A. Prime Subtraction

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given two integers x and y (it is guaranteed that $x > y$). You may choose any *prime* integer p and subtract it any number of times from x. Is it possible to make x equal to y?

Recall that a *prime* number is a positive integer that has exactly two positive divisors: 1 and this integer itself. The sequence of prime numbers starts with 2, 3, 5, 7, 11.

Your program should solve t independent test cases.

Input

The first line contains one integer t ($1 \le t \le 1000$) — the number of test cases.

Then t lines follow, each describing a test case. Each line contains two integers x and y ($1 \le y < x \le 10^{18}$).

Output

For each test case, print YES if it is possible to choose a prime number p and subtract it any number of times from x so that x becomes equal to y. Otherwise, print NO.

You may print every letter in any case you want (so, for example, the strings yEs, yes, Yes, and YES will all be recognized as positive answer).

Example

input

```
4
100 98
42 32
1000000000000000000 1
41 40
```

output

```
YES
YES
YES
NO
```

Note

In the first test of the example you may choose $p = 2$ and subtract it once.

In the second test of the example you may choose $p = 5$ and subtract it twice. Note that you cannot choose $p = 7$, subtract it, then choose $p = 3$ and subtract it again.

In the third test of the example you may choose $p = 3$ and subtract it 333333333333333333 times.

B. Kill 'Em All

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Ivan plays an old action game called Heretic. He's stuck on one of the final levels of this game, so he needs some help with killing the monsters.

The main part of the level is a large corridor (so large and narrow that it can be represented as an infinite coordinate line). The corridor is divided into two parts; let's assume that the point $x = 0$ is where these parts meet.

The right part of the corridor is filled with n monsters — for each monster, its initial coordinate x_i is given (and since all monsters are in the right part, every x_i is positive).

The left part of the corridor is filled with crusher traps. If some monster enters the left part of the corridor or the origin (so, its current coordinate becomes **less than or equal** to 0), it gets instantly killed by a trap.

The main weapon Ivan uses to kill the monsters is the Phoenix Rod. It can launch a missile that explodes upon impact, obliterating every monster caught in the explosion and throwing all other monsters away from the epicenter. Formally, suppose that Ivan launches a missile so that it explodes in the point c. Then every monster is either killed by explosion or pushed away. Let some monster's current coordinate be y, then:

- if $c = y$, then the monster is killed;
- if $y < c$, then the monster is pushed r units to the left, so its current coordinate becomes $y - r$;
- if $y > c$, then the monster is pushed r units to the right, so its current coordinate becomes $y + r$.

Ivan is going to kill the monsters as follows: choose some integer point d and launch a missile into that point, then wait until it explodes and all the monsters which are pushed to the left part of the corridor are killed by crusher traps, then, if at least one monster is still alive, choose another integer point (probably the one that was already used) and launch a missile there, and so on.

What is the minimum number of missiles Ivan has to launch in order to kill all of the monsters? You may assume that every time Ivan fires the Phoenix Rod, he chooses the impact point optimally.

You have to answer q independent queries.

Input

The first line contains one integer q ($1 \le q \le 10^5$) — the number of queries.

The first line of each query contains two integers n and r ($1 \le n, r \le 10^5$) — the number of enemies and the distance that the enemies are thrown away from the epicenter of the explosion.

The second line of each query contains n integers x_i ($1 \le x_i \le 10^5$) — the initial positions of the monsters.

It is guaranteed that sum of all n over all queries does not exceed 10^5.

Output

For each query print one integer — the minimum number of shots from the Phoenix Rod required to kill all monsters.

Example

input

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

output

```
2
2
```

Note

In the first test case, Ivan acts as follows:

- choose the point 3, the first monster dies from a crusher trap at the point -1, the second monster dies from the explosion, the third monster is pushed to the point 7;
- choose the point 7, the third monster dies from the explosion.

In the second test case, Ivan acts as follows:

- choose the point 5, the first and fourth monsters die from the explosion, the second monster is pushed to the point 1, the third monster is pushed to the point 2;
- choose the point 2, the first monster dies from a crusher trap at the point 0, the second monster dies from the explosion.

C. Standard Free2play

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are playing a game where your character should overcome different obstacles. The current problem is to come down from a cliff. The cliff has height h, and there is a moving platform on each height x from 1 to h.

Each platform is either hidden inside the cliff or moved out. At first, there are n moved out platforms on heights $p_1, p_2, \dots, p_n$. The platform on height h is moved out (and the character is initially standing there).

If you character is standing on some moved out platform on height x, then he can pull a special lever, which switches the state of **two platforms: on height x and $x - 1$**. In other words, the platform you are currently standing on will hide in the cliff and the platform one unit below will change it state: it will hide if it was moved out or move out if it was hidden. In the second case, you will safely land on it. **Note that this is the only way to move from one platform to another**.

Your character is quite fragile, so it can safely fall from the height no more than 2. In other words falling from the platform x to platform $x - 2$ is okay, but falling from x to $x - 3$ (or lower) is certain death.

Sometimes it's not possible to come down from the cliff, but you can always buy (for donate currency) several magic crystals. Each magic crystal can be used to change the state of any single platform (except platform on height h, which is unaffected by the crystals). After being used, the crystal disappears.

What is the minimum number of magic crystal you need to buy to safely land on the 0 ground level?

Input

The first line contains one integer q ($1 \le q \le 100$) — the number of queries. Each query contains two lines and is independent of all other queries.

The first line of each query contains two integers h and n ($1 \le h \le 10^9$, $1 \le n \le \min(h, 2 \cdot 10^5)$) — the height of the cliff and the number of moved out platforms.

The second line contains n integers $p_1, p_2, \dots, p_n$ ($h = p_1 > p_2 > \dots > p_n \ge 1$) — the corresponding moved out platforms in the descending order of their heights.

The sum of n over all queries does not exceed $2 \cdot 10^5$.

Output

For each query print one integer — the minimum number of magic crystals you have to spend to safely come down on the ground level (with height 0).

Example

input

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

output

```
0
1
2
0
```

D. AB-string

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The string $t_1 t_2 \dots t_k$ is good if each letter of this string belongs to at least one palindrome of length **greater** than 1.

A palindrome is a string that reads the same backward as forward. For example, the strings A, BAB, ABBA, BAABBBAAB are palindromes, but the strings AB, ABBBAA, BBBA are not.

Here are some examples of good strings:

- t = AABBB (letters t_1, t_2 belong to palindrome $t_1 \dots t_2$ and letters t_3, t_4, t_5 belong to palindrome $t_3 \dots t_5$);
- t = ABAA (letters t_1, t_2, t_3 belong to palindrome $t_1 \dots t_3$ and letter t_4 belongs to palindrome $t_3 \dots t_4$);
- t = AAAAA (all letters belong to palindrome $t_1 \dots t_5$);

You are given a string s of length n, consisting of **only** letters A and B.

You have to calculate the number of good substrings of string s.

Input

The first line contains one integer n ($1 \le n \le 3 \cdot 10^5$) — the length of the string s.

The second line contains the string s, consisting of letters A and B.

Output

Print one integer — the number of good substrings of string s.

Examples

input

```
5
AABBB
```

output

```
6
```

input

```
3
```

```
AAA
```

output

```
3
```

input

```
7
AAABABB
```

output

```
15
```

Note

In the first test case there are six good substrings: $s_1 \dots s_2$, $s_1 \dots s_4$, $s_1 \dots s_5$, $s_3 \dots s_4$, $s_3 \dots s_5$ and $s_4 \dots s_5$.

In the second test case there are three good substrings: $s_1 \dots s_2$, $s_1 \dots s_3$ and $s_2 \dots s_3$.

E. Keyboard Purchase

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have a password which you often type — a string s of length n. Every character of this string is one of the first m lowercase Latin letters.

Since you spend a lot of time typing it, you want to buy a new keyboard.

A keyboard is a permutation of the first m Latin letters. For example, if $m = 3$, then there are six possible keyboards: abc, acb, bac, bca, cab and cba.

Since you type your password with one finger, you need to spend time moving your finger from one password character to the next. The time to move from character s_i to character s_{i+1} is equal to the distance between these characters on keyboard. The total time you have to spend typing the password with a keyboard is called the *slowness* of this keyboard.

More formaly, the slowness of keyboard is equal to $\sum_{i=2}^{n} |pos_{s_{i-1}} - pos_{s_i}|$, where pos_x is position of letter x in keyboard.

For example, if s is aacabc and the keyboard is bac, then the total time of typing this password is $|pos_a - pos_a| + |pos_a - pos_c| + |pos_c - pos_a| + |pos_a - pos_b| + |pos_b - pos_c|$ = $|2-2| + |2-3| + |3-2| + |2-1| + |1-3|$ = $0 + 1 + 1 + 1 + 2 = 5$.

Before buying a new keyboard you want to know the minimum possible slowness that the keyboard can have.

Input

The first line contains two integers n and m ($1 \le n \le 10^5, 1 \le m \le 20$).

The second line contains the string s consisting of n characters. Each character is one of the first m Latin letters (lowercase).

Output

Print one integer – the minimum slowness a keyboard can have.

Examples

input

```
6 3
aacabc
```

output

```
5
```

input

```
6 4
aaaaaa
```

output

```
0
```

input

```
15 4
abacabadabacaba
```

output

```
16
```

Note

The first test case is considered in the statement.

In the second test case the slowness of any keyboard is 0.

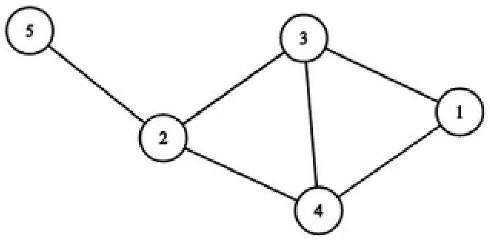
In the third test case one of the most suitable keyboards is bacd.

F. The Maximum Subtree

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Assume that you have k one-dimensional segments $s_1, s_2, \dots s_k$ (each segment is denoted by two integers — its endpoints). Then you can build the following graph on these segments. The graph consists of k vertexes, and there is an edge between the i-th and the j-th vertexes ($i \neq j$) if and only if the segments s_i and s_j intersect (there exists at least one point that belongs to both of them).

For example, if $s_1 = [1, 6], s_2 = [8, 20], s_3 = [4, 10], s_4 = [2, 13], s_5 = [17, 18]$, then the resulting graph is the following:



A tree of size m is good if it is possible to choose m one-dimensional segments so that the graph built on these segments coincides with this tree.

You are given a tree, you have to find its good subtree with maximum possible size. Recall that a subtree is a connected subgraph of a tree.

Note that you have to answer q independent queries.

Input

The first line contains one integer q ($1 \le q \le 15 \cdot 10^4$) — the number of the queries.

The first line of each query contains one integer n ($2 \le n \le 3 \cdot 10^5$) — the number of vertices in the tree.

Each of the next $n - 1$ lines contains two integers x and y ($1 \le x, y \le n$) denoting an edge between vertices x and y. It is guaranteed that the given graph is a tree.

It is guaranteed that the sum of all n does not exceed $3 \cdot 10^5$.

Output

For each query print one integer — the maximum size of a good subtree of the given tree.

Example

input

```
1
10
1 2
1 3
1 4
2 5
2 6
3 7
3 8
4 9
4 10
```

output

```
8
```

Note

In the first query there is a good subtree of size 8. The vertices belonging to this subtree are $9, 4, 10, 2, 5, 1, 6, 3$.

G. Adilbek and the Watering System

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Adilbek has to water his garden. He is going to do it with the help of a complex watering system: he only has to deliver water to it, and the mechanisms will do all the remaining job.

The watering system consumes one liter of water per minute (if there is no water, it is not working). It can hold no more than c liters. Adilbek has already poured c_0 liters of water into the system. He is going to start watering the garden right now and water it for m minutes, and the watering system should contain at least one liter of water at the beginning of the i-th minute (for every i from 0 to $m-1$).

Now Adilbek wonders what he will do if the watering system runs out of water. He called n his friends and asked them if they are going to bring some water. The i-th friend answered that he can bring no more than a_i liters of water; he will arrive at the beginning of the t_i-th minute and pour all the water he has into the system (if the system cannot hold such amount of water, the excess water is poured out); and then he will ask Adilbek to pay b_i dollars for each liter of water he has brought. You may assume that if a friend arrives at the beginning of the t_i-th minute and the system runs out of water at the beginning of the same minute, the friend pours his water fast enough so that the system does not stop working.

Of course, Adilbek does not want to pay his friends, but he has to water the garden. So he has to tell his friends how much water should they bring. Formally, Adilbek wants to choose n integers k_1, k_2, ..., k_n in such a way that:

- if each friend i brings exactly k_i liters of water, then the watering system works during the whole time required to water the garden;
- the sum $\sum\limits_{i=1}^{n} k_i b_i$ is minimum possible.

Help Adilbek to determine the minimum amount he has to pay his friends or determine that Adilbek not able to water the garden for m minutes.

You have to answer q independent queries.

Input

The first line contains one integer q ($1 \le q \le 5 \cdot 10^5$) – the number of queries.

The first line of each query contains four integers n, m, c and c_0 ($0 \le n \le 5 \cdot 10^5, 2 \le m \le 10^9, 1 \le c_0 \le c \le 10^9$) — the number of friends, the number of minutes of watering, the capacity of the watering system and the number of liters poured by Adilbek.

Each of the next n lines contains three integers t_i, a_i, b_i ($0 < t_i < m, 1 \le a_i \le c, 1 \le b_i \le 10^9$) — the i-th friend's arrival time, the maximum amount of water i-th friend can bring and the cost of 1 liter from i-th friend.

It is guaranteed that sum of all n over all queries does not exceed $5 \cdot 10^5$.

Output

For each query print one integer — the minimum amount Adilbek has to pay his friends, or -1 if Adilbek is not able to water the garden for m minutes.

Example

input

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

output

```
6
0
-1
4
```

Codeforces (c) Copyright 2010-2022 Mike Mirzayanov
The only programming contests Web 2.0 platform