The 2023 Asia Jinan Regional Contest

Contest Session

December 3, 2023

Problem List

A	Many Many Heads
B	Graph Partitioning 2
C	Turn on the Light 2
D	Largest Digit
E	I Just Want... One More...
F	Say Hello to the Future
G	Gifts from Knowledge
H	Basic Substring Structure
I	Strange Sorting
J	Computational Intelligence
K	Rainbow Subarray
L	Ticket to Ride
M	Almost Convex

This problem set should contain 13 (thirteen) problems on 19 (nineteen) numbered pages. Please inform a runner immediately if something is missing from your problem set.

Hosted by

Problem Set Prepared by

It's against the rules to open non-contest websites during the contest.
If you're interested (which is our pleasure),
please scan the QR code only after the contest.

Problem A. Many Many Heads

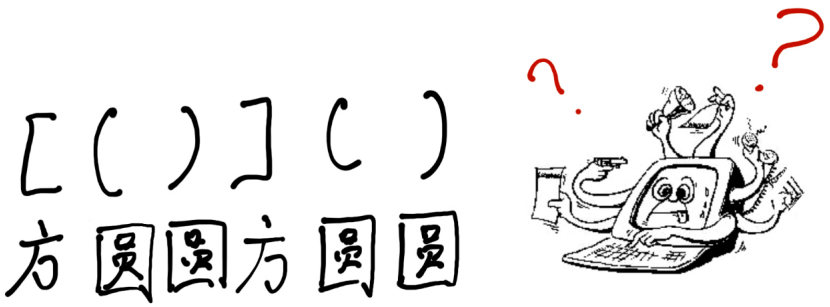
Multi-Heads Cup, or MHC for short, is a worldwide programming contest made for the participants with **many many heads**. The Chief Judge of the competition, Little Cyan Fish, is considering to design an identification number for each participant.

"And that's it," Little Cyan Fish thought, "Let's use some bracket sequences!" He assigned a unique balanced bracket sequence to each participant, with two kinds of the brackets - the round brackets (also known as parentheses), and the square brackets. To make sure you understand the concepts of balanced bracket sequence, Little Cyan Fish prepared a formal definition of the balanced bracket sequence:

- ε (an empty string) is a balanced bracket sequence.
- If A is a balacned bracket sequence, then (A) and $[A]$ are both balanced bracket sequences.
- If A and B are balanced bracket sequences, then AB is also a balanced bracket sequence.

For example, "`()`", "`[()]`" and "`[()]()`" are balanced bracket sequences, but "`)(`", "`[(])`" and "`[)`" are not.

For our multi-headed participants, memorizing a bracket sequence is not a difficult task. However, the challenge lies in their unique ability: because they have too much heads, they can't discern the direction of each bracket! Consequently, compared to the original balanced bracket sequence, they might recall the sequence with some brackets changing their directions. For example, the bracket sequence "`[()]()`" might be memorized as "`]))]))`" or "`](()]))`". Fortunately, the type of brackets are guaranteed to remain unchanged.



On the contest day, as Little Cyan Fish receives each participant's bracket sequence, a question arises: Can the original bracket sequence be deduced uniquely? In other words, Little Cyan Fish needs to determine if the provided bracket sequence maps to exactly one balanced bracket sequence.

Please, help Little Cyan Fish to finish this task, so our multi-heads friends will be able to participate in the contest!

Input

There are multiple test cases. The first line of the input contains an integer T, indicating the number of test cases. For each test case:

The first line contains a string S consisting of '`(`', '`)`', '`[`', and '`]`' ($1 \le |S| \le 10^5$), indicating the bracket sequence.

It is guaranteed that:

- The sum of $|S|$ over all test cases does not exceed 10^6.
- Each bracket sequence is obtained by changing the directions of some brackets in a balanced bracket sequence.

Output

For each test case:

- If the provided bracket sequence maps to more than one balanced bracket sequence, output a single line `No`.
- Otherwise, output a single line `Yes`.

Example

standard input	standard output
6)) ((([()] ()[()]() ([()]) ([])([])	Yes No Yes No Yes No

Note

In the first test case, the bracket sequence maps to exactly one balanced bracket sequence: `()`. So the answer is `Yes`.

In the second test case, the bracket sequence corresponds to two distinct balanced bracket sequences: `(())` and `()()`. So the answer is `No`.

In the third test case, the bracket sequence maps to exactly one balanced bracket sequence: `[()]`. So the answer is `Yes`.

In the fourth test case, the bracket sequence corresponds to two distinct balanced bracket sequences: `(([()]))` and `()[()]()`. So the answer is `No`.

In the fifth test case, the bracket sequence maps to exactly one balanced bracket sequence: `([()])`. So the answer is `Yes`.

In the sixth test case, the bracket sequence corresponds to three distinct balanced bracket sequences: `([])([])`, `([]()[])` and `([[()]])`. So the answer is `No`.

Problem B. Graph Partitioning 2

After successfully solving the problem *Cut Cut Cut!*, Little Cyan Fish seeks to further hone his skills in partitioning connected components within graphs.

One day, Little Cyan Fish is challenged by a mysterious sage with a unique problem. He is presented with an unrooted tree containing n vertices and is given an integer k. Let E represent the set of all edges in the tree. Little Cyan Fish's objective is to identify a subset $E' \subseteq E$. Upon removing all edges in E', the graph should split into several connected components, each with a size of either k or $(k+1)$.

As an expert in partitioning, Little Cyan Fish adeptly solves this problem. However, the mysterious sage's curiosity extends beyond mere mastery. He seeks to explore all potential outcomes. Consequently, he tasks Little Cyan Fish with determining the total number of different ways to select $E' \subseteq E$ while adhering to the stated condition. Two ways are considered different if the selected subsets of edges are different.

Please help Little Cyan Fish to finish this challenge. As the answer may be large, you only need to provide the answer modulo 998 244 353.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains two integers n and k ($2 \le n \le 10^5$, $1 \le k \le n$) indicating the number of vertices in the tree and the target size of the smaller connected components.

For the following $(n-1)$ lines, the i-th line contains two integers u_i and v_i ($1 \le u_i, v_i \le n$) indicating an edge connecting vertices u_i and v_i in the tree.

It's guaranteed that the sum of n of all test cases will not exceed 3×10^5.

Output

For each test case, output one line containing one integer indicating the number of ways to choose a subset E' modulo 998 244 353.

Example

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

Note

Let (u, v) be an edge connecting vertices u and v. For the first sample test case, the two valid subsets of edges are $\{(2,4),(3,5)\}$ and $\{(1,2),(3,5)\}$.

Problem C. Turn on the Light 2

Lux et Veritas
(Light and Truth)

The much-anticipated Universal Cup Final is around the corner! Little Cyan Fish is busy preparing the venue for the competition. To make the venue gorgeous, Little Cyan Fish is planning to hang some light bulbs.

Little Cyan Fish has m wires and plans to connect n light bulbs using these wires. Each wire needs to connect two distinct bulbs and make all the bulbs form a single connected component. To maintain safety, for any two bulbs there can be at most one wire connecting them directly, and no more than d wires can be attached to a single bulb.

Once the bulbs are connected, Little Cyan Fish wants to illuminate some of them. Given that active bulbs generate heat, positioning two lit bulbs adjacent to each other could be dangerous. Therefore, if two bulbs are directly connected by a wire, they must not be illuminated at the same time. On the other hand, he doesn't want too few lights on, so he does not want to see an unlit bulb, such that all bulbs directly connected to it are also unlit.

This leads Little Cyan Fish to wonder about the number of different plans to turn on those bulbs to fulfill the requirements. More so, he is keen on finding the optimal way to connect all the light bulbs to maximize the number of the feasible lighting plans.

Given the integers m and d, your goal is to assist Little Cyan Fish in determining the optimal way to connect n bulbs using all m wires, with the intent of maximizing the number of the ways to illuminate the bulbs. Note that you have to determine the value of n by yourself.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 200$), indicating the number of test cases. For each test case:

The first and only line contains two integers m and d ($2 \le m \le 20$, $2 \le d \le m$).

Output

For each test case:

In the first line, output an integer w ($1 \le w \le 2^{m+1}$), indicating the maximum number of ways to illuminate the bulbs.

In the second line, output an integer n ($1 \le n \le m+1$), indicating the number of bulbs Little Cyan Fish should use.

Then output m lines, where the i-th line contains two integers u_i and v_i ($1 \le u_i, v_i \le n, u_i \ne v_i$) separated by a space, indicating a wire connecting the u_i-th and the v_i-th bulb.

Example

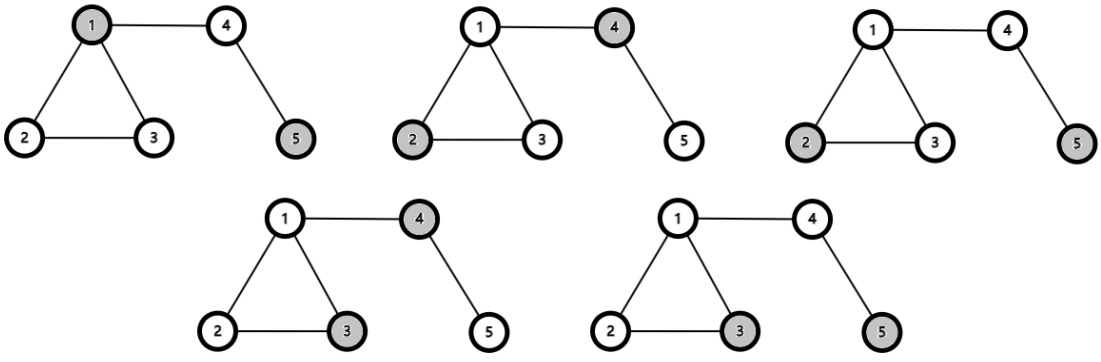
standard input	standard output
3 2 2 5 4 6 2	2 3 1 2 2 3 5 5 1 2 1 3 2 3 1 4 4 5 7 7 1 2 2 3 3 4 4 5 5 6 6 7

Note

We use colored circles to represent lit bulbs.

For the first sample test case, the 2 ways to illuminate the bulbs are illustrated below.

For the second sample test case, the 5 ways to illuminate the bulbs are illustrated below.



Problem D. Largest Digit

Let $f(x)$ be the largest digit in the decimal representation of a positive integer x. For example, $f(4523) = 5$ and $f(1001) = 1$.

Given four positive integers l_a, r_a, l_b and r_b such that $l_a \le r_a$ and $l_b \le r_b$, calculate the maximum value of $f(a + b)$, where $l_a \le a \le r_a$ and $l_b \le b \le r_b$.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 10^3$) indicating the number of test cases. For each test case:

The first and only line contains four integers l_a, r_a, l_b and r_b ($1 \le l_a \le r_a \le 10^9$, $1 \le l_b \le r_b \le 10^9$).

Output

For each test case output one line containing one integer indicating the maximum value of $f(a + b)$.

Example

standard input	standard output
2 178 182 83 85 2 5 3 6	7 9

Note

For the first sample test case, the answer is $f(182 + 85) = f(267) = 7$.

For the second sample test case, the answer is $f(4 + 5) = f(9) = 9$.

Problem E. I Just Want... One More...

After writing the paper *Sandpile Prediction on Structured Undirected Graphs*, Little Cyan Fish wants everyone to solve more graph theory tasks. "We cannot survive without graph theory problems, and everyone should come to do the sandpile prediction!"

A bipartite graph is a graph whose vertices can be divided into two disjoint sets U and V, such that every edge in the graph connects a vertex in U to one in V. If the number of vertices in U and V are the same, then this graph is called a balanced bipartite graph.

A matching in an undirected graph is a set of edges, in which any two edges have no common vertices. A maximum matching of the graph is a matching containing the largest possible number of edges. The matching number of a graph is the number of edges in its maximum matching.

Now, Little Cyan Fish gives you a balanced bipartite graph. You are asked to count the number of ways to add exactly one edge between a vertex in U and a vertex in V, such that the matching number of the graph is increased.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains two integers n and m ($1 \le n, m \le 10^5$) indicating number of vertices in U and V, and the number of edges.

For the following m lines, the i-th line contains two integers u_i and v_i ($1 \le u_i, v_i \le n$) indicating the i-th edge connects the u_i-th vertex in U and the v_i-th vertex in V. The graph may contain multiple edges.

It's guaranteed that the sum of $(n + m)$ of all test cases will not exceed 4×10^5.

Output

For each test case output one line containing one integer indicating the answer.

Example

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

Note

For the first sample test case, the matching number of the original graph is 2. By adding edge $(1, 1)$, $(1, 4)$, $(2, 1)$, $(2, 4)$, $(3, 1)$ or $(3, 4)$, we can increase the matching number to 3. So the answer is 6.

For the second sample test case, the matching number of the original graph is 3. Obviously we cannot increase the matching number since all vertices are in the matching, so the answer is 0.

For the third sample test case, the matching number of the original graph is 1. By adding edge $(2, 1)$, $(2, 3)$, $(3, 1)$ or $(3, 3)$, we can increase the matching number to 2. So the answer is 4.

Problem F. Say Hello to the Future

"Ugh... Well, can we restart our friendship?"
"What do you mean... You said, restart?"
"..."

A long time ago, Little Cyan Fish prepared a programming contest with his best friend Little $\mathcal{F}$. They proposed n problems in total, indexed with the integers from 1 to n. The i-th problem ($1 \le i \le n$) has a difficulty rating a_i.

Time flies fast. It has been fifteen months since their contest. Instead of being a participant in the Informatics Olympiad, Little Cyan Fish has transitioned into a coach. But they once had a pact to run a series of championships together.

Little Cyan Fish didn't forget about it.

Now, Little Cyan Fish would like to group these n problems into several training activities. To ensure the stories in the statements of the tasks are consistent, Little Cyan Fish wants to divide these n tasks into several intervals. A dividing plan of the problems can be represented by a sequence of integers $0 = r_0 < r_1 < r_2 < \cdots < r_k = n$, meaning that there will be k training activities, and the i-th training activity will contain all tasks whose indices are between $(r_{i-1} + 1)$ and r_i (both inclusive).

Furthermore, Little Cyan Fish doesn't want an activity to be too unbalanced. If there is a hard problem in an activity, then the activity needs to contain more problems. Formally, if task j is in the i-th activity (i.e. $r_{i-1} < j \le r_i$), then the inequality $r_i - r_{i-1} \ge a_j$ must hold.

Little Cyan Fish is interested in how many ways he could divide the problems to satisfy all the requirements above, denoted by $f(a)$. This question is easy for him, so Little Cyan Fish calculated it easily.

One day before the activities, Little Cyan Fish suddenly realized that those problems were too hard for the participants. Therefore, he came up with a new easy problem with a difficulty rating of only 1. He is wondering, for each $1 \le j \le n$, if we define the sequence $a^{(j)}$ in the following way, what is the value of $f(a^{(j)})$.

$$a_i^{(j)} = \begin{cases} 1 & i = j \\ a_i & \text{otherwise} \end{cases}$$

Since the value of $f(a^{(j)})$ can be extremely large, you only need to output it modulo 998 244 353.

Input

There is only one test case in each test file.

The first line of the input contains an integer n ($1 \le n \le 2 \times 10^5$) indicating the number of problems.

The second line of the input contains n integers $a_1, a_2, \cdots, a_n$ ($1 \le a_i \le n$), where a_i is the difficulty rating of the i-th problem.

Output

Output one line contains n integers separated by a space, where the i-th integer denotes the value of $f(a^{(i)})$, modulo 998 244 353.

Please, DO NOT output extra spaces at the end of each line, or your solution may be considered incorrect!

Example

standard input	standard output
5 1 3 2 1 2	3 6 3 3 6

Note

In the sample test case, for $j = 1$, we have $a^{(j)} = [1, 3, 2, 1, 2]$. There are 3 ways to assign the problems to the activities, all as follows:

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

For $j = 2$, we have $a^{(j)} = [1, 1, 2, 1, 2]$. There are 6 ways to assign the problems to the activities, all as follows:

- $[1]$, $[1]$, $[2, 1, 2]$
- $[1]$, $[1, 2]$, $[1, 2]$
- $[1]$, $[1, 2, 1, 2]$
- $[1, 1]$, $[2, 1, 2]$
- $[1, 1, 2]$, $[1, 2]$
- $[1, 1, 2, 1, 2]$

For $j = 3$ and $j = 4$, all the plans are identical to the plans in $j = 1$.

For $j = 5$, we have $a^{(j)} = [1, 3, 2, 1, 1]$. There are 6 ways to assign the problems to the activities, all as follows:

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

Problem G. Gifts from Knowledge

After studying the paper *Solving Sparse Linear Systems Faster than Matrix Multiplication* by Richard Peng, and Santosh Vempala, Little Cyan Fish becomes obsessed with anything that is sparse. For example, a sparse matrix. Here, a sparse matrix refers to a matrix in which the number of zero elements is much higher than the number of non-zero elements. Now, Little Cyan Fish comes up with a problem about binary sparse matrices and he wants you to try solving it.

Given a binary matrix (a matrix containing only 0s and 1s) with r rows and c columns, you can choose whether to reverse each row or not. Find the number of ways to choose a set of rows to reverse (it is allowed not to choose any row), so that every column has at most one 1. Two ways are considered different if a row is chosen in one of them but not the other.

By reversing a row, we mean this: Let the elements on the i-th row be $b_{i,1}, b_{i,2}, \cdots, b_{i,c}$ from the first column to the last column. If you reverse the i-th row, it becomes $b_{i,c}, b_{i,c-1}, \cdots, b_{i,1}$.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains two integers r and c ($1 \le r, c \le 10^6$, $1 \le r \times c \le 10^6$) indicating the number of rows and columns of the matrix.

For the following r lines, the i-th line contains a string $b_{i,1}b_{i,2}\cdots b_{i,c}$ ($b_{i,j} \in \{0, 1\}$) where $b_{i,j}$ is the element on the i-th row and the j-th column of the matrix.

It's guaranteed that the sum of $r \times c$ of all test cases does not exceed 10^6.

Output

For each test case output one line containing one integer indicating the number of ways. As the answer might be large, print the answer modulo $(10^9 + 7)$.

Example

standard input	standard output
3 3 5 01100 10001 00010 2 1 1 1 2 3 001 001	4 0 2

Note

For the first sample test case, the set of selected rows can be empty, $\{1, 3\}$, $\{2\}$ or $\{1, 2, 3\}$. So the answer is 4.

Problem H. Basic Substring Structure

After writing the paper *Faster Algorithms for Internal Dictionary Queries*, Little Cyan Fish and Kiwihadron decided to write this problem.

Let $\mathrm{lcp}(s,t)$ be the length of the longest common prefix of two strings $s = s_1 s_2 \dots s_n$ and $t = t_1 t_2 \dots t_m$, which is defined as the maximum integer k such that $0 \le k \le \min(n,m)$ and $s_1 s_2 \dots s_k$ equals $t_1 t_2 \dots t_k$.

Little Cyan Fish gives you a non-empty string $s = s_1 s_2 \dots s_n$. Let $f(s) = \sum_{i=1}^{n} \mathrm{lcp}(s, \mathrm{suf}(s,i))$, where $\mathrm{suf}(s,i)$ is the suffix of s starting from s_i (i.e. $\mathrm{suf}(s,i) = s_i s_{i+1} \dots s_n$). Note that in this problem, the alphabet contains n letters, not just 26.

For each $i = 1, 2, \cdots, n$, you are asked to answer the following query: if you MUST change s_i to another different character c ($c \neq s_i$), choose the best character c and calculate the maximum value of $f(s^{(i)})$, where $s^{(i)} = s_1 \dots s_{i-1} c s_{i+1} \dots s_n$.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains an integer n ($2 \le n \le 2 \times 10^5$) indicating the length of the string.

The second line contains n integers $s_1, s_2, \dots, s_n$ ($1 \le s_i \le n$) where s_i indicates that the i-th character of the string is the s_i-th letter in the alphabet.

It's guaranteed that the sum of n over all test cases doesn't exceed 2×10^5.

Output

Let $m(i)$ be the maximum value of $f(s^{(i)})$. To decrease the size of output, for each test case output one line containing one integer which is $\sum_{i=1}^{n} (m(i) \oplus i)$, where $\oplus$ is the bitwise exclusive or operator.

Example

standard input	standard output
2 4 2 1 1 2 12 1 1 4 5 1 4 1 9 1 9 8 10	15 217

Note

For the first sample test case, let's first calculate the value of $m(1)$.

- If you change s_1 to 1, then $f(s^{(1)}) = 4 + 2 + 1 + 0 = 7$.
- If you change s_1 to 3 or 4, then $f(s^{(1)}) = 4 + 0 + 0 + 0 = 4$.

So $m(1) = 7$.

Similarly, $m(2) = 6$, $m(3) = 6$ and $m(4) = 4$. So the answer is $(7 \oplus 1) + (6 \oplus 2) + (6 \oplus 3) + (4 \oplus 4) = 15$.

Problem I. Strange Sorting

> We present an extremely simple sorting algorithm. It may look like it is obviously wrong, but we prove that it is in fact correct. [a]
>
> [a] Stanley P. Y. Fung. Is this the simplest (and most surprising) sorting algorithm ever? arXiv:2110.01111

After learning the strange sorting algorithm in the problem *Paimon Sorting* of The 2021 ICPC Asia Nanjing Regional Contest, Little Cyan Fish comes up with the following task.

Given a sequence $a_1, a_2, \cdots, a_n$ which is a permutation of n, your task is to sort the permutation in ascending order by applying the following operation for at most $\lfloor \frac{n}{2} \rfloor$ times: Choose two indices l and r satisfying $1 \le l < r \le n$ and $a_l > a_r$, and then sort $a_l, a_{l+1}, \cdots, a_r$ in ascending order.

Recall that a permutation of n is a sequence of length n, in which each integer from 1 to n (both inclusive) appears exactly once. Also recall that $\lfloor x \rfloor$ indicates the largest integer less than or equal to x.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains an integer n ($1 \le n \le 100$) indicating the length of the permutation.

The second line contains n distinct integers $a_1, a_2, \cdots, a_n$ ($1 \le a_i \le n$) indicating the given permutation.

It's guaranteed that the sum of n of all test cases will not exceed 10^4.

Output

For each test case, first output one line containing one integer k ($0 \le k \le \lfloor \frac{n}{2} \rfloor$) indicating the number of operations you're going to use. Then output k lines, where the i-th line contains two integers l_i and r_i separated by a space, indicating the two indices you choose for the i-th operation.

It can be proven that the answer always exists. If there are multiple valid answers, you can output any of them.

Example

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

Note

For the first sample test case, after the 1-st operation the permutation becomes $\{2, 3, 1, 4, 5, 6\}$, and after the 2-nd operation the permutation becomes $\{1, 2, 3, 4, 5, 6\}$, which is in ascending order.

Problem J. Computational Intelligence

Given two straight line segments on a two-dimensional Cartesian plane, you need to pick a point uniformly at random on each of them and calculate the expected Euclidean distance between the two picked points.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 10^5$) indicating the number of test cases. For each test case:

The first line contains four integers x_1, y_1, x_2 and y_2 ($-10^3 \le x_1, y_1, x_2, y_2 \le 10^3$) indicating that the two endpoints of the first segment are (x_1, y_1) and (x_2, y_2).

The second line contains four integers x_3, y_3, x_4 and y_4 ($-10^3 \le x_3, y_3, x_4, y_4 \le 10^3$) indicating that the two endpoints of the second segment are (x_3, y_3) and (x_4, y_4).

It is guaranteed that the two segments both have positive lengths.

Output

For each test case, output one line containing one number indicating the expected distance between the two randomly picked points.

Your answer will be considered correct if its absolute or relative error does not exceed 10^{-9}. Formally speaking, suppose that your output is a and the jury's answer is b, your output is accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-9}$.

Example

standard input	standard output
3 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 1 0 0 1 1 1	0.333333333333333333 0.765195716464212691 1.076635732895178009

Note

Thanks to "Computational Intelligence", we can know that:

For the first sample case, the expected distance is

$$\int_0^1 \int_0^1 |x_0 - x_1| \, \mathrm{d}x_0 \, \mathrm{d}x_1 = \frac{1}{3} \approx 0.33333333333333333;$$

For the second sample case, the expected distance is

$$\int_0^1 \int_0^1 \sqrt{x^2 + y^2} \, \mathrm{d}x \, \mathrm{d}y = \frac{\sqrt{2} + \ln(1 + \sqrt{2})}{3} \approx 0.765195716464212691;$$

For the third sample case, the expected distance is

$$\int_0^1 \int_0^1 \sqrt{(x_0 - x_1)^2 + 1} \, \mathrm{d}x_0 \, \mathrm{d}x_1 = \frac{2 - \sqrt{2} + 3\ln(1 + \sqrt{2})}{3} \approx 1.076635732895178009.$$

Problem K. Rainbow Subarray

TEN is an exciting push-your-luck and auction board game. Players may push their luck to draw more cards and use currency to buy additional cards in their attempt to build the longest number sequence in each color. Here let's consider a related problem to the game.

Photo by @freethemeeple on Instagram

Given a sequence $a_1, a_2, \cdots, a_n$ of length n, we say its continuous subarray $a_l, a_{l+1}, a_{l+2}, \cdots, a_r$ is a rainbow subarray if $a_{i+1} - a_i = 1$ for all $l \le i < r$. Specifically, a subarray of length 1 is always a rainbow subarray.

You can perform at most k operations. Each operation allows you to increase or decrease an element in the sequence by one. Calculate the maximum possible length of the longest rainbow subarray after performing the operations.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains two integers n and k ($1 \le n \le 5 \times 10^5$, $0 \le k \le 10^{15}$) indicating the length of the sequence and the maximum number of operations you can perform.

The second line contains n integers $a_1, a_2, \cdots, a_n$ ($1 \le a_i \le 10^9$) indicating the sequence.

It's guaranteed that the sum of n of all test cases will not exceed 5×10^5.

Output

For each test case output one line containing one integer indicating the maximum possible length of the longest rainbow subarray after performing at most k operations.

Example

standard input	standard output
5 7 5 7 2 5 5 4 11 7 6 0 100 3 4 5 99 100 5 6 1 1 1 1 1 5 50 100 200 300 400 500 1 100 3	4 3 5 1 1

Note

For the first sample test case, we can perform 4 operations and change the sequence to $\{7, 3, 4, 5, 6, 11, 7\}$. The longest rainbow subarray is $\{3, 4, 5, 6\}$, so the answer is 4.

For the second sample test case, we cannot perform any operation. The longest rainbow subarray is $\{3, 4, 5\}$, so the answer is 3.

For the third sample test case, we can perform 6 operations and change the sequence to $\{-1, 0, 1, 2, 3\}$. The whole sequence is a rainbow subarray, so the answer is 5.

Problem L. Ticket to Ride

Please note the UNUSUAL MEMORY LIMIT of this problem.

Ticket to Ride is a railway-themed German-style board game. In the game, players collect and play train cards to build railroads across the map. Points are earned based on the length of the built routes, and whether the player can connect distant cities which are determined by drawing ticket cards.

Photo by @garyjames on BoardGameGeek

Consider a one-dimensional version of the game. There are $(n + 1)$ cities arranged in a row, numbered from 0 to n from left to right. For each $1 \le i \le n$, you can connect city $(i - 1)$ and city i by placing a railroad between them.

There are m ticket cards which reward the player for connecting cities. The i-th card can be represented by three integers l_i, r_i and v_i, indicating that if city l_i and r_i are connected through railroads (that is, for all $l_i < j \le r_i$, there is a railroad placed between city $(j - 1)$ and j), you will gain v_i points.

For each $1 \le k \le n$, calculate the maximum points you can gain if you place exactly k railroads. If you fail to gain any reward then you get 0 points.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains two integers n and m ($1 \le n, m \le 10^4$) indicating the maximum number of railroads you can place and the number of ticket cards for rewarding.

For the following m lines, the i-th line contains three integers l_i, r_i and v_i ($0 \le l_i < r_i \le n$, $1 \le v_i \le 10^9$) indicating that if city l_i and r_i are connected through railroads, you will gain v_i points.

It's guaranteed that neither the sum of n nor the sum of m of all test cases will exceed 10^4.

Output

For each test case output one line containing n integers separated by a space, where the i-th integer indicates the maximum points you can gain if you place exactly i railroads.

Please, DO NOT output extra spaces at the end of each line, or your solution may be considered incorrect!

Example

standard input	standard output
2 4 3 0 2 3 3 4 2 0 3 1 3 1 1 3 100	2 3 5 6 0 100 100

Note

Let $(i-1, i)$ be a railroad between city $(i-1)$ and i. For the first sample test case:

- If you place 1 railroad, you can place $(3, 4)$, then get the second reward. The answer is 2.
- If you place 2 railroads, you can place $(0, 1)$ and $(1, 2)$, then get the first reward. The answer is 3.
- If you place 3 railroads, you can place $(0, 1)$, $(1, 2)$ and $(3, 4)$, then get the first and the second reward. The answer is $3 + 2 = 5$.
- If you place all 4 railroads you can get all rewards. The answer is $3 + 2 + 1 = 6$.

Problem M. Almost Convex

This is a story about Kevin, a friend of Little Cyan Fish.

Kevin is the chief judge of the International Convex Polygon Championship (ICPC). He proposed a geometry task for the contest. However, since he is inexperienced in computational geometry, he couldn't generate a correct convex polygon for the tests of the task.

Kevin was very saddened by this. His good friend, Little Cyan Fish, consoled him by saying, "Although the data you generated is not a convex polygon, you can call it an almost-convex polygon!"

Given a set of points S (containing at least 3 points) in a two-dimensional plane, where no two points coincide and no three points are collinear. Little Cyan Fish calls a polygon P an almost-convex polygon, if and only if:

- The polygon P is simple, i.e., its vertices are distinct and no two edges of the polygon intersect or touch, except that consecutive edges touch at their common vertex.
- The vertices of the polygon belong to S, and all points in S are either inside or on the boundary of the polygon.

Let $\mathbb{U}$ be the set consisting of all almost-convex polygons. It can be shown that $\mathbb{U}$ is a finite set and is not empty. Therefore, there exists a polygon R such that $|R|$ is the minimum among all polygons in $\mathbb{U}$ ($|R|$ is the number of vertices of polygon R).

Kevin and Little Cyan Fish want you to calculate the number of polygons $Q \in \mathbb{U}$ such that $|Q| \leq |R| + 1$.

Input

There is only one test case in each test file.

The first line contains an integer n ($3 \leq n \leq 2 \times 10^3$) indicating the number of points in set S.

For the following n lines, the i-th line contains two integers x_i and y_i ($-10^6 \leq x_i, y_i \leq 10^6$) indicating a point (x_i, y_i) in set S.

It is guaranteed that no two points in S coincide and no three points are collinear.

Output

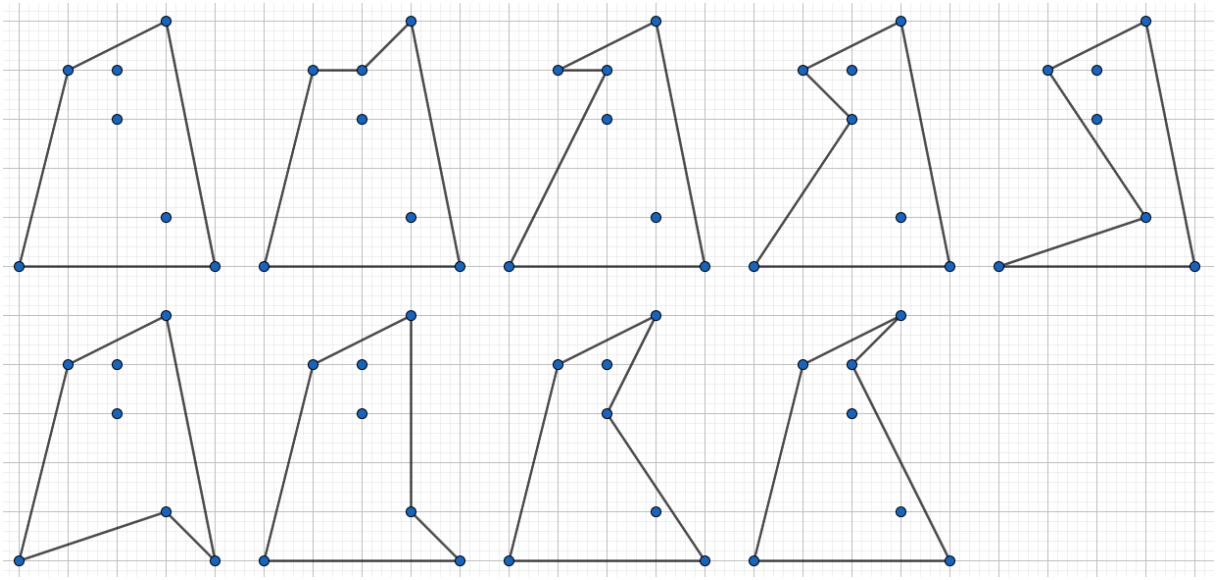
Output one line containing one integer indicating the number of polygons Q.

Examples

standard input	standard output
7 1 4 4 0 2 3 3 1 3 5 0 0 2 4	9
5 4 0 0 0 2 1 3 3 3 1	5
3 0 0 3 0 0 3	1

Note

For the first sample test case, $|R| = 4$. All polygons Q are shown below.



For the second sample test case, $|R| = 3$. All polygons Q are shown below.