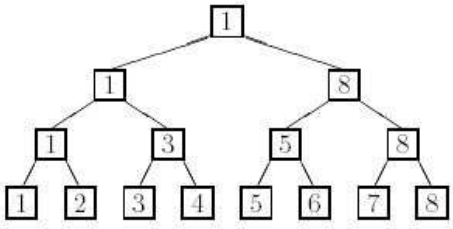
IUPC

A. Tournament Simulation

2 seconds, 256 megabytes

T knockout tournaments are planned. Your task is to simulate them and find the tournament winning probability for each player.

In a single knockout tournament there are 2^M players. One loss and a player is out of the tournament. Winners then play each other with the new winners advancing until there is only one winner left. If we number the players $1, 2, 3, \ldots, 2^M$, with the first round pairings $2k-1$ vs $2k$, for $k = 1, 2, \ldots, 2^{M-1}$, then we could give the results of the tournament in a complete binary tree. The winners are indicated in the internal nodes of the tree. Below is an example of a tournament with $M = 3$.



It is given that in a match between two players i and j, the winning probability of i is given by $a_i / (a_i + a_j)$, similarly, the winning probability of j is given by $a_j / (a_i + a_j)$, where a_i and a_j are their respective abilities and, after the match the winner's ability gets updated as follows: $a_{winner} := a_{winner} + a_{loser}$.

Input

The first line contains a single integer T $(1 \le T \le 1000)$, the number of tournaments. The description of T tournaments follow.

The first line of each description contains a single positive integer $M (1 \le M \le 17)$.

The second line of each description contains 2^M space separated integers $a_1, a_2, \ldots, a_{2^M}$ $(1 \le a_i \le 1000)$, the abilities of the 2^M players participating in the tournament.

It is guaranteed that the total number of players over all the tournaments does not exceed $2 * 10^5$, i.e., $\sum 2^M \le 2 * 10^5$.

Output

For each tournament, print one line containing 2^M integers, w_i, such that $w_i = p * q^{-1}$, computed modulo $10^9 + 7$, where p and q are co-prime positive integer coefficients representing the winning probability p/q for the i^{th} player.

input
1 2 1 2 3 4
output
700000005 400000003 100000001 800000006

In the given example, the first and only tournament has $2^2 = 4$ players.

In one possible scenario, 1 wins against 2 and 3 wins against 4. Then the only surviving players being 1 and 3, they are paired against each other and compete with their updated abilities.

B. Magical Numbers

1.5 seconds, 256 megabytes

Since the input-output is large, prefer using fast input-output methods.

Numbers have magic, that's common knowledge. To his surprise, young Sherlock has just found out that the pairing between composite numbers and prime numbers yields profound magic called conjunction!

Formally, given a prime number p, and a composite number q, their conjunction value c_v is given by: $c_v = p * q$

Given a permutation P of integers between 1 and N, his curiosity brings him to ask you certain queries. The detective is busy with an investigation so he asks for your aid. Help him and impress Ms. Sherlocked (͡° ͜ʖ ͡°)

Each query describes a contiguous subsection of the permutation. You are tasked with finding the greatest possible value of c_v achievable using the numbers present in the specified subsection.

Input

The first line of input contains a single integer $N (2 \le N \le 40,000)$, the size of the permutation. The second line contains N space-separated integers, $P_0, P_1, P_2, \ldots, P_N$, a valid permutation of integers between 1 and N.

The third line contains a single integer, $Q (1 \le Q \le 2,500,000)$, the number of queries. The next Q lines contain two integers each, $L (1 \le L \le N)$ and $R (L \le R \le N)$, describing the subsection for the query.

Output

For every query, output a single line containing one integer each, the greatest value of c_v achievable using only numbers in the specified subsection of the permutation.

input
5 5 2 4 1 3 3 1 3 2 4 4 5
output
20 8 0

For any query, it will always be possible to attain $c_v = 0$, as, even if we do not choose any suitable pair, we would still have nothing.

In the given example:

In the first query, the subsection considered is [5, 2, 4]. Possible values of c_v are 0, 8 and 20.

In the second query, the subsection considered is [2, 4, 1]. Possible values of c_v are 0 and 8.

In the third query, the subsection considered is [1, 3]. Only possible value of c_v is 0.

C. Trek Paths

5 seconds, 256 megabytes

You and your friends are going trekking.

The trek can be seen as N safe areas at increasing altitudes, connected by M paths of equal length.

Each path has a limit on the number of people it can accommodate at once. Travelling the i^{th} path has a Risk of Injury of the form $RoI(n) = x_i + y_i * (n-1)$, where n is the number of people traveling the path.

Ironically, the "safe" areas are not free of Risk either, visiting the i^{th} safe area has a Risk of Injury of the form $RoI(n) = (i-1) * n$.

All of you start travelling from the 1^{st} safe area. You can split into any number of groups to continue traveling. However you always travel simultaneously, if a group of people has to be left behind they do not continue traveling towards the N^{th} safe area.

Find the maximum number of people that can reach the N^{th} safe area and the minimum possible sum of RoI over all paths and safe areas traveled on the way.

`Note: While calculating the RoI of any path or safe area we only take into account the people who managed to reach the N^th safe area`

Input

The first line of input contains N and M ($2 \leq N \leq 5 * 10^4$, $1 \leq M \leq min(\frac{N(N-1)}{2}, 5 * 10^4)$).

The next M lines of input contain 5 integers: u_i v_i c_i x_i y_i ($1 \leq u_i, v_i \leq N, 1 \leq c_i \leq N, 1 \leq x_i \leq y_i \leq 10^6$) meaning the i^{th} path is a one directional path from safe area u_i to safe area v_i, the maximum number of people it can accommodate is c_i and it's risk of injury is described by x_i and y_i.

Output

Print the maximum number of people that can reach the N^{th} safe area and the minimum sum of RoI over all paths and safe areas traveled on the way.

input
9 11 1 2 3 2 5 2 3 5 6 6 2 4 2 1 1 3 4 9 7 7 8 9 3 2 3 4 7 1 5 10 3 6 4 1 2 6 8 8 9 20 7 9 3 2 9 5 9 9 1 100 5 1 9 1000 2000
output
3 133

D. Two Squirrels

1 second, 256 megabytes

You lead a peaceful life tending to your tree with N vertices.

But sometimes an annoying friend visits your garden, here she comes once more. This time she proposes this problem:

Lets say there are two squirrels on the tree, Chintu and Minku, Chintu likes large numbers and Minku likes small numbers. They are going to follow their passion to the very end, literally.

They are initially at the same vertex and they do not visit any vertex more than once. Now they start moving around the tree, when choosing which vertex to travel to, Chintu always chooses the vertex with the largest number and Minku always chooses the vertex with the smallest number, among all adjacent vertices. This continues until both of them arrive at a leaf vertex (independent of each other).

Your friend wants to know the final locations of Chintu and Minku if they start at vertex 1. However when she describes the problem to you and your face is not filled with despair, she decides to be even more annoying and demands to know the final location of Chintu and Minku if they start at the i^{th} vertex for all $i \ \epsilon \ [1, N]$.

Now to make her leave you alone, you have to solve this problem.

`Note: The starting vertex is not considered a leaf.`

Input

The first line of input contains a single integer N ($2 \leq N \leq 10^5$).

The following $N-1$ lines of input each contain 2 integers u and v, representing that there is an edge between vertex u and vertex v.

It is guaranteed that the given graph is a tree.

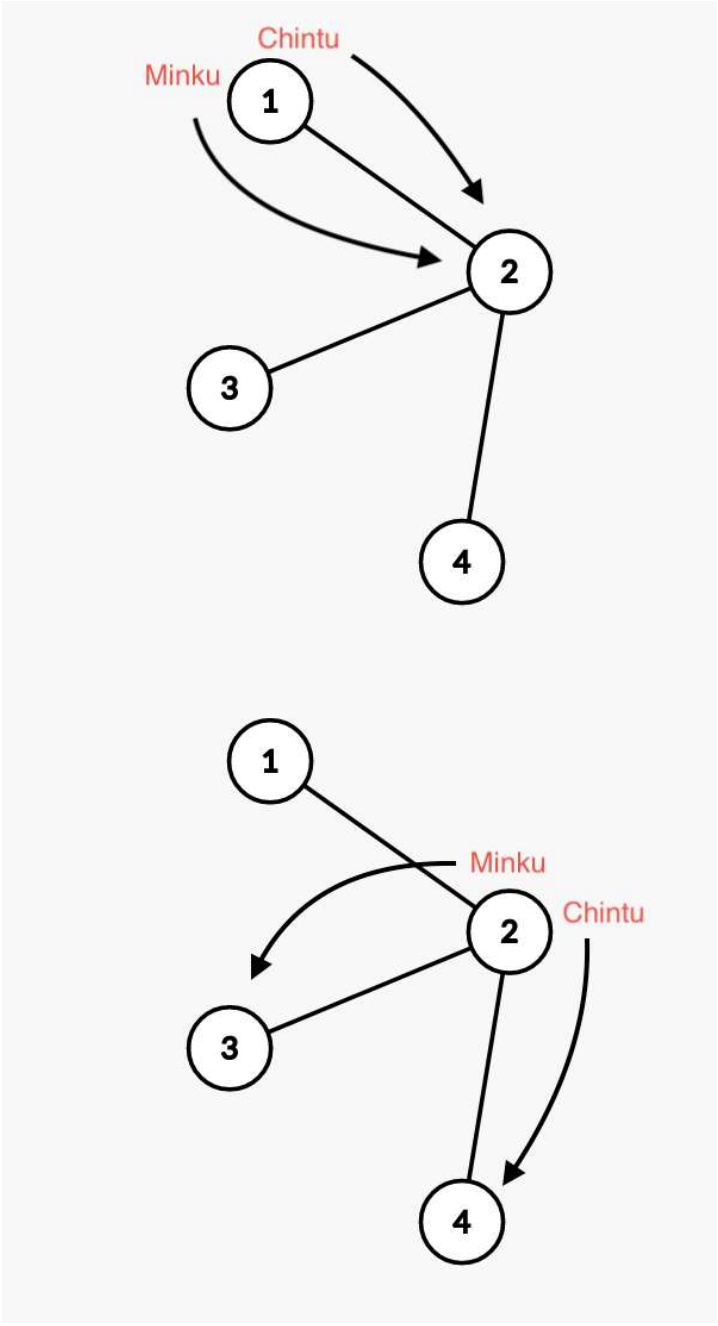
Output

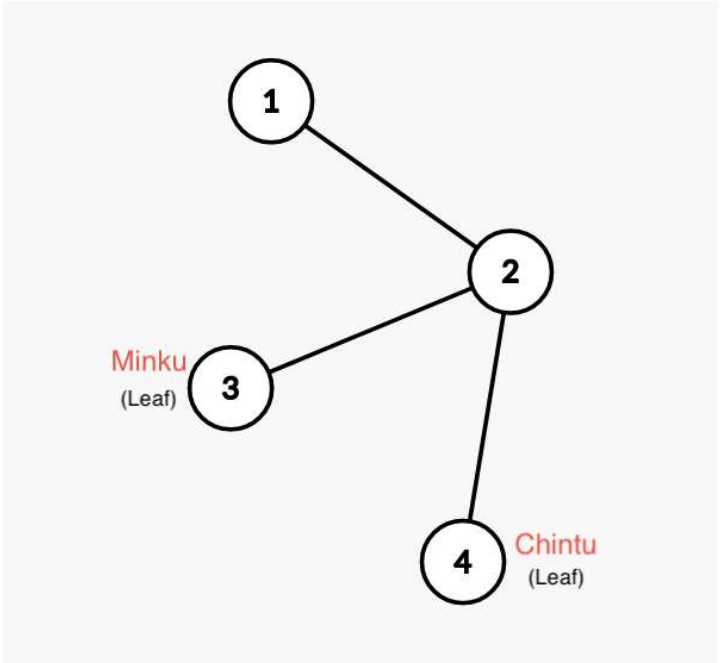
Print N lines, the i^{th} line consisting of 2 integers, the final position of Chintu and Minku, respectively, if they start at vertex i.

input
4 1 2 2 3 2 4
output
4 3 4 1 4 1 3 1

input
10 1 2 1 3 2 4 3 5 3 6 1 7 6 8 6 9 4 10
output
7 10 10 5 9 10 10 5 9 10 9 10 9 10 9 10 8 10 7 5

The following images show the process of Chintu and Minku traversing the tree given in the first sample test, starting at vertex 1.





E. Darkness

1 second, 256 megabytes

Alice walks through a forest for n days, each time she walks through it she loses some money. Bob is cursed with the power of $darkness$ and each day he can make Alice's lost money be multiplied by the current value of $darkness$.

You are given an array a consisting of n integers, a_i representing the money lost by Alice on the i^{th} day without bob's influence.

Initially the value of $darkness$ is k. Each time the power is used the value decreases by 1, a value of 0 implies that Bob no longer has the power.

Bob being a nice person does not want Alice to lose more money but if he still has the curse when the n days are over, he will have to face the god of $darkness$ and probably die.

Find the minimum total money lost by Alice.

Input

The first line of input contains two integers n and k. ($1 \leq k \leq n \leq 10^4$)

The second line of input contains n integers, the elements of array a. ($1 \leq a_i \leq 10^9$)

Output

Print the minimum total money lost by Alice.

input
5 3 5 2 3 1 4
output
20

input
9 4 99 10 30 20 30 30 30 30 100
output
479

F. POP OR POP

1 second, 256 megabytes

You are given a integer X and array of integers a and a integer s which is initially 0. You can perform these two operations ᓚᘏᗢ

- Pop the first element from the array a and add it to s.
- Pop the last element from the array a and add it to s.

You can perform the following 0 or any number of times. Find if it is possible to make $s = X$.

Input

The first line of input contains one integer T, the number of test cases ($2 \leq T \leq 5 * 10^3$).

The next $2 * T$ line contains two integers n size of array a and X ($1 \leq N \leq 2 * 10^5$), ($-10^9 \leq X \leq 10^9$) and $x \neq 0$

The next line contains n items of array a ($-10^9 \leq a_i \leq 10^9$)

Output

Print for each test case t_i

"YES" if it is possible to make X

"NO" otherwise

separated by new line

input
3 3 1 1 2 -1 4 -2 -3 2 -2 5 4 -2 -3 2 -2 3
output
YES NO YES

Elements of the array can be negative ie ≤ 0

G. Another Minimizing Problem

3 seconds, 256 megabytes

You are given an array a consisting of n positive integers and q queries to this array.

Let's define sum(l, r) of array to be $\sum a_i$ for all $l \leq i \leq r$.

Let compliment(l, r) of array = sum(1, $l-1$) + sum($r+1$, n).

There are two types of queries:

`1 l r x` — for each index i such that $l \leq i \leq r$ set $a_i = x$.

`2 l` — find $r \geq l$ such that absolute difference of compliment(l, r) and sum(l, r) is minimized.

Input

The first line contains two integers n ($1 \leq n \leq 10^5$) and q ($1 \leq q \leq 10^5$) — the number of elements in array and the number of queries.

The second line contains n integers — elements of the array a ($1 \leq a_i \leq 10^9$).

Then q lines follow, each representing a query. Each query is given either as $1\ l\ r\ x$ or $2\ l$. ($1 \leq l \leq n$) , ($l \leq r \leq n$) , ($1 \leq x \leq 10^9$).

Output

For each query of type 2 print r - answer to this query.

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

input
5 4 6 7 3 9 8 2 3 1 4 4 12 2 5 1 1 3 7
output
5 5

H. divmin

1 second, 256 megabytes

There are n stones weighing 1 Kg,2 Kg....n Kg. Your task is to make 2 sets of stones such that every stone should belong to exactly one of the sets and there should be at least 1 stone in both sets. While making the sets you must remember the 2 rules.

Rule 1 : Every stone of the first set should weigh less than every stone of the second set.

Rule 2 : The absolute difference in the total weights of both sets must be the minimum possible.

Print the absolute difference in the total weights of both sets.

Input

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

The next T lines contain an integer N, the number of stones. ($2 \leq N \leq 10^7$).

Output

Print T lines, each containing a single integer, the absolute difference in the total weights of both sets.

input
5 2 3 4 5 6

output
1 0 2 3 1

For test 2,N = 3 so we can put weight 1 and 2 in one set and weight 3 in other.

For test 5,N = 6 set1 : {1, 2, 3, 4} and set2 : {5, 6}.

I. INAZUMA's magical city

1 second, 256 megabytes

In INAZUMA's magical city there are 'N' towns and 'M' one-way roads that connect these towns. Towns are numbered from $1, 2, 3...N$.

AXEL is a pro footballer and wishes to visit each town.

AXEL knows a MAGICIAN who can help him, you are given an array A of length 'n', the magician charges him 'A_i' rupees to magically teleport him to the i^{th} town.

AXEL can then travel for free on the ONE-WAY roads between towns. He can use each road any number of times, but he only uses them if it is possible to use these roads and come back to the town from which he started.

AXEL needs to visit each town at least once, Help AXEL by telling him the minimum money he needs to spend to visit all the towns.

Input

The first line contains two integers N and M $(1 \leq N \leq 10^5)$ $(0 \leq M \leq 10^5)$— the number of towns and the number of one way roads.

The next line contains N integers $a_1, a_2, \ldots, a_n (0 \leq a_i \leq 10^9)$ — where a_i is the cost of teleporting to the i^{th} town.

This is followed by M lines describing the roads. Each road is characterized by two integers u and v $(1 \leq u, v \leq N, u \neq v)$ — the one-way road from town 'u' to town 'v'. It is guaranteed that each one-way road (u, v) is mentioned exactly once in the input.

Output

Print a single integer, representing the minimum amount of money that AXEL needs to spend.

input
6 6 1 2 3 4 5 6 1 2 2 3 3 4 4 5 5 6 6 1

output
1

input
3 2 10 10 20 1 2 2 1

output
30

For the first sample test case, the road map is depicted in the photo above problem statement. AXEL will pay the magician to teleport him to the 1^{st} town. He will then travel to the 2^{nd}, 3^{rd}, 4^{th}, 5^{th}, and 6^{th} towns by road and return to the 1^{st} town.

J. XOR

1 s., 256 MB

After putting a lot of effort in searching the new ways to cheat and pass online exams as "Modern Problems Require Modern Solutions", cooldude gets exhausted and now just wants to rest till the next exams. So he wants you to solve this problem, on his behalf.

You are given an array Arr containing N integers.

You will be given M queries, each query contains two integers A and B.

Find xor of the resultant array after replacing all occurrences of A in the array by B.

$Note$: All queries are dependent on each other i.e. you need to perform the second query on the resultant array of the first query, 3rd query on the resultant of 2nd query, and so on.

Input

The first line of input consists of two space-separated integers N and M.

The second line of input contains N integers $(A_1, A_2, \ldots, A_n)$, where A_i is the ith element of the array

For each of the M queries, you were given two integers A and B.

Output

For each query print the XOR of the whole Array after replacing all A by B

$Constraints$:

$0 \leq N \leq 10^5$

$0 \leq M \leq 10^5$

$0 \leq Ai \leq 10^9$

input
5 6 1 2 3 4 5 1 2 2 3 3 4 4 5 5 1 1 5

output
2 2 5 5 1 5

Resultant Array after $1st$ query: 22345 After $2nd$ query: 33345 After $3rd$ query: 44445 After $4th$ query: 55555