Problem A. Code Lock

Time limit: 1 second
Memory limit: 256 megabytes

You have a lock with an integer array a of length n written on it. The numbers in the array can be repeated. The lock opens only when the array is sorted in non-decreasing order.

To change the state of the array there is only operation available which uses a permutation p of numbers from 1 to n. When this operation is applied the elements of the array change positions according to this permutation, meaning the number a_i moves from position i to position p_i for each i.

In the beginning the array on the lock was sorted in non-decreasing order but then this operation was applied to it once. Your goal is to unlock the lock again but there's a trick: you can independently restore any position of the array to the value that was on this position before. In general, you can do the following:

1. First, you record the current state of the array.
2. Then you apply the operation to the array moving its elements according to the permutation p.
3. If for every i there exists a previously recorded state in which a_i equals to the initial value of a_i, you stop, otherwise you repeat the first two steps.

How many iterations of this algorithm will be performed until you'll be able to restore the initial values? Notice that you strictly follow the algorithm, for example if the array is already sorted in non-decreasing order you still perform one iteration before you stop.

Input

The first line contains a single integer n — the length of the array on the lock ($1 \le n \le 5 \cdot 10^5$).

The second line contains n space-separated integers a_i — the initial (sorted) state of the array a ($1 \le a_i \le 10^9$; $a_i \le a_{i+1}$).

The last line contains n distinct space-separated integers p_i — the elements of the permutation ($1 \le p_i \le n$).

Output

Output a single integer — the number of states of the array that will be recorded before you can unlock the lock.

Examples

standard input	standard output
3 1 1 1 3 1 2	1
5 1 1 2 3 3 2 4 3 5 1	3
6 1 1 1 1 2 2 2 3 5 6 4 1	4

Note

In the third example, the array states will be equal to

1. $2, 1, 1, 2, 1, 1$
2. $1, 2, 1, 1, 1, 2$
3. $2, 1, 2, 1, 1, 1$
4. $1, 2, 1, 1, 2, 1$

As can be seen, 2 does not appear in the fifth position until the fourth iteration. On all other positions the required number appears at least once during these four iterations, so the answer is 4.

Problem B. Performance

Time limit: 2 seconds
Memory limit: 512 megabytes

There is a theatrical performance which will be watched by n regular spectators and m theater critics. The spectators will sit on the first row on places numbered from 1 to n while the critics will watch the performance from their own section.

Critics analyze not only the performance itself but also the response of the audience. Critic number i will observe the spectators on the seats from l_i to r_i inclusive. Initially the spectator on a seat number j has a level of *criticism* equal to a_j.

Before the performance the promoters can approach some viewers and spend no more than k items of merchandize. Each item given to a viewer will decrease their criticism by 1. However, the level of criticism of any viewer can't become negative.

We define a *criticism* of the i-th critic as a total criticism level of the viewers he's observing. What is the minimum possible sum of critics' levels of criticism that can be achieved?

Input

The first line contains three numbers n, m, and k — the number of spectators, critics, and the amount of merchandize ($1 \le n \le 10^5$; $1 \le m \le 10^6$; $0 \le k \le 10^{12}$).

The second line contains n space-separated integers a_i — the initial values of the criticism levels of the spectators ($0 \le a_i \le 10^7$).

The next m lines contain two numbers l_i and r_i each indicating that the i-th critic is observing the spectators with numbers from l_i to r_i inclusive ($1 \le l_i \le r_i \le n$).

Output

Output a single integer — the minimum sum of the levels of criticism that the critics can have about the performance.

Examples

standard input	standard output
4 2 2 1 2 3 4 1 4 3 4	13
4 2 5 1 2 0 0 1 4 3 4	0

Problem C. Protective Field

Time limit: 10 seconds
Memory limit: 256 megabytes

There are n creatures on the plane, creature number i is a point with coordinates (x_i, y_i).

Your goal is to establish a protective field in a shape of a circle with the minimal possible radius that will cover at least half of those creatures, i.e. $\lceil \frac{n}{2} \rceil$.

Given that you can choose any point (possibly with non-integers coordinates) as a center of the field, determine the optimal position and radius of the field.

Input

The first line of the input contains an integer n — the number of creatures ($1 \le n \le 400$).

The next n lines contain n pairs of real numbers x_i and y_i — the coordinates of each creature ($-10^9 \le x_i, y_i \le 10^9$). The creatures are lazy and do not move, so their positions are constant.

Output

Output three numbers — the center of the protective field of the minimum radius that covers at least half of the creatures, and the radius itself. All numbers should be printed with an absolute or relative error not exceeding 10^{-6}.

Examples

standard input	standard output
8 4 3 3 4 -3 4 -4 3 -4 -3 -3 -4 3 -4 4 -3	0 3 4
4 1.5 1.5 1.5 -1.5 -1.5 -1.5 -1.5 1.5	1.5 0 1.5

Problem D. Company Network

Time limit: 2 seconds
Memory limit: 256 megabytes

In a particular company there is a hierarchy in the form of a rooted tree. Group's management consists of n people:

- the CEO (leader, root of the tree) holds the number 1;
- each of the remaining $n-1$ members has their direct supervisor: the supervisor of member number i has the number p_i;
- in addition, the i-th of the n managers has a_i ordinary "subordinates" in direct subordination (the sets of subordinates of different managers do not intersect).

You need to deliver an important message that should reach as many people in the company as possible. It is known that as soon as a person receives the message, they pass it on to all their direct subordinates. Ordinary subordinates receive the message from their manager instantly, and the manager number i receives the message from their supervisor in t_i minutes.

You have exactly T minutes and only one call to any of the n managers. Choose who to call in order to reach as many people as possible within those T minutes.

Input

The first line contains two integers n and T — the number of managers and the time limit for spreading the message ($1 \le n \le 10^5$; $0 \le T \le 10^9$).

The next line contains $n-1$ integers $p_2, \ldots, p_n$ — the numbers of the direct supervisors of the managers from 2 to n ($1 \le p_i \le n$). It is guaranteed that the hierarchy is a tree rooted by node 1.

The third line contains $n-1$ integers $t_2, \ldots, t_n$ — the time it takes for the message to reach the corresponding managers from their direct supervisor ($0 \le t_i \le 10^9$).

The last line contains n integers a_i in the same format — the number of ordinary subordinates of each manager ($0 \le a_i \le 10^9$).

Output

Print two integers separated by a space: the number of the manager that you call, and the total number of people who will receive the message within T minutes.

If there are multiple optimal answers, print any of them.

Examples

standard input	standard output
3 10 1 1 9 11 100 49 51	1 151
7 15 1 1 2 2 3 3 9 8 6 9 16 5 400 100 200 50 1000 1100 300	2 1153

Note

In the first example you should call the CEO. In 10 minutes the message will reach him, the manager number 2, and another $100 + 49$ of their subordinates: 151 people in total.

In the second example the manager with the largest number of subordinates (1100) will not receive the message in 15 minutes if he is not called directly, and the person with 1000 subordinates will not receive the message if the CEO is called. The optimal answer is achieved by calling the manager number 2: then he will receive the message, as well as two of his subordinates (3 and 4) and another $100 + 50 + 1000$ of their ordinary subordinates, for a total of 1153 people.

Problem E. Fireworks Show

Time limit: 1 second
Memory limit: 256 megabytes

At the next autumn festival, a grand fireworks show is planned, during which n different colored rockets will be launched. Each rocket has two charges of the same type (one charge is strictly below the other, it is not possible to remove the lower one without first removing the upper one).

Unfortunately, someone had swapped some of the charges — now some rockets have charges of different types. However, the overall composition of the fireworks has not changed — in all the rockets together there are still two charges of each of the n types.

Your goal is to rearrange the charges between the rockets in order to once again have n rockets, each with two charges of the same type. For this you have an additional $n+1$-th rocket at your disposal, which does not contain any charges. In one action you can:

1. select rocket number i, which has at least one charge;
2. select rocket number $j \neq i$, which has strictly less than two charges;
3. transfers the **upper** charge from rocket i to the **lower** slot in the rocket number j.

To clarify, upper or lower in this context refers not to the "upper or lower of two positions" but to the last occupied or first free slot going from bottom to top. For example, if there's only one charge in the rocket then it's located on the bottom but it still is considered "upper" (since there is nothing above it) thus can be transferred to another rocket using the described action.

Find a way to obtain n rockets with pairs of identical charges in no more than $2n$ such actions.

Input

The first line contains an integer n — the number of rockets prepared for the fireworks show ($1 \leq n \leq 10^5$).

In the i-th of the next n lines the current state of the i-th rocket is described: the numbers of the lower and upper charges types are given separated by a space ($1 \leq x_{i_1}, x_{i_2} \leq n$). It is guaranteed that each number from 1 to n appears exactly twice in the descriptions of the rockets. Rocket number $n+1$ is initially empty.

Output

The first line should contain an integer k — the number of actions you need ($0 \leq k \leq 2n$).

In the next k lines describe the actions in the order they are performed. Each action is described by the numbers of the rockets (from 1 to $n+1$) between which the upper charge should be transferred. It is not allowed to put more than two charges in a rocket and it is not allowed to transfer a charge from a rocket to the same rocket.

Note that you are **not required to minimize the number of actions** — it is enough to simply ensure that there are no more than $2n$ of them.

Examples

standard input	standard output
3 2 1 3 3 1 2	3 1 4 3 1 4 3
5 1 5 2 3 3 5 4 2 1 4	6 1 6 3 6 2 3 4 2 5 4 5 1

Problem F. Recombination

Time limit: 1 second
Memory limit: 512 megabytes

Scientists discovered a fascinating new bacteria and observe its colony. There are n individuals in the colony (n is even). The genetic code of the i-th individual is obtained by cyclically shifting a string s of length n by i positions, which means it contains the characters $s_i, s_{i+1}, \ldots, s_n, s_1, s_2, \ldots, s_{i-1}$.

New individuals are created through recombination: the genetic codes of two individuals from the first generation are taken and written down character by character in alternation: the first character from the first individual, the second from the second individual, the third character from the first individual, and so on. Thus, when recombining the i-th and j-th Avatars, the resulting string is $s_i, s_{(j+1) \bmod n}, s_{(i+2) \bmod n}, s_{(j+3) \bmod n}, \ldots, s_{(i+n-2) \bmod n}, s_{(j+n-1) \bmod n}$.

Determine how many pairs of individuals from the first generation exist, such that when recombined, they produce a new genetic code that is not represented in the first generation. If there are two pairs of individuals (i_1, j_1) and (i_2, j_2) for which the same new genetic code is obtained, both pairs should be counted in the answer.

Input

The first line of the input contains a single even integer n — the length of the genetic code and the number of individuals in the first generation ($2 \le n \le 10^6$).

The second line contains the original string s, consisting of lowercase Latin letters.

Output

Output a single integer — the number of pairs of individuals from the first generation, whose recombination results in a genetic code that is not represented in the first generation.

Examples

standard input	standard output
4 abcd	12
4 abab	8
12 aabbccaabbcc	48

Problem G. Heaps Rearrangement

Time limit: 2.5 seconds
Memory limit: 256 megabytes

There are n heaps of data, heap i consists of c_i clusters and each of these clusters contain w_i information.

Your task is to emulate an algorithm used by data scientists in an IT company "Goggle". This algorithm is used for redistribution of n heaps of data between n data scientists and it operates as follows.

1. Each data scientist will receive the same number of clusters as there are in the corresponding heap initially, meaning that exactly c_i clusters will be assigned to the i-th data scientist.
2. The algorithm iterates through all of the data scientists from first to last and doesn't move to the data scientist $i + 1$ until exactly c_i clusters are assigned to the i-th data scientist.
3. The first cluster ever assigned is taken from the first heap. Each following cluster is taken from the next non-empty heap relative to the previous one. Order on the heaps is cyclic which means that the "next" for the last heap is the first one.

In other words, the algorithm maintains a sequence of heaps that have not been fully distributed yet, and a pointer indicating the "current" heap. When it assigns the next cluster to the current data scientist, one cluster is taken from the current heap and the pointer moves to the next one in cyclic order. When the i-th data scientist has exactly c_i clusters assigned to them the algorithm moves to the next data scientist.

Because running the entire algorithm could take too much time, determine the total information in all of the clusters each data scientist has when the algorithm finishes.

Input

The first line contains an integer n — the number of heaps and data scientists ($1 \le n \le 5 \cdot 10^5$).

In the i-th line among the following n lines two integers c_i and w_i are given separated by a space. They represent the number of clusters in the i-th heap and the amount of information in each of them ($1 \le c_i, w_i \le 10^9$).

Output

In a single line output n numbers separated by spaces where the i-th number represents the total amount of information assigned to the i-th data scientist.

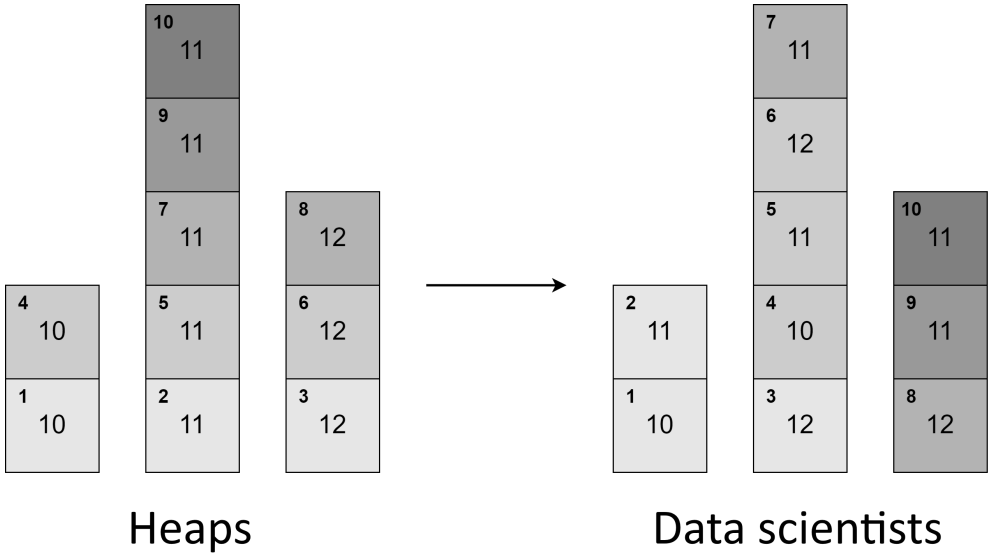
Examples

standard input	standard output
3 3 1 2 10 4 100	111 11 301
1 10 10	100
3 2 10 5 11 3 12	21 56 34

Note

You can see the illustration for the third example below. On the left side, the clusters that make up the heaps are shown, while on the right side, the clusters assigned to each data scientist are displayed.

The amount of information in each cluster is indicated in the center of the cluster, and the number in the upper left corner represents the order of cluster selection. For clarity clusters are selected from heaps in a left-to-right, then bottom-to-top order, and they are assigned in a bottom-to-top, then left-to-right order.



Problem H. Garbage

Time limit:	3 seconds
Memory limit:	256 megabytes

If you spend too much time programming, garbage starts to pile up. It's even more true if there are a lot of programmers involved. After the world record breaking hackathon that lasted n days the organizers were left with n piles of garbage that had to be cleaned up.

Piles are numbered from 1 to n and in the i-th pile there is exactly a_i garbage. Because throwing away this amount of garbage is challenging, it was immediately decided to organize another hackathon for this particular task. To make it more interesting for the competitors, they were asked to perform the following operations:

1. Calculate the sum $\sum_{i=l}^{r} a_i \oplus i$, where $x \oplus y$ represents the XOR of two numbers.
2. Assign the value x to all elements of the array in the range $[l, r]$.
3. Apply the bitwise AND, OR, or XOR operation with the number x to all numbers in the range $[l, r]$.

Since the task was solved and a winner received immense praise, we offer you an opportunity to solve it as a training for future hackathons you may participate in.

Input

The first line contains two integers n and m which represent the number of piles of garbage and the number of queries, respectively ($1 \le n, m \le 10^5$).

The next line contains n space-separated integers $a_1, \ldots, a_n$ — the amount of garbage in each pile ($0 \le a_i < 2^{15}$).

The following m lines describe the queries each in one of the following formats.

1. A query of the first type in the format "`1` l r".
2. A query of the second type in the format "`2` l r x".
3. A query of the third type in the format "`3` l r x c" where the symbol c denotes the logical operation to be applied: AND ('`&`'), OR ('`|`'), or XOR ('`^`').

In each query $1 \le l \le r \le n$ and $0 \le x < 2^{15}$.

Output

For each query of the first type, output the required sum on a new line.

Example

standard input	standard output
5 6 3 0 11 21 17 1 2 5 2 1 3 9 1 1 4 3 3 5 23 ^ 3 2 4 19 & 1 1 5	47 46 37

Problem I. Collecting Artifacts

Time limit: 1.5 seconds
Memory limit: 512 megabytes

Imagine that you have found a treasure map depicting paths between n locations. Path i connects locations with numbers u_i and v_i and has a length of c_i. The number of paths is exactly $n-1$ and it is possible to reach any site from any other. In other words, the structure of paths and locations forms a tree.

Before exploring the locations you have investigated the map and determined that there a total of k types of artifacts, and location number i contains an artifact of type a_i. Your goal is to collect all k different types of artifacts meaning you should visit at least one location containing each type of artifacts.

You can choose any two locations as your starting and finishing points and travel any (not necessarily simple) path between them. Calculate the smallest distance you'll have to travel in order to collect all k types of artifacts.

Input

The first line contains two space-separated integers n and k which represent the number of locations and the number of types of artifacts ($2 \le n \le 10^5$; $1 \le k \le 6$).

The next line contains n integers a_i separated by a space, representing the types of artifacts in each site ($0 \le a_i \le k$). If $a_i = 0$, the i-th location doesn't contain artifacts at all.

The following $n-1$ lines contain triplets of integers u_i, v_i, c_i, indicating the existence of a path of length c_i between sites u_i and v_i ($1 \le u_i, v_i \le n$; $1 \le c_i \le 10^9$). It is guaranteed that the graph structure forms a tree.

Output

If it is impossible to collect k different types of artifacts, output "`-1`" (without quotes). Otherwise, output the **minimum** distance that needs to be traveled to collect all types of artifacts.

Examples

standard input	standard output
5 4 1 3 2 4 4 1 3 1 2 3 1 3 4 1 4 5 1	4
5 5 1 3 2 4 4 1 3 1000 2 3 5123 3 4 3341 4 5 7197	-1
4 3 0 1 2 3 1 2 10 2 3 1 3 4 50	51

Note

In the first example, one of the optimal paths would be $1 \to 3 \to 2 \to 3 \to 4$.

In the second example, there is no site that contains artifact number 5, so it is impossible to collect all five artifacts.

In the third example, one of the optimal paths would be $2 \to 3 \to 4$.

Problem J. Histogram Matching

Time limit: 1 second
Memory limit: 256 megabytes

A large histogram s of length m is available to you. Histogram is represented by n integers where the i-th integer s_i represents the height of the i-th column of the histogram.

There is a standard procedure called "*extraction*" that goes as follows:

1. First you select a subsegment of the histogram from the l-th column to the r-th inclusive.
2. Then you draw a horizontal "baseline" at height h (which can be negative) on the histogram between columns l and r.
3. You select all columns $i_1, \ldots, i_n$ with $s_i > h$ (strictly above the line) on the segment $[l, r]$ and draw a new histogram t of size n in which $t_j = s_{i_j} - h$.

Your task is to implement the reverse operation. Given the initial histogram s and the resulting histogram t determine from which columns $i_1, \ldots, i_n$ of s could t be "*extracted*".

Input

The first line contains an integer n — the length of the resulting histogram t ($1 \le n \le 10^5$). The second line consists of n space-separated integers t_i representing the heights of the columns in the histogram t ($1 \le t_i \le 10^6$).

In the third and the fourth lines the initial histogram s is given in the same format ($n \le m \le 10^5$; $n \cdot m \le 10^7$; $1 \le s_i \le 10^6$).

Output

In the first line output "`YES`" (without quotes) if the provided t can be extracted from s by the given procedure, and "`NO`" otherwise.

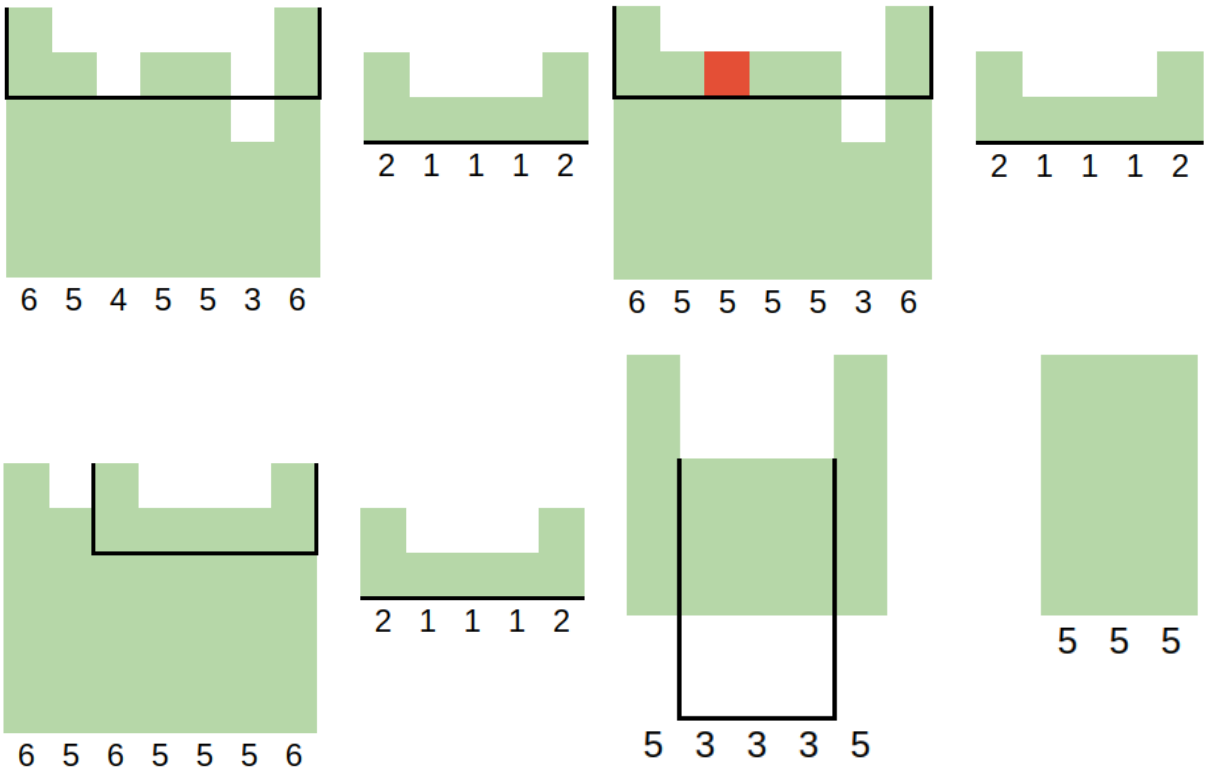
If you printed "`YES`" on the first line, the second line should contain n space-separated integers $i_1, \ldots, i_n$ in increasing order which represent the columns of s forming t ($1 \le i_j \le m$).

Examples

standard input	standard output
5 2 1 1 1 2 7 6 5 4 5 5 3 6	YES 1 2 4 5 7
5 2 1 1 1 2 7 6 5 5 5 5 3 6	NO
5 2 1 1 1 2 7 6 5 6 5 5 5 6	YES 3 4 5 6 7
3 5 5 5 5 5 3 3 3 5	YES 2 3 4

Note

Below you can see the illustrations for the examples. Each illustration contains an initial histogram (on the left) with highlighted segment $[l, r]$ and baseline h and the resulting histogram t (on the right):



In all the given examples, except the second one, it is possible to extract a subset of columns (while the remaining columns are left below the baseline) to get the desired result. However in the second example it is not possible since there is an extra column above the baseline.

Note that it is not necessary for $l = 1$ and $r = n$ to hold true.

Problem K. Monster Transformer

Time limit: 1 second
Memory limit: 256 megabytes

Everyone loves toys that you can transform at will. This year "Le'go" announced a new project in collaboration with famous cartoon "Monsters Company". After its launch kids will be able to create unique monster toys and assemble large collections of them.

One line of products in this project features monsters with exactly five limbs and three heads. Company designed n types of limbs and m types of heads which can be purchased and attached to the body of a monster independently.

Since the parts are reusable the company wants to find how many different monsters can be created from different pools of components. To determine that they conduct the following test.

1. First, they select a limbs and b heads. Each limb is chosen independently and uniformly at random, meaning that each limb type could appear with a probability of $\frac{1}{n}$, and likewise for the heads.
2. Then they calculate the *expected value* of the number of **different** monsters they could assemble from the selected parts. Two monsters are considered different if they differ in at least one limb or at least one head, taking their order into account (e.g., "leg, arm" is not the same as "arm, leg").

Since they don't have many talented programmers among their staff you are tasked with finding this value instead.

Because obtaining the exact value may be impossible with floating point arithmetics, perform all arithmetic operations in the field of remainders modulo a prime number $p = 1073676287$. This means that if the exact answer can be represented as a rational fraction $\frac{x}{y}$, you should output an integer z such that $z \cdot y \equiv x \mod p$.

Input

The first line contains two integers n and m separated by a space, representing the number of types of limbs and heads, respectively ($1 \le n, m \le 670$).

The second line contains two integers a and b separated by a space, representing the number of limbs and heads in the pool that the company has selected ($5 \le a \le 10^9$; $3 \le b \le 10^9$).

Output

Output a single integer, which is the expected value of the number of different monsters that could be assembled from a pool of parts of a given size, assuming that any combination of limb and head types in the pool is equally likely.

The resulting integer should be the result of calculating the rational fraction representing the expected value in the field of remainders modulo p.

Examples

standard input	standard output
1 1 5 3	1
1 2 5 3	536838146
5 3 5 3	629317602

Note

The answer for the second example in the problem statement is $\frac{20}{8}$ or $\frac{5}{2}$. It can be verified that $536838146 \cdot 2 = p + 5$.

Problem L. Fence Decoration

Time limit: 2 seconds
Memory limit: 512 megabytes

A house with m residents has a fence. The front segment of the fence should be decorated but the residents can't agree on where to put the decorations. There are n possible slots for decorations: the i-th slot is located at a distance of x_i from the beginning of the fence.

The decorations can be put into any subset of the slots that includes the first slot and the last slot. Among all such subsets the residents decided to choose one that maximizes their total *satisfaction*. Satisfaction is calculated as follows:

1. First, each resident picks their favorite number. If the i-th resident picked a number d_i then for each two **consecutive** decorations the total satisfaction **increases** by $|d - d_i|$ where d is the distance between said decorations.
2. For every occupied slot i the total satisfaction **decreases** by c_i.

In other words if the decorations are placed in the slots $i_1, \ldots, i_k$ (in order of increasing distance from the beginning of the fence), the total satisfaction is calculated as

$$\left(\sum_{r=1}^{m}\sum_{t=2}^{k}|x_{i_t} - x_{i_{t-1}} - d_r|\right) - \left(\sum_{t=1}^{k} c_{i_t}\right).$$

Find the maximum total satisfaction that can be achieved by optimally selecting the slots for decorations.

Input

The first line of input consists of two space-separated integers n and m representing the number of slots for decorations and the number of residents in the house, respectively ($2 \le n \le 10^5$; $1 \le m \le 10^5$).

The second line contains m space-separated integers $d_1, \ldots, d_n$ denoting the favorite numbers of each resident ($0 \le d_i \le 10^7$).

Each of the following n lines contains two numbers x_i and c_i: the distance of the i-th slot from the beginning of the fence and its impact on the total satisfaction of the residents ($0 \le x_i \le 10^7$; $0 \le |c_i| \le 10^{12}$). It is guaranteed that $x_1 < x_2 < \ldots < x_n$.

Output

Output a single number, which represents the maximum possible total satisfaction of the residents in the house.

Examples

standard input	standard output
2 1 10 0 5 20 3	2
3 3 3 7 10 2 20 5 4 10 -3	-1
9 5 30 64 2 93 67 0 81 1 256 6 251 13 256 23 180 52 256 72 94 77 256 97 12	137