A. Moving Chips

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 512 megabytes

There is a ribbon divided into n cells, numbered from 1 to n from left to right. Each cell either contains a chip or is free.

You can perform the following operation any number of times (possibly zero): choose a chip and move it to the **closest free cell to the left**. You can choose any chip that you want, provided that there is at least one free cell to the left of it. When you move the chip, the cell where it was before the operation becomes free.

Your goal is to move the chips in such a way that **they form a single block, without any free cells between them**. What is the minimum number of operations you have to perform?

Input

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

Each test case consists of two lines:

- the first line contains one integer n ($2 \le n \le 50$) — the number of cells;
- the second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 1$); $a_i = 0$ means that the i-th cell is free; $a_i = 1$ means that the i-th cell contains a chip.

Additional constraint on the input: in each test case, at least one cell contains a chip.

Output

For each test case, print one integer — the minimum number of operations you have to perform so that all chips form a single block without any free cells between them.

Standard Input	Standard Output
5 8 0 1 1 1 0 1 1 0 6 0 1 0 0 0 0 6 1 1 1 1 1 1 5 1 0 1 0 1 9 0 1 1 0 0 0 1 1 0	1 0 0 2 3

Note

In the first example, you can perform the operation on the chip in the 7-th cell. The closest free cell to the left is the 5-th cell, so it moves there. After that, all chips form a single block.

In the second example, all chips are already in a single block. Same for the third example.

,

B. Monsters Attack!

Input file: standard input
Output file: standard output
Time limit: 2.5 seconds
Memory limit: 256 megabytes

You are playing a computer game. The current level of this game can be modeled as a straight line. Your character is in point 0 of this line. There are n monsters trying to kill your character; the i-th monster has health equal to a_i and is initially in the point x_i.

Every second, the following happens:

- first, you fire up to k bullets at monsters. Each bullet targets exactly one monster and decreases its health by 1. For each bullet, you choose its target arbitrary (for example, you can fire all bullets at one monster, fire all bullets at different monsters, or choose any other combination). Any monster can be targeted by a bullet, regardless of its position and any other factors;
- then, all alive monsters with health 0 or less die;
- then, all alive monsters move 1 point closer to you (monsters to the left of you increase their coordinates by 1, monsters to the right of you decrease their coordinates by 1). If any monster reaches your character (moves to the point 0), you lose.

Can you survive and kill all n monsters without letting any of them reach your character?

Input

The first line of the input contains one integer t ($1 \le t \le 3 \cdot 10^4$) — the number of test cases.

Each test case consists of three lines:

- the first line contains two integers n and k ($1 \le n \le 3 \cdot 10^5$; $1 \le k \le 2 \cdot 10^9$);
- the second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$);
- the third line contains n integers $x_1, x_2, \ldots, x_n$ ($-n \le x_1 < x_2 < x_3 < \cdots < x_n \le n$; $x_i \neq 0$).

Additional constraint on the input: the sum of n over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, print YES if you can kill all n monsters before they reach your character, or NO otherwise.

You can output each letter of the answer in any case (upper or lower). For example, the strings yEs, yes, Yes, and YES will all be recognized as positive responses.

Standard Input	Standard Output
5 3 2 1 2 3 -1 2 3 2 1 1 1 -1 1 4 10 3 4 2 5 -3 -2 1 3	YES NO YES YES NO

5 3 2 1 3 2 5 -3 -2 3 4 5 2 1 1 2 1 2	

Note

In the first example, you can act as follows:

- during the 1-st second, fire 1 bullet at the 1-st monster and 1 bullet at the 3-rd monster. Then the 1-st monster dies, the 2-nd and the 3-rd monster move closer;
- during the 2-nd second, fire 2 bullets at the 2-nd monster. Then the 2-nd monster dies, the 3-rd monster moves closer;
- during the 3-rd second, fire 2 bullets at the 3-rd monster. Then the 3-rd monster dies.

In the second example, you can fire only 1 bullet, so you can kill only one of the two monsters during the 1-st second. Then, the remaining monster moves closer and kills your character.

,

C. Find B

Input file: standard input
Output file: standard output
Time limit: 3 seconds
Memory limit: 256 megabytes

An array a of length m is considered good if there exists an integer array b of length m such that the following conditions hold:

1. $\sum_{i=1}^{m} a_i = \sum_{i=1}^{m} b_i$;
2. $a_i \neq b_i$ for every index i from 1 to m;
3. $b_i > 0$ for every index i from 1 to m.

You are given an array c of length n. Each element of this array is greater than 0.

You have to answer q queries. During the i-th query, you have to determine whether the subarray $c_{l_i}, c_{l_i+1}, \ldots, c_{r_i}$ is good.

Input

The first line contains one integer t ($1 \le t \le 10^4$) — the number of test cases.

The first line of each test case contains two integers n and q ($1 \le n, q \le 3 \cdot 10^5$) — the length of the array c and the number of queries.

The second line of each test case contains n integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le 10^9$).

Then q lines follow. The i-th of them contains two integers l_i and r_i ($1 \le l_i \le r_i \le n$) — the borders of the i-th subarray.

Additional constraints on the input: the sum of n over all test cases does not exceed $3 \cdot 10^5$; the sum of q over all test cases does not exceed $3 \cdot 10^5$.

Output

For each query, print YES if the subarray is good. Otherwise, print NO.

You can output each letter of the answer in any case (upper or lower). For example, the strings yEs, yes, Yes, and YES will all be recognized as positive responses.

Standard Input	Standard Output
1 5 4 1 2 1 4 5 1 5 4 4 3 4 1 3	YES NO YES NO

,

D. Slimes

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

There are n slimes placed in a line. The slimes are numbered from 1 to n in order from left to right. The size of the i-th slime is a_i.

Every second, the following happens: **exactly one** slime eats one of its neighbors and increases its size by the eaten neighbor's size. A slime can eat its neighbor only if it is strictly bigger than this neighbor. If there is no slime which is strictly bigger than one of its neighbors, the process ends.

For example, suppose $n = 5$, $a = [2, 2, 3, 1, 4]$. The process can go as follows:

- first, the 3-rd slime eats the 2-nd slime. The size of the 3-rd slime becomes 5, the 2-nd slime is eaten.
- then, the 3-rd slime eats the 1-st slime (they are neighbors since the 2-nd slime is already eaten). The size of the 3-rd slime becomes 7, the 1-st slime is eaten.
- then, the 5-th slime eats the 4-th slime. The size of the 5-th slime becomes 5, the 4-th slime is eaten.
- then, the 3-rd slime eats the 5-th slime (they are neighbors since the 4-th slime is already eaten). The size of the 3-rd slime becomes 12, the 5-th slime is eaten.

For each slime, calculate the minimum number of seconds it takes for this slime to be eaten by another slime (among all possible ways the process can go), or report that it is impossible.

Input

The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases.

The first line of each test case contains a single integer n ($1 \le n \le 3 \cdot 10^5$) — the number of slimes.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$).

The sum of n over all test cases doesn't exceed $3 \cdot 10^5$.

Output

For each test case, print n integers. The i-th integer should be equal to the minimum number of seconds it takes for the i-th slime to be eaten by another slime or -1 if it is impossible.

Standard Input	Standard Output
4 4 3 2 4 2 3 1 2 3 5 2 2 3 1 1 7 4 2 3 6 1 1 8	2 1 2 1 1 1 -1 2 1 -1 1 2 2 1 1 3 1 1 4

,

E. Count Paths

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 512 megabytes

You are given a tree, consisting of n vertices, numbered from 1 to n. Every vertex is colored in some color, denoted by an integer from 1 to n.

A simple path of the tree is called *beautiful* if:

- it consists of at least 2 vertices;
- the first and the last vertices of the path have the same color;
- no other vertex on the path has the same color as the first vertex.

Count the number of the *beautiful* simple paths of the tree. Note that paths are considered undirected (i. e. the path from x to y is the same as the path from y to x).

Input

The first line contains a single integer t ($1 \le t \le 10^4$) — the number of testcases.

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

The second line contains n integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le n$) — the color of each vertex.

The i-th of the next $n - 1$ lines contains two integers v_i and u_i ($1 \le v_i, u_i \le n$; $v_i \neq u_i$) — the i-th edge of the tree.

The given edges form a valid tree. The sum of n over all testcases doesn't exceed $2 \cdot 10^5$.

Output

For each testcase, print a single integer — the number of the *beautiful* simple paths of the tree.

Standard Input	Standard Output
4 3 1 2 1 1 2 2 3 5 2 1 2 1 2 1 2 1 3 3 4 4 5 5 1 2 3 4 5 1 2 1 3 3 4	1 3 0 3

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

,

F. Shrink-Reverse

Input file: standard input
Output file: standard output
Time limit: 3 seconds
Memory limit: 256 megabytes

You are given a binary string s of length n (a string consisting of n characters, and each character is either `0` or `1`).

Let's look at s as at a binary representation of some integer, and name that integer as the *value* of string s. For example, the value of `000` is 0, the value of `01101` is 13, "`100000`" is 32 and so on.

You can perform at most k operations on s. Each operation should have one of the two following types:

- SWAP: choose two indices $i < j$ in s and swap s_i with s_j;
- SHRINK-REVERSE: delete all leading zeroes from s and reverse s.

For example, after you perform SHRINK-REVERSE on `000101100`, you'll get `001101`.
What is the minimum value of s you can achieve by performing at most k operations on s?

Input

The first line contains two integers n and k ($2 \le n \le 5 \cdot 10^5$; $1 \le k \le n$) — the length of the string s and the maximum number of operations.

The second line contains the string s of length n consisting of characters `0` and/or `1`.

Additional constraint on the input: s contains **at least one** `1`.

Output

Print a single integer — the minimum value of s you can achieve using no more than k operations. Since the answer may be too large, print it modulo $10^9 + 7$.

Note that you need to minimize the original value, not the remainder.

Standard Input	Standard Output
8 2 10010010	7
8 2 01101000	7
30 30 111111111111111111111111111111	73741816
14 1 10110001111100	3197

Note

In the first example, one of the optimal strategies is the following:

1. $\underline{1}0010\underline{0}10 \xrightarrow{\text{SWAP}} 00010110$;

2. $\texttt{000}\underline{\texttt{1}}\texttt{011}\underline{\texttt{0}} \xrightarrow{\text{SWAP}} \texttt{00000111}$.

The value of $\texttt{00000111}$ is 7.

In the second example, one of the optimal strategies is the following:

1. $\texttt{01101000} \xrightarrow{\text{SHRINK}} \texttt{1101000} \xrightarrow{\text{REVERSE}} \texttt{0001011}$;
2. $\texttt{000}\underline{\texttt{10}}\texttt{11} \xrightarrow{\text{SWAP}} \texttt{0000111}$.

The value of $\texttt{0000111}$ is 7.