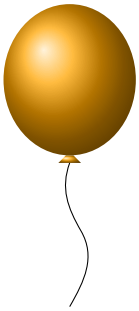
XAPC Algorithmic Programming Contest > badas

A McCarthy function



Time limit: 1.0s
Memory limit: 256MB

Last night, you had a strange dream. In it, you discovered a mysterious function $f(n)$ defined over integers. Upon waking up, the definition was still clear in your mind:

$$f(n) = \begin{cases} n + 20 & \text{if } n < 48, \\ f\big(f(n - 21)\big) & \text{if } n \geq 48. \end{cases}$$

Curious about what this function actually computes, you decide to evaluate it for various values of n.

Input

The first line contains a single integer t ($1 \leq t \leq 10^5$) — the number of test cases. Each of the next t lines contains a single integer n ($1 \leq n \leq 10^{18}$).

Output

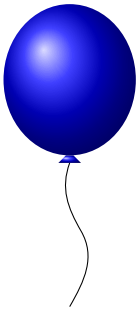
For each test case, output a single integer — the value of $f(n)$.

Each answer should be printed on its own line.

Samples

Sample input 1	Sample output 1
4 1 2 25 48	21 22 45 67

B Walking the Cube



Time limit: 3.0s
Memory limit: 256MB

This is an interactive problem.

Alice and Bob are playing a game on a k-dimensional Boolean cube. Its vertices are all binary strings of length k. There is an edge between two vertices if their strings differ at exactly one position.

Initially, the token is at the vertex $00\ldots0$. Alice and Bob make moves alternately. In one move, a player must choose one bit and flip it. In other words, from the current vertex, the token can be moved to any vertex connected by an edge.

It is forbidden to move the token to a vertex that has already been visited(the initial vertex $00\ldots0$ counts as visited). The player who cannot make a move loses.

You may choose whether you want to play as Alice or as Bob. After that, you have to play the game and win.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 100$). The description of the test cases follows.

The only line of each test case contains a single integer k ($3 \le k \le 10$) — the dimension of the Boolean cube.

Then interaction follows.

Interaction

First, you need to output one of the following lines:

- `Alice` — if you want to play first;
- `Bob` — if you want to play second.

After that, the game starts at the vertex $00\ldots0$.

If you chose Alice, you should make the first move. If you chose Bob, the jury will make the first move.

To make a move, output a single integer i ($1 \le i \le k$) — the position of the bit you want to flip.

After each move of the jury, you should read a single integer i ($1 \le i \le k$) — the position of the bit flipped by the jury.

If you cannot make a move, you should print 0. The interactor will print -1 and finish the interaction.

If the jury cannot make a move and loses, you should read 0 and proceed to the next test case.

After printing each move, do not forget to output the end of line and flush the output. Otherwise, you will get *Idleness limit exceeded* verdict.

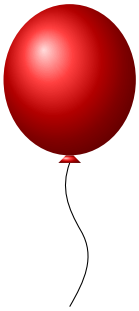
If, at any interaction step, you read -1, your solution must terminate immediately. This means that your solution has made an invalid move, you lost, or some other error happened. Failing to terminate can result in an arbitrary verdict.

The jury program is adaptive, meaning it's behavior might be different on the same test based on the participant's choices.

Samples

Sample input 1	Sample output 1
1 3 random	

C Educational Round



Time limit: 1.0s
Memory limit: 256MB

You wrote a contest where you competed against n other participants, and you were given p problems to solve.

You solved exactly x problems. For each problem i, you know that it was solved by exactly k_i of your competitors.

You do not know which participants solved which problems. Find the minimum possible number of participants who solved strictly more problems than you.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 1000$). The description of the test cases follows.

The first line of each test case contains three integers n, p, and x ($1 \le n, p \le 10^6$, $0 \le x \le p$) — the number of your competitors, the number of problems, and the number of problems you solved.

The second line of each test case contains p integers $k_1, k_2, \ldots, k_p$ ($0 \le k_i \le n$), where k_i is the number of other competitors who solved problem i.

It is guaranteed that the sum of n and the sum of p over all test cases do not exceed 10^6.

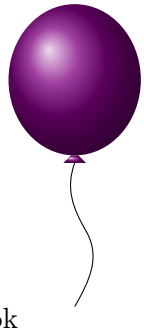
Output

For each test case, output a single integer — the minimum possible number of participants who could have solved strictly more problems than you.

Samples

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

D Directed 67



Time limit: 1.0s
Memory limit: 256MB

As a waterpark director, you have to plan the rides. You already know that your rides will look like a directed graph with n vertices and m edges. However, you don't know yet how long each of the edges will be, which is determined by the edge's weight. Based on the materials you have, each weight should be one of the digits $d_1, d_2, \ldots, d_k$.

The weight of a directed path is the sum of the digits assigned to the edges of the path (edges in a path are allowed to repeat).

You know that if there is a directed path of weight w divisible by 67, it will become the only ride used, and you want to avoid that. Provide the assignment, where there is no path with weight divisible by 67, or determine there is no such assignment.

Input

The first line contains two integers n and m ($1 \le n, m \le 200000$).

The second line contains a non-empty string s consisting of distinct characters from `1` to `9`. These are the allowed digits d_i ($1 \le d_i \le 9$).

Each of the next m lines contains two integers u_i and v_i ($1 \le u_i, v_i \le n$). This means that there is a directed edge from u_i to v_i. Multiple edges and self-loops are allowed.

Output

If there is no valid assignment, print `NO`.

Otherwise, print `YES` on the first line.

On the second line print m integers $a_1, a_2, \ldots, a_m$, where a_i is the digit assigned to the i-th edge. Each a_i must be one of the allowed digits.

If there are several valid assignments, print any of them.

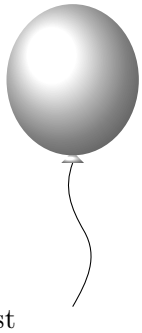
Samples

Sample input 1	Sample output 1
2 1 3 1 2	YES 3

Sample input 2	Sample output 2
4 3 679 1 2 1 3 3 4	YES 6 7 7

Sample input 3	Sample output 3
3 3 123 1 2 2 3 3 1	NO

E Ready... Set... Go!



Time limit: 1.0s
Memory limit: 256MB

In an orienteering competition, a course is described by a sequence of controls. A competitor must visit the controls in the given order. Each control is denoted by a unique positive integer. The pair of adjacent controls in a sequence is called a leg.

For a mass-start race, course setters often make the race pass in two laps. Different competitors may receive different combinations of parts of these laps, so that they can't simply follow each other. Such variations are made in such way that after both laps every runner completed the same set of legs, but did it in a different order.

You are given two sequences of controls, which describe two laps of one of the runners. It is guaranteed that:

- Each lap is a sequence of control points that starts in the start control point and ends in the finish control point.
- There is no control point that is visited twice during the single lap
- There are some control points that are common between laps. The relative order of common control points is the same for the two laps.
- Each leg is present only once in the union of the two laps.

Given two laps for one of the competitors, determine the total number of variations possible for the race. Each variation should respect the given constraints, and union of the laps should have the same set of legs.

Compute the number of possible variations modulo 998244353.

Input

The first line contains two integers n and m ($2 \le n, m \le 200000$) — the lengths of the two laps.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the control points of the first lap.

The third line contains m integers $b_1, b_2, \ldots, b_m$ ($1 \le b_i \le 10^9$) — the control points of the second lap.

It is guaranteed that:

- All a_i are distinct.
- All b_i are distinct.
- $a_1 = b_1$ and $a_n = b_m$.

- If a value appears in both sequences, then the relative order of such values is the same in both sequences. There is no pair of adjacent values in the common sequence that is adjacent in both a and b.

Output

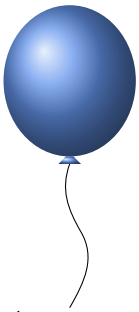
Output a single integer — the number of valid variations modulo 998244353.

Samples

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

Sample input 2	Sample output 2
2 3 6 7 6 1 7	2

F Aux Champs Élysées



Time limit: 5.0s
Memory limit: 512MB

Maximilien is a student officer at Polytechnique and is preparing for the traditional *défilé du 14 juillet* on the Champs-Élysées. However, Maximilien prefers chaos over order, and thus plans to organize a prank that will definitely make an impression on the big day!

For the prank, Maximilien settles on the following idea: for each row of students parading, he will choose some number a_i of students in that row to wear their hats backward.

Not satisfied with this alone, Maximilien adds an extra constraint: for every student wearing their hat backward, the immediate neighbors in the same row must wear their hats straight. More precisely, if such a student has a neighbor directly to their left in the row, that neighbor must wear their hat straight; similarly, if they have a neighbor directly to their right in the row, that neighbor must also wear their hat straight.

As a loyal comrade of Maximilien and a zealous mathematician, you decide to help him determine how many different valid configurations of hats for each row satisfy this rule. As this number can be large, output it modulo $10^9 + 7$.

Input

The input consists of the following lines: · The first line contains two integers n and m ($1 \leq n, m \leq 10^5$), the dimensions of the parade block (there are n rows, each containing m students). · Each of the next n lines contains a single integer a_i ($0 \leq a_i \leq m$), the number of students in row i who must wear their hats backward.

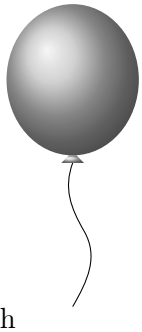
Output

Output n integers in n different lines where the i-th integer represents the number of valid hat configurations for the i-th row that satisfy all the given constraints. As this number can be large, output it modulo $10^9 + 7$.

Samples

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

G Space Invaders



Time limit: 2.0s
Memory limit: 512MB

A certain species of space invaders were found on the streets of Paris, taking a mosaic form. Each invader has a fixed shape represented by the following 8×13 pattern:

```
#..#..#..#..#
#.#########.#
###.#####.###
###.#####.###
#####...#####
..#########..
..###...###..
.##....##....
```

Each `#` denotes an occupied cell, and each `.` denotes empty cell.

For a positive integer k, a *scaled invader* is obtained by replacing every cell of the pattern with a $k \times k$ block. Occupied cells become filled $k \times k$ blocks, and empty cells become empty $k \times k$ blocks.

You want to code an «invader detector». Your program will be given a photo of a grid that may or may not contain space invaders. You need to determine if there exists such configuration of space invaders that:

- There is at least one space invader.
- Every occupied cell belongs to exactly one space invader,
- All space invaders have the same scale k,
- It is allowed for one space invader's occupied cells to overlap with empty cells of another space invader.

Input

The first line contains two integers n and m ($1 \le n, m \le 4000$) — dimensions of the grid.

Each of the next n lines contains a row of a grid, that has exactly m characters, each being "#" or ".", that represent occupied or empty cell correspondingly.

Output

Print `YES` if the photo is valid, otherwise print `NO`.

Samples

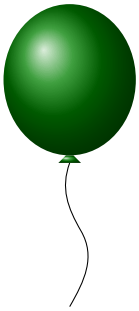
Sample input 1	Sample output 1
8 13 #..#..#..#..# #.#########.# ###.#####.### ###.#####.### #####...##### ..#########.. ..###...###.. .##....##....	YES

Sample input 2	Sample output 2
4 5 .#.#. #.#.# .#.#. #.#.#	NO

Sample input 3	Sample output 3
8 30 ..#..#..#..#..#..#..#..#..#..# ..#.#########.#..#.#########.# ..###.#####.###..###.#####.### ..###.#####.###..###.#####.### ..#####...#####..#####...##### ....#########......#########.. ....###...###......###...###.. ...##....##.......##....##....	YES

Sample input 4	Sample output 4
9 16 #######..####### ##.......####### ##...........##. ##..........##.. ######.....##... #######...##.... ##...##..##..... ##...##.##...... #########.......	NO

H Ash-e-emö



Time limit: 1.0s
Memory limit: 256MB

You are given a clock where a day consists of h hours and every hour consists of m minutes. The time is written in the format $a : b$ ($0 \le a < h$, $0 \le b < m$). **Note** that the leading zeros are preserved. So for $h = 240$, $m = 100$, $a = 9$, $b = 41$ the time would look like `009:41`.

We call a time `a:b` *nice* if both of the following values are perfect squares[1]:

1. The total number of minutes since the beginning of the day: $a \cdot m + b$.
2. The concatenation of a and b written as a decimal number, where m is written with leading zeros to always have the same number of digits.

More formally, let d be the number of digits in $m - 1$. Then define: $\text{concat}(a, b) = a \cdot 10^d + b$

For example, for $h = 24$, $m = 60$, the time $1 : 21$ is *nice*, because:

- $1 \cdot 60 + 21 = 81 = 9^2$
- $\text{concat}(1, 21) = 121 = 11^2$

Your task is to determine how many *nice* times there are in a day.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 1000$). The description of the test cases follows.

The only line of each test case contains two integers h and m ($2 \le h, m \le 5 \cdot 10^5$).

It is guaranteed that the sum of h and the sum of m across all test cases do not exceed $5 \cdot 10^5$.

Output

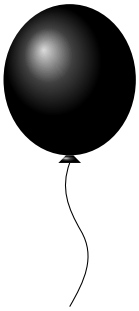
For each test case, output a single integer — the number of *nice* times in the corresponding day.

[1] A perfect square is an integer that is the square of another integer.

Samples

Sample input 1	Sample output 1
7 24 60 24 91 24 99 24 100 90000 90000 10 400000 400000 10	10 11 10 49 327 640 2000

I Fishing deal with ziplines



Time limit: 2.0s
Memory limit: 256MB

High above the valleys, the legendary Mount Thalassa feeds a vast river system. From a single source at the summit, the river splits into smaller and smaller streams, forming a tree-like structure flowing down the mountain.

Along these waterways lie isolated fishing spots. The fishing spots correspond exactly to the nodes of this tree. Each spot produces a fixed amount of fish every year. Because of the rugged terrain, these locations are difficult to access, and the transport infrastructure is limited to what was built long ago: downstream rails that allow fish to be transported downriver, and ziplines that allow fish to be transported upriver.

Each transport connection has a maximum yearly capacity.

Two rival companies, AquaDyn Fisheries and BlueCurrent Co., have been instructed to split the mountain's fishing resources evenly.

They agree on the following:

- Every fishing spot must belong to exactly one company.
- The fishing spots owned by each company must form a connected component of the tree.
- Each company will ensure it can collect fish from its own fishing spots to its warehouses, but will not build new infrastructure to redistribute fish between spots.
- Any transfer of fish between the two companies must rely solely on the existing transport network.

Fishing spots produce a fixed amount of fish, but they can store and handle any amount of incoming fish.

After choosing a valid repartition of the fishing spots, the two companies compare the total yearly fish production of their respective regions.

- If both sides already have the same total, the agreement is immediate.
- Otherwise, fish must be transferred from one side to the other using the existing network.

Your task is to determine a valid repartition of the fishing spots so that the two companies can end up with equal total amounts of fish, while minimizing the amount of fish that needs to be transferred between them. The transfer can only be done on integer quantities.

If multiple choices are possible, any one minimizing the required transfer is acceptable.

Input

The first line contains an integer n ($1 \leq n \leq 10^5$), the number of fishing spots.

The second line contains n integers $a_1, a_2, \ldots, a_n$, where $a_i \leq 10^9$ is the yearly fish production of spot i.

The following $n-1$ lines describe the transport network. Each line contains four integers: u v c_{uv} c_{vu}

- u and v are connected fishing spots
- $c_{uv} \leq 10^9$ is the capacity for transporting fish from u to v
- $c_{vu} \leq 10^9$ is the capacity for transporting fish from v to u

It is guaranteed that the graph formed by the fishing spots and connections is a tree.

Output

Output a single integer — the minimum amount of fish that must be transferred between the two companies after choosing an optimal repartition.

If it is impossible to balance the total amount of fish for any valid repartition, output -1.

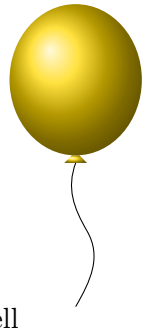
Samples

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

Sample input 2	Sample output 2
2 1 9 1 2 5 0	-1

Sample input 3	Sample output 3
2 1 9 1 2 0 5	4

J Dangerous Maze



Time limit: 3.0s
Memory limit: 256MB

In an abandoned industrial complex, represented by a grid of M rows and N columns, each cell (i, j) has an altitude $h_{i,j}$. A toxic gas begins to fill the complex. If the gas reaches a height H, all cells whose altitude is strictly less than H become impassable ($h_{i,j} < H$). Cells where $h_{i,j} \geq H$ remain safe.

You are initially at cell $(1, 1)$ (top left) and must reach the exit located at cell (M, N) (bottom right). You can move horizontally or vertically between two adjacent cells, provided that both cells are passable.

Your goal is to determine the maximum value of H such that there still exists a path between the entrance and the exit. Note that the entrance and the exit themselves must also be passable.

Input

The first line contains two integers M and N ($1 \leq M, N \leq 500$). The next M lines each contain N integers $h_{i,j}$ ($0 \leq h_{i,j} \leq 10^9$).

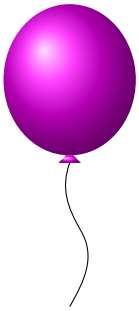
Output

A single integer: the maximum gas height H.

Samples

Sample input 1	Sample output 1
3 3 10 5 8 2 3 12 15 10 10	5

K Caps and Sneakers



Time limit: 1.0s
Memory limit: 256MB

You are moving from one apartment to another.

There are n caps and n pairs of sneakers in the old apartment. You also have a suitcase of size W that you can use to transport caps and sneakers.

For every integer i from 1 to $W-1$, consider the following moving scenario:

- every cap has size i;
- every pair of sneakers has size $W-i$;
- regardless of the scenario, the suitcase has capacity W.

In each scenario, you may make several trips from the old apartment to the new one. After every such trip except the last one, you return back to the old apartment.

During every trip from the old apartment to the new one:

- you may put some items into the suitcase, with total size at most W;
- you may wear at most one cap, which does not occupy suitcase capacity;
- you must wear exactly one pair of sneakers, which also does not occupy suitcase capacity.

During every return trip from the new apartment to the old one you must wear exactly one pair of sneakers and you carry the empty suitcase back. You don't have to wear the cap, so if you had one on during the forward trip, you can leave it at the new apartment.

Let $f(i)$ be the minimum possible number of trips from the old apartment to the new one required to move all n caps and all n pairs of sneakers to the new apartment in the scenario where cap has size i.

Your task is to compute $\sum_{i=1}^{W-1} f(i)$ modulo 998244353.

Input

The only line contains two integers n and W ($2 \le n,\ W \le 5 \times 10^{17}, n \times W \le 10^{18}$)

Output

Print one integer: the value of $\sum_{i=1}^{W-1} f(i)$ modulo 998244353.

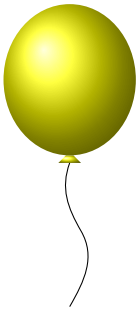
Samples

Sample input 1	Sample output 1
3 5	8

Sample input 2	Sample output 2
5 5	14

Sample input 3	Sample output 3
67 69	3602

L Tree Game



Time limit: 5.0s
Memory limit: 256MB

Alice and Bob play a game on an undirected tree. The players start at two different nodes of the tree. The players take turns and Alice plays first. At their turn, the player must move to an empty adjacent node and can optionally decide to destroy the node they just left.

Destroyed nodes can no longer be used by the players. Note also that the tree can become a forest when nodes are destroyed.

If the two players end up in two different connected components, the game immediately ends and the player that is in the biggest component wins (if the two components are of the same size, this is considered a draw). If a player cannot move, he or she loses. Assuming both players play optimally, given the tree and the starting nodes of each player, who will win?

Input

The first three lines contain the following integers:

- N ($4 \leq N \leq 10^6$) , the number of tree nodes, numbered from 1 to N.
- A ($1 \leq A \leq N$), Alice's start node.
- B ($1 \leq B \leq N$ and $B \neq A$), Bob's start node.

The next $N - 1$ lines contain the edges of the tree. Each edge is represented by the two nodes it connects.

Output

A single line with either:

- ALICE WINS
- BOB WINS
- DRAW

Samples

Sample input 1	Sample output 1
4 1 4 1 2 2 3 3 4	BOB WINS

Sample input 2	Sample output 2
6 1 6 1 4 2 4 2 3 4 5 5 6	DRAW