Codeforces Round #659 (Div. 1)

A. String Transformation 1

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

Note that the only difference between `String Transformation 1` and `String Transformation 2` is in the move Koa does. In this version the letter y Koa selects must be strictly greater alphabetically than x (read statement for better understanding). You can make hacks in these problems independently.

Koa the Koala has two strings A and B of the same length n ($|A| = |B| = n$) consisting of the first 20 lowercase English alphabet letters (ie. from `a` to `t`).

In one move Koa:

1. selects some subset of positions $p_1, p_2, \ldots, p_k$ ($k \geq 1; 1 \leq p_i \leq n; p_i \neq p_j$ if $i \neq j$) of A such that $A_{p_1} = A_{p_2} = \ldots = A_{p_k} = x$ (ie. all letters on this positions are equal to some letter x).
2. selects a letter y (from the first 20 lowercase letters in English alphabet) such that $y > x$ (ie. letter y is **strictly greater** alphabetically than x).
3. sets each letter in positions $p_1, p_2, \ldots, p_k$ to letter y. More formally: for each i ($1 \leq i \leq k$) Koa sets $A_{p_i} = y$.
   Note that you can only modify letters in string A.

Koa wants to know the smallest number of moves she has to do to make strings equal to each other ($A = B$) or to determine that there is no way to make them equal. Help her!

Input

Each test contains multiple test cases. The first line contains t ($1 \leq t \leq 10$) — the number of test cases. Description of the test cases follows.

The first line of each test case contains one integer n ($1 \leq n \leq 10^5$) — the length of strings A and B.

The second line of each test case contains string A ($|A| = n$).

The third line of each test case contains string B ($|B| = n$).

Both strings consists of the first 20 lowercase English alphabet letters (ie. from `a` to `t`).

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

Output

For each test case:

Print on a single line the smallest number of moves she has to do to make strings equal to each other ($A = B$) or -1 if there is no way to make them equal.

Example

input
5 3 aab bcc 4 cabc abcb 3 abc tsr 4 aabd cccd 5 abcbd bcdda
output
2 -1 3 2 -1

Note

- In the 1-st test case Koa:
  1. selects positions 1 and 2 and sets $A_1 = A_2 = \texttt{b}$ ($aab \rightarrow bbb$).
  2. selects positions 2 and 3 and sets $A_2 = A_3 = \texttt{c}$ ($bbb \rightarrow bcc$).
- In the 2-nd test case Koa has no way to make string A equal B.
- In the 3-rd test case Koa:
  1. selects position 1 and sets $A_1 = \texttt{t}$ ($abc \rightarrow tbc$).
  2. selects position 2 and sets $A_2 = \texttt{s}$ ($tbc \rightarrow tsc$).
  3. selects position 3 and sets $A_3 = \texttt{r}$ ($tsc \rightarrow tsr$).

B. GameGame

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

Koa the Koala and her best friend want to play a game.

The game starts with an array a of length n consisting of non-negative integers. Koa and her best friend move in turns and each have initially a score equal to 0. Koa starts.

Let's describe a move in the game:

- During his move, a player chooses any element of the array and removes it from this array, xor-ing it with the current score of the player.
  More formally: if the current score of the player is x and the chosen element is y, his new score will be $x \oplus y$. Here $\oplus$ denotes bitwise XOR operation.

  Note that after a move element y is removed from a.
- The game ends when the array is empty.

At the end of the game the winner is the player with the maximum score. If both players have the same score then it's a draw.

If both players play optimally find out whether Koa will win, lose or draw the game.

Input

Each test contains multiple test cases. The first line contains t ($1 \leq t \leq 10^4$) — the number of test cases. Description of the test cases follows.

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

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

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

Output

For each test case print:

- `WIN` if Koa will win the game.
- `LOSE` if Koa will lose the game.
- `DRAW` if the game ends in a draw.

Examples

input
3 3 1 2 2 3 2 2 3 5 0 0 0 2 2
output
WIN LOSE DRAW

input
4 5 4 1 5 1 3

4
1 0 1 6
1
0
2
5 4

output

WIN
WIN
DRAW
WIN

Note

In testcase 1 of the first sample we have:

$a = [1, 2, 2]$. Here Koa chooses 1, other player has to choose 2, Koa chooses another 2. Score for Koa is $1 \oplus 2 = 3$ and score for other player is 2 so Koa wins.

C. String Transformation 2

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

Note that the only difference between `String Transformation 1` and `String Transformation 2` is in the move Koa does. In this version the letter y Koa selects can be any letter from the first 20 lowercase letters of English alphabet (read statement for better understanding). You can make hacks in these problems independently.

Koa the Koala has two strings A and B of the same length n ($|A| = |B| = n$) consisting of the first 20 lowercase English alphabet letters (ie. from `a` to `t`).

In one move Koa:

1. selects some subset of positions $p_1, p_2, \ldots, p_k$ ($k \geq 1; 1 \leq p_i \leq n; p_i \neq p_j$ if $i \neq j$) of A such that $A_{p_1} = A_{p_2} = \ldots = A_{p_k} = x$ (ie. all letters on this positions are equal to some letter x).
2. selects **any** letter y (from the first 20 lowercase letters in English alphabet).
3. sets each letter in positions $p_1, p_2, \ldots, p_k$ to letter y. More formally: for each i ($1 \leq i \leq k$) Koa sets $A_{p_i} = y$.
   Note that you can only modify letters in string A.

Koa wants to know the smallest number of moves she has to do to make strings equal to each other ($A = B$) or to determine that there is no way to make them equal. Help her!

Input

Each test contains multiple test cases. The first line contains t ($1 \leq t \leq 10$) — the number of test cases. Description of the test cases follows.

The first line of each test case contains one integer n ($1 \leq n \leq 10^5$) — the length of strings A and B.

The second line of each test case contains string A ($|A| = n$).

The third line of each test case contains string B ($|B| = n$).

Both strings consists of the first 20 lowercase English alphabet letters (ie. from `a` to `t`).

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

Output

For each test case:

Print on a single line the smallest number of moves she has to do to make strings equal to each other ($A = B$) or -1 if there is no way to make them equal.

Example

input

5
3
aab
bcc
4
cabc
abcb
3
abc
tsr
4
aabd
cccd

5
abcbd
bcdda

output

2
3
3
2
4

Note

- In the 1-st test case Koa:
  1. selects positions 1 and 2 and sets $A_1 = A_2 =$ b ($aab \rightarrow bbb$).
  2. selects positions 2 and 3 and sets $A_2 = A_3 =$ c ($bbb \rightarrow bcc$).
- In the 2-nd test case Koa:
  1. selects positions 1 and 4 and sets $A_1 = A_4 =$ a ($cabc \rightarrow aaba$).
  2. selects positions 2 and 4 and sets $A_2 = A_4 =$ b ($aaba \rightarrow abbb$).
  3. selects position 3 and sets $A_3 =$ c ($abbb \rightarrow abcb$).
- In the 3-rd test case Koa:
  1. selects position 1 and sets $A_1 =$ t ($abc \rightarrow tbc$).
  2. selects position 2 and sets $A_2 =$ s ($tbc \rightarrow tsc$).
  3. selects position 3 and sets $A_3 =$ r ($tsc \rightarrow tsr$).

D. Rearrange

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

Koa the Koala has a matrix A of n rows and m columns. Elements of this matrix are distinct integers from 1 to $n \cdot m$ (each number from 1 to $n \cdot m$ appears exactly once in the matrix).

For any matrix M of n rows and m columns let's define the following:

- The i-th row of M is defined as $R_i(M) = [M_{i1}, M_{i2}, \ldots, M_{im}]$ for all i ($1 \leq i \leq n$).
- The j-th column of M is defined as $C_j(M) = [M_{1j}, M_{2j}, \ldots, M_{nj}]$ for all j ($1 \leq j \leq m$).

Koa defines $S(A) = (X, Y)$ as the spectrum of A, where X is the set of the maximum values in rows of A and Y is the set of the maximum values in columns of A.

More formally:

- $X = \{\max(R_1(A)), \max(R_2(A)), \ldots, \max(R_n(A))\}$
- $Y = \{\max(C_1(A)), \max(C_2(A)), \ldots, \max(C_m(A))\}$

Koa asks you to find some matrix A' of n rows and m columns, such that each number from 1 to $n \cdot m$ appears exactly once in the matrix, and the following conditions hold:

- $S(A') = S(A)$
- $R_i(A')$ is bitonic for all i ($1 \leq i \leq n$)
- $C_j(A')$ is bitonic for all j ($1 \leq j \leq m$)

An array t ($t_1, t_2, \ldots, t_k$) is called bitonic if it first increases and then decreases.
More formally: t is bitonic if there exists some position p ($1 \leq p \leq k$) such that: $t_1 < t_2 < \ldots < t_p > t_{p+1} > \ldots > t_k$.

Help Koa to find such matrix or to determine that it doesn't exist.

Input

The first line of the input contains two integers n and m ($1 \leq n, m \leq 250$) — the number of rows and columns of A.

Each of the ollowing n lines contains m integers. The j-th integer in the i-th line denotes element A_{ij} ($1 \leq A_{ij} \leq n \cdot m$) of matrix A. It is guaranteed that every number from 1 to $n \cdot m$ appears exactly once among elements of the matrix.

Output

If such matrix doesn't exist, print -1 on a single line.

Otherwise, the output must consist of n lines, each one consisting of m space separated integers — a description of A'.

The j-th number in the i-th line represents the element A'_{ij}.

Every integer from 1 to $n \cdot m$ should appear exactly once in A', every row and column in A' must be bitonic and $S(A) = S(A')$

must hold.

If there are many answers print any.

Examples

input
3 3 3 5 6 1 7 9 4 8 2
output
9 5 1 7 8 2 3 6 4

input
2 2 4 1 3 2
output
4 1 3 2

input
3 4 12 10 8 6 3 4 5 7 2 11 9 1
output
12 8 6 1 10 11 9 2 3 4 5 7

Note

Let's analyze the first sample:

For matrix A we have:

- Rows:
  - $R_1(A) = [3, 5, 6]; \max(R_1(A)) = 6$
  - $R_2(A) = [1, 7, 9]; \max(R_2(A)) = 9$
  - $R_3(A) = [4, 8, 2]; \max(R_3(A)) = 8$
- Columns:
  - $C_1(A) = [3, 1, 4]; \max(C_1(A)) = 4$
  - $C_2(A) = [5, 7, 8]; \max(C_2(A)) = 8$
  - $C_3(A) = [6, 9, 2]; \max(C_3(A)) = 9$

- $X = \{\max(R_1(A)), \max(R_2(A)), \max(R_3(A))\} = \{6, 9, 8\}$
- $Y = \{\max(C_1(A)), \max(C_2(A)), \max(C_3(A))\} = \{4, 8, 9\}$
- So $S(A) = (X, Y) = (\{6, 9, 8\}, \{4, 8, 9\})$

For matrix A' we have:

- Rows:
  - $R_1(A') = [9, 5, 1]; \max(R_1(A')) = 9$
  - $R_2(A') = [7, 8, 2]; \max(R_2(A')) = 8$
  - $R_3(A') = [3, 6, 4]; \max(R_3(A')) = 6$
- Columns:
  - $C_1(A') = [9, 7, 3]; \max(C_1(A')) = 9$
  - $C_2(A') = [5, 8, 6]; \max(C_2(A')) = 8$
  - $C_3(A') = [1, 2, 4]; \max(C_3(A')) = 4$

- Note that each of this arrays are bitonic.
- $X = \{\max(R_1(A')), \max(R_2(A')), \max(R_3(A'))\} = \{9, 8, 6\}$
- $Y = \{\max(C_1(A')), \max(C_2(A')), \max(C_3(A'))\} = \{9, 8, 4\}$
- So $S(A') = (X, Y) = (\{9, 8, 6\}, \{9, 8, 4\})$

E. Strange Operation

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

Koa the Koala has a binary string s of length n. Koa can perform no more than $n-1$ (possibly zero) operations of the following form:

In one operation Koa selects positions i and $i+1$ for some i with $1 \le i < |s|$ and sets s_i to $max(s_i, s_{i+1})$. Then Koa deletes position $i+1$ from s (after the removal, the remaining parts are concatenated).

Note that after every operation the length of s decreases by 1.

How many different binary strings can Koa obtain by doing no more than $n-1$ (possibly zero) operations modulo 10^9+7 (1000000007)?

Input
The only line of input contains binary string s ($1 \le |s| \le 10^6$). For all i ($1 \le i \le |s|$) $s_i = 0$ or $s_i = 1$.

Output
On a single line print the answer to the problem modulo 10^9+7 (1000000007).

Examples

input
000
output
3

input
0101
output
6

input
0001111
output
16

input
00101100011100
output
477

Note
In the first sample Koa can obtain binary strings: 0, 00 and 000.

In the second sample Koa can obtain binary strings: 1, 01, 11, 011, 101 and 0101. For example:

- to obtain 01 from 0101 Koa can operate as follows: $0101 \to 0(10)1 \to 011 \to 0(11) \to 01$.
- to obtain 11 from 0101 Koa can operate as follows: $0101 \to (01)01 \to 101 \to 1(01) \to 11$.

Parentheses denote the two positions Koa selected in each operation.

F. Special Edges

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Koa the Koala has a **directed** graph G with n nodes and m edges. Each edge has a capacity associated with it. Exactly k edges of the graph, numbered from 1 to k, are special, such edges initially have a capacity equal to 0.

Koa asks you q queries. In each query she gives you k integers $w_1, w_2, \ldots, w_k$. This means that capacity of the i-th special edge becomes w_i (and other capacities remain the same).

Koa wonders: what is the maximum flow that goes from node 1 to node n after each such query?

Help her!

Input

The first line of the input contains four integers n, m, k, q ($2 \le n \le 10^4$, $1 \le m \le 10^4$, $1 \le k \le \min(10, m)$, $1 \le q \le 2 \cdot 10^5$) — the number of nodes, the number of edges, the number of special edges and the number of queries.

Each of the next m lines contains three integers u, v, w ($1 \le u, v \le n$; $0 \le w \le 25$) — the description of a directed edge from node u to node v with capacity w.

Edges are numbered starting from 1 in the same order they are listed in the input. The first k edges are the special edges. It is guaranteed that $w_i = 0$ for all i with $1 \le i \le k$.

Each of the next q lines contains k integers $w_1, w_2, \ldots, w_k$ ($0 \le w_i \le 25$) — the description of the query. w_i denotes the capacity of i-th edge.

Output

For the i-th query, print one integer res_i — the maximum flow that can be obtained from node 1 to node n given the i-th query's special edge weights.

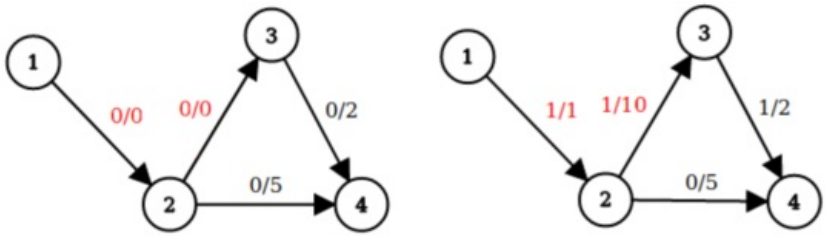
Examples

input
2 1 1 3 1 2 0 0 1 2
output
0 1 2

input
4 4 2 5 1 2 0 2 3 0 2 4 5 3 4 2 0 0 1 10 10 0 7 1 7 2
output
0 1 5 6 7

Note

For the second sample, the following images correspond to the first two queries (from left to right respectively). For each edge there is a pair `flow/capacity` denoting flow pushed through the edge and edge's capacity. The special edges are colored in red.



As you can see in first query maximum flow from node 1 to node 4 equals 0 and in second query equals 1.