The 3rd Universal Cup

Stage 5: Moscow

July 27-28, 2024

This problem set should contain 13 problems on 21 numbered pages.

Problem A. Counting Permutations

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

You are given an array a of size n and two integers m_1 and m_2. We call array p *good* if the following conditions are met:

- p is a permutation of size n, consisting of different integers from 1 to n.
- The array $a_{p_1} \bmod m_1,\ a_{p_2} \bmod m_1,\ \ldots,\ a_{p_n} \bmod m_1$ is a non-decreasing array.
- The array $a_{p_1} \bmod m_2,\ a_{p_2} \bmod m_2,\ \ldots,\ a_{p_n} \bmod m_2$ is a non-increasing array.

Count the number of good permutations p by modulo 998244353.

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10\,000$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains three integers n, m_1 and m_2 ($1 \le n \le 100\,000$, $1 \le m_1, m_2 \le 10^4$) — the length of the array and two modules.

The next line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the array.

It is guaranteed that the sum of n for all test cases does not exceed 100 000.

Output

For each test case, print a single integer — the answer to the problem by modulo 998244353.

Example

standard input	standard output

3
5 2 3
1 2 3 4 10
4 2 4
1 2 3 4
3 8 9
1 1 1
	2
	0
	6

Note

In the first test case, there are two good permutations $p = [2, 4, 5, 1, 3]$ and $p = [2, 5, 4, 1, 3]$.

In the second test case, there are no good permutations.

In the third test case, all 6 permutations are good.

Problem B. Bookshelf Tracking

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Professor E.D. Pryanik, the head of the department of programming history and philosophy (PHP), is absolutely convinced that only a student who has thoroughly studied all n volumes of his fundamental work "The Science of Programming" can write a term paper under his guidance. Student should study all books, and only then E.D. Pryanik conducts the interview with him. If the professor is satisfied with the results, he assigns a topic for the term paper to the student.

Professor wants you to help track the order of books on the bookshelf in his office. The bookshelf contains n books numbered with integers from 1 to n. It is guaranteed that n is an even number. The bookshelf can be represented as a permutation of integers from 1 to n — the order of books on the bookshelf.

There are two operations that can happen with the bookshelf:

- Professor swaps two books on the bookshelf.
- A new student comes to the office to read all n books.

  He reads books in order from 1 to n:

  - n times the student takes the next book. An empty space remains in the position of the book.
  - If the number of books to the left of the empty space is smaller than the number of books to the right of the empty space, he moves all the books on the left of the empty space one position to the right. Otherwise, he moves all books on the right of the empty space one position to the left. As a result, the empty space goes to the first, or to the last position.
  - The student returns the book to the empty space.

Given the initial order of books on the bookshelf and the sequence of operations. Tell the professor the final order of books after applying all operations.

Input

The first line contains two integers n, q ($2 \le n, q \le 3 \cdot 10^5$). It is guaranteed that n is an even number.

The second line contains a permutation of numbers from 1 to n — the initial order of books on the bookshelf.

Each of the following q lines contains a description of the next operation. Each line starts with the symbol "`E`" or "`R`" describing the type of the operation.

- If the symbol is "`E`" the line contains two integers i, j ($1 \le i, j \le n$, $i \ne j$) — the indices of books that should be swapped (book numbers, not positions in the permutation).
- If the symbol is "`R`" the student comes to the office to read all books.

Output

Print n numbers — the final order of books on the bookshelf.

Example

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

Problem C. Painting Fences

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

After a huge defeat in ICPC competition, Matvey decided to go paint fences.

His task is to paint a "small" (as was said for him) fence into black color. The fence is a checkered rectangle of size $n \times m$. Each cell can be either white or black. Initial colors are given.

Matvey can paint using the following operation:

- Select some straight horizontal or vertical line going along the edges and intersecting the rectangle.
- On one side with respect to the selected line, do not change the colors.
- On the other side with respect to the selected line, firstly paint all cells white. After that, on this side, paint black all cells for which the symmetrical cell with respect to the selected line is black.

What is the minimum number of operations that should be used to paint all cells black?

Input

The first line contains two integers n, m ($1 \le n \cdot m \le 10^6$) — the size of the fence.

Each of the next n lines contains a string consisting of m symbols 0 or 1. If the j-th symbol in the i-th string is equal to 0, the cell in the i-th row and j-th column is white. Otherwise, it is black.

It is guaranteed that at least one black cell exists.

Output

Print a single integer — the minimum number of described operations that should be used to paint all cells black.

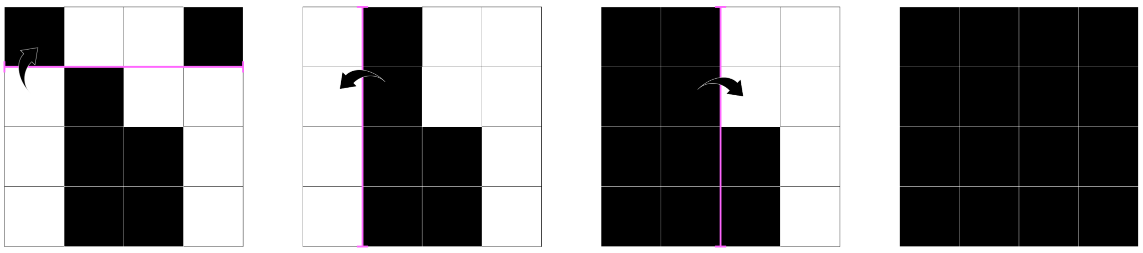
Example

standard input	standard output

4 4
1001
0100
0110
0110
	3

Note

The way to paint all cells black for the first test is presented in the picture below.



Problem D. Function with Many Maximums

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

For a positive integer $a > 0$, let us define the function $f_a : \mathbb{Z}_{\geq 0} \to \mathbb{Z}$:

$$f_a(x) = \begin{cases} a + x, \text{ if } x \leq a \\ 0, \text{ otherwise} \end{cases}$$

You are given a positive integer b. Construct the non-empty set of distinct integers $a_1, a_2, \ldots, a_n$, such that the function $f(x) = \sum\limits_{i=1}^{n} f_{a_i}(x)$ has at least b points with the maximum function value. Formally:

$$\left|\left\{x \in \mathbb{Z}_{\geq 0} : f(x) = \left(\max_{y \in \mathbb{Z}_{\geq 0}} f(y)\right)\right\}\right| \geq b$$

Input

The only line contains a single integer b ($1 \leq b \leq 100\,000$).

Output

On the first line, print a single integer n ($1 \leq n \leq 500\,000$) — the size of the array a.

On the next line, print n distinct integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^{12}$).

It is guaranteed that under given constraints the answer exists.

If there are multiple answers, you can print any.

Example

standard input	standard output
4	5 2 3 5 10 12

Note

In the first test, $f(x) = f_2(x) + f_3(x) + f_5(x) + f_{10}(x) + f_{12}(x)$. Maximum value is $\max\limits_{y \in \mathbb{Z}_{\geq 0}} f(y) = 42$. We have 4 points with maximum value $f(2) = f(3) = f(5) = f(10) = 42$.

Problem E. Building a Fence

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

There is a flower bed that was recently arranged in the yard of Nikodim's house. The neighbor's dogs often ran through the flower bed, and Nikodim decided to protect the flowers. He planned to put a small fence around the flower bed.

In the store "All for garden" you can buy a plastic section of a fence. Each section has a length s.

After the purchase, if the customer wants, the salespersons can cut some initial sections into **exactly** two smaller sections of any size (not necessarily integer length). The new sections can not be cut again. Different initial sections can be cut into new sections of different length.

The flower bed is a rectangle of size $w \times h$. Nikodim wants to make a fence of exactly this size. To do that, each side of the rectangle should be represented as a sum of some sections (each section can be used at most once). Note that it is not necessary to use all parts of sections.

Determine the minimum number of sections of length s he should buy.

Input

The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases.

Each of the next t lines contains three integers w, h, s ($1 \le w,\ h,\ s \le 10^8$) – the dimensions of the flower bed and the length of the section.

Output

For each test case, print the minimum number of sections of length s Nikodim should buy to build the fence.

Example

standard input	standard output

7
7 9 4
1 1 2
1 1 4
4 6 2
3 3 5
10 6 4
1 11 5
	8
	2
	4
	10
	4
	8
	5

Note

In the first test case, 8 sections should be bought. After that, two sections can be divided into smaller sections of length 1 and 3. After that, fence can be built: two times $7 = 4+3$ and two times $9 = 4+4+1$.

Problem F. Teleports

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

There are n chests arranged in a row, numbered from 1 to n. One of the chests contains treasure, and all the rest are empty. You want to find the treasure, but opening all the chests takes too much time, so you want to determine the chest that contains the treasure.

You also have n teleports, the i-th of them is located on top of chest i. You can activate any teleport, and the treasure will be symmetrically teleported to the other side of the teleport. So, if you activate the teleport a, and the treasure is located inside chest b, the new position for the treasure will be $b+(a-b)\cdot 2$. If there is a chest with that number, the treasure will be teleported to the chest with number $b+(a-b)\cdot 2$. If there is no chest with that number, the treasure will remain in chest b and you will know that teleportation was not successful.

Each teleport has it's own cost to use, for a single use of teleport i, you have to pay c_i. Find the smallest amount of money you have to spend, so that after using multiple teleports you can determine the position of the treasure.

Input

The first line contains a single integer n ($1 \le n \le 500$) — the number of chests.

The second line contains n integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le 10^9$) — the cost of a single use of each teleport.

Output

Output the minimum amount of money you need, so that you can determine the position of the treasure.

Examples

standard input	standard output
3 5 2 1	4
12 18 19 11 2 20 15 18 1 14 1 1 1	6

Note

In the first example, first you can activate teleport 3. If teleportation was successful, the only valid chest for that is 3. If teleportation was not successful, you will know that treasure is inside chests 1 or 2. Then you can activate teleport 2, after that the treasure will be inside chests 2 or 3. After that you can activate teleport 3, if teleportation was successful, the treasure is inside chest 3, otherwise it's inside chest 2.

Problem G. Exponent Calculator

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Kirill wants to make his own calculator. However, he faced a serious problem — he should add buttons that calculate some complex functions.

His calculator has 16 registers $\$0, \$1, \ldots, \$15$, each containing 64-bit floating point number (in double-precision floating-point format).

Calculator operations should be represented by the sequence of commands using registers:

- $\$i =$ operand1 operation operand2
- $\$i =$ value

In the first command, operand1 and operand2 can be either real numbers or one of the registers. At least one of the operands should be a register, and operation should be one of `+`, `-`, `*`, which correspond to sum, subtraction, and multiplication, respectively.

In the second command, value should be some real number (registers are not allowed).

To calculate this operation, the initial value of register \$1 is equal to x (input of the operation), initial values of other registers are equal to 0. After that, all commands are executed. Firstly, the value on the right hand side is calculated and assigned to the register on the left hand side. Output of the operation is the value of register \$0, and the values of other registers are not taken into account.

Write a sequence of at most 25 commands that will calculate the function e^x with good precision for all $x \in [-20, 20]$ (e is Euler's number and the base of natural logarithm).

Input

You can consider this problem as an output-only problem.

The problem has exactly one test which contains a single string "`input`".

Output

The first line contains a single integer k ($0 \le k \le 25$) — the number of commands. The next k lines should describe a sequence of commands that will calculate e^x with good precision.

Each of the next k lines should contain a command in one of two given formats. Conditions described above should be satisfied. For all registers $\$i$ the condition $0 \le i \le 15$ should be satisfied.

Note that in this problem, it is important to use the first line only for the number of commands and print exactly one operation in each of the next k lines (without any blank lines in between). All operands, operation, and $=$ sign should be separated by space.

For all $x = \frac{n}{10^5}$ for an integer n ($-2 \cdot 10^6 \le n \le 2 \cdot 10^6$), the sequence of commands will be executed.

Suppose the answer equals $y_1 = e^x$, and the calculated answer equals y_2. The answer will be considered correct if $\dfrac{|y_1 - y_2|}{y_1} \le 10^{-9}$ is satisfied.

Example

standard input	standard output

`input`	`6`
	`$0 = $0 * $0`
	`$0 = 2.718281828459`
	`$0 = $1 + $0`
	`$15 = 8.1000000737 * $0`
	`$12 = $15 * 0.123456789`
	`$0 = $12 - $1`

Note

The presented answer is given as an example and is incorrect.

It is easy to see that for all x, it calculates e approximately (so the answer is correct for $x = 1$).

Problem H. Ancient Country

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	512 megabytes

In the ancient world, there were many wars. It was determined that in order for a city to be protected, it needs to be bounded with a fortification wall that forms a convex polygon. The king of some ancient country decided to build some cities and protect each of them with fortification walls. Please help him to achieve the highest protection of the country.

A territory of the ancient country is a simple polygon $P_1P_2\dots P_n$. All points strictly inside or on the boundary of this polygon are controlled by the country. All vertices of the polygon are distinct, no two edges of the polygon intersect or touch, except that consecutive edges touch at their common vertex. Every two consecutive edges are not collinear.

There is a tower at each point P_i of the polygon.

Initially, there are no cities in the country. The king can make a city $P_{i_1}P_{i_2}\dots P_{i_k}$ in the country if the following properties hold:

- The points $P_{i_1}P_{i_2}\dots P_{i_k}$ are distinct vertices of the polygon that forms a country.
- The polygon $P_{i_1}P_{i_2}\dots P_{i_k}$ is convex and has positive area.
- Points $P_{i_1}, P_{i_2}, \dots, P_{i_k}$ occur in this order in a counterclockwise traversal of the city boundary.
- There are no other towers P_i on the boundary of the city, except points $P_{i_1}, P_{i_2}, \dots, P_{i_k}$.
- All points inside or on the boundary of the city are controlled by the country, i.e. all points inside or on the boundary of the polygon $P_{i_1}P_{i_2}\dots P_{i_k}$ lie inside or on the boundary of the polygon $P_1P_2\dots P_n$.

You are given two non-negative integers w and c. The protection level of a city is equal to $2 \cdot \text{area}(P_{i_1}P_{i_2}\dots P_{i_k}) + w \cdot k + c$. The protection level of the country is the sum of the protection levels of all its cities.

The king wants to select some cities in the country in such a way that no two cities share a common point (including boundaries). Note that some points of the country may not be covered by any city.

What is the maximum possible protection level of the country?

Input

The first line contains a single integer n ($3 \le n \le 200$) — the number of vertices in the country.

Each of the next n lines contains two integers x_i, y_i ($|x_i|, |y_i| \le 10^6$) — coordinates of point P_i. It is guaranteed that the vertices are given in the counter-clockwise order.

The next line contains two integers w, c ($0 \le w, c \le 10^{13}$).

Output

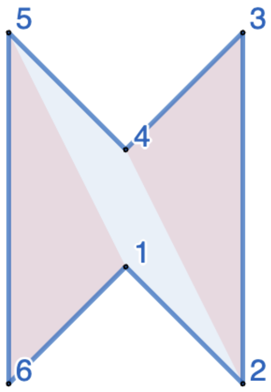
Print a single integer — the maximum possible protection level of the country.

Examples

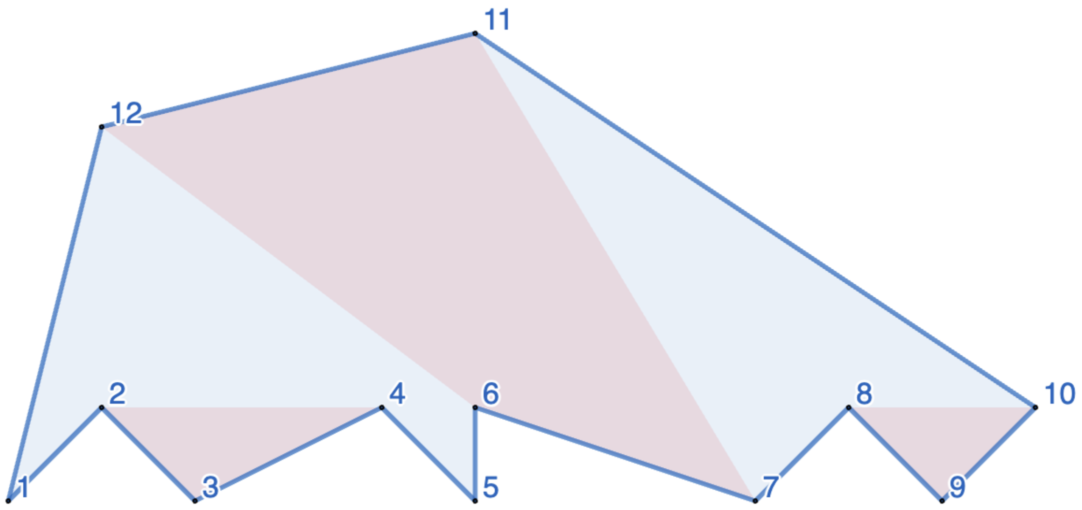
standard input	standard output
6	16
0 0	
1 -1	
1 2	
0 1	
-1 2	
-1 -1	
1 2	
12	3000063
0 0	
1 1	
2 0	
4 1	
5 0	
5 1	
8 0	
9 1	
10 0	
11 1	
5 5	
1 4	
3 1000000	
12	61
0 0	
1 1	
2 0	
4 1	
5 0	
5 1	
8 0	
9 1	
10 0	
11 1	
5 5	
1 4	
0 9	

Note

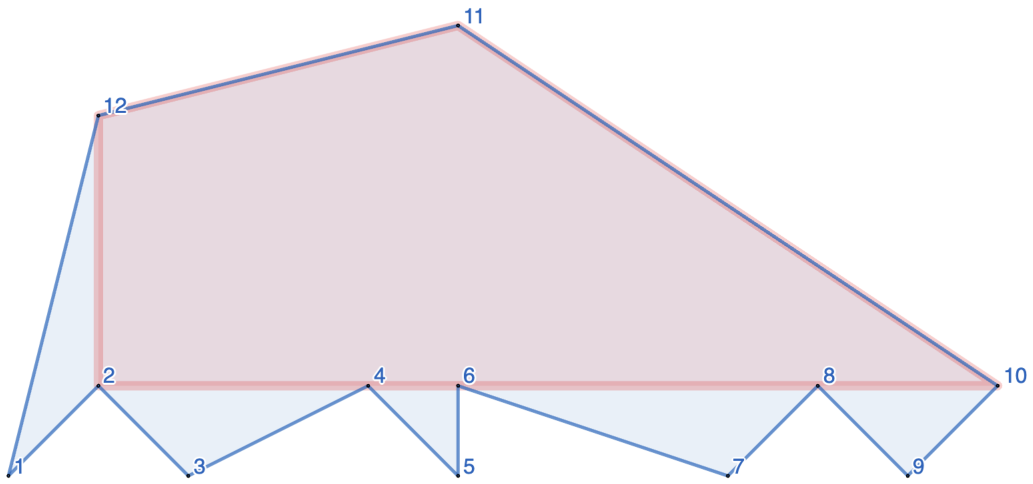
The images below display cities for the first three examples (cities are highlighted in red).



Sample 1 (possible optimal answer: $\{1, 5, 6\}, \{2, 3, 4\}$)



Sample 2 (possible optimal answer: $\{2, 3, 4\}, \{6, 7, 11, 12\}, \{8, 9, 10\}$)



Sample 3 (possible optimal answer: $\{2, 4, 6, 8, 10, 11, 12\}$)

Problem I. Marks Sum

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

This is a runtwice problem. Your solution will be executed twice.

Ivan studies very badly. During his studies, he got only marks 1 and 2. To present his results to his parents, he decided to use a sum of marks (in this case, they won't be able to recognize his bad marks).

Ivan's marks can be represented by the string s, consisting of characters 1 and 2. He started downloading it, but unfortunately, the connection to the internet was lost and he downloaded only some prefix of the string $s[1..i]$ (for some $1 \leq i < |s|$, so this prefix is not empty and is not equal to the full string). After reconnection, the remaining suffix of the string will be downloaded.

Your solution will be executed twice:

1. During the first execution, you will be given the downloaded prefix: $s[1..i]$.

   You should give some number $0 \leq d < \min(i+1, 2000)$ and $0 \leq info < 2000$.

2. During the second execution, you will be given the number $info$ and the remaining suffix of the string with d last symbols of the downloaded prefix: $s[(i+1-d)..|s|]$.

   You should give the sum of all symbols in s.

Input

The first line contains a single integer *type* ($1 \leq type \leq 2$). In the first execution, $type = 1$, and in the second execution, $type = 2$.

The input consists of multiple independent test cases. The second line contains a single integer t ($1 \leq t \leq 1000$) — the number of test cases. The description of the test cases follows.

- If $type = 1$, each of the next t lines contains a single string, consisting of characters 1 and 2 — downloaded prefix $s[1..i]$.
- If $type = 2$, each of the next t lines contains an integer $info$ ($0 \leq info < 2000$), that your solution was given in the first execution for this test case, and a string, consisting of characters 1 and 2 — downloaded suffix $s[(i+1-d)..|s|]$ for d, that your solution was given in the first execution for this test case.

It is guaranteed, that all strings s and numbers i for them are fixed in advance.

It is guaranteed, that the sum of $|s|$ for all test cases does not exceed 10^6.

Note, that test cases can be reordered for the second execution.

Output

- If $type = 1$, for each test case, your solution should print two integers $0 \leq d < \min(i+1, 2000)$, $0 \leq info < 2000$. They will be used for the second execution on this test case.
- If $type = 2$, for each test case, your solution should print a single integer — the sum in the string s.

Example

Note, that given outputs of the solution in the first execution are given as example and may be different for your solution.

First execution.

standard input	standard output

1
3
1
222
1212111122
	0 42
	1 11
	4 22

Second execution.

standard input	standard output

2
3
11 22211
22 11222221
42 2
	12
	21
	3

Note

In the first test, there are 3 test cases and strings s for them are:

```
12
2222211
12121111222221
```

Note, that for the second execution, test cases were reordered.

Problem J. Prefix Divisible by Suffix

Input file:	`standard input`
Output file:	`standard output`
Time limit:	8 seconds
Memory limit:	512 megabytes

You are given two positive integers n and c.

Consider a positive integer x and its decimal representation $x = \overline{(x_{k-1}x_{k-2}\dots x_1x_0)}_{10}$ (without leading zeros). We call the number x *good* if its decimal representation can be divided into non-empty prefix $p = \overline{(x_{k-1}\dots x_i)}_{10}$ and non-empty suffix $s = \overline{(x_{i-1}\dots x_0)}_{10}$ (for some $0 < i \le k-1$), such that p is divisible by $s + c$. Note, that $x_{i-1} = 0$ is possible.

Calculate the number of good integers from 1 to n.

Input

The only line contains two positive integers n and c ($1 \le n \le 10^{14}$, $1 \le c \le 10^4$).

Output

Print a single integer — the number of good integers from 1 to n.

Examples

standard input	standard output
`20 1`	`2`
`111 4`	`9`
`1111 10`	`75`
`1000000 7`	`111529`

Note

In the first test, good numbers are 10, 20.

In the second test, good numbers are 40, 51, 62, 73, 80, 84, 95, 101, 106.

Problem K. Train Depot

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

You are the chief of the train depot. The depot has n railroad switches and $n-1$ railroad tracks between them, so that all switches are connected by a single railroad network. The i-th railroad track connects switches a_i and b_i and has a length of c_i meters.

There are m trains that are going to be stored at the depot. The i-th train is going to enter the depot at switch 1 and go by the shortest path to switch s_i until it's first wagon reaches the switch s_i. All other wagons should follow each other in the consecutive order.

Unfortunately, each part of railroad track can be occupied by only one train, and all the trains may not fit inside the depot. You know that the i-th train has k_i wagons numbered from 1 to k_i. The j-th wagon has a value $v_{i,j}$ and a length of $l_{i,j}$ meters. When the wagon is inside the depot, it occupies $l_{i,j}$ meters of railroad (possibly of multiple tracks). For each train, only some (possibly zero or all) of it's first wagons can enter the depot. If the first t_i wagons of train i will enter the depot, they will occupy $\sum_{j=1}^{t_i} l_{i,j}$ consecutive meters of tracks directly from the switch s_i towards the switch 1. Note again, that all entered wagons should fit inside the depot, two wagons can touch, but cannot occupy the same part of railroad track.

You can choose the order in which trains will enter the depot, and you can choose how many first wagons of each train should enter the depot. Find the maximum total value of all the wagons that can be fit inside the depot.

Input

The first line contains two integers n and m ($2 \le n \le 200\,000$, $1 \le m \le 200\,000$) — the number of switches in the depot and the number of trains.

The next $n-1$ lines describe railroad tracks. Each line contains three integers a_i, b_i, and c_i ($1 \le a_i, b_i \le n$, $1 \le c_i \le 10^9$) — the switches that are connected by the railroad track and the length of the track.

The next $3m$ lines describe the trains. For each train, the first line contains two integers k_i and s_i ($1 \le k_i \le 200\,000$, $1 \le s_i \le n$) — the number of wagons in the train and the switch where the train will stop.

The second line contains k_i integers $v_{i,1}, v_{i,2}, \ldots, v_{i,k_i}$ ($1 \le v_{i,j} \le 10^9$) – the values of wagons for the i-th train.

The third line contains k_i integers $l_{i,1}, l_{i,2}, \ldots, l_{i,k_i}$ ($1 \le l_{i,j} \le 10^9$) — the lengths of wagons for the i-th train.

It is guaranteed that $\sum k_i \le 200\,000$.

Output

Output the maximum total value of all the wagons that can be fit inside the depot.

Examples

standard input	standard output

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

6 4
1 2 2
2 3 1
2 4 2
4 5 1
4 6 2
2 3
1 1
2 1
1 5
3
2
1 4
5
4
3 6
1 1 10
1 2 2
	12

Problem L. Array Spread

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

Vasya wants to construct an array a of size n consisting of non-negative real numbers.

He has m segments $[l_1, r_1], [l_2, r_2], \ldots, [l_m, r_m]$ $(1 \le l_i \le r_i \le n)$, defining m subarrays of array a.

Let us define *spread* of array a as follows:

- Calculate subarray sums $s_i = \sum\limits_{j=l_i}^{r_i} a_j$.
- Spread of array a is equal to $\frac{\max\limits_{j=1}^{m} s_j}{\min\limits_{j=1}^{m} s_j}$. If $\min\limits_{j=1}^{m} s_j = 0$, spread is equal to $+\infty$.

Find the minimum spread among all possible arrays a. It is guaranteed that the minimum exists. You should find the answer by modulo 998244353.

We can show that each answer can be written in the form $\frac{p}{q}$ where p and q are relatively prime integers and $q \not\equiv 0 \pmod{998244353}$. The answer by modulo 998244353 is equal to $(p \cdot q^{-1})$ modulo 998244353.

Input

The input consists of multiple test cases. The first line contains a single integer t $(1 \le t \le 2000)$ — the number of test cases. The description of the test cases follows.

The first line of each test case contains two integers n, m $(1 \le n \le 10^9$, $1 \le m \le 2000)$ — array size and the number of segments.

Each of the next m lines contains two integers l_i, r_i $(1 \le l_i \le r_i \le n)$ — description of the i-th segment.

It is guaranteed that the sum of m for all test cases does not exceed 2000.

Output

For each test case print a single integer — the answer to the problem by modulo 998244353.

Example

standard input	standard output

3
3 3
1 3
2 3
1 2
12 6
2 3
5 7
1 9
4 8
1 2
7 11
4 5
3 4
2 3
1 2
4 4
1 1
	1
	2
	499122178

Note

In the first test case, the minimum spread is 1 and it can be achieved, for example, for $a = [0, 3, 0]$. Subarray sums are $s = [3, 3, 3]$.

In the second test case, the minimum spread is 2 and it can be achieved, for example, for $a = [0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 2, 0]$. Subarray sums are $s = [1, 1, 2, 1, 1, 2]$.

In the third test case, the minimum spread is $\frac{3}{2}$ and it can be achieved, for example, for $a = [2, 1, 1, 2]$. Subarray sums are $s = [3, 2, 3, 2, 2]$.

Problem M. Uniting Amoebas

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

There are n amoebas placed on a circle. They are numbered with integers from 1 to n. For each i, amoebas i and $i+1$ are neighbors. Also, amoebas 1 and n are neighbors. The i-th amoeba has volume a_i.

Two neighboring amoebas can unite. As a result, both of them disappear and a new amoeba appears at their place. The volume of a new amoeba is equal to the sum of volumes of two amoebas. The cost of this operation is equal to the minimum volume of two amoebas. The amoebas will be uniting until one amoeba is left.

What is the minimum total cost of all operations until one amoeba is left?

Input

The input consists of multiple test cases. The first line contains a single integer t $(1 \le t \le 10^5)$ — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n $(2 \le n \le 2 \cdot 10^5)$ — the number of amoebas.

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

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

Output

For each test case, print a single integer — the minimum total cost of all operations until one amoeba is left.

Example

standard input	standard output

3
3
1 1 1
4
0 1 0 2
2
100 42
	2
	1
	42