Problem Set 14
15-295 Spring 2018

A. Genta Game

time limit per test: 1.0 s
memory limit per test: 256 MB
input: standard input
output: standard output

Kojima Genta is one of the best friends of Conan, and the fattest one!

Everyone believes that Genta is just thinking about food. So, he wants to prove the opposite. So, his friends challenged him in a game. Genta's friends will give him a string s of length n, and m update operations. At each update operation, an integer p $(1 \leq p \leq n)$ and a lowercase English letter c will be given to Genta, and he is asked to change the p^{th} letter in the string s to the letter c.

Conan explained to Genta that an update operation is said to be beautiful if the string s was a palindrome string after the update operation has been executed.

Genta task is to count the number of beautiful update operations. Genta wants to win in this game no matter what this will cost because his friends promised him that the food will be at their expense throughout the week if he solved the task. Can you help Genta by solving his task?

Input

The first line contains an integer T $(1 \leq T \leq 50)$, in which T is the number of test cases.

The first line of each test cases contains two integers n and m $(1 \leq n, m \leq 10^5)$, in which n is the length of the string s and m is the number of update operations. The second line of each test cases contains a string s of length n consisting of lowercase English letters only. Then m lines follow, each line contains an integer p and a lowercase English letter c $(1 \leq p \leq n)$, giving the update operations.

The sum of n and m overall test cases does not exceed 7×10^5 for each.

Output

For each test case, print a single line containing the number of beautiful update operations.

Example

input

```
1
10 7
abcdefdcba
5 x
6 x
4 d
2 d
3 y
8 y
9 d
```

output

```
3
```

Note

A `palindrome` is a word, phrase, number, or other sequence of characters which reads the same backward as forward, such as "`madam`" or "`racecar`".

In the first test case, the string s will be updated as follow:

abcdefdcba $\longrightarrow$ abcdxfdcba $\longrightarrow$ abcdxxdcba $\longrightarrow$ abcdxxdcba $\longrightarrow$ adcdxxdcba $\longrightarrow$ adydxxdcba $\longrightarrow$ adydxxdyba $\longrightarrow$ adydxxdyda.

There are 3 beautiful update operations, which are the 2^{rd}, 3^{th}, and 7^{th} operations.

Problem B
Robots

Input File: G.in
Output File: standard output
Time Limit: *0.2* seconds (C/C++)
Memory Limit: *128* megabytes

You're the leading designer of a complex robot, that will explore human unreachable locations. Your job is to design a robot that will go as far as possible. To do this, you have **n** available energy sources. The **i**th source is capable of accelerating the robot by a rate of $\mathbf{a_i}$ (m/s^2) and can do this for a total of $\mathbf{s_i}$ seconds. The robot is initially at rest (its initial velocity is zero). You have to decide the order in which to use the sources in order to maximize the total distance traveled by the robot. You will use one source until $\mathbf{s_i}$ seconds have elapsed, then immediately switch to another unused source (the switch is instantaneous). Each source can be used only once.

Given the accelerations and durations of each source, write an efficient program to determine the optimal order of the sources, in order to maximize the total distance traveled. Your program must compute the difference between the traveled distance in the optimal case and in the default case (the order given by the input data).

Physics background: if the velocity is **v** before you start using a source whose acceleration is **a** then, after **t** seconds, the robot has traveled a total $\mathbf{vt+1/2at^2}$ meters, and the final velocity will be **v' = v+at**.

Input
The input file starts with the number **n** **(**$1 \le \mathbf{n} \le 10^4$**)** of sources. Starting from a different line follows the **n** space-separated acceleration and duration for each source (positive integer numbers).

Output
The output file contains the computed difference between the traveled distance in the optimal case and in the default case (the order given by the input data), with one decimal.

Sample input	Sample output

2
2 1
30 2
	56.0

C. Stripe Bishops

time limit per test: 1.0 s
memory limit per test: 512 MB
input: standard input
output: standard output

Ignatiy is a huge fan of chess game. But even more than that, he is a professional problem author and solver, that is why he creates lots of problems involving chess. You are going to face one of his problems.

Consider a chessboard of an unusual form. It looks like a horizontal stripe consisting of h rows and infinite number of columns. Rows are numbered from bottom to top with consecutive integers from 0 to $h - 1$. Columns are numbered with consecutive integers as well, one of them has index 0, columns to the right have indexes $1, 2$, and so on, while columns to the left have indexes $-1, -2$, and so on.

Bishop is a chess piece that, when located in a cell (x, y), attacks all cells that share the same diagonal with the cell (x, y) (both $/$-shaped diagonal and $\backslash$-shaped diagonal).

Ignatiy put n bishops in different cells of a board in such way that **no two bishops attack each other** (i.e. no bishop is located in a cell that is under attack of another bishop). Now he asks you the total number of cells that are under attack of at least one bishop. Any cell occupied by one of the bishops is not considered to be under attack of this bishop.

Input

In the first line of input there are two integers n and h ($1 \le n \le 200\,000$, $1 \le h \le 10^9$), the number of bishops and the height of a board.

In the i-th of the following n lines there are two integers x_i and y_i ($-10^9 \le x_i \le 10^9$, $0 \le y_i \le h - 1$), the coordinates of the i-th bishop.

It is guaranteed that no two bishops are located at the same cell and that no two bishops attack each other.

Output

Output the number of unoccupied cells attacked by at least one bishop.

Example

input

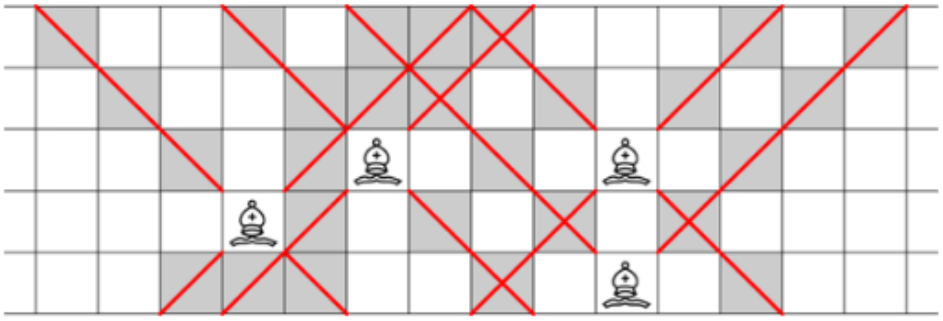
```
4 5
-1 1
1 2
5 2
5 0
```

output

```
27
```

Note

An illustration for the sample test is given below:



Problem D
Escape Room

Input File: K.in
Output File: standard output
Time Limit: *1* second (C/C++)
Memory Limit: *64* megabytes

As you know, escape rooms became very popular since they allow you to play the role of a video game hero. One such room has the following quiz. You know that the locker password is a permutation of **N** numbers. A permutation of length **N** is a sequence of distinct positive integers, whose values are at most **N**. You got the following hint regarding the password - the length of the longest increasing subsequence starting at position **i** equals A_i. Therefore you want to find the password using these values. As there can be several possible permutations you want to find the lexicographically smallest one. Permutation **P** is lexicographically smaller than permutation **Q** if there is an index **i** such that **Pi** < **Qi** and $P_j = Q_j$ for all **j < i**. It is guaranteed that there is at least one possible permutation satisfying the above constraints.
Can you open the door?

Input
The first line of the input contains one integer **N** ($1 \leq N \leq 10^5$).
The next line contains **N** space-separated integer A_i ($1 \leq A_i \leq N$).
It's guaranteed that at least one possible permutation exists.

Output
Print in one line the lexicographically smallest permutation that satisfies all the conditions.

Sample input	Sample output

`4`
`1 2 2 1`
	`4 2 1 3`

`1` `1`	`1`

E. Unsmooth Tree

time limit per test: 2.0 s
memory limit per test: 512 MB
input: standard input
output: standard output

Dmitriy likes trees and does not like when things change fast. He even created a notion of tree unsmoothness. A tree consisting of n vertices indexed from 1 to n with each vertex i assigned some real value x_i has a value of *unsmoothness* equal to the largest of *roughness* values of its edges. A *roughness* value of an edge connecting vertices i and j is defined as $|x_i - x_j|$.

Dmitriy had a tree on a table that was pretty smooth. One day he returned to his table back from lunch and found out that his colleague Andrew erased some of the numbers x_i. Dmitriy does not remember the exact values of lost x_i, but he sees this as an opportunity to make his tree even more smooth!

Find out the minimum possible value of unsmoothness of his the tree that Dmitriy may achieve if he fills the erased values of x_i properly.

Input

The first line of input contains an integer n ($2 \le n \le 100\,000$), the number of vertices in the tree.

The second line of input contains n tokens $x_1, x_2, \ldots, x_n$ (either x_i is an integer and $-10^6 \le x_i \le 10^6$ or x_i = '*' which denotes the erased number).

The i-th of the following $n - 1$ lines contains two integers u_i, v_i ($1 \le u_i, v_i \le n, u_i \ne v_i$), indices of the vertices connected by the i-th edge.

Output

Output the only number, the minimum possible smoothness of the resulting tree after Dmitry fills all the erased values.

Your answer will be considered correct if its relative or absolute error does not exceed 10^{-6}.

Examples

input Copy

```
4
-1 4 * 2
1 3
4 3
3 2
```

output Copy

```
2.5
```

input Copy

```
3
* 2 *
3 1
2 1
```

output Copy

```
0
```

Note

In the first sample test the optimal way to fill the only erased value is to put 1.5 there. In such way the roughness of the edge $(1, 3)$ is 2.5, the roughness of the edge $(4, 3)$ is 0.5 and the roughness of the edge $(3, 2)$ is 2.5. Thus, the unsmoothness of the tree equals to 2.5.

In the second sample test the optimal way to fill the erase values is to put 2 for both vertices. In this way both edges have roughness of 0, and the unsmoothness of the tree also equals to 0.

Problem F
Binary Transformations

Input File: F.in
Output File: standard output
Time Limit: *1* second (C/C++)
Memory Limit: *256* megabytes

There are **n** bits. Each bit **i** has a value $\mathbf{a_i}$ (0 or 1) and an associated cost $\mathbf{c_i}$. We can change the value of bit **i** with a cost computed as the sum of all the costs $\mathbf{c_j}$ of the bits **j** such that $\mathbf{a_j = 1}$ **AFTER** bit **i** is changed. What is the minimum amount that should be paid to set each bit **i** to a specified value $\mathbf{b_i}$.

Input
The first line contains the integer **n** ($1 \le \mathbf{n} \le 5 \times 10^3$) - the number of bits
The second line contains *n* integers $\mathbf{c_i}$ ($1 \le \mathbf{c_i} \le 10^9$) - the costs associated with the bits
The third line contains the original **n** values of the bits $\mathbf{a_i}$ - the original values of the bits
The fourth line contains the required **n** values of the bits $\mathbf{b_i}$ - the required values of the bits

Output
Print one number - the minimum cost.

Sample input	Sample output

`5`
`5 2 6 1 5`
`01110`
`10011`
	`21`

G Max $\mathcal{B}$-Matching

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Max $\mathcal{B}$ is an expert in combinatorial optimization and bitwise operations. He recently read a paper about so-called b-matchings, but it appeared to be too complicated and unnatural for him, so he decided to create his own kind of matchings and to call it $\mathcal{B}$-matching. He came up with a following problem.

Recall that the bitwise AND of two non-negative integers x and y is such a number $x \,\&\, y$ that it has ones in a binary representation in exactly those positions, in which both x and y also have ones in a binary representation. Define a function $\mathcal{B}(z)$ to be equal to a largest power of two not exceeding z if z is positive and to zero if $z = 0$.

You are given a complete bipartite graph (i.e. such bipartite graph that any two vertices from different parts are connected with an edge). Vertices of the first part of the graph are numbered from 1 to n_1, vertices of the second part of the graph are numbered from 1 to n_2. The i-th vertex of the first part of the graph is assigned a non-negative integer x_i, the j-th vertex of the second part of the graph is assigned a non-negative integer y_j.

Recall that the matching in a graph is such a set M of edges that no two distinct edges $e_1, e_2 \in M$ share a common endpoint. Define a weight of an edge connecting vertices u and v (from the first part and the second part of a graph respectively) to be $\mathcal{B}(x_u \,\&\, y_v)$ (that is why it is valid to call such an object $\mathcal{B}$-matching in the future!)

Your task is to find a matching of a maximum possible total weight of edges.

Input

The first line of input contains two integers n_1 and n_2 ($1 \le n_1, n_2 \le 100\,000$).

The second line of input contains n_1 integers $x_1, x_2, \ldots, x_{n_1}$ ($0 \le x_i < 2^{20}$) that are assigned to the corresponding vertices of the first part of the graph.

The third line of input contains n_2 integers $y_1, y_2, \ldots, y_{n_2}$ ($0 \le y_i < 2^{20}$) that are assigned to the corresponding vertices of the second part of the graph.

Output

Output the maximum possible total weight of edges of a matching in a given graph.

Example

g. max $\mathcal{B}$-matching Copy

```
4 5
5 7 5 2
1 2 3 4 2
```

output Copy

```
9
```

Note

In the sample test the best possible matching consists of edges $(1, 1)$, $(2, 5)$, $(3, 4)$ and $(4, 3)$ (where the first number in pair denotes the index of a vertex of a first part, and the second number denotes the index of a vertex of a second part).