

Rutgers Programming Contest

Fall 2026

Piscataway, NJ

October 25, 2026

Problem A. Balanced Tripartition

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

You are given a positive integer $n > 1$. Your task is to partition the set $\{1, 2, \dots, 3n\}$ into three sets A , B , and C , each containing n elements. The sets A , B , and C should have the same sum. It can be shown that under the given constraints, a valid partition always exists.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 50$). The description of the test cases follows.

Each test case consists of a single line containing an integer n ($2 \leq n \leq 1000$) — the required size of each set.

It is guaranteed that the sum of n over all test cases does not exceed 1000.

Output

For each test case, output a single line consisting of $3n$ characters. Each character should be either A, B, or C (upper or lower case). The character at position i indicates which set the number i belongs to (A , B , or C).

If there are multiple solutions, you may print any.

Example

standard input	standard output
2	ABBCCCAABACB
4	ABABCCBCACCBAAAB
5	

Note

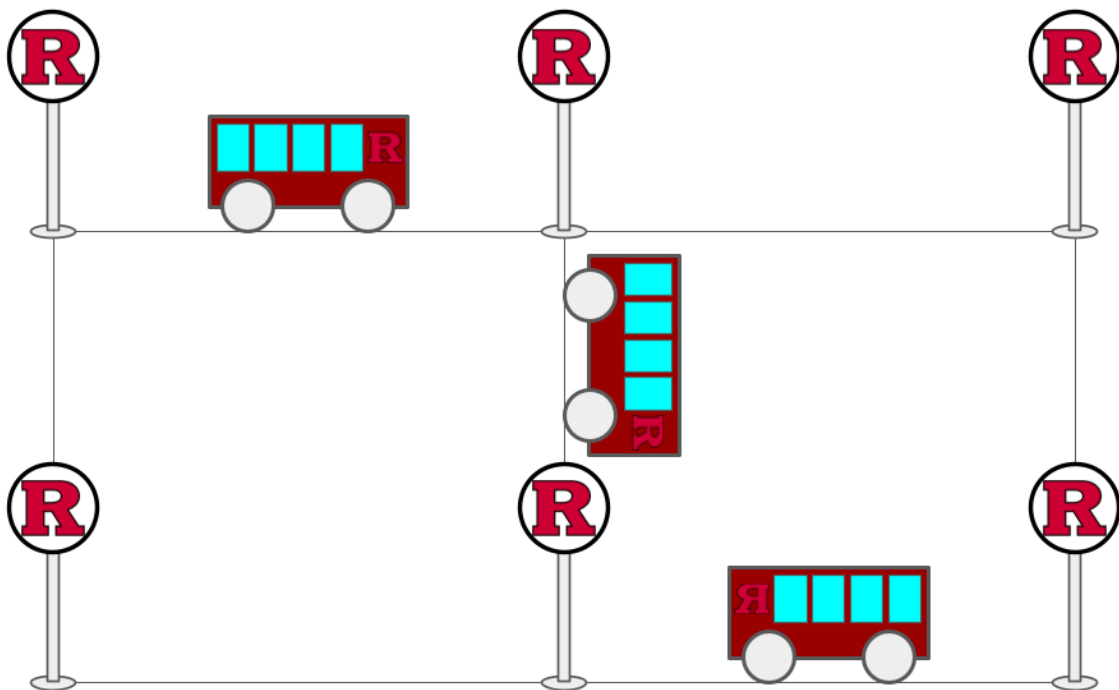
In the first test case, $n = 4$, so we are splitting up the numbers from 1 to $3 \cdot 4 = 12$. One solution is to partition these numbers into $A = \{1, 7, 8, 10\}$, $B = \{2, 3, 9, 12\}$, and $C = \{4, 5, 6, 11\}$. Each of these sets contains 4 numbers and sums to 26. Since 1 is in A , the first character of our output is A. Since 2 is in B , the second character of our output is B. Carrying on this process, we obtain the output ABBCCCAABACB.

Problem B. Rutgers Bus Budgeting

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

Due to budget reasons (the football team needs more DoorDash), Rutgers is working on updating its bus system. Currently, the bus system operates on an n by m rectangular grid, with bus stops in every cell and bus routes connecting horizontally or vertically adjacent stops. To do this and RUScrew over as many people as possible, Rutgers needs to remove 2 bus stops and all bus routes using those stops so that it's no longer possible to use the bus system to travel between every pair of remaining bus stops. Count the number of pairs of bus stops Rutgers can remove to do this.

Formally, given an n by m rectangular grid graph, count the number of pairs of vertices such that removing them and all incident edges makes the remaining graph disconnected.



Example of a 2 by 3 Rutgers bus system.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 2500$). The description of the test cases follows.

Each test case consists of a single line containing two integers n and m ($1 \leq n, m \leq 250$, $n \cdot m \geq 3$) — the dimensions of the Rutgers bus system.

Note that there is no limit on the sum of n or m over all test cases.

Output

For each test case, output a single integer — the number of pairs of bus stops Rutgers can remove to make it no longer possible to use the bus system to travel between every pair of remaining bus stops.

Example

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

Note

In the first test case, there are 6 ways to choose 2 bus stops to remove. 4 of these choices don't work. Removed stops are shown without a Rutgers logo.



The other 2 choices do work.



Problem C. Lucyxx Never Clear

Input file: **standard input**
Output file: **standard output**
Time limit: 10 seconds
Memory limit: 1024 megabytes

We define the function $F(N, M)$ to be the number of ordered pairs of integers (a, b) such that $1 \leq a \leq N$, $1 \leq b \leq M$, and $\gcd\left(\frac{a}{\gcd(a,b)}, b\right) = \gcd(a, b)$.

You are given that $F(3, 3) = 7$, $F(10, 10^3) = 6947$, and $F(10^5, 10^3) = 69278063$.

Find $F(n, m)$ modulo 998 244 353.

Input

The first and only line contains two integers n and m ($1 \leq n, m \leq 10^{10}$) — the values for which you must calculate $F(n, m)$.

Output

Output a single integer — the value $F(n, m)$ modulo 998 244 353.

Examples

standard input	standard output
3 3	7
10 1000	6947
100000 1000	69278063

Note

In the first test case, we have that $\gcd\left(\frac{3}{\gcd(3,1)}, 1\right) = 1 = \gcd(3, 1)$, so $(3, 1)$ is counted as a valid ordered pair. We also have that $\gcd\left(\frac{2}{\gcd(2,2)}, 2\right) = 1 \neq 2 = \gcd(2, 2)$, so $(2, 2)$ is not counted as a valid ordered pair. The 7 valid ordered pairs are $(1, 1)$, $(1, 2)$, $(1, 3)$, $(2, 1)$, $(2, 3)$, $(3, 1)$, and $(3, 2)$.

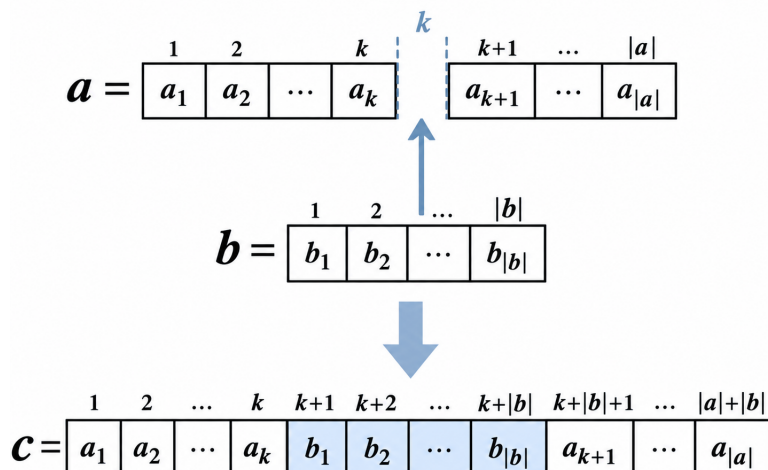
Problem D. Generatable Sequences

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

You enter the prompt “Build me a generator for infinite energy. Make no mistakes.” Unfortunately, your free trial of astra has ended, and you have been given a generator for infinite sequences instead.

Define G_m as the set of all sequences that can be generated using the integer m . A sequence c belongs to G_m if at least one of the following conditions holds:

1. The sequence $c = [1, 2, \dots, m]$.
2. There exist two (not necessarily distinct) sequences $a = [a_1, a_2, \dots, a_{|a|}] \in G_m$ and $b = [b_1, b_2, \dots, b_{|b|}] \in G_m$, and an integer k with $0 \leq k \leq |a|$, such that the sequence is obtained by inserting b into a after its first k elements. In other words, the resulting sequence $c = [a_1, a_2, \dots, a_k, b_1, b_2, \dots, b_{|b|}, a_{k+1}, a_{k+2}, \dots, a_{|a|}]$.



You are given an integer m and a sequence $a_1, a_2, \dots, a_n$. Find the maximum length of a subarray* of a that appears as a subarray of some sequence in G_m .

Formally, find the maximum integer k such that there exist a sequence $x \in G_m$ and indices $1 \leq l_1 \leq r_1 \leq |a|, 1 \leq l_2 \leq r_2 \leq |x|$ satisfying $r_1 - l_1 + 1 = r_2 - l_2 + 1 = k$ and $a_{l_1+i} = x_{l_2+i}$ for every $0 \leq i < k$.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 2 \cdot 10^5$). The description of the test cases follows.

The first line of each test case contains two integers m and n ($1 \leq m \leq 10^9, 1 \leq n \leq 2 \cdot 10^5$) — the generator and the length of the given sequence, respectively.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \leq a_i \leq m$).

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

*An array a is a subarray of an array b if a can be obtained from b by the deletion of several (possibly zero or all) elements from the beginning and several (possibly zero or all) elements from the end.

Output

For each test case, output a single integer — the maximum length of a subarray of a that occurs as a subarray of at least one sequence in G_m .

Example

standard input	standard output
5	5
5 5	2
1 2 3 4 5	7
1000000000 5	1
1 5 1 5 1	5
3 8	
1 2 1 2 3 3 1 3	
1 1	
1	
10 5	
4 5 6 7 8	

Note

For the first and fourth test cases, the entire sequence is generatable.

For the third test case, $[1, 2, 1, 2, 3, 3, 1]$ is a valid subarray.

For the fifth test case, $[4, 5, 6, 7, 8]$ is a subarray of $[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$.

Problem E. Non-Similar Paths

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

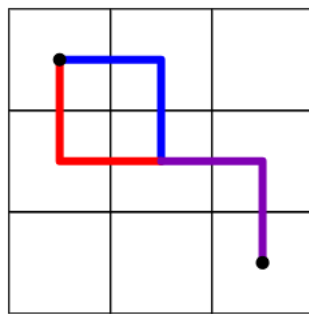
After years of research, you (Nick Belov) finally find your favorite biking place. You want to explore as many different bike paths as possible. However, you don't see a difference in exploring two bike paths that are very similar. You want to know how many bike paths you can explore such that each path is non-similar to any of your previously explored paths.

You are given an $n \times m$ grid. Each cell is either **passable** or **impassable**.

A **valid path** starts at the top-left cell $(1, 1)$ and ends at the bottom-right cell (n, m) . At each step, you may move either **right** or **down**, and every cell visited by the path must be passable.

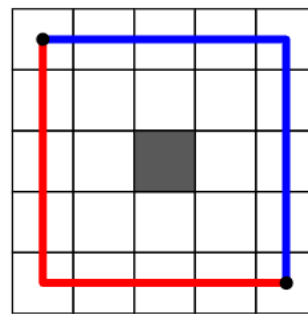
Represent each valid path by a string containing $m - 1$ characters 'R' and $n - 1$ characters 'D', denoting moves right and down, respectively.

Two valid paths a and b are called **similar** if there exists a sequence of valid paths $a = q_1, q_2, \dots, q_{|q|} = b$ such that, for every $1 \leq j < |q|$, the paths q_j and q_{j+1} differ in exactly 2 adjacent positions.



Similar paths

RDRD DRRD



Non-similar paths

RRRRDDD DDDDRRRR

Let S be a set of valid paths such that no two distinct paths in S are similar.

Find the maximum possible size of S . Since the answer may be large, output it modulo 998 244 353.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 10^5$). The description of the test cases follows.

The first line of each test case contains two integers n and m ($1 \leq n, m \leq 10^6, 2 \leq n \cdot m \leq 10^6$) — the number of rows and columns of the grid.

Each of the next n lines contains a string of length m describing one row of the grid.

A '.' denotes a passable cell, and a '#' denotes an impassable cell.

It is guaranteed that the sum of $n \cdot m$ over all test cases does not exceed 10^6 .

Output

For each test case, output one integer — the maximum possible size of S , modulo 998 244 353.

Example

standard input	standard output
4	2
5 5	0
.....	1
.....	1
..#..	
.....	
.....	
2 2	
#.	
..	
3 2	
.#	
..	
#.	
1 10	
.....	

Note

In the first test case (same as second picture), here is a possible set S :

$\{\text{RRRRDDDD}, \text{DDDDRRRR}\}$

In the second test case, the starting cell is impassable so there are no valid paths.

In the third and fourth test cases there is only 1 valid path.

Problem F. Permutation Edit Distance

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

You are given two permutations[†] and must compute the edit distance between them using only the following edits:

- Given a length n permutation, insert $n + 1$ anywhere into the array, resulting in a permutation of length $n + 1$.
- Given a length n permutation, delete n from the array, resulting in a permutation of length $n - 1$.

The edit distance between two permutations p and q is the minimum possible number of edits it takes to turn p into q or vice versa. It can be shown that, given the edits above, it does not matter which permutation you start with.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 2 \cdot 10^5$). The description of the test cases follows.

The first line of each test case contains two integers n and m ($1 \leq n, m \leq 2 \cdot 10^5$) — the lengths of the permutations.

The second line of each test case contains n integers $p_1, p_2, \dots, p_n$ ($1 \leq p_i \leq n$) — the permutation p .

The last line of each test case contains m integers $q_1, q_2, \dots, q_m$ ($1 \leq q_i \leq m$) — the permutation q .

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

Output

For each test case, output a single integer — the minimum number of valid edits to transform p into q .

Example

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

Note

In the first test case, inserting 5 into p in the second position will result in q .

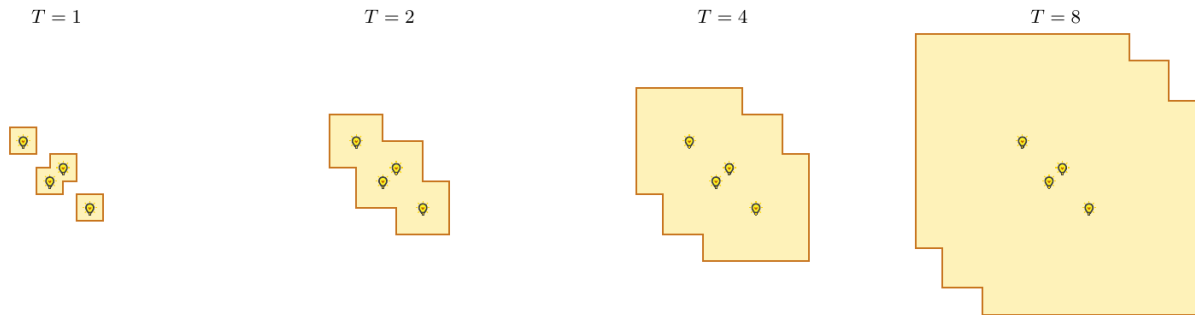
[†]A permutation of length n is an array consisting of n distinct integers from 1 to n in arbitrary order. For example, $[2, 3, 1, 5, 4]$ is a permutation, but $[1, 2, 2]$ is not a permutation (2 appears twice in the array), and $[1, 3, 4]$ is also not a permutation ($n = 3$ but there is 4 in the array).

In the second test case, first delete 3 and 2 from the end of p . Then insert 2 into the beginning, and after that insert 3 into the beginning as well. This will result in q , and it can be shown that this is a minimal solution.

In the third test case, the permutations are already equal.

Problem G. Illuminating Lights

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



You are given the positions of a collection of lights in the plane. At time 0 they are all simultaneously activated, and at time T each light illuminates an axis-aligned square of side length $2T$ centered on itself. You must decide whether there is a time $T > 0$ when the illuminated area is a rectangle.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 2 \cdot 10^5$). The description of the test cases follows.

The first line of each test case contains an integer n ($1 \leq n \leq 2 \cdot 10^5$) — the number of lights.

Each of the next n lines of each test case contains two integers x_i and y_i ($-10^9 \leq x_i, y_i \leq 10^9$) — the coordinates of the i -th light. The positions of the lights are guaranteed to be distinct.

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 **YES** if the illuminated area is at some point a rectangle, and **NO** otherwise.

You can output **YES** and **NO** in any case (for example, **yES**, **yes**, and **Yes** will be accepted).

Example

standard input	standard output
4	YES
4	NO
0 0	YES
0 1	NO
1 0	
1 1	
4	
2 2	
0 5	
5 0	
3 3	
3	
6 1	
6 2	
6 3	
2	
-1000 -1000	
1000 1000	

Note

In the second test case, which is shown in the statement, it can be shown that regardless of how much time passes, the illuminated area will never be a rectangle.

Problem H. Stringmaster

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

After years of training, you have finally awakened your true power and become a stringmaster, allowing you to perform the following operation on a string:

Choose a subsequence[‡] that is a palindrome[§] and arbitrarily permute the characters at the selected positions.

Here is an example of such an operation where we select the palindrome **tst**:

testing → s**e**tting

Now you have a string x , and you want to know if you can turn it into a string y by performing the above operation any number of times (possibly zero).

Input

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

The first line of each test case contains a single integer n ($1 \leq n \leq 2 \cdot 10^5$) — the length of the strings x and y .

The second line of each test case contains a string x of length n consisting of lowercase Latin letters.

The third line of each test case contains a string y of length n consisting of lowercase Latin 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, output **YES** if you can turn x into y , and **NO** otherwise.

You can output **YES** and **NO** in any case (for example, **yES**, **yes**, and **Yes** will be accepted).

[‡]A string a is a subsequence of a string b if a can be obtained from b by the deletion of several (possibly zero or all) characters.

[§]A string is called a palindrome if it is the same when written in reverse order. For example, **eertree** and **abba** are palindromes.

Example

standard input	standard output
6	YES
7	NO
testing	YES
setting	NO
5	YES
trees	NO
steer	
6	
thomas	
thomas	
5	
beleb	
volov	
10	
pactisaywa	
piscataway	
12	
brunswicknew	
newbrunswick	

Note

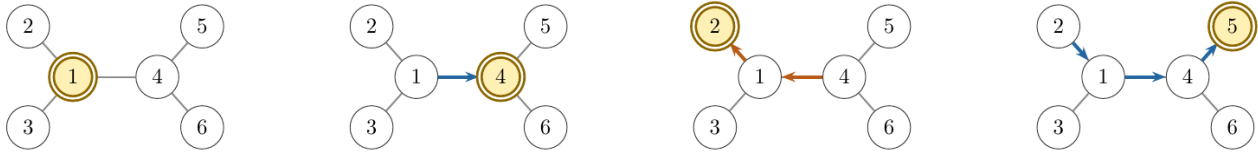
The solution to the first test case is shown above.

In the second test case, it is not possible to change the string.

In the third test case, no operations are needed.

Problem I. Increasing Tree Jumps

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



Alice and Bob play a game by taking turns moving a token in a tree. Alice goes first. On their turn, the player moves the token from its current vertex a to a vertex b . The length of the move is the number of edges on the unique simple path from a to b . Each move must be strictly longer than the previous move, and Alice's first move must have positive length.

A player who has no legal move on their turn loses, and the other player wins. Given the tree and initial position of the token, determine who will win the game if both players play optimally.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 2 \cdot 10^5$). The description of the test cases follows.

The first line of each test case contains two integers n and k ($1 \leq k \leq n \leq 2 \cdot 10^5$) — the number of vertices in the tree and the initial position of the token.

The next $n - 1$ lines each contain two integers u_i and v_i ($1 \leq u_i, v_i \leq n$) — the i -th edge in the tree.

It is guaranteed that the given edges form a tree.

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

Output

For each test case, output **Alice** if Alice wins with optimal play, or **Bob** if Bob wins.

Example

standard input	standard output
3	Bob
1 1	Alice
6 1	Bob
1 2	
1 3	
1 4	
4 5	
4 6	
15 8	
1 2	
2 3	
2 4	
1 5	
5 6	
5 7	
1 8	
8 9	
9 10	
9 11	
11 12	
8 13	
13 14	
14 15	

Note

In the first test case, Alice has no move since there are no nodes with a nonzero distance away from the token.

In the second test case, which is shown in the statement, Alice can move the token a distance of 1 to node 4. This leaves nodes 2 and 3 as Bob's only moves. In either case, she can respond with a move to node 5 and win.

In the third test case, it can be shown that Bob has a winning strategy.

Problem J. John Buck Legends

Input file: `standard input`
Output file: `standard output`
Time limit: 2 seconds
Memory limit: 1024 megabytes

The problemsetters of the Gennady Needs Your Rating contest have written n problems, but they forgot that they still have to write solutions! You know that for the i -th problem, it takes them a_i minutes to write a correct solution and b_i minutes to write an incorrect solution.

You know that the problemsetters will write exactly one solution for each problem and spend at most k minutes in total writing solutions for all problems. For how many problems is it possible that the problemsetters write a correct solution?

More formally, count the number of values i among $1, 2, \dots, n$ such that there exists some arrangement of correct and incorrect solutions for each problem, where a correct solution is written for problem i , and the total time spent writing all solutions does not exceed k . Note that the arrangements may differ for different values of i .

Input

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

The first line of each test case contains two integers n and k ($1 \leq n \leq 2 \cdot 10^5$, $n \leq k \leq 10^9$) — the number of problems and the maximum amount of time spent, respectively.

The next n lines each contain two integers a_i and b_i ($1 \leq a_i, b_i \leq 10^4$) — the times needed to write a correct solution and an incorrect solution, respectively.

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

Additional constraint on the input: there exists some arrangement of correct and incorrect solutions such that all solutions are written in at most k minutes.

Output

For each test case, output a single integer — the number of problems that might have a correct solution.

Example

standard input	standard output
3	3
5 20	0
6 7	3
15 5	
6 5	
100 1	
2 1	
1 1	
2 1	
3 8	
2 1	
3 2	
4 5	

Note

In the first test case, problems 1, 3, and 5 might have a correct solution. For example,

- Writing correct solutions to problems 1 and 5 and incorrect solutions to everything else takes $6 + 5 + 5 + 1 + 2 = 19$ minutes.
- Writing a correct solution to problem 3 and incorrect solutions to everything else takes $7 + 5 + 6 + 1 + 1 = 20$ minutes.

In the second test case, it is not possible to write the correct solution since writing the correct solution takes 2 minutes and $k = 1$.

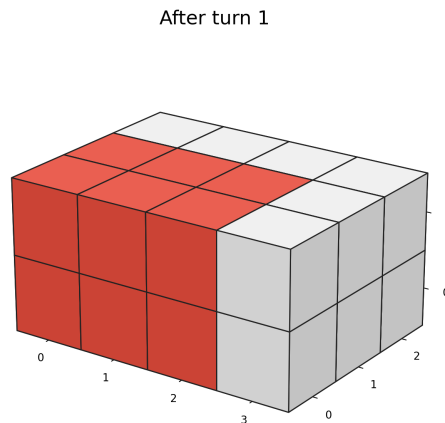
Problem K. 3D Turf War

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

In a four-dimensional world, video games would take place on three-dimensional screens in the same way we humans play video games on two-dimensional screens. There are k four-dimensional beings playing their favorite game on an $m \times n \times h$ grid. Each player has their own color.

The game has r turns. Each turn results in some player coloring a rectangular prism subgrid with their color. If any of these cells had been previously colored, they are overwritten to have the new color.

Formally, a turn is defined by a player number i and two triples (x_1, y_1, z_1) and (x_2, y_2, z_2) , where $x_1 \leq x_2$; $y_1 \leq y_2$; and $z_1 \leq z_2$. Player i colors all (x, y, z) such that $x_1 \leq x \leq x_2$; $y_1 \leq y \leq y_2$; and $z_1 \leq z \leq z_2$. Below is an example of a turn for $m = 4, n = 3, h = 2$. The player has colored the subgrid defined by corners $(0, 0, 0)$ and $(2, 1, 1)$.



Your task is to calculate, at the end of the game, how many cells in the grid were only ever colored by one player. Note that a cell might get colored by the same player multiple times across different turns.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 50$). The description of the test cases follows.

The first line of each test case consists of five integers, m, n, h, k and r ($1 \leq mn, k, r \leq 4 \cdot 10^5$) — this represents a k -player game lasting r turns played on an $m \times n \times h$ grid.

The next r lines of each test case consist of seven integers each: $i, x_1, y_1, z_1, x_2, y_2, z_2$ ($0 \leq i, x_1, y_1, z_1, x_2, y_2, z_2 < 4 \cdot 10^5$). The following conditions are guaranteed: $i < k$; $x_1 \leq x_2 < m$; $y_1 \leq y_2 < n$; and $z_1 \leq z_2 < h$. This line represents a turn in which player number i colors the entire subgrid defined by (x_1, y_1, z_1) and (x_2, y_2, z_2) .

It is guaranteed that the sum of mnh over all test cases is at most $4 \cdot 10^5$ and the sum of r over all test cases is at most $4 \cdot 10^5$.

Output

For each test case, output a single integer — the number of cells in the grid that were only ever colored by exactly one player.

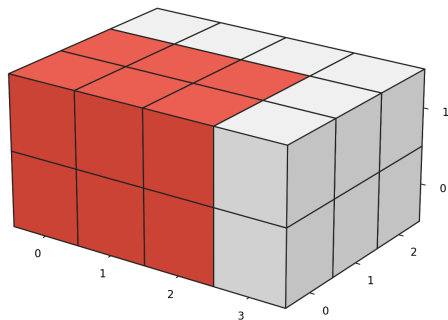
Example

standard input	standard output
3	11
4 3 2 2 3	4
0 0 0 0 2 1 1	0
1 1 0 0 1 2 1	
0 0 1 0 2 2 0	
2 2 1 4 4	
0 0 0 0 0 0 0	
1 0 1 0 0 1 0	
2 1 0 0 1 0 0	
3 1 1 0 1 1 0	
1 2 2 3 3	
0 0 0 0 0 1 1	
1 0 0 0 0 0 1	
2 0 1 0 0 1 1	

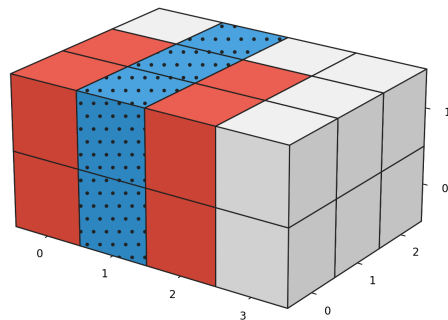
Note

In the first test case, there are two players and three turns. Here are the game states after each turn.

After turn 1

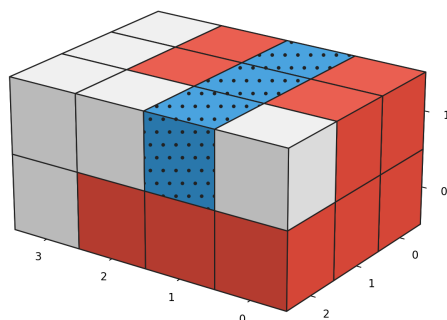


After turn 2



The changes to the state after the third move are not visible from the previous angle, so a rotated view is below.

After turn 3 — rotated view



In total, 16 cells were colored over the course of the game. Of these, the five cells $\{(1, 0, 0), (1, 1, 0), (1, 2, 0), (1, 0, 1), (1, 1, 1)\}$ were colored by both players at some point. The other 11 were only touched by one player (10 by the first player and 1 by the second player).

Problem L. Skyline

Input file: **standard input**
Output file: **standard output**
Time limit: 7 seconds
Memory limit: 1024 megabytes



Busch is actually sort of a campus of New Brunswick, so Little Keshstein had time to appreciate the Busch skyline. However, he can't really figure out whether some buildings belong to Busch or Livingston, so he naturally comes up with the following problem.

You are given a function $f(b)$ that represents the maximum skyline achievable for an array b of length m , defined as:

$$f(b) = \max(\min(b_i, b_{i+1}, \dots, b_j) \cdot (j - i + 1)), \text{ where } 1 \leq i \leq j \leq m.$$

Given an array a of length n , partition its elements into two nonempty subsequences x and y such that each element is contained in exactly one of x or y , and $f(x) + f(y)$ is maximized.

A sequence x is a subsequence of a sequence a if x can be obtained from a by the deletion of several (possibly zero or all) elements from arbitrary positions.

Input

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

The first line of each test case contains a single integer n ($2 \leq n \leq 7500$) — the length of the array.

The second line of each test case contains n integers $a_1, a_2, \dots, a_n$ ($1 \leq a_i \leq n$) — the elements of the array.

It is guaranteed that the sum of n^2 over all test cases does not exceed 7500^2 .

Output

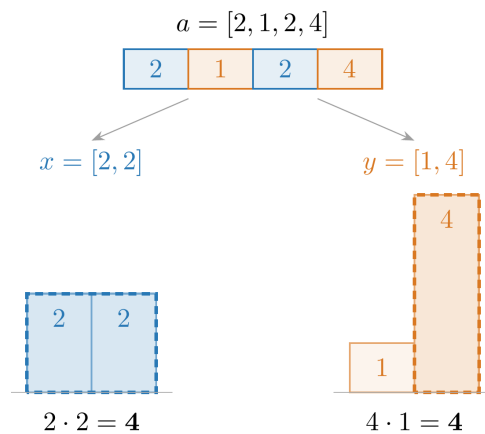
For each test case, output a single integer — the maximum value of $f(x) + f(y)$ that can be achieved by partitioning a into two nonempty subsequences x and y .

Example

standard input	standard output
6	11
5	8
3 1 4 1 5	8
4	8
1 2 3 4	9
4	15
2 1 2 4	
4	
2 1 4 2	
5	
2 3 1 2 3	
6	
4 1 3 6 2 3	

Note

In the third test case, one optimal partition is $x = [a_1, a_3] = [2, 2]$ and $y = [a_2, a_4] = [1, 4]$. We can show that $f(x) + f(y) = 2 \cdot (2 - 1 + 1) + 4 \cdot (2 - 2 + 1) = 8$. We illustrate below.



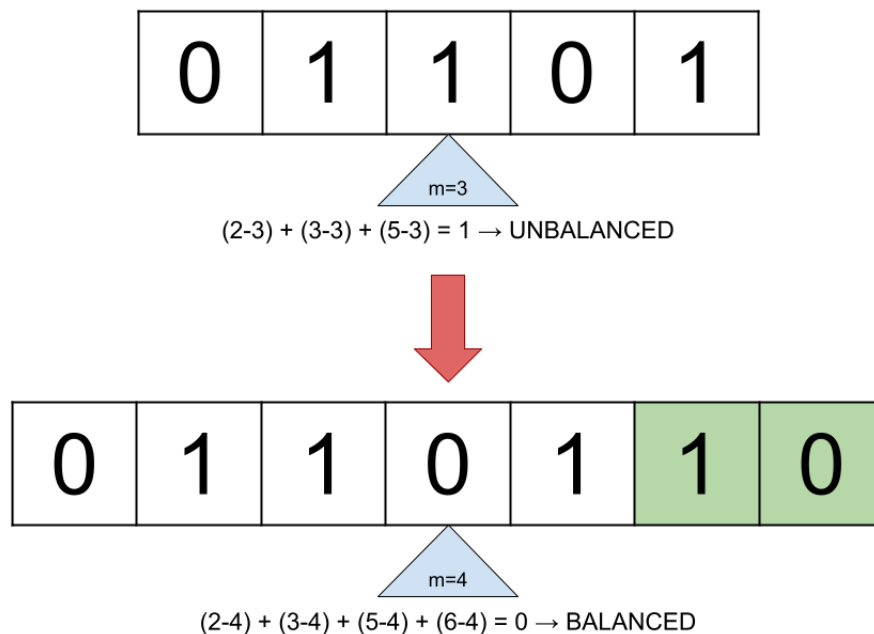
Problem M. Balancing Thomas

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

Thomas has been studying physics, and recently he learned about levers and torque. As practice, he has been experimenting by placing identical weights on a lever to see what balances. Since the weights are identical, he can keep track of his setup with a binary string of odd length, where the 1s represent where weights are located, and the middle index is the position under which the lever's fulcrum is located. Thomas's current configuration of weights might not be balanced, and he wants to put in as little effort as possible to extend it to a configuration that will balance.

Formally, we define a binary string to be an equilibrium string if it has odd length and satisfies $\sum_{i \in W} (i - m) = 0$ where m is the index of the midpoint of the string and W is the set of indices having 1s. Given a binary string representing Thomas's current configuration of weights, your task is to find the length of the shortest equilibrium string of which the given string is a prefix.

Below is an example calculation of a binary string that is not an equilibrium string. However, by adding two characters onto the end, it can be made into one. In fact, we cannot turn the first string into an equilibrium string with any shorter extension.



Input

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

The first line of each test case consists of a single integer n ($1 \leq n \leq 2 \cdot 10^5$) — the length of the binary string.

The second line of each test case consists of a single string of length n , where each character is either a 0 or 1.

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

Output

For each test case, output a single integer — the length of the shortest equilibrium string of which the given string is a prefix.

Example

standard input	standard output
5	7
5	7
01101	3
6	15
100011	11
3	
111	
10	
1001010110	
8	
10010000	

Note

For the first test case, the shortest equilibrium string having the input as a prefix is of length 7, and one example is shown in the picture above.

In the second test case, we can convert 100011 into 1000110.

The third test case is already an equilibrium string, so we need not extend it.

Problem N. MEX of XOR

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

The nonempty prefix XORs of an array of length m are the values $a_1 \oplus a_2 \oplus \cdots \oplus a_k$ where $1 \leq k \leq m$ and $\oplus$ denotes the bitwise XOR operation.

You are given an integer n . Output a permutation[¶] of the integers from 0 to $2^n - 1$ such that the MEX[‡] of all nonempty prefix XORs is maximized.

Input

The only line of input contains a single integer n ($2 \leq n \leq 20$).

Output

Output 2^n integers which are a permutation such that the MEX of all nonempty prefix XORs is maximized.

If there are multiple solutions, you may print any.

Examples

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

Note

In the first test case, the prefix XORs are $[0, 1, 2, 0]$, so the MEX is 3, which is the maximum possible.

In the second test case, the prefix XORs are $[0, 1, 3, 6, 5, 2, 4, 0]$, so the MEX is 7, which is the maximum possible.

[¶]A permutation of the integers from 0 to $2^n - 1$ is an array consisting of 2^n distinct integers from 0 to $2^n - 1$ in arbitrary order. For example, $[2, 3, 1, 0]$ is a permutation, but $[1, 1]$ is not a permutation (1 appears twice in the array), and $[1, 2, 4, 0]$ is also not a permutation (the length is 4 but there is 4 in the array).

[‡]The minimum excluded (MEX) of a collection of integers $c_1, c_2, \dots, c_k$ is defined as the smallest non-negative integer x which does not occur in the collection c .

Problem O. Squaremas

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

You are given two integers m and n . Print a grid of integers with m rows and n columns such that the four corners of any rectangular subgrid add up to a perfect square. Each integer must lie in the range $[1, 10^9]$. Note that we are only interested in rectangular subgrids that have four corners. Your solution must contain at least two different numbers. It can be shown that under the given constraints, a solution always exists.

An example on a 3×3 grid is shown below. Several examples of rectangular subgrids are highlighted. The four corners of each rectangle are colored red. Note that the entire grid is also considered a subgrid.

3	2	29
31	13	37
1	19	31

$$3 + 2 + 31 + 13 = 49$$

3	2	29
31	13	37
1	19	31

$$3 + 29 + 31 + 37 = 100$$

$$3 + 29 + 1 + 31 = 64$$

3	2	29
31	13	37
1	19	31

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 50$). The description of the test cases follows.

Each test case consists of a single line containing two integers m and n ($2 \leq m, n \leq 100$) — the required number of rows and columns of the grid.

It is guaranteed that the sum of m and the sum of n over all test cases are at most 100.

Output

For each test case, output m rows, each containing n integers satisfying the above conditions.

If there are multiple solutions, you may output any.

Example

standard input	standard output
2	5 4 5
2 3	3 4 3
3 3	3 2 29
	31 13 37
	1 19 31

Note

The first test case is shown in the image below.

5	4	5
3	4	3

$$5 + 4 + 3 + 4 = 16$$

5	4	5
3	4	3

$$5 + 4 + 3 + 4 = 16$$

$$5 + 5 + 3 + 3 = 16$$

5	4	5
3	4	3

The second test case is from the image in the problem statement. There are 9 rectangles to consider. In all, we have

- $3 + 2 + 31 + 13 = 49 = 7^2$
- $2 + 29 + 13 + 37 = 81 = 9^2$
- $31 + 13 + 1 + 19 = 64 = 8^2$
- $13 + 37 + 19 + 31 = 100 = 10^2$
- $3 + 29 + 31 + 37 = 100 = 10^2$
- $31 + 37 + 1 + 31 = 100 = 10^2$
- $3 + 2 + 1 + 19 = 25 = 5^2$
- $2 + 29 + 19 + 31 = 81 = 9^2$
- $3 + 29 + 1 + 31 = 64 = 8^2$