2023「钉耙编程」中国大学生算法设计超级联赛 #9

Problem List

A	Capoo on tree
B	Shortest path
C	Reasoning
D	Broken Dream
E	List Reshape
F	Peek
G	Average
H	Coins
I	Sequence
J	MST problem
K	Cargo
L	Inference

2023/8/15

Problem A. Capoo on tree

Input file:	`standard input`
Output file:	`standard output`
Time limit:	20 seconds
Memory limit:	512 megabytes

Bugcat Capoo planted an apple tree at his doorstep, which has a total of n nodes, and on the i-th node, an apple of size a_i grows. Capoo observes and records the growth of the apples every day. He discovers that:

On the i-th day, if it's raining, all apples of size x_i on the chain from node u_i to node v_i on the tree would grow by 1 unit;

If it's cloudy, all apples of size x_i on the chain from node u_i to node v_i on the tree would shrink by 1 unit;

If it's sunny, then Capoo goes out to check the quality of the apples on the tree. Capoo believes that only apples exceeding size y_i are up to standard apples. He would climb from node u_i to node v_i on the tree, checking all the apples along the path. When Capoo continuously encounters l_i qualifying apples, he gets very happy and records the distance from the starting position p_i to the starting node u_i, then stops checking. Otherwise, if there is no set of l_i consecutive up-standard apples on the entire path, Capoo gets disappointed and creates a scene.

However, Capoo inadvertently lost all the records of p_i. Now, he comes to you with growth records of a total of m days, hoping to find out the distance dis_i from the recorded position p_i to the starting point that he recorded each sunny day he went out to inspect, or to confirm if such a position does not exist.

Formally speaking, for an unrooted tree (V, E) of size n with node i having a weight of a_i, you need to support the following operations:

If we represent the path from u_i to v_i on the tree as

$$path_T(u_i, v_i) = \{p_0, p_1, \cdots, p_k | p_0 = u, p_k = v, (p_i, p_{i+1}) \in E\}$$

That is, we use p_i to represent the i-th point on the path, then operations can be expressed as:

- 1 u_i v_i x_i, for each point p_j satisfying $a_{p_j} = x_i$, update a_{p_j} to $a_{p_j} + 1$
- 2 u_i v_i x_i, for each point p_j satisfying $a_{p_j} = x_i$, update a_{p_j} to $a_{p_j} - 1$
- 3 u_i v_i l_i y_i, seek the minimum j that satisfies $\forall k \in [0, l_i), a_{p_{j+k}} \geq y_i$

Input

The first line contains a positive integer $T (1 \leq T \leq 100)$, indicating the number of test cases.

For each of the next T test cases, the format is:

The first line contains two positive integers $n(2 \leq n \leq 10^5), m(1 \leq m \leq 10^5)$, which represent the total number of nodes on the tree and the number of operations, respectively.

The next line contains n integers, where the i-th integer a_i $(1 \leq a_i \leq n)$ represents the weight of node i.

The next $n-1$ lines consist of two integers $u_i, v_i(1 \leq u_i, v_i \leq n)$ each, indicating an edge on the tree.

The following m lines represent m operations, where the first integer $op(1 \leq op \leq 3)$ of each line indicates the type of operation.

- If $op = 1$, then the next three integers $u_i, v_i, x_i(1 \leq u_i, v_i \leq n, 1 \leq x_i < n)$ represent increasing the weight of all nodes with value x_i on the path from u_i to v_i by one.
- If $op = 2$, then the next three integers $u_i, v_i, x_i(1 \leq u_i, v_i \leq n, 1 < x_i \leq n)$ represent decreasing the weight of all nodes with value x_i on the path from u_i to v_i by one.
- If $op = 3$, then the next four integers $u_i, v_i, l_i, y_i(1 \leq u_i, v_i, l_i, y_i \leq n)$ ask for the distance from node u_i to the closest chain with a length of l_i and node weights greater than or equal to y_i on the path from u_i to v_i.

It is guaranteed that $\sum n \leq 10^6, \sum m \leq 10^6$.

Output

For each query, output a line with a positive integer dis indicating the calculated distance. If the corresponding chain does not exist, output -1.

Example

standard input	standard output

2
5 5
1 2 3 4 5
1 2
1 3
2 4
2 5
3 4 3 4 2
1 4 3 1
3 4 3 4 2
2 4 3 2
3 4 3 4 2
5 5
1 1 1 1 1
1 2
2 3
3 4
4 5
1 1 5 1
2 2 3 2
1 1 5 2
2 2 4 3
3 1 5 2 2
	-1
	0
	-1
	3

Problem B. Shortest path

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	64 megabytes

Forever-chicken has recently been studying a lot of single-source shortest path algorithms, such as Dijkstra's algorithm, Bellman-Ford algorithm, and so on. To test the accuracy and efficiency of these algorithms, he generated his own directed graph with n vertices(numbered from 1 to n) using the following method:

- For each vertex $i(1 \le i \le n/2)$, there is a directed edge from i to $2 * i$.
- For each vertex $i(1 \le i \le n/3)$, there is a directed edge from i to $3 * i$.
- For each vertex $i(1 \le i \le n-1)$, there is a directed edge from i to $i+1$.

Now he wants to know the length of the shortest path from vertex 1 to vertex n. Can you help him with this?

Input

Each test contains multiple test cases. The first line of input contains a single integer $t(1 \le t \le 2000)$ – the number of test cases.

Each test case consists of an integer $n(1 \le n \le 10^{18})$, representing the number of vertices of the graph.

Output

For each test case, output a positive integer representing the distance from vertex 1 to vertex n.

Example

standard input	standard output
4 7 114514 1919810 2147483648	3 19 20 31

Problem C. Reasoning

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	64 megabytes

The reasoning system consists of the following symbols:

1. Parentheses: (and);
2. Logical connectives: $\neg$ and $\rightarrow$;
3. Universal quantifier: $\forall$;
4. Variables: $u - z$;
5. Constants: $a - e$;
6. Functions: $f - h$;
7. Predicates: $P - T$.

This reasoning system also includes concepts such as term, formula, free occurrence, and replacement. Based on these concepts, we can define whether a certain term t can replace a certain variable x without contradiction. This is one of the basis for reasoning, and You wants to solve this problem first.

Term is defined as follows:

1. Every variable v is a term.
2. Every constant c is a term.
3. If $t_1, \dots, t_n$ are terms and f is a function, then $ft_1 \dots t_n$ is a term.

Formula (well-formed formula) is defined as follows:

1. If $t_1, \dots, t_n$ are terms and P is a predicate, then $Pt_1 \dots t_n$ is a formula. This kind of formula is also called the atomic formula.
2. If φ and ψ are formulas, then $(\neg\varphi)$ and $(\varphi \rightarrow \psi)$ are both formulas.
3. If φ is a formula, and v is a variable, then $\forall v \varphi$ is also a formula.

Free occurrence of a variable x in formula φ is defined as follows:

1. If φ is an atomic formula, then x occurs free in φ if and only if x occurs in φ (which means there is a char x inside the string φ).
2. If φ is $(\neg\psi)$, then x occurs free in φ if and only if x occurs free in ψ.

3. If φ is $(\psi \to \gamma)$, then x occurs free in φ if and only if x occurs free in ψ or x occurs free in γ.
4. If φ is $\forall v\psi$, then x occurs free in φ if and only if x occurs free in ψ and $x \neq v$.

For every formula φ, every variable x, and every term t, the replacement φ^x_t is defined as:

1. If φ is an atomic formula, then φ^x_t is the expression formed by simply replacing every char x to string t.
2. If φ is $\neg\psi$, then $(\neg\psi)^x_t = (\neg\psi^x_t)$.
3. If φ is $\psi \to \gamma$, then $(\psi \to \gamma)^x_t = (\psi^x_t \to \gamma^x_t)$
4. If φ is $\forall y\psi$, then $(\forall y\psi)^x_t = \begin{cases} \forall y(\psi^x_t), & \text{if } x \neq y; \\ \forall y\psi, & \text{if } x = y. \end{cases}$

And finally, we can now define the rule of zero-contradiction replacement:

1. For atomic formula φ, t can always replace x in φ without contradiction.
2. If φ is $\neg\psi$, then t can replace x in φ without contradiction if and only if t can replace x without contradiction in ψ.
3. If φ is $\psi \to \gamma$, then t can replace x in φ without contradiction if and only if t can replace x without contradiction in both ψ and γ.
4. If φ is $\forall y\psi$, then t can replace x in φ without contradiction if and only if
   (a) x doesn't occur free in φ, or
   (b) y doesn't occur in t, and t can replace x in ψ without contradiction.

Now, by programming, Jack wants to judge whether x is replaceable by the term t without contradiction.

Input

At the beginning of the input section, there is an integer $T(1 \leq T \leq 10)$, indicating the number of test cases.

For every test case, the first line gives a formula A, which has length $L(1 \leq L \leq 100)$. In the input formula, $\neg$ will be replaced by char N, $\to$ will be replaced by char I, and $\forall$ will be replaced by char A.

The next line consists of an integer m, denoting the number of function and predicate.

Then there followed m lines, each line consists of a char and an integer, representing the parameter number of the function or predicate.

The next line consists of an integer $q(1 \leq q \leq 100)$, denoting q queries.

There follow q lines, each of which consists of a term t and a variable x, denoting the query of whether the term t can replace the variable x in the formula A without replacement. It is guaranteed that the length of string t is no more than 100.

Output

The output consists of q sections.

In each section, you need to print a character Y/N to answer whether the variable is replaceable without contradiction.

If the answer is yes, you should print the formula A' after replacement.

Example

standard input	standard output

```
1
AxAy(PzvfxygxyI(NQ))
5
f 3
g 2
P 3
Q 0
h 2
4
hxx x
hxy y
hzz z
hxz z
```
	```
	Y
	AxAy(PzvfxygxyI(NQ))
	Y
	AxAy(PzvfxygxyI(NQ))
	Y
	AxAy(PhzzvfxygxyI(NQ))
	N
	```

Note

The given formula is $\forall x \forall y(P(z, v, f(x, y, g(x, y))) \to (\neg Q))$.

For the first and the second query, x and y don't occur free in the formula, because they are constrained by $\forall$. Thus the answer is the original formula.

For the third query, z occurs free in the formula, because it isn't constrained by $\forall$. Thus the answer is to replace z by $h(z, z)$.

For the last query, z occurs free in the formula, and after the replacement, x in $h(x, z)$ will be constrained by $\forall x$, making the x in $h(x, z)$ no longer occur free. Thus the replacement is illegal.

Problem D. Broken Dream

Input file:	`standard input`
Output file:	`standard output`
Time limit:	8 seconds
Memory limit:	256 megabytes

People comfort and hug each other along the way, where they struggle desperately to chase their broken and dying dream, which has long been an old friend of theirs, since the good old days.

So does Akura. His dream is also broken now, and he wanted to make it reunited so that he can finally achieve it. From his perspective, his dream has already broken into several islands in the ocean, and there are bridges between some pair of islands, and every bridge has its own length. Akura's task is to find some bridges that connect all these islands while minimizing the total length of the chosen bridges. However, not every bridge is available, there are possibilities that some bridges are broken. So he wants you to help him find the expectation of the total length of bridges you have to choose so that all islands are connected. If in some situations there were no such selection to do so, you don't need to choose any bridges.

Formally speaking, there are n islands and m bridges. The $i-$th bridge connects the u_i-th and v_i-th island and has the length of w_i and the possibility of p_i that the bridge is **available**. Your task is to find the expectation of the total length of the bridges in the MST(minimum spanning tree) modulo 998244353. Note that if in some situations there is no MST in the graph, we consider the total length of bridges in MST to be 0.

Input

The input consists of multiple test cases. The first line contains a single integer $T(1 \leq T \leq 7)$ — the number of test cases.

The first line of each test case contains 2 integers $n, m(n \leq 15, m \leq 50)$.

The following are m lines.

The $i-$th line contains 4 integers u_i, v_i, w_i, p_i $(1 \leq u_i, v_i \leq n, 0 \leq w_i < 998244353, 2 \leq p_i < 998244353)$, where p_i is the possibility modulo 998244353.

Output

An integer, the expectation modulo 998244353.

Example

standard input	standard output

1
3 3
1 2 1 499122177
2 3 2 332748118
3 1 1 499122177
	1

Note

In the example, $p_1 = p_3 = \dfrac{1}{2}, p_2 = \dfrac{1}{3}$.

Problem E. List Reshape

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

ProtectEMmm is learning to use NumPy.

She is curious about the `reshape` function and wants to implement it by hand. For simplicity, hers `reshape` function reads a non-nested list (which means, all elements in the list are numbers, not a list), and the size of two dimensions of the output list, and then outputs a 2-dimension list with the data, the order and the number of elements unchanged, but with the specified shape.

A well-formatted Python list can be written as a list of comma-separated values (items) between square brackets `[]`. See the sample input and output for more information.

The size of each dimension describes the shape of the list. For example, a 2×3 list like `[[1, 2, 3], [4, 5, 6]]`, consists of two lists, each of which consists of three numbers.

ProtectEMmm finished implementing it quickly. Could you implement it as fast as she can?

Input

The first line contains an integer T $(1 \le T \le 50)$, indicating the number of test cases.

In each test case:

The first line contains a string s, indicating the list in Python.

The second line contains two numbers x, y, the size of each dimension of the output list. Your output list need to consist of x lists, each of which consists of y numbers.

It is guaranteed that the number of elements in s equals to $x \cdot y$, all elements in the list are in the range 0 to 1000, and the sum of the number of elements in all test cases does not exceed 5×10^5.

Output

For each test case, you should print a line of string, the result of your `reshape` operation.

Note that you need to print exactly one space between every comma and the next item in your output. **Do not print any more space at the end of line**.

Example

standard input	standard output
4 [3, 1, 4, 1, 5, 9, 2, 6] 2 4 [998, 244, 3, 5, 3] 5 1 [1, 1, 2, 3, 5, 8, 13, 21, 34] 1 9 [2, 3, 5, 7, 11, 13, 17, 19, 23] 3 3	[[3, 1, 4, 1], [5, 9, 2, 6]] [[998], [244], [3], [5], [3]] [[1, 1, 2, 3, 5, 8, 13, 21, 34]] [[2, 3, 5], [7, 11, 13], [17, 19, 23]]

Problem F. Peek

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

There is a large cave with words written on its inner wall, and Bob wants to see these words. Unfortunately, the interior of the cave has a large number of traps, thus making it difficult to enter; fortunately, the cave has been opened up with many windows at the corners, through which one can peek at the texts.

In the process of looking in through a particular window, the edge of the polygon might be parallel to the line of sight, and then it would be difficult to see the writing on the inner wall. At this moment, you can make the inner wall no longer parallel to the line of sight, by making small movements by the window.

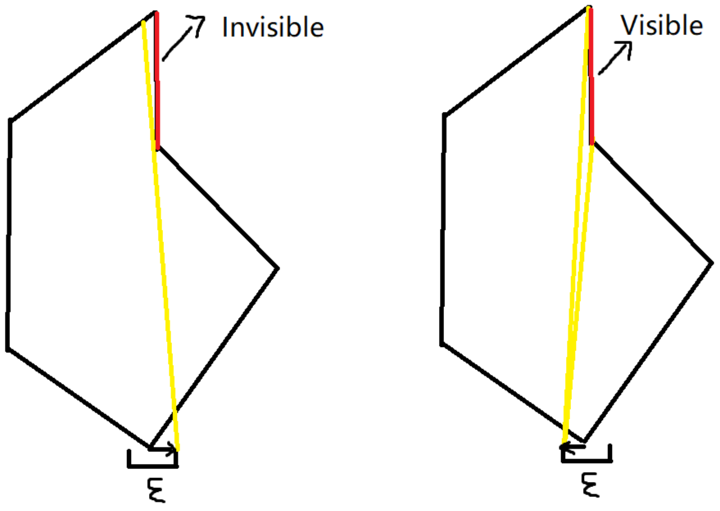
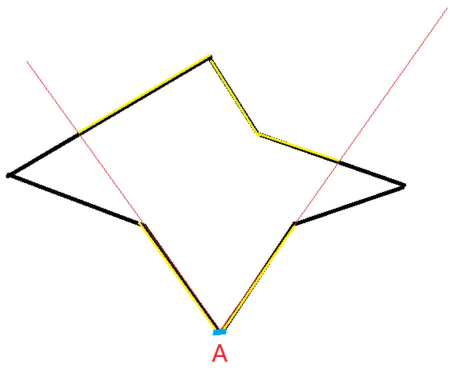


Fig. 1. Making slight movements by the window.

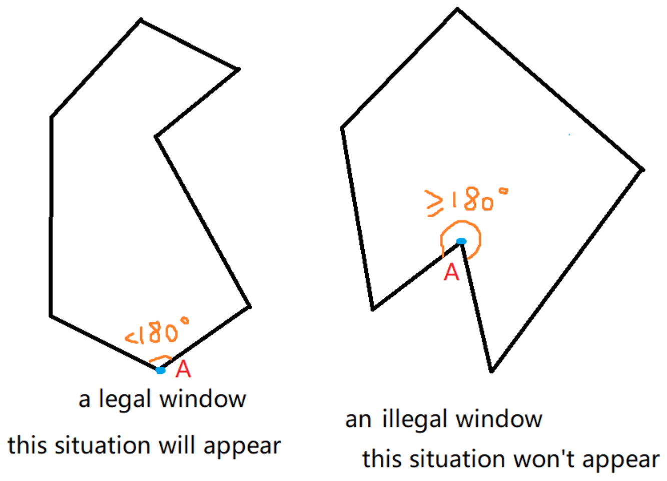
Formally, The large cave can be viewed as a polygon P which doesn't intersect with itself, with many of its vertices $\{P_i\}$ serving as windows. When observing from the outer area of a polygon into a vertice A serving as a window, the observer's line of sight will be restricted by two edges adjacent to A, which is shown in the following graph. Segments observable from A will contribute to the total length of the answer. In addition, the observer can make a tiny movement ε parallel to the window, thus making some of the edge observable.

It is guaranteed that the point which is chosen as a window will form an angle $\alpha \leq 180°$ with its two adjacent edges.

Now, Bob wants to know what is the total inner wall length that can be seen by peering through all the windows. Please note that ε can be seen as a very small number, and it multiplied with any constant k won't make contribute to the final answer.



(a) The highlighted lines are visible sections.



(b) There will be such a window in the left graph, but won't be such a window in the right graph.

Input

The first line consists of an integer $T(1 \le T \le 10)$, denoting the number of test cases.

In each test case, the first line gives two integers $n(1 \le n \le 100)$, $m(1 \le m \le n)$, indicating the number of vertices of the polygon and the number of windows.

The next section consists of n lines, each line giving two integers $x(-10^9 \le x \le 10^9)$, $y(-10^9 \le y \le 10^9)$, indicating the coordinates of the corresponding points.

The next m line gives m integers, indicating the subscripts of windows.

It is guaranteed that the two adjacent points will have adjacent subscripts. For example, if point A and point B are adjacent in the polygon, and A is P_2, then B will be P_1 or P_3.

Output

Outputs one float number with 3 decimal places, representing the total length that can be seen.

Example

standard input

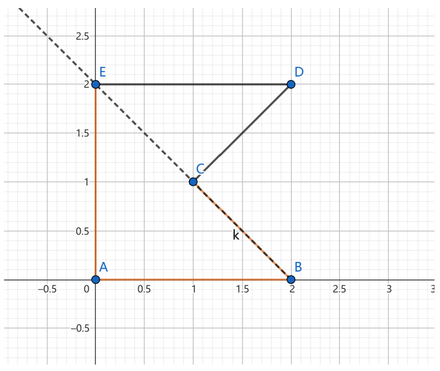
```
2
5 1
0 0
2 0
1 1
2 2
0 2
2
12 2
3 0
5 0
5 3
3 3
3 5
6 5
6 6
0 6
0 2
3 2
3 1
0 1
1
12
```

standard output

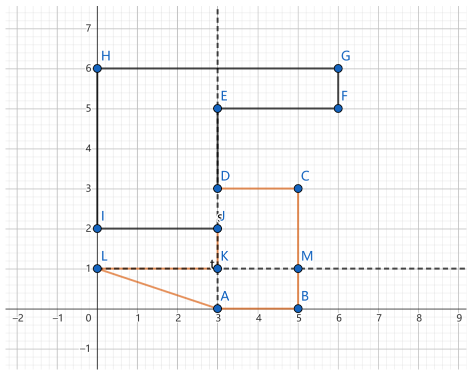
```
5.414
14.162
```

Note

The examples are illustrated in the following graphs.



(a) Example 1



(b) Example 2

Example 1: The final answer is $2 + 2 + \sqrt{2} = 4 + \sqrt{2}$.

Example 2: The final answer is $2 + 3 + 2 + 1 + 3 + \sqrt{10} = 11 + \sqrt{10}$.

Problem G. Average

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

You are given a tree with n nodes. Every node has its value a_u. The value of a $path(u, v)$ on the tree is equal to

$$\frac{\sum_{w \in path(u,v)} a_w}{len(u, v)}$$

where $path(u, v)$ is the shortest path from u to v, $len(u, v)$ means the number of nodes in $path(u, v)$.

You have to answer questions like what is the maximum possible value of a path with length greater than or equal to k. Unfortunately, the value of nodes will increase. So you need to answer the question after each change.

Input

Each test contains multiple test cases. The first line contains an integer $T(1 \le T \le 10^5)$, indicating the number of test cases. The description of test cases follows.

The first line contains two integers n,k $(1 \le n \le 10^5, 1 \le k \le 30)$.

The second line contains n integers $a_i(1 \le a_i \le 10^9)$.

The following $n - 1$ lines contains 2 integers u_i and v_i $(1 \le u_i, v_i \le n, u_i \ne v_i)$, indicating an edge between vertices u_i and v_i. It is guaranteed that the given edges form a tree.

The next line contains a integer q $(1 \le q \le 10^4)$.

The following $n - 1$ lines contains 2 integers u_i, x $(1 \le u_i \le n, 1 \le x \le 10^9)$, indicating that the value of node u_i will increase x.

It is guaranteed that the sum of n over all test cases does not exceed 2.5×10^5, and the sum of q over all test cases does not exceed 2.5×10^4.

Output

Output the answer A/B, indicating the value is $\frac{A}{B}$ and $\gcd(A, B) = 1$. If no such path, output ”0/1”.

Example

standard input	standard output

2
6 2
6 16 6 15 5 4
1 2
2 3
2 4
2 5
4 6
2
1 4
3 5
14 1
12 19 9 17 9 10 11 18 9 19 10 14 18 8
1 2
2 3
1 4
1 5
4 6
2 7
7 8
8 9
6 10
6 11
1 12
11 13
3 14
5
10 3
2 8
7 9
5 6
2 1000000000
	31/2
	31/2
	22/1
	27/1
	27/1
	27/1
	1000000027/1

Problem H. Coins

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3.5 seconds
Memory limit:	256 megabytes

There are n people playing a game, where $i-$th person has a_i coins. In each round, they randomly choose two players. Then the first one should give one coin to the second. If someone leaves no coin after that, he leaves the game and the rest players continue the game until a player owns all the coins. F wants to know the expected number of rounds.

Input

The first line contains integer t $(1 \le t \le 10^2)$ — the number of test cases.

For each of the next t test cases, the format is:

The first line contains an integer n $(2 \le n \le 10^5)$, representing the number of people.

The next line contains n integers a_i $(1 \le a_i \le 10^6)$, representing the number of coins that each person has.

Output

For each case, output the expected number of rounds.

Example

standard input	standard output

1
3
1 1 1
	3

Problem I. Sequence

Input file:	`standard input`
Output file:	`standard output`
Time limit:	12.5 seconds
Memory limit:	512 megabytes

A **good** sequence $a_0, a_1, ..., a_{tn-1}$ satisfies following criteria:

- $a_0 = 0, a_{tn-1} = 2tn - 1, 1 \le a_{k+1} - a_k \le d \ (0 \le k < tn - 1)$.
- $\forall\ i, j,\ a_j - a_i \ne kn$ (k is any odd number, $0 \le i < j < tn$)

F wants to know the number of **good** sequence.

Input

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

The first line of each test case contains three integers t, n, d $(1 \le d \le 5{\cdot}10^4, 1 \le t \le 10^5, 1 \le n \le 10^{15})$.

It is guaranteed that the sum of d does not exceed $2 \cdot 10^5$.

Output

For each case, output the number of **good** sequence modulo 998244353.

Example

standard input	standard output

2
1 5 2
2 5 3
	0
	2

Problem J. MST problem

Input file:	`standard input`
Output file:	`standard output`
Time limit:	18 seconds
Memory limit:	256 megabytes

For a tree, we define its values as $\max\{i \times c_i\}$, where c_i means the number of edges where weights i. For example, if a tree with edges weight $3, 3, 3, 6$, its value equals $\max(3 \times 3, 6 \times 1) = 9$.

You're given a connected graph with n vertices and m edges. Try to find a spanning tree with minimal value. Output its value.

Input

The first line contains one integer $T (1 \le T \le 500)$, representing the number of test cases.

For each test case, the first line contains two integers n, m $(1 \le n, m \le 500)$.

The following m lines, each line contains three integer u, v, w $(1 \le u, v \le n, 1 \le w \le 10^9)$, represent an edge connecting vertex u and v, weights w.

It is guaranteed that the graph is connected and contains no self-loops (but may contain multi-edges).

The number of test cases where $n \ge 50$ or $m \ge 50$ will not exceed 50.

Output

For each test case, output one integer, representing the minimal value of a spanning tree.

Example

standard input	standard output

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

Problem K. Cargo

Input file:	`standard input`
Output file:	`standard output`
Time limit:	18 seconds
Memory limit:	256 megabytes

There're n stores selling m kinds of items. Each store only sells one kind of item, the i-th store sells the a_i-th kind of item.

You're going to buy some items. You do the following operation k times: choose a store at random, and buy one item from that store.

Let there are c_i stores selling item i. After all the operations, you will get unsatisfied, if and only if the following condition is satisfied:

- There exists an item i, where you hold exact c_i of them, and they are all from different stores.

For example, if store 1 and 3 are selling item 1, and after operations, you hold exactly two items 1, and one of them is from store 1, where the other one is from store 3, you will get unsatisfied.

You want to know the possibility of **not** getting unsatisfied after k operations. Output it modulo 998244353.

Input

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

For each test case:

The first line contains three integers: $n, m, k (1 \le m \le n \le 2 \times 10^5, 1 \le k < 998244353)$.

The following line contains n integers $a_1, a_2 \ldots a_n (1 \le a_i \le m)$. It is guaranteed that each of the m kinds of items will appear at least once.

There will be no more than 10 test cases where $n \ge 10^4$.

Output

One single integer, represents the answer.

Example

standard input	standard output

4
3 2 3
1 1 2
1 1 1
1
6 3 4
1 1 1 2 3 3
8 4 19908
1 1 2 2 3 3 3 4
	221832079
	0
	465231165
	665765722

Problem L. Inference

Input file:	`standard input`
Output file:	`standard output`
Time limit:	8 second
Memory limit:	256 megabytes

Given m features, you want to infer the last feature's value of Alice by the value of her first $m-1$ features, based on massive data.

The relationship between features can be represented as a directed acyclic graph, where a node A pointing to a node B indicates that B is a random variable dependent on A.

The following formula can calculate the probability of the last feature's value of Alice based on other data, in which x_i represents the $i-$th feature, $\pi(x_i)$ represents nodes that point to x_i, and cnt represents their number of occurrences:

$$P(x_m|x_1,\ldots,x_{m-1}) = cP(x_1,\ldots,x_m) = \prod_{i=1}^{m} P(x_i|\pi(x_i))$$

$$P(x_i|\pi(x_i)) = \frac{P(\pi(x_i),x_i)}{P(\pi(x_i))} = \begin{cases} \frac{cnt(\pi(x_i),x_i)}{cnt(\pi(x_i))}, \text{ if } cnt(\pi(x_i)) \neq 0 \\ 0, \text{ if } cnt(\pi(x_i)) = 0 \end{cases}$$

It is guaranteed that the last feature has zero out-degree, and there exists at least one case of value such that $cnt(\pi(x_i)) \neq 0$.

Now, given n people's data, what is the most likely value of the $m-$th feature of Alice, given the value of her first $m-1$ features.

Input

At the beginning of the input section, there is an integer $T(1 \leq T \leq 10)$, indicating the number of test cases.

For every test case, the first line consists of three integers $n(1 \leq n \leq 10^4)$, $m(1 \leq m \leq 100)$, $k(1 \leq k \leq 300)$, indicating the number of data, the number of features, and the number of the relationship between the features.

In the next k rows, each row contains two integers $x, y(1 \leq x, y \leq m)$, indicating node x point to node y, thus $y-$th feature is dependent on $x-$th feature. Every point has an in-degree less than 5.

In the next n rows, each row contains m integers indicating one person's values of each feature. Every feature's value is among $0, 1, 2$.

In the last line, there are $m-1$ integers indicating Alice's values of the first $m-1$ features.

Output

An integer denoting the most probable value of Alice's $m-$th feature.

It is guaranteed that there exists only 1 value which has the biggest probability.

Example

standard input	standard output

1
10 5 4
1 4
2 4
3 5
4 5
0 1 0 1 0
0 1 1 1 1
0 0 0 0 0
0 0 1 0 0
1 1 1 0 1
1 0 0 1 1
1 1 0 1 1
1 0 1 0 0
1 1 1 1 1
0 1 0 1 1
1 1 0 0
	0

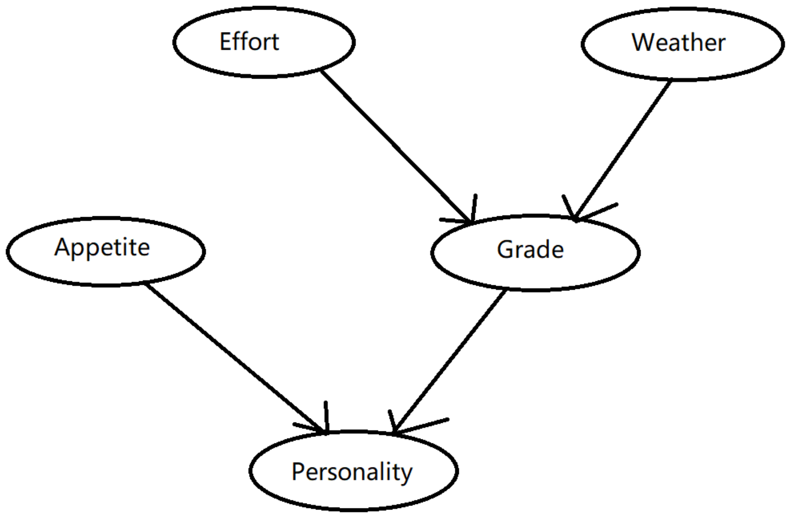
Note

In the example, there are 10 people's data, 5 features, and 4 edges between the features. The table of data is shown in the following table.

Effort	Weather	Appetite	Grade	Personality
0	1	0	1	0
0	1	1	1	1
0	0	0	0	0
0	0	1	0	0
1	1	1	0	1
1	0	0	1	1
1	1	0	1	1
1	0	1	0	0
1	1	1	1	1
0	1	0	1	1
1	1	0	0	**?**

The relation between features is shown in the following graph.

To calculate the possibility that the 5-th feature has a value 0 in the last line of data, we use the formula:



$$P(x_m|x_1,\ldots,x_{m-1}) = cP(x_1,\ldots,x_m) = \prod_{i=1}^{m} P(x_i|\pi(x_i))$$

$$= P(x_1 = 1)\cdot P(x_2 = 1)\cdot P(x_3 = 0)\cdot P(x_4 = 0|x_1 = 1, x_2 = 1)\cdot P(x_5 = 0|x_3 = 0, x_4 = 0)$$

$$= \frac{5}{10}\cdot\frac{6}{10}\cdot\frac{5}{10}\cdot\frac{1}{3}\cdot\frac{1}{1} = 0.05$$

The possibility that the $5-$th feature has a value 1 or 2 is 0, which is less than 0.05. So we take 0 as the value of 5-th feature.