements a_i and a_j $(1 \leq i, j \leq n, i \neq j)$

Can you help Ashish guess the elements of the array?

In this version, each element takes a value in the range $[0, n-1]$ (inclusive) and Ashish can ask no more than $n+1$ queries.

Input

The first line of input contains one integer n ($4 \leq n \leq 2^{16}$) — the length of the array. It is guaranteed that n is a **power of two**.

Interaction

To ask a query print a single line containing one of the following (without quotes)

- "AND i j"
- "OR i j"
- "XOR i j"

where i and j $(1 \le i, j \le n, i \ne j)$ denote the indices being queried. For each query, you will receive an integer x whose value depends on the type of query. If the indices queried are invalid or you exceed the number of queries however, you will get $x = -1$. In this case, you should terminate the program immediately.

When you have guessed the elements of the array, print a single line "! " (without quotes), followed by n space-separated integers — the elements of the array.

Guessing the array does **not** count towards the number of queries asked.

The interactor is not adaptive. The array a does not change with queries.

After printing a query do not forget to output the end of the line and flush the output. Otherwise, you will get Idleness limit exceeded. To do this, use:

- fflush(stdout) or cout.flush() in C++;
- System.out.flush() in Java;
- flush(output) in Pascal;
- stdout.flush() in Python;
- see the documentation for other languages.

Hacks

To hack the solution, use the following test format:

On the first line print a single integer n $(4 \le n \le 2^{16})$ — the length of the array. It **must** be a power of 2. The next line should contain n space-separated integers in the range $[0, n-1]$ — the array a.

input
4 0 2 3
output
OR 1 2 OR 2 3 XOR 2 4 ! 0 0 2 3

The array a in the example is $[0, 0, 2, 3]$.

L. Enormous XOR

1 second, 256 megabytes

You are given two integers l and r in binary representation. Let $g(x, y)$ be equal to the bitwise XOR of all integers from x to y inclusive (that is $x \oplus (x+1) \oplus \cdots \oplus (y-1) \oplus y$). Let's define $f(l, r)$ as the maximum of all values of $g(x, y)$ satisfying $l \le x \le y \le r$.

Output $f(l, r)$.

Input

The first line contains a single integer n $(1 \le n \le 10^6)$ — the length of the binary representation of r.

The second line contains the binary representation of l — a string of length n consisting of digits 0 and 1 $(0 \le l < 2^n)$.

The third line contains the binary representation of r — a string of length n consisting of digits 0 and 1 $(0 \le r < 2^n)$.

It is guaranteed that $l \le r$. The binary representation of r does not contain any extra leading zeros (if $r = 0$, the binary representation of it consists of a single zero). The binary representation of l is preceded with leading zeros so that its length is equal to n.

Output

In a single line output the value of $f(l, r)$ for the given pair of l and r in binary representation without extra leading zeros.

input
7 0010011 1111010
output
1111111

input
4 1010 1101
output
1101

In sample test case $l = 19$, $r = 122$. $f(x, y)$ is maximal and is equal to 127, with $x = 27$, $y = 100$, for example.

M. Make It Ascending

7 seconds, 512 megabytes

You are given an array a consisting of n elements. You may apply several operations (possibly zero) to it.

During each operation, you choose two indices i and j $(1 \le i, j \le n; i \ne j)$, increase a_j by a_i, and remove the i-th element from the array (so the indices of all elements to the right to it decrease by 1, and n also decreases by 1).

Your goal is to make the array a strictly ascending. That is, the condition $a_1 < a_2 < \cdots < a_n$ should hold (where n is the resulting size of the array).

Calculate the minimum number of actions required to make the array strictly ascending.

Input

The first line contains one integer T $(1 \le T \le 10000)$ — the number of test cases.

Each test case consists of two lines. The first line contains one integer n ($1 \le n \le 15$) — the number of elements in the initial array a.

The second line contains n integers a_1, a_2, ..., a_n $(1 \le a_i \le 10^6)$.

It is guaranteed that:

- the number of test cases having $n \ge 5$ is not greater than 5000;
- the number of test cases having $n \ge 8$ is not greater than 500;
- the number of test cases having $n \ge 10$ is not greater than 100;
- the number of test cases having $n \ge 11$ is not greater than 50;
- the number of test cases having $n \ge 12$ is not greater than 25;
- the number of test cases having $n \ge 13$ is not greater than 10;
- the number of test cases having $n \ge 14$ is not greater than 3;
- the number of test cases having $n \ge 15$ is not greater than 1.

Output

For each test case, print the answer as follows:

In the first line, print k — the minimum number of operations you have to perform. Then print k lines, each containing two indices i and j for the corresponding operation. Note that the numeration of elements in the array changes after removing elements from it. If there are multiple optimal sequences of operations, print any one of them.

input
4 8 2 1 3 5 1 2 4 5 15 16384 8192 4096 2048 1024 512 256 128 64 32 16 8 4 2 1 2 3 3 14 1 2 3 4 5 6 7 8 9 10 11 12 13 14

output

```
3
6 8
1 6
4 1
7
1 15
1 13
1 11
1 9
1 7
1 5
1 3
1
2 1
0
```

In the first test case, the sequence of operations changes a as follows:

$[2, 1, 3, 5, 1, 2, 4, 5] \rightarrow [2, 1, 3, 5, 1, 4, 7] \rightarrow [1, 3, 5, 1, 6, 7] \rightarrow [2, 3, 5, 6, 7]$

.