# Problem A. City

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 1 second |
| Memory limit: | 256 mebibytes |

Hi ICPCer, welcome to Xi'an.

Being a beautiful ancient city, Xi'an is the capital city of Zhou, Qin, Han, and Tang Dynasties. With a long history, the streets in Xi'an have a grid pattern.

Attracted by the streets' structure, Coach *Pang* would like to conduct his research on them. He draws an $n \times m$ grid on the board. The grid consists $n + 1$ vertical line segments and $m + 1$ horizontal line segments. The vertical and horizontal line segments intersect at exactly $(n+1) \times (m+1)$ points, forming $n \times m$ unit squares. We call the $(n + 1) \times (m + 1)$ intersections *grid points*. Output the number of line segments(not only vertical or horizontal) $l$ satisfying the following three conditions:

1. The length is not zero.
2. Both endpoints of $l$ are grid points.
3. The midpoint of $l$ is a grid point.

## Input

The only line contains two integers $n, m (1 \leq n, m \leq 1000)$.

## Output

Print the answer in a single line.

## Examples

| standard input | standard output |
|---|---|
| 1 1 | 0 |
| 2 3 | 14 |

## Problem B. Black and White

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 1 second |
| Memory limit: | 256 mebibytes |

*Master Pang* walks from the bottom-left corner of a $n \times m$ chessboard to the top-right corner. The chessboard contains $n+1$ horizontal line segments and $m+1$ vertical line segments. The horizontal line segments are numbered from 0 to $n$ from bottom to top and the vertical ones are numbered from 0 to $m$ from left to right. The intersection of horizontal line segment $r$ and vertical segment $c$ is denoted by $(r, c)$. The bottom-left corner is $(0, 0)$ and the top-right corner is $(n, m)$. At each step, he can only walk from $(x, y)$ to $(x, y+1)$ or from $(x, y)$ to $(x+1, y)$.

Each of the $n \times m$ cells is colored white or black. A cell with corners $(i, j), (i+1, j), (i, j+1), (i+1, j+1)$ $(0 \le i < n, 0 \le j < m)$ is colored white if and only if $i \equiv j \pmod 2$.

Given *Pang*'s walking path from $(0, 0)$ to $(n, m)$, his score is $a - b$ where $a$ is the number of white cells to the left of his walking path and $b$ is the number of black cells to the left of his walking path.

Help *Master Pang* count the number of walking paths with score $k$ modulo 998244353.

## Input

The first line contains a single integer $T$ — the number of test cases $(1 \le T \le 100)$.

Each of the next $T$ lines contains three integers $n$, $m$ and $k$ $(1 \le n \le 100000, 1 \le m \le 100000, -100000 \le k \le 100000)$.

## Output

For each test case, output a single integer — the answer modulo 998244353.

## Example

| standard input | standard output |
|---|---|
| 5<br>1 1 0<br>1 1 -1<br>2 2 1<br>2 2 0<br>4 4 1 | 1<br>0<br>1<br>4<br>16 |

## Problem C. Dirichlet $k$-th root

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 1 second |
| Memory limit: | 256 mebibytes |

*Mathematician Pang* learned Dirichlet convolution during the previous camp. However, compared with deep reinforcement learning, it's too easy for him. Therefore, he did something special.

If $f, g : \{1, 2, \ldots, n\} \to \mathbb{Z}$ are two functions from the positive integers to the integers, the Dirichlet convolution $f * g$ is a new function defined by:

$$(f * g)(n) = \sum_{d|n} f(d)g(\frac{n}{d}).$$

We define the $k$-th power of an function $g = f^k$ by

$$f^k = \underbrace{f * \cdots * f}_{k\ \text{times}}.$$

In this problem, we want to solve the inverse problem: Given $g$ and $k$, you need to find a function $f$ such that $g = f^k$.

Moreover, there is an additional constraint that $f(1)$ and $g(1)$ must equal to 1. And all the arithmetic operations are done on $\mathbb{F}_p$ where $p = 998244353$, which means that in the Dirichlet convolution, $(f * g)(n) = \left(\sum_{d|n} f(d)g(\frac{n}{d})\right) \bmod p$.

## Input

The first line contains two integers $n$ and $k$ $(2 \le n \le 10^5, 1 \le k < 998244353)$ .

The second line contains n integers $g(1), g(2), ..., g(n)$ $(0 \le g(i) < 998244353, g(1) = 1)$.

## Output

If there is no solution, output $-1$.

Otherwise, output $f(1), f(2), ..., f(n)$ $(0 \le f(i) < 998244353, f(1) = 1)$. If there are multiple solutions, print anyone.

## Example

| standard input | standard output |
|---|---|
| `5 2`<br>`1 8 4 26 6` | `1 4 2 5 3` |

## Problem D. Fire

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 1 second |
| Memory limit: | 256 mebibytes |

*Pang* lives on a tree with $n$ vertices. The vertices are labelled as $1, 2, \ldots, n$ and *Pang* is in vertex 1. Each vertex has a temperature. On the morning of each day after day 0, the temperature of each vertex decreases by 1. The temperature doesn't decrease on day 0. On the afternoon of each day, *Pang* can travel to an adjacent vertex, provided that he is at a vertex with positive temperature and his destination vertex has a non-negative temperature. On the evening of each day, if the temperature is higher than or equal to 0, *Pang* can cast magic which increases the temperature of the vertex he is in by $k$. For each pair of adjacent vertices $a$ and $b$, *Pang* can travel from vertex $a$ to vertex $b$ at most once (and from $b$ to $a$ at most once). He can choose not to travel and stay in the current vertex.

*Pang* wants to cast his magic on each vertex exactly once. He also tries to stay at vertex 1 as long as possible, before traveling to any other city. Given the temperature of each vertex right before the morning of the day 1, on which day must *Pang* prepare for departing? If *Pang* prepares on day $i$, he can cast his magic on that day and will make his first move on day $i + 1$. If he cannot cast his magic on each vertex exactly once even if he prepares for departing on the day 0, output $-1$.

### Input

The first line contains two integers $n$ and $k$ ($2 \leq n \leq 100000, 0 \leq k \leq 1000000000$).

Each of the next $n - 1$ lines contains two integers $x$ and $y$, indicating an edge between vertices $x$ and $y$ ($1 \leq x, y \leq n$).

The $(n + 1)$-th line contains $n$ integers $a_1, a_2, \ldots, a_n$ — the temperature of vertex $i$ right before the morning of day 1 ($0 \leq a_i \leq 1000000000$).

It's guaranteed that the input is a tree structure.

### Output

If he cannot cast his magic on each vertex exactly once, output $-1$.

Otherwise, output a single integer $x$ — he must prepare for departing from vertex 1 on day $x$. Day 1 is the day after day 0, and so on.

### Examples

| standard input | standard output |
|---|---|
| 3 1<br>1 2<br>1 3<br>4 3 5 | 1 |
| 3 1<br>1 2<br>1 3<br>2 10 10 | -1 |

## Problem E. Flow

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 1 second |
| Memory limit: | 256 mebibytes |

One of *Pang*'s research interests is the maximum flow problem.

A directed graph $G$ with $n$ vertices is *universe* if the following condition is satisfied:

- $G$ is the union of $k$ vertex-independent simple paths from vertex 1 to vertex $n$ of the same length.

A set of paths is vertex-independent if they do not have any internal vertex in common.

A vertex in a path is called internal if it is not an endpoint of that path.

A path is simple if its vertices are distinct.

Let $G$ be a *universe* graph with $n$ vertices and $m$ edges. Each edge has a non-negative integral capacity. You are allowed to perform the following operation any (including 0) times to make the maximum flow from vertex 1 to vertex $n$ as large as possible:

Let $e$ be an edge with positive capacity. Reduce the capacity of $e$ by 1 and increase the capacity of another edge by 1.

*Pang* wants to know what is the minimum number of operations to achieve it?

## Input

The first line contains two integers $n$ and $m$ ($2 \le n \le 100000, 1 \le m \le 200000$).

Each of the next $m$ lines contains three integers $x, y$ and $z$, denoting an edge from $x$ to $y$ with capacity $z$ ($1 \le x, y \le n$, $0 \le z \le 1000000000$).

It's guaranteed that the input is a *universe* graph without multiple edges and self-loops.

## Output

Output a single integer — the minimum number of operations.

## Examples

| standard input | standard output |
|---|---|
| `4 3`<br>`1 2 1`<br>`2 3 2`<br>`3 4 3` | `1` |
| `4 4`<br>`1 2 1`<br>`1 3 1`<br>`2 4 2`<br>`3 4 2` | `1` |

## Problem F. Game

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 2 seconds |
| Memory limit: | 256 mebibytes |

Alice and Bob are playing *Luzhanqi*. Each of them has a *permutation* of the following 24 pieces:

- one Field Marshal, order 9
- one General, order 8
- two Major Generals, order 7
- two Brigadier Generals, order 6
- two Colonels, order 5
- two Majors, order 4
- three Captains, order 3
- three Lieutenants, order 2
- three Engineers, order 1
- two Bombs
- three Landmines

To determine the winner, we repeat the following process until someone wins the game or the game ends in a draw:

- If both permutations are empty, the game ends in a draw.
- If Alice's permutation is empty, Bob wins the game.
- If Bob's permutation is empty, Alice wins the game.
- Let the first piece in Alice's permutation be $A$ and the first piece in Bob's permutation be $B$. The following is the outcome of the battle between $A$ and $B$:
  1. If $A$ and $B$ are the same types of pieces, or if one of $A$ and $B$ is Bomb, they are both removed.
  2. Otherwise, if one of $A$ and $B$ is Landmine and the other is Engineer, the Landmine is removed and the Engineer stays alive.
  3. Otherwise, if one of $A$ and $B$ is Landmine and the other's order is greater than 1, the Landmine stays alive and the other one is removed.
  4. Otherwise, we compare the order of $A$ and $B$ and the piece with smaller order is removed.

Bob knows Alice's permutation in advance and can decide his permutation based on that information. After Bob deciding his permutation, Alice can swap two pieces in Bob's permutation. Can Bob construct a permutation that wins against Alice's permutation no matter which pair of pieces she swaps?

### Input

The first line contains one integer $T$ denoting the number of test cases ($1 \le T \le 100$).

Each of the next $T$ lines contains 24 integers denoting Alice's permutation:

- 40 represents Field Marshal
- 39 represents General
- 38 represents Major Generals
- 37 represents Brigadier Generals
- 36 represents Colonels
- 35 represents Majors
- 34 represents Captains
- 33 represents Lieutenants
- 32 represents Engineers
- 31 represents Landmines
- 30 represents Bombs

It is guaranteed that all permutations are chosen uniformly at random and contains exactly the 24 pieces described in the statement.

## Output

Output one line for each test case.

If Bob cannot construct the required permutation, print −1.

Otherwise, print 24 integers representing Bob's permutation in the same format as in the input. If there are multiple solutions, print any. Bob's permutation must contain exactly the 24 pieces described in the statement.

## Example

standard input
```
4
40 39 38 38 37 37 36 36 35 35 34 34 34 33 33 33 32 32 32 31 31 31 30 30
34 31 36 33 31 39 37 38 35 32 32 35 36 31 34 32 38 40 30 33 30 34 33 37
37 30 40 38 36 38 32 34 36 35 37 32 34 33 31 30 33 31 35 34 33 39 31 32
30 33 32 39 37 38 35 40 34 30 31 37 31 33 31 33 34 32 36 36 35 34 32 38
```

standard output
```
34 36 30 39 33 38 37 31 34 30 33 35 38 31 37 33 40 31 35 32 32 36 32 34
34 32 32 38 40 33 33 30 31 34 31 35 37 32 34 36 33 31 38 30 36 37 35 39
38 33 32 31 36 34 30 34 33 40 32 37 38 30 37 35 33 35 32 31 34 31 39 36
37 34 33 36 34 35 31 38 32 38 31 32 37 30 30 31 33 36 32 33 40 39 34 35
```

## Problem G. Happiness

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 5 seconds |
| Memory limit: | 256 mebibytes |

*Pang* has graduated from college 3 years and he really misses the time he spent with ICPC (Interspecies Collegiate Pokemon Camp).

There are 10 problems in one contest in ICPC. $n$ participating teams have 300 minutes to solve them. After the contest, teams are ranked according to the most problems solved. Teams who solve the same number of problems are ranked by least total time. The total time is the sum of the time consumed for each problem solved. The time consumed for a solved problem is the time elapsed from the beginning of the contest to the submittal of the first accepted run plus 20 penalty minutes for every previously rejected run for that problem. There is no time consumed for a problem that is not solved. If two teams tie, their *solution time list*s are calculated. A team's solution time list is a list consisting of solution times of all problems solved by that team, sorted in descending order. The solution time of one problem is the time elapsed from the beginning of the contest to the submittal of the first accepted run of that problem. (We do not add a penalty for the solution time.) The team with a lexicographically smaller solution time list has a better rank. A list $(a_1, \ldots, a_k)$ is lexicographically smaller than $(b_1, \ldots, b_k)$ if there exists an integer $i \in [1, k]$ such that $a_i < b_i$ and $a_j = b_j$ for all integers $j \in [1, i)$. If teams still tie, *Pang*'s team is assumed to have a better rank.

After determining the rank, prizes will be awarded. Initially, a team with rank $r$ will get $\lfloor 5000/r \rfloor$ happiness. Then medals are awarded: Teams with rank 1 to $\lfloor n/10 \rfloor$ are awarded gold medal. The *happiness* of receiving a gold medal is 1200. Teams with rank $\lfloor n/10 \rfloor + 1$ to $3\lfloor n/10 \rfloor$ are awarded silver medal. The *happiness* of receiving a silver medal is 800. Teams with rank $3\lfloor n/10 \rfloor + 1$ to $6\lfloor n/10 \rfloor$ are awarded bronze medal. The *happiness* of receiving a bronze medal is 400. In addition to medals, for each problem, the team solved it first gets 800 happiness. The team with at least one solution and the smallest solution time overall teams and all problems gets an extra 700 happiness. The team with at least one solution and the largest solution time overall teams and all problems gets an extra 500 happiness. In the case of a tie, *Pang*'s team can always get happiness.

There were $n$ teams in a contest *Pang* participated. He remembers all the submissions (time and verdict) of all other teams. For each problem, he also remembers if he knew the solution to that problem and the number of rejected runs and times he needed to solve it.

If *Pang* solved problems in the wisest order, what is the maximum happiness he could get? Note that *Pang* cannot solve any problem after 300 minutes from the beginning of the contest (He can solve problems at exactly 300 minutes). Once *Pang* solves a problem, he needs to submit it immediately and solve another one. He can't postpone his submission to get the last submission happiness.

### Input

The first line contains an integer $n$ denoting the number of teams ($10 \le n \le 300$, $n$ is a multiple of 10).

Each of the next $n - 1$ lines describes one team and contains the statuses of the 10 problems. For each problem, if it is not solved by the team, the status contains a single character "-". Otherwise, the status contains two integers $t$ and $w$ separated by a single space denoting the solution time and the number of rejected runs before the solution time ($1 \le t \le 300, 0 \le w \le 10$). Statuses of different problems are separated by ",".

The last line describes *Pang*'s team. For each problem, if *Pang* did not know how to solve it, the status contains a single character "-". Otherwise, the status contains two integers $x$ and $y$ separated by a single space denoting the required time and the number of rejected runs before *Pang* could solve it ($1 \le x \le 300, 0 \le y \le 10$). Statuses of different problems are separated by ",".

There are no extra spaces and other characters in the statuses of *Pang* and other teams.

## Output

Output one integer — the maximum happiness.

## Example

| standard input | standard output |
|---|---|
| 10<br>233 1,-,-,7 7,257 4,173 5,117 1,-,-,<br>85 3<br>-,231 0,167 0,257 7,-,-,122 4,283 0,<br>215 4,-<br>41 1,-,290 8,-,-,-,-,246 7,120 3,184<br> 9<br>142 8,243 7,69 0,-,41 9,-,279 1,264<br> 4,-,74 9<br>53 8,-,187 9,60 1,48 8,99 10,-,-,55<br> 7,259 5<br>250 0,-,-,-,166 0,16 3,-,82 4,73 0,<br>184 3<br>-,-,-,-,105 3,-,-,-,152 4,-<br>-,84 5,98 8,-,120 8,241 3,94 1,-,28<br> 7,109 8<br>280 6,246 5,58 9,-,-,-,-,-,-,-<br>38 10,-,227 10,187 9,182 1,-,203 9<br>,254 7,-,- | 1800 |

## Note

Note that the sample input and sample output contain wrapped lines to fit in the width of page.

# Problem H. King

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 2 seconds |
| Memory limit: | 256 mebibytes |

As we all know, the number of *Pang*'s papers follows exponential growth. Therefore, we are curious about *King* sequence.

You are given a prime $p$. A sequence $(a_1, a_2, \ldots, a_n)$ is a *King* sequence if and only if there is an integer $1 \le q < p$ such that for all integers $i \in [2, n]$, $qa_{i-1} \equiv a_i \pmod{p}$.

Given a sequence $B = (b_1, \ldots, b_m)$, what is the length of the longest *King* subsequence of $B$?

A subsequence is a sequence that can be derived from another sequence by deleting some elements without changing the order of the remaining elements.

*Pang* is super busy recently, so the only thing he wants to know is whether the answer is greater than or equal to $\frac{n}{2}$.

If the length of the longest *King* sequence is less than $\frac{n}{2}$, output $-1$. Otherwise, output the length of the longest *King* subsequence.

## Input

The first line contains an integer $T$ denoting the number of test cases ($1 \le T \le 1000$).

The first line in a test case contains two integers $n$ and $p$ ($2 \le n \le 200000$, $2 \le p \le 1000000007$, $p$ is a prime). The sum of $n$ over all test cases does not exceed 200000.

The second line in a test case contains a sequence $b_1, \ldots, b_n$ ($1 \le b_i < p$).

## Output

For each test case, output one line containing the answer which is $-1$ or the length of the longest *King* subsequence.

## Example

standard input
```
4
6 1000000007
1 1 2 4 8 16
6 1000000007
597337906 816043578 617563954 668607211 89163513 464203601
5 1000000007
2 4 5 6 8
5 1000000007
2 4 5 6 7
```

standard output
```
5
-1
3
-1
```

# Problem I. Moon

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 2 seconds |
| Memory limit: | 256 mebibytes |

Let $S$ be a sphere with radius 1 and center $(0, 0, 0)$. Let $a_0, a_1, \ldots, a_n$ be $n+1$ points on the surface of $S$. The positions of $a_1, \ldots, a_n$ are fixed while the position of $a_0$ is a uniform random point on the surface of $S$. Let $f$ be 1 if there exists a hemisphere of $S$ that contains $a_0, \ldots, a_n$ and 0 otherwise. Calculate the expected value of $f$.

## Input

The first line contains an integer $n$ denoting the number of points ($0 \le n \le 100000$).

The $i$-th line of the next $n$ lines contains three integers $x, y, z$ denoting the point $a_i = \left(\frac{x}{\sqrt{x^2+y^2+z^2}}, \frac{y}{\sqrt{x^2+y^2+z^2}}, \frac{z}{\sqrt{x^2+y^2+z^2}}\right)$ ($-1000000 \le x, y, z \le 1000000, x^2 + y^2 + z^2 \ne 0$).

It is guaranteed that $a_1, \ldots, a_n$ are distinct.

## Output

Output the answer.

The answer will be considered correct if its absolute or relative error doesn't exceed $10^{-6}$.

## Example

| standard input | standard output |
|---|---|
| 3<br>1 0 0<br>0 1 0<br>0 0 1 | 0.875000000000 |

## Problem J. Permutation

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 1 second |
| Memory limit: | 256 mebibytes |

You are given a permutation $p_1, p_2, \ldots, p_n$. You can do the following operations repeatedly:

- Choose an interval $p_l, p_{l+1}, \ldots, p_{l+c} (l \geq 1, l+c \leq n)$ where $p_l$ is the smallest element in this interval, you can permutate $p_{l+1}, \ldots, p_{l+c}$ in arbitrary way.
- Choose an interval $p_l, p_{l+1}, \ldots, p_{l+c} (l \geq 1, l+c \leq n)$ where $p_{l+c}$ is the smallest element in this interval, you can permutate $p_l, \ldots, p_{l+c-1}$ in arbitrary way.

You want to know how many distinct permutations you can get using operations. The answer can be large, output the answer modulo 998244353.

### Input

The first line contains an integer $T$ denoting the number of test cases ($1 \leq T \leq 100000$).

The first line in a test case contains two integers $n$ and $c$ ($2 \leq c \leq 500000$, $2 \leq n \leq 500000$). The sum of $n$ over all test cases does not exceed 500000.

The second line in a test case contains a permutation $p_1, \ldots, p_n$ ($1 \leq p_i \leq n$).

### Output

For each test case, output one line containing the answer modulo 998244353.

### Example

| standard input | standard output |
|---|---|
| 5<br>5 3<br>3 4 2 1 5<br>5 4<br>4 2 1 3 5<br>5 2<br>4 5 3 1 2<br>5 3<br>4 3 2 1 5<br>5 2<br>2 3 1 5 4 | 6<br>1<br>4<br>6<br>4 |

# Problem K. All Pair Maximum Flow

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 6 seconds |
| Memory limit: | 256 mebibytes |

You are given an undirected graph. You want to compute the maximum flow from each vertex to every other vertex.

The graph is special. You can regard it as a convex polygon with $n$ points (vertices) and some line segments (edges) connecting them. The vertices are labeled from 1 to $n$ in the clockwise order. The line segments can only intersect each other at the vertices.

Each edge has a capacity constraint.

Denote the maximum flow from $s$ to $t$ by $f(s,t)$. Output $\left(\sum_{s=1}^{n}\sum_{t=s+1}^{n} f(s,t)\right) \bmod 998244353$.

## Input

The first line contains two integers $n$ and $m$, representing the number of vertices and edges ($3 \le n \le 200000, n \le m \le 400000$).

Each of the next $m$ lines contains three integers $u, v, w$ denoting the two endpoints of an edge and its capacity ($1 \le u, v \le n, 0 \le w \le 1000000000$).

It is guaranteed there are no multiple edges and self-loops.

It is guaranteed that there is an edge between vertex $i$ and vertex $(i \bmod n) + 1$ for all $i = 1, 2, \ldots, n$.

## Output

Output the answer in one line.

## Example

| standard input | standard output |
|---|---|
| 6 8<br>1 2 1<br>2 3 10<br>3 4 100<br>4 5 1000<br>5 6 10000<br>6 1 100000<br>1 4 1000000<br>1 5 10000000 | 12343461 |

# Problem L. Travel

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 2.5 seconds |
| Memory limit: | 256 mebibytes |

"I'm tired of seeing the same scenery in the world." — *Philosopher Pang*

*Pang*'s world can be simplified as a directed graph $G$ with $n$ vertices and $m$ edges.

A *path* in $G$ is an ordered list of vertices $(v_0, \ldots, v_{t-1})$ for some non-negative integer $t$ such that $v_i v_{i+1}$ is an edge in $G$ for all $0 \le i < t-1$. A *path* can be empty in this problem.

A *cycle* in $G$ is an ordered list of distinct vertices $(v_0, \ldots, v_{t-1})$ for some positive integer $t \ge 2$ such that $v_i v_{(i+1) \bmod t}$ is an edge in $G$ for all $0 \le i < t$. All circular shifts of a cycle are considered the same.

$G$ satisfies the following property: Every vertex is in at most one cycle.

Given a fixed integer $k$, count the number of pairs $(P_1, P_2)$ modulo 998244353 such that

1. $P_1, P_2$ are paths;
2. For every vertex $v \in G$, $v$ is in $P_1$ or $P_2$;
3. Let $c(P, v)$ be the number of occurrences of $v$ in path $P$. For every vertex $v$ of $G$, $c(P_1, v) + c(P_2, v) \le k$.

## Input

The first line contains 3 integers $n$, $m$ and $k$ ($1 \le n \le 2000, 0 \le m \le 4000, 0 \le k \le 1000000000$).

Each of the next $m$ lines contains two integers $a$ and $b$, denoting an edge from vertex $a$ to $b$ ($1 \le a, b \le n, a \ne b$).

No two edges connect the same pair of vertices in the same direction.

## Output

Output one integer — the number of pairs $(P_1, P_2)$ modulo 998244353.

## Examples

| standard input | standard output |
|---|---|
| 2 2 1<br>1 2<br>2 1 | 6 |
| 2 2 2<br>1 2<br>2 1 | 30 |
| 3 3 3<br>1 2<br>2 1<br>1 3 | 103 |

# Problem M. Value

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 1 second |
| Memory limit: | 256 mebibytes |

*Pang* believes that one cannot make an omelet without breaking eggs.

For a subset $A$ of $\{1, 2, \ldots, n\}$, we calculate the score of $A$ as follows:

1. Initialize the score as 0.
2. For any $i \in A$, add $a_i$ to the score.
3. For any pair of integers $(i, j)$ satisfying $i \geq 2$, $j \geq 2$, $i \in A$ and $j \in A$, if there exists positive integer $k > 1$ such that $i^k = j$, subtract $b_j$ from the score.

Find the maximum possible score over the choice of $A$.

## Input

The first line contains a single integer $n$ ($1 \leq n \leq 100000$).

The second line contains $n$ integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 1000000000$).

The third line contains $n$ integers $b_1, b_2, \ldots, b_n$ ($1 \leq b_i \leq 1000000000$).

## Output

Print a single integer $x$ — the maximum possible score.

## Examples

| standard input | standard output |
|---|---|
| 4<br>1 1 1 2<br>1 1 1 1 | 4 |
| 4<br>1 1 1 1<br>1 1 1 2 | 3 |