Problem A. Divisor Difference

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Given a positive integer n. Over all positive integers x, y satisfying

$$xy = n$$

find the maximum possible value of $|x - y|$ (where $|a|$ denotes the absolute value - see notes for more clarification).

Input

The first line of input contains t $(1 \le t \le 2 \cdot 10^5)$ the number of test cases. The description of each test case follows.

Each test case contains one line of one integer n $(1 \le n \le 10^9)$, the integer described in the problem.

Output

For each test case output one line containing one integer, the maximum value of $|x - y|$ as described in the problem.

Example

standard input	standard output
2 4 5	3 4

Note

The absolute value of an integer a - denoted $|a|$ - is either a if a is non-negative, or $-a$ if a is negative.

Problem B1. Longest Common Suffix

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

A string t is a *suffix* of a string s if and only if t is obtainable by deleting several (possibly none) characters from the **front** of s.

For example, the string `cde` is a suffix of the string `abcde` since we can delete `a` then delete `b`. However, `bcd` is not a suffix of `abcde`.

You are given two strings a and b of lengths n and m respectively, find the length of the longest common suffix (that is the length of the longest string that is a suffix of both strings)

Input

The first line of input contains an 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 m ($1 \le n, m \le 10^5$), the lengths of the two strings a and b respectively.

The second line of each test case contains the string a of n lowercase Latin letters.

The third line of each test case contains the string b of m lowercase Latin letters.

The sum of $n + m$ over all test cases doesn't exceed 10^6.

Output

For each test case, output one line containing one integer, the length of the longest common suffix of a and b.

Example

standard input	standard output
3 3 4 beh efeh 5 4 aaaaa aaaa 5 5 fjekl mesth	2 4 0

Note

In the first test case, the longest common suffix is `eh`.

In the second test case, the longest common suffix is `aaaa`.

In the third test case, there is no common suffix, so the answer is 0.

Problem B2. Mina and Ayman

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

Mina was doing some research and during his research, he wrote down an array of non-negative integers $a_1, a_2, \ldots, a_n$, and he calculated the value

$$x = a_1 \oplus a_2 \oplus \cdots \oplus a_n$$

and wrote down the number x, where $\oplus$ denotes the "Bitwise XOR" (look at notes for more clarification).

However, his evil and distracting friend Ayman was teasing him, so he took **exactly one** index j and erased a_j, and Mina didn't remember it, but the rest of the array and the number x are still written down. Help Mina recover the value of a_j, so that he can proceed with his research.

Input

The first line of input starts with an 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 x ($2 \le n \le 10^5$, $0 \le x < 2^{30}$, the size of the array and the number Mina wrote down).

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i < 2^{30}$, and there is exactly one j such that $a_j = -1$, the integer that Ayman erased).

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

Output

For each test case, output one line containing one integer, the value of a_j, where j is the index such that $a_j = -1$ in the input.

It is guaranteed that a_j satisfies $0 \le a_j < 2^{30}$, and it can be proven that there is only one unique integer a_j satisfying the conditions given in the problem.

Example

standard input	standard output
4 4 3 4 3 -1 1 5 6 7 1 -1 6 3 4 7 -1 1 2 4 2 0 1 -1	5 5 0 1

Note

In the first test case, we note that $4 \oplus 3 \oplus 5 \oplus 1 = 3$, which is the given x.

The XOR of two bits x and y is 1 if and only if $x \ne y$. For example 1 XOR 0 is 1 and 1 XOR 1 is 0. The bitwise XOR of any two integers x and y can be found by converting each integer into binary and taking the XOR of each two corresponding bits. For example, $5 \oplus 3 = 6$ since 3 in binary is `011` and 5 in binary is `101`, so the result is `110` which is 6.

Problem C1. Sets and Integers

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

For any two non-negative integers x and y.

We say that x is a *supermask* of y if and only if $(x \mid y) = x$ (where $\mid$ denotes the bitwise OR operation - see notes for more clarification).

You are given an array of n non-negative integers $a_1, a_2, \ldots, a_n$. In one operation, you can choose two elements and replace them with any number that is a supermask of both elements. More formally, choose any two indices i, j such that $1 \le i < j \le n$, and replace a_i and a_j with an integer x where x is a supermask of both a_i and a_j.

You keep doing this operation until the array contains one element (that is, do it $n-1$ times). Find the minimum possible value of such element.

Input

The first line of input starts with an integer t ($1 \le t \le 10^4$), the number of test cases.

The first line of each test case contains an integer n ($2 \le n \le 10^5$), the size of the array.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i < 2^{30}$).

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

Output

For each test case, output one line containing one integer, the minimum possible value achievable by doing the described operation $n-1$ times.

Example

<table>
<tr><th>standard input</th><th>standard output</th></tr>
<tr><td><pre>2
4
5 4 2 6
5
1 2 4 8 16</pre></td><td><pre>7
31</pre></td></tr>
</table>

Note

In the first test case, you can replace 5 and 6 with 7, since $5 =$ `101` and $6 =$ `110` and $7 =$ `111` which is a supermask of both since $(5|7) = 7$ and $(6|7) = 7$. So, the array becomes $7, 4, 2$. Afterwards, you can replace 7 and 4 with 7, and 7 and 2 with 7. It can be proven that this is the minimum possible value achievable.

In the second test case, you can replace 1 and 16 with 17, and 2 and 4 with 6 making the array $6, 8, 17$. Afterwards you can replace 6 and 17 with 23 and 23 and 8 with 31. It can be proven that this is the minimum possible value achievable.

The OR of two bits x and y is 0 if and only if both x and y are 0. So, for example, 1 OR 0 is 1.

The bitwise OR of any two integers x and y can be found by converting each integer into binary and taking the OR of each two corresponding bits. For example, $(5|3) = 7$ since 3 in binary is `011` and 5 in binary is `101`, so the result is `111` which is 7.

Problem C2. Flipping Cards

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

Proofy with playing with n cards numbered from 1 to n. The ith card has the value a_i on its front face and the value b_i at its back face.

Now, Proofy places the cards on the table with their front faces up. He can flip **at most** k cards (choose at most k cards and make their back faces up). Afterward, he looks at the table from above and calculates the sum of values appearing face-up on the cards.

Find the maximum sum that Proofy can achieve.

Input

The first line of input contains an 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 k ($1 \le k \le n \le 10^5$), the number of cards, and the maximum number of cards that can be flipped.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$), the values displayed on the front faces of the cards.

The third line of each test case contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 10^9$), the values displayed on the back faces of the cards.

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

Output

For each test case, output one line containing one integer, the maximum sum Proofy can achieve by flipping at most k cards.

Please note, that the answer for some test cases won't fit into 32-bit integer type, so you should use at least 64-bit integer type in your programming language (like `long long` for `C++`).

Scoring

Group 1: (10 points)

$1 \le k \le n \le 10$.

Group 2: (20 points)

$1 \le k \le n \le 2000$, the sum of n over all test cases doesn't exceed 10000.

Group 3: (70 points)

No additional constraints.

Example

standard input	standard output
3 4 4 1 2 3 4 5 6 7 8 4 3 5 6 7 8 1 2 3 4 5 3 8 7 1 5 3 9 8 6 9 5	26 26 35

Note

In the first test case, Proofy can - and it will be optimal - to flip all cards; so, the answer will be $5 + 6 + 7 + 8 = 26$.

In the second test case, Proofy - while he can flip **at most** 3 - will not need to flip any cards; so, the answer will be $5 + 6 + 7 + 8 = 26$.

In the third test case, it is optimal to flip the last three cards (since he can flip at most 3). So, the answer will be $8 + 7 + 6 + 9 + 5 = 35$.

Problem C3. Nested Sum (Easy Version)

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

Given an array of n integers $a_1, a_2, \ldots, a_n$, find the value of

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} a_i a_j$$

Input

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

The first line of each test case contains an integer n ($1 \le n \le 10^5$), the size of the array.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^4$), the elements of the array.

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

Output

For each test case output one line containing one integer, the value of the sum described in the problem.

Please note, that the answer for some test cases won't fit into 32-bit integer type, so you should use at least 64-bit integer type in your programming language (like `long long` for `C++`).

Scoring

Group 1: (30 points)

$1 \le n \le 2000$, the sum of n over all test cases doesn't exceed 10^4.

Group 2: (70 points)

No additional constraints.

Example

standard input	standard output
3 3 1 1 1 4 1 2 3 4 5 5 4 3 1 2	3 35 85

Problem C4. Polynomial Convolution

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

Given two arrays $a_0, a_1, \ldots, a_{n-1}$ and $b_0, b_1, \ldots, b_{m-1}$, and an integer k, where $k < n + m - 1$. Find the coefficient of x^k in the expansion of

$$(a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1})(b_0 + b_1x + b_2x^2 + \cdots + b_{m-1}x^{m-1})$$

Input

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

The first line of each test case contains three integers n, m, k, ($1 \le n, m \le 10^5$, $0 \le k < n + m - 1$)

The second line of each test case contains n integers $a_0, a_1, \ldots, a_{n-1}$ ($1 \le a_i \le 100$).

The third line of each test case contains m integers $b_0, b_1, \ldots, b_{m-1}$ ($1 \le b_i \le 100$).

The sum of $n + m$ over all test cases doesn't exceed 10^6.

Output

For each test case output one line containing one integer, the coefficient of x^k, as described in the problem.

Scoring

Group 1: (30 points)

$1 \le n, m \le 1000$, the sum of $n + m$ over all test cases doesn't exceed 2000.

Group 2: (70 points)

No additional constraints.

Example

standard input	standard output
3 2 2 1 1 1 1 1 3 4 2 2 5 7 3 4 1 3 5 5 5 2 1 3 4 2 3 2 4 2 5	2 43 31

Note

In the first test case, we note that the two given arrays correspond to the multiplication

$$(1 + x)(1 + x) = (1 + x)^2 = 1 + 2x + x^2$$

and since $k = 1$, the coefficient of x^k is the coefficient of x which is 2.

In the second test case, the two given arrays correspond to the multiplication

$$(2 + 5x + 7x^2)(3 + 4x + x^2 + 3x^3) = 6 + 23x + 43x^2 + 39x^3 + 22x^4 + 21x^5$$

and the coefficient of x^2 is 43.

Problem D1. Looks Divisible To Me

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	256 megabytes

Given a positive integer n, we say that a positive integer x is n-mersenne if and only if $2^x - 1$ divides $2^n - 1$.

Find all n-mersenne positive integers.

Input

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

Each test case contains one line of one integer n ($1 \le n \le 10^9$), the integer described in the problem.

Output

For each test case containing an integer n, let $x_1, x_2, \ldots, x_k$ (for some positive integer k) be all distinct n-mersenne positive integers where $x_1 < x_2 < \cdots < x_k$. Output $x_1, x_2, \ldots, x_k$ space-separated **in ascending order**.

It is guaranteed that the sum of k over all test cases would not exceed $2 \cdot 10^6$.

Example

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

Note

In the first test case, we note that $2^1 - 1 = 1$, and only 1-mersenne number if 1 itself.

In the second test case, we note that $2^4 - 1 = 15$, and the divisors of 15 are $1, 3, 5, 15$, making the 4-mersenne numbers only $1, 2$, and 4, since $2^1 - 1 = 1$, $2^2 - 1 = 3$, and $2^4 - 1 = 15$, all of which are divisors of 15.

In the third test case, we note that $2^6 - 1 = 63$, and the divisors of 63 are $1, 3, 7, 9, 21, 63$, and of those, the 6-mersenne numbers are only

- 1 since $2^1 - 1 = 1$
- 2 since $2^2 - 1 = 3$
- 3 since $2^3 - 1 = 7$
- 6 since $2^6 - 1 = 63$

Problem D2. Nested Sum (Hard Version)

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

Given an array of n positive integers $a_1, a_2, \ldots, a_n$ find the value of

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=j+1}^{n} a_i a_j a_k$$

modulo $10^9 + 7$.

Input

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

The first line of each test case contains an integer n ($1 \le n \le 10^5$), the size of the array.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$), the elements of the array.

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

Output

For each test case output one line containing one integer, the sum described in the problem modulo $10^9 + 7$.

Scoring

Group 1: (30 points)

$1 \le n \le 500$, the sum of n over all test cases doesn't exceed 1000.

Group 2: (70 points)

No additional constraints.

Example

<table>
<tr><th>standard input</th><th>standard output</th></tr>
<tr><td><pre>3
3
1 1 1
4
1 2 3 4
5
5 4 3 1 2</pre></td><td><pre>1
50
225</pre></td></tr>
</table>

Problem D3. Rotating Strings

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

A cyclic shift of size k of the string

$$s_0, s_1, \ldots, s_{n-1}$$

is the string

$$s_{n-k}, s_{n-k+1}, \ldots, s_{n-1}, s_0, s_1, \ldots, s_{n-k-1}$$

More formally, it is the string t where $t_i = s_{(i-k+n) \bmod n}$.

For example, a cyclic shift of size 2 of the string `abbcd` is the string `cdabb`, and the cyclic shift of size 4 of the string `abcdefgh` is the string `efghabcd`.

We say that a string t is k-*indifferent* if and only if the cyclic shift of size k of t is t. That is, if we perform a cyclic shift of size k, we will obtain the same string.

You are given a string s of size n, find the maximum non-negative integer r such that $r < n$, and you can **reorder the letters** (or possibly keep the string the same) of s to make it r-indifferent.

Input

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

The first line of each test case consists of one integer n ($1 \le n \le 10^5$), the length of the string.

The second line of each test case consists of a string s of n lowercase Latin letters.

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

Output

For each test case, output one line containing one integer, the maximum non-negative integer $r < n$ such that s can be reordered to be r-indifferent.

Example

standard input	standard output
3 5 abcde 12 aaaabbbbcccc 12 aaaaaabbccbb	0 9 6

Note

In the first test case, there is no way to reorder the string to make it r-indifferent for any positive r, and so the answer is 0.

In the second test case, we can reorder the letters of the string to make it `abcabcabcabc`, and so the answer will be 9, since when shifting it by 9 letters it remains the same. It can be shown that this is the maximum possible.

In the third test case, we can reorder the letters to make it `aaabbcaaabbc`, and so the answer will be 6. It can be shown that this is the maximum possible.

Problem D4. The Dilworth Tree

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

You are given a tree consisting of n vertices numbered from 1 to n, *rooted at* 1 (see notes for more clarification on the convention of rooted trees).

We say that a vertex u is an *ancestor* of vertex v if u exists on the simple (shortest) path from v to the root (which is the vertex numbered 1).

We say that a set of vertices is *dilworthy* if no vertex is an ancestor of the other. More formally, a set S is *dilworthy* if and only if for any $u, v \in S$ where $u \neq v$, we have that u is NOT an ancestor of v.

Find the maximum size of a dilworthy set in the given tree, and find the number of such sets modulo $10^9 + 7$. (Two sets of vertices are different if there exists one vertex in one of them that is not in the other).

Input

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

The first line of each test case contains an integer n ($2 \leq n \leq 10^5$), the number of vertices in the tree.

The second line of each test case contains $n - 1$ integers $p_2, p_3, \ldots, p_n$ ($1 \leq p_i < i$), where p_i is the parent of the ith vertex.

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

Output

For each test case, output one line containing two integers, the maximum size of a dilworthy set, and the number of dilworthy sets of maximum size modulo $10^9 + 7$.

Example

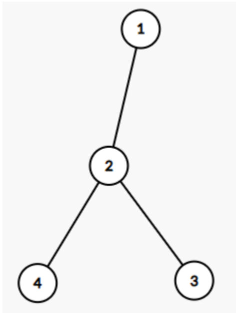
<table>
<tr><th>standard input</th><th>standard output</th></tr>
<tr><td><pre>4
4
1 2 2
5
1 2 1 4
10
1 1 2 2 3 3 7 8 6
4
1 2 3</pre></td><td><pre>2 1
2 4
4 6
1 4</pre></td></tr>
</table>

Note

A *tree* is a connected undirected graph without cycles. A *rooted tree* is a tree with a selected vertex, which is called the *root*.

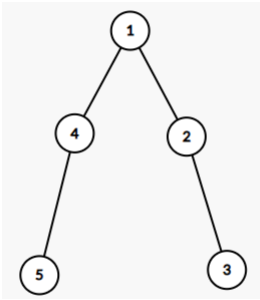
The tree is specified by an array of parents p containing n numbers: p_i is a parent of the vertex with the index i. The *parent* of a vertex u is a vertex that is the next vertex on the shortest path from u to the root.

In the first test case, the tree looks like this:



Here, the maximum size of a dilworthy set is 2, and there is only one dilworthy set of size 2: $\{3, 4\}$.

In the second test case, the tree looks like this:



Here, the maximum size of a dilworthy set is 2, and there are 4 dilworthy sets of size 2: $\{2, 4\}, \{2, 5\}, \{3, 4\}, \{3, 5\}$.

Problem E1. Blips and Chitz

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

Rick is a genius scientist that can travel across the galaxy through teleportation. Rick went to his favorite place on an alien planet called "Blips and Chitz" where he can play all kinds of games that are much more fun than the ones on Earth!

There is a particular game Rick likes that makes him win Flerbos (the currency in the alien planet). The game keeps count of two values, a *score*, and a *weight*.

When Rick starts playing, his score and weight are initially 0. The game also gives Rick a positive integer m.

The game has n buttons. Rick can press one button **at most once**. The ith button is labeled with two positive integers w_i and p_i, where if Rick presses this button, his weight will increase by w_i and p_i points will be added to his score. For each button, Rick can press it or not, he decides.

When Rick decides to stop playing, he rings a bell and the game evaluates his weight and score, and if the weight is divisible by m, he will win Flerbos equal to his points.

Help Rick win the maximum number of Flerbos.

Input

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

The first line of each test case contains two integers n, m ($1 \le n, m \le 2000$), the number of bags, and the divisor of weights of our thief's peculiar sack, respectively.

The second line of each test case contains n integers $w_1, w_2, \ldots, w_n$ ($1 \le w_i \le 10^9$), the labels of the buttons that add to Rick's weight.

The second line of each test case contains n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le 10^9$), the labels of the buttons that add to Rick's points.

The sum of $n + m$ over all test cases doesn't exceed $2 \cdot 10^4$.

Output

For each test case output one line containing one integer, the maximum amount of Flerbos Rick can get.

Please note, that the answer for some test cases won't fit into 32-bit integer type, so you should use at least 64-bit integer type in your programming language (like `long long` for `C++`).

Scoring

Group 1: (20 points)

$1 \le n \le 10$

Group 2: (80 points)

No additional constraints.

Example

standard input	standard output
3 4 10 4 6 13 17 1 2 3 4 5 2 1 1 1 2 2 4 3 6 10 10 4 100 10 20 3 40 10 20 3 40	10 30 0

Note

In the first test case, Rick can press all buttons and obtain a weight of 40 and 10 points, and the weight is divisible by 10 (the value of m), so he rings the bell, and collects 10 Flerbos.

In the second test case, Rick can press the first, third, fourth, and fifth buttons, to obtain a weight of 6 - divisible by $m = 2$ - and 30 points. He rings the bell and collects 30 Flerbos.

In the third test case, there is no way Rick can obtain a weight that is divisible by 100, so the answer is 0.

Problem E2. Make Them Equivalent

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

Given an array of n integers $a_1, a_2, \ldots, a_n$, and a positive integer $k < n$, and a positive integer x.

You can perform any one of two operations several times (possibly zero):

- Choose an integer i where $1 \le i \le n$, and decrement a_i (that is, perform the assignment $a_i := a_i - 1$). This operation costs x.
- Choose an integer i where $1 \le i \le n$, and increment a_i (that is, perform the assignment $a_i := a_i + 1$). This operation costs x.

Find the minimum cost to achieve $a_i = a_{i+k}$ for all i such that $1 \le i \le n - k$.

Input

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

The first line of each test case contains three integers n, k, x ($1 \le k < n \le 10^5$, $1 \le x \le 10^5$).

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_1 \le a_2 \le \cdots \le a_n \le 10^4$).

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

Output

For each test case, output one line containing one integer, the minimum cost to achieve $a_i = a_{i+k}$ for all i such that $1 \le i \le n - k$.

Please note, that the answer for some test cases won't fit into 32-bit integer type, so you should use at least 64-bit integer type in your programming language (like `long long` for `C++`).

Example

standard input	standard output
3 8 2 4 1 3 5 7 9 11 13 15 6 3 10 1 1 1 2 2 2 5 1 1 1 2 3 9 100	128 30 106

Note

In the first test case, you can:

- Increment the first four values of the array by the following values $6, 6, 2, 2$ (that is, for example, increment the first element 6 times), making the array in the form $7, 9, 7, 9, 9, 11, 13, 15$.
- Decrement the last four values of the array by the following values $2, 2, 6, 6$ (that is, for example, decrement the last element 6 times), making the array in the form $7, 9, 7, 9, 7, 9, 7, 9$, which satisfies the requirement with $k = 2$.

This will cost $(6 + 6 + 2 + 2 + 2 + 2 + 6 + 6) \cdot 4 = 128$.

Problem F. Proofy and the cat

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

Proofy wanted to make his cat a fun game, so he drew with his pencil on a big piece of paper a tree of n vertices numbered from 1 to n that is rooted at vertex 1 (see notes for more clarification on the convention of rooted trees). He gave each edge a weight - which is a positive integer written on the edge. He also gave each vertex a value (he wrote it down), where the value of the ith vertex is a_i.

Now, he gave his cat a chip.

She can start playing *a round* by placing the chip on **any** vertex u.

In one round, she can play several moves (possibly none); that is, she stops the round whenever she likes.

On each move, she can move the chip from the vertex u along any edge (u, v) to another vertex v such that v is NOT the parent of u.

In the end of the round, she records all the weights of the edge she moved the chip along, and all the vertices that her chip visited.

Proofy tells her that the *profit of a round* is the **sum** of all values a_v for each **vertex** v her chip visited (including the starting one), and the *cost of a round* is the **maximum** of all **weights** of edges that her chip moved along (if the cat hasn't moved the chip along any edge, the cost is 0).

Now, the game is to find the **minimum cost** of a round such that its **profit** is **at least** k, for a positive integer k that Proofy gave his cat.

Can you help the cat find this minimum cost?

Important Note: if no round exists that has profit at least k, the cat should answer -1.

Input

The first line of input contains an 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 k ($2 \le n \le 10^5, 1 \le k \le 10^9$), the number of vertices in the tree that Proofy drew and the lower-bound on the sum of a round that the cat has to collect.

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

The third line of each test case contains $n-1$ integers $p_2, p_3, \ldots, p_n$ ($1 \le p_i < i$), where p_i is the parent of the ith vertex.

The fourth line of each test case contains $n-1$ integers $w_2, w_3, \ldots, w_n$ ($1 \le w_i \le 10^9$), where w_i is the weight of the edge from i to the parent of the ith vertex (i.e. the weight of the edge (p_i, i)).

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

Output

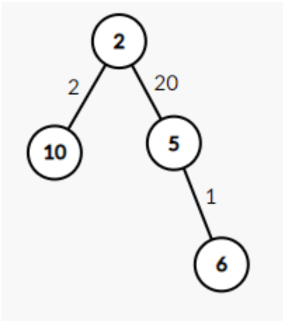
For each test case, output one line containing one integer, the minimum cost of a round such that its value is at least k, or -1 if such round doesn't exist.

Example

standard input	standard output
2 4 11 2 5 6 10 1 2 1 20 1 2 8 11 2 6 3 7 8 4 9 10 1 1 2 2 3 6 6 2 20 10 10 2 3 5	1 3

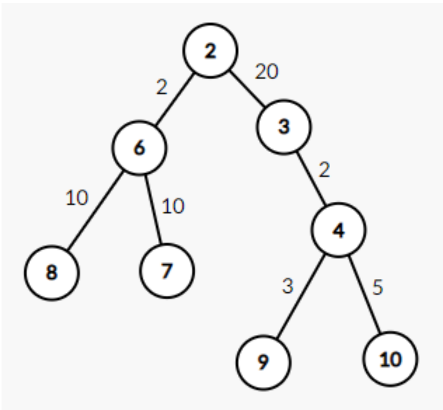
Note

In the first test case, the tree looks like this: (the labels are replaced by values for simplicity)



The optimal solution would be placing the chip on the node with value 5 and moving to the node with value 6, achieving a profit of a round of 11, which is at least $k = 11$, and the cost is 1.

In the second test case, the tree looks like this:



An optimal solution would be placing the chip on the node with value 3 and moving it to the node with value 4 and then to the node with value 9, and so the profit is 16, which is at least $k = 11$, and the cost is 3.

A *tree* is a connected undirected graph without cycles. A *rooted tree* is a tree with a selected vertex, which is called the *root*.

The tree is specified by an array of parents p containing n numbers: p_i is a parent of the vertex with the index i. The *parent* of a vertex u is a vertex that is the next vertex on the shortest path from u to the root.