Problem A. Two Spanning Trees

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

Alice and Bob are participating in a competition. The competition consists of several independent rounds. The rounds go as follows. Each player is given a simple (without loops and parallel edges) connected undirected graph with n vertices and m edges. The values n and m are the same for both players, but the graphs might be different. The first player to find a spanning tree of their graph is declared to be the winner. In case both players have found spanning trees relatively fast, the round is declared to be tied. For simplicity, the edges are numbered from 1 to m, and the players are required to present just the indices of the edges in the spanning tree they have found.

Alice has recently learned many algorithms for finding a spanning tree, and she is sure the game will be easy. On his side, Bob doesn't know any algorithms for finding a spanning tree, but he has his own strategy. He hopes that the organizers are too lazy to prepare two different graphs, so he will just copy Alice's answer and present it to the jury. In the past, it worked decently well. However, Alice has recently begun to suspect Bob of cheating, so she is now trying to expose Bob's dirty strategy.

There still remain t rounds till the end of the competition, and Alice tries to win as much as possible. To achieve that, each round she will search for such a spanning tree in her graph that the corresponding edges in Bob's graph do not form a spanning tree. It turns out that this is a bit difficult, so now Alice wonders if there even exists a suitable spanning tree.

Input

The first line contains an integer t ($1 \le t \le 2500$), the number of remaining rounds.

Each round description starts with a line containing two integers n and m ($2 \le n \le 5000$, $n-1 \le m \le \min\left(5000, \frac{n(n-1)}{2}\right)$): the number of vertices and edges in the graphs. Then follow m lines denoting the edges. The i-th of these lines contains four integers: u_A, v_A, u_B, and v_B ($1 \le u_A, u_B, v_A, v_B \le n$, $u_A \ne v_A$, $u_B \ne v_B$). They mean that the i-th edge in Alice's graph connects vertices u_A and v_A, while the i-th edge in Bob's graph connects vertices u_B and v_B. Both graphs are simple and connected.

Both the sum of all n and the sum of all m across all rounds do not exceed 5000.

Output

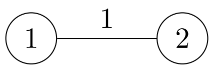
For each round, determine if there exists a spanning tree in Alice's graph such that the corresponding edges in Bob's graph do not form a spanning tree.

If that's the case, print a single line containing "`YES`", otherwise, print "`NO`".

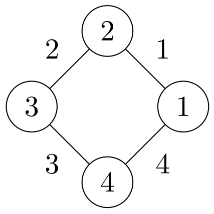
Example

standard input	*standard output*
`3`	`NO`
`2 1`	`YES`
`1 2 2 1`	`NO`
`4 4`	
`1 2 1 2`	
`2 3 2 3`	
`3 4 3 4`	
`1 4 1 3`	
`4 4`	
`1 2 1 2`	
`2 3 2 3`	
`3 4 3 4`	
`1 3 1 4`	

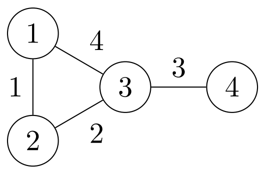
Note



Graph A



Graph B



Graph C

In the example, in the first round, both players are given graph A.

In the second round, Alice is given graph B and Bob is given graph C. In that case, Alice can present $[1, 2, 4]$ as an answer: these edges form a spanning tree in graph B, but not in graph C.

In the third round, Alice is given graph C and Bob is given graph B.

Problem B. Christmas Tree

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

There is a Christmas Tree in your room. A Christmas Tree is a connected undirected graph with n vertices and $n - 1$ edges. Each vertex of the tree has a light bulb in it. The *luminance* (or *brightness*) of the bulb in vertex v is a_v units.

The tree is lovely, and it shines really strong. However, you feel rather tired and want to take a nap. It would be nice to dim the lights a bit, but there is no such option. Instead, you can direct the edges to make the tree less illuminated. For each edge (u, v), choose exactly one direction, $u \to v$ or $u \leftarrow v$, in which the light can pass. For a vertex v, define its *contrast* as the sum of a_u over all vertices u such that there exists a directed path from u to v.

What is the minimum possible sum of contrasts of all the vertices over all possible ways to direct the tree edges?

Input

The input contains several test cases. The first line contains a single integer t ($1 \le t \le 250$), the number of test cases. The description of the test cases follows.

The first line of each test case contains an integer n ($2 \le n \le 500$), the number of vertices in the tree.

The second line contains n integers $a_1, \ldots, a_n$ ($0 \le a_i \le 10^{12}$): the luminances of the light bulbs in the vertices.

Each of the next $n - 1$ lines contains two integers u_i and v_i ($1 \le u_i \ne v_i \le n$), representing an undirected edge between nodes u_i and v_i. The edges form a tree.

The sum of all n across all test cases does not exceed 500.

Output

For each test case, output a line with an integer: the minimum possible sum of contrasts.

Example

standard input	*standard output*

2
2
10 20
1 2
4
10 30 60 100
1 2
2 3
2 4
	40
	290

Note

In the first test case, you should direct the edge as $1 \to 2$. Then the contrast of vertex 1 is 10 (it is lit only by itself, and $a_1 = 10$), and the contrast of vertex 2 is 30 (it is lit by vertices 1 and 2, and $a_1 + a_2 = 30$).

In the second test case, you can direct the edges as $1 \to 2$, $2 \to 3$, and $2 \to 4$.

Problem C. Roman Numerals

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

Consider the following generalization of Roman numerals. There are n digits in total, and each digit has its own *priority* and *value*. In this problem, we will denote the numerals by strings of lowercase and uppercase English letters. Let us denote the priority, value, and string representation of the i-th Roman numeral by p_i, v_i, and d_i, respectively.

A *number* in Roman numerals is an array consisting of Roman numerals. The *value* of a number $d_1 \ldots d_m$ is calculated recursively. First, choose a position i such that the digit d_i has the highest priority. In case there are multiple digits with the highest priority, choose one of them with the lowest position i. Then

$$\text{value}(d_1 \ldots d_m) = v_i - \text{value}(d_1 \ldots d_{i-1}) + \text{value}(d_{i+1} \ldots d_m),$$

where v_i is the value of d_i. The value of an empty number is conventionally set to zero.

For example, consider the following triples (p_i, v_i, d_i): $(1, 1, \mathtt{I})$, $(2, 5, \mathtt{V})$, and $(3, 10, \mathtt{X})$. They assign the usual values to the usual Roman numerals: for example, $\text{value}(\mathtt{II}) = 2$, $\text{value}(\mathtt{IX}) = 9$, and $\text{value}(\mathtt{XIV}) = 14$. On the other hand, the representation now is not unique: $\text{value}(\mathtt{IXI}) = \text{value}(\mathtt{IIIIIXV}) = \text{value}(\mathtt{X}) = 10$.

You are given an array $s_1 \ldots s_n$ of Roman numerals and q independent queries of the form (ℓ, r). For each query, calculate the value of the subarray $s_\ell \ldots s_r$.

Input

The first line contains three integers m, n, and q ($1 \le m \le 10^5$, $1 \le n, q \le 3 \cdot 10^5$): the number of Roman numerals, the length of the given array, and the number of queries.

Then follow m lines. The i-th of them contains two integers, p_i and v_i ($1 \le p_i \le m$, $1 \le v_i \le 10^9$), followed by a string d_i of lowercase and uppercase English letters ($1 \le |d_i| \le 7$). These are the priority, value, and string representation of the i-th Roman numeral, respectively. All d_i are distinct.

The next line contains n space-separated representations of Roman numerals $s_1 \ldots s_n$.

Then follow q lines, each of them containing two integers ℓ and r ($1 \le \ell \le r \le n$).

Output

For each query, output a line with a single integer: the value of the number $s_\ell \ldots s_r$.

Examples

standard input	*standard output*
3 5 3 1 1 I 2 5 V 3 10 X I X I V X 1 2 1 3 3 5	9 10 6
3 6 1 1 6 six 1 16 sixteen 1 60 sixty six sixty six sixteen six sixteen 1 6	110

Problem D. Disjoint Set Splitting

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2.5 seconds
Memory limit:	1024 mebibytes

You are given an undirected graph consisting of n vertices and m edges. The vertices are numbered from zero: $0, 1, \ldots, n-1$. You are required to process q queries. Each query provides a pair (u, v) and asks you to erase the edge (u, v) from the graph. If there is no such edge in the graph (for example, if $u = v$), do nothing. After each query, print whether the graph is connected or not.

Input

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

The first line of each test case contains three integers n, m, and q ($2 \le n \le 10^6$, $1 \le m, q \le 10^6$): the number of vertices, edges, and queries, respectively.

Each of the next m lines contains two integers u_i and v_i ($0 \le u_i, v_i < n$), representing an edge between u_i and v_i. There are no loops and no parallel edges in the graph.

Each of the next q lines describes a query. However, the query parameters u_j and v_j are not provided directly. Instead, the j-th of these lines contains two integers a_j and b_j ($0 \le a_j, b_j < n$). Suppose that the answer to the k-th query is $c_k = 1$ if the graph is connected or $c_k = 0$ otherwise. Then u_j and v_j are generated according to the formulae:

$$u_j = \left(2^{j-2}c_1 + 2^{j-3}c_2 + \ldots + 2^1 c_{j-2} + 2^0 c_{j-1} + a_j\right) \bmod n,$$
$$v_j = \left(3^{j-2}c_1 + 3^{j-3}c_2 + \ldots + 3^1 c_{j-2} + 3^0 c_{j-1} + b_j\right) \bmod n.$$

The sum of all n, the sum of all m, and the sum of all q across all test cases do not exceed 10^6.

Output

For each test case, print q lines with integers $c_1, \ldots, c_q$: the answers to the queries.

Example

standard input	*standard output*

2
3 1 2
0 1
0 0
0 1
4 4 4
0 1
1 2
2 3
0 3
0 1
1 3
0 0
2 3
	0
	0
	1
	1
	0
	0

Note

In the first test case, the queries ask to delete the edges $(0, 0)$ (it does not exist) and $(0, 1)$.

In the second test case, the queries ask to delete the edges $(0, 1)$, $(2, 0)$ (it does not exist), $(3, 0)$, and $(0, 3)$ (it is already deleted by the previous operation).

Problem E. Maximum Segment Sum

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1.5 seconds
Memory limit:	1024 mebibytes

Calculate the number of sequences of -1 and 1 of length n with maximum subsegment sum k. Solve this task for all k from 0 to n, and print the answers modulo $998\,244\,353$. The empty subsegment is considered too.

Input

The input contains a single integer n ($1 \le n \le 5 \cdot 10^5$).

Output

Print $n + 1$ integers modulo $998\,244\,353$: the answers for $k = 0, 1, \ldots, n$.

Examples

standard input	*standard output*
`1`	`1 1`
`2`	`1 2 1`
`3`	`1 4 2 1`
`7`	`1 33 41 28 14 8 2 1`

Note

In the third example where $n = 3$, the maximum subsegment sum:

- of $(-1, -1, -1)$ is 0;
- of $(1, -1, -1)$, $(-1, 1, -1)$, $(-1, -1, 1)$, and $(1, -1, 1)$ is 1;
- of $(1, 1, -1)$ and $(-1, 1, 1)$ is 2;
- of $(1, 1, 1)$ is 3.

Problem F. This Time I Will Be Lucky

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	1024 mebibytes

In a casino, there is a bag with a white balls and b black balls. Until the bag is empty, we can pick one ball randomly from the bag, and if it is white, we gain 1 dollar; otherwise, we lose 1 dollar. After that, we trash this ball. We can stop at any time, including the time before picking any balls. What maximum expected profit can we get?

Input

The input contains two space-separated integers, a and b ($0 \le a, b \le 100\,000$).

Output

Print one real number: the maximum expected profit. The answer will be considered correct if the absolute or relative error is at most 10^{-6}.

Examples

standard input	*standard output*
0 0	0
3 4	0.342857142857142857
2 2	0.666666666666666666
3 0	3

Note

Don't forget `setprecision` if you use `C++`.

Problem G. Far Away

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

There is an undirected graph with n vertices and m edges. The graph does not contain self-loops or parallel edges and is not necessarily connected. The vertices are numbered from 1 to n.

There are q queries. Each query is a pair (x, y). You need to determine whether the distance between x and y is strictly more than 20 000 or not.

The distance between two vertices is the number of edges on a shortest path between them. If there is no shortest path, the distance is considered to be $+\infty > 20\,000$.

Input

The first line contains three integers n, m, and q ($1 \le n \le 10^5$, $0 \le m \le 3 \cdot 10^5$, $1 \le q \le 3 \cdot 10^5$).

Then follow m lines, each containing two integers u_i and v_i, representing the edges ($1 \le u_i < v_i \le n$, no two edges coincide).

After that follow q lines, each containing two integers x_i and y_i ($1 \le x_i, y_i \le n$, x_i and y_i may coincide).

Output

Print q lines, each with either "`YES`" or "`NO`": the answers to the queries.

Example

standard input	*standard output*

10 5 10
1 2
1 7
3 10
4 10
5 6
3 3
5 6
5 2
10 6
9 4
6 6
1 7
10 1
10 5
8 7
	NO
	NO
	YES
	YES
	YES
	NO
	NO
	YES
	YES
	YES

Note

In the example, there are just 10 vertices in the graph, so we are checking whether the vertices are in different connected components or not.

Problem H. Absolutely Flat

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	1024 mebibytes

You are given an array a consisting of n positive integers. However, some elements of a are missing, and they are replaced by zeroes.

Define the *oscillation* of a on a segment $[\ell, r]$ to be $\max(a_\ell, \ldots, a_r) - \min(a_\ell, \ldots, a_r)$.

You are given q segments in the form $[\ell_i, r_i]$. Replace each zero in a with a positive integer to minimize the sum of oscillations of a on all q segments.

Input

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

The second line contains n integers $a_1, \ldots, a_n$ ($0 \le a_i \le 10^9$): the elements of the array. Here, $a_i = 0$ means that the i-th element is missing.

The next q lines describe the segments. Each of them contains two integers ℓ_i and r_i ($1 \le \ell_i \le r_i \le n$).

Output

Print a single integer: the minimum sum of oscillations of a on all the given segments.

Examples

standard input	*standard output*

5 2
2 1 0 3 2
1 3
3 5
	2

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

7 4
4 4 0 1 0 3 4
1 3
1 3
3 5
4 7
	6

Note

In the first example, the only optimal array is $a = [2, 1, 2, 3, 2]$.

In the second example, there are several optimal arrays. For example, $a = [1, 2, 3, 2, 1]$ is optimal.

In the third example, there are several optimal arrays. For example, $a = [4, 4, 4, 1, 2, 3, 4]$ is optimal.

Problem I. Two Permutations

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

You are given two permutations p and q of length n. You can do the following operation with the first permutation zero or more times:

Choose $1 \le i < j \le n$ such that $p_i < p_j$ and swap p_i with p_j.

You need to determine if it is possible to make p equal to q. If it is possible, you need to find any suitable sequence of operations.

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 first line of each test case contains an integer n ($1 \le n \le 2000$).

The second line contains n different integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$).

The third line contains n different integers $q_1, q_2, \ldots, q_n$ ($1 \le q_i \le n$).

The given p and q are both permutations of $\{1, 2, \ldots, n\}$.

The sum of n over all test cases does not exceed 2000.

Output

For each test case, determine if there exists a way to make p equal to q.

If that's the case, print "`YES`" on the first line. On the next line, print an integer k, the number of operations you wish to perform. On the next lines, you should print the operations.

To describe an operation that swaps p_i with p_j ($1 \le i < j \le n$), print a line formatted as "i j".

Otherwise, print "`NO`" on a single line.

In case there are several suitable answers, print any one of them.

Example

standard input	*standard output*

6
2
1 2
2 1
3
1 2 3
3 2 1
5
3 5 1 2 4
4 5 1 3 2
6
2 3 6 4 1 5
3 4 5 6 2 1
6
2 3 6 4 1 5
5 3 6 4 2 1
1
1
1
	YES
	1
	1 2
	YES
	3
	1 2
	1 3
	2 3
	YES
	2
	1 5
	4 5
	NO
	YES
	6
	1 2
	1 4
	1 6
	2 4
	4 6
	5 6
	YES
	0

Note

In the second test case, p is 1 2 3. One of the possible sequences of swaps is: “1 2”, “1 3”, “2 3”.

After the first swap, p becomes 2 1 3; after the second swap, it becomes 3 1 2; and after the third, 3 2 1.

Problem J. One Permutation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	1024 mebibytes

You are given one permutation p of length n. For a continuous subsegment of p, define its cost as the length of the longest increasing subsequence of this subsegment. For a partition of p into disjoint subsegments, define its cost as the sum of the costs of these subsegments.

Define a_k as the maximum cost among partitions of p into k non-empty disjoint subsegments.

You need to find a_k for each k from 1 to n.

Input

The input contains several test cases. The first line contains a single integer t ($1 \le t \le 10^3$), the number of test cases. The description of the test cases follows.

The first line of each test case contains an integer n ($1 \le n \le 10^5$).

The second line contains n different integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$). Together, they define a permutation of $1, 2, \ldots, n$.

The sum of n over all test cases does not exceed 10^5.

Output

For each test case, print $a_1, a_2, \ldots, a_n$ in one line.

Example

standard input	*standard output*

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

Note

Consider the third test case with the permutation $p = [2, 4, 1, 3, 6, 5, 8, 7]$:

- For $k = 1$, the answer is just the length of the longest increasing subsequence in p, which is $a_1 = 4$.
- For $k = 2$, the partition into $[2, 4]$ and $[1, 3, 6, 5, 8, 7]$ has the maximum cost: $a_2 = 2 + 4 = 6$.
- For $k = 3$, the partition into $[2, 4]$, $[1, 3, 6]$, and $[5, 8, 7]$ has the maximum cost: $a_3 = 2 + 3 + 2 = 7$.
- For $k = 4$, the partition into $[2, 4]$, $[1, 3, 6]$, $[5, 8]$, and $[7]$ has the maximum cost: $a_4 = 2+3+2+1 = 8$.
- For $k > 4$, it is easy to see that $a_k = 8$.

Problem K. Game on Board

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

On the board, there are n different positive integers $a_1, \ldots, a_n$. Three players are making moves in the order 123123123.... In a move, the player chooses two different numbers x and y on the board such that $|x - y|$ is not on the board, and writes $|x - y|$ onto the board (x and y remain on the board as well). The player who can't make a move loses. Can players 2 and 3 cooperate so that player 1 loses?

Input

The first line contains an integer n ($1 \le n \le 10^5$), the number of integers on the board.

The second line contains n distinct integers $a_1, \ldots, a_n$ ($1 \le a_i \le 10^9$): the numbers that are on the board initially.

Output

Print "YES" if players 2 and 3 can cooperate and force player 1 to lose, or "NO" otherwise.

Examples

standard input	*standard output*
3 1 2 3	YES
3 1 4 3	NO