A. Farmer John's Challenge

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

Trade Winds - Patrick Deng

Let's call an array a *sorted* if $a_1 \le a_2 \le \ldots \le a_{n-1} \le a_n$.

You are given two of Farmer John's favorite integers, n and k. He challenges you to find any array $a_1, a_2, \ldots, a_n$ satisfying the following requirements:

- $1 \le a_i \le 10^9$ for each $1 \le i \le n$;
- Out of the n total cyclic shifts of a, exactly k of them are sorted.[†]

If there is no such array a, output -1.

[†]The x-th ($1 \le x \le n$) **cyclic shift** of the array a is $a_x, a_{x+1} \ldots a_n, a_1, a_2 \ldots a_{x-1}$. If $c_{x,i}$ denotes the i'th element of the x'th cyclic shift of a, exactly k such x should satisfy $c_{x,1} \le c_{x,2} \le \ldots \le c_{x,n-1} \le c_{x,n}$.

For example, the cyclic shifts for $a = [1, 2, 3, 3]$ are the following:

- $x = 1$: $[1, 2, 3, 3]$ (sorted);
- $x = 2$: $[2, 3, 3, 1]$ (not sorted);
- $x = 3$: $[3, 3, 1, 2]$ (not sorted);
- $x = 4$: $[3, 1, 2, 3]$ (not sorted).

Input

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

Each test case contains two integers n and k ($1 \le k \le n \le 10^3$) — the length of a and the number of sorted cyclic shifts a must have.

It is guaranteed that the sum of n over all test cases does not exceed 10^3.

Output

For each test case, print a single line:

- if there is a valid array a, output n integers, representing $a_1, a_2, \ldots, a_n$;
- otherwise, output -1.

If there are multiple solutions, print any of them.

Standard Input	Standard Output

3
2 2
3 1
3 2
	1 1
	69420 69 420
	-1

Note

In the first testcase, $a = [1, 1]$ satisfies $n = 2, k = 2$:

The two cyclic shifts of a are $[a_1, a_2]$ and $[a_2, a_1]$, which are both $[1, 1]$ and are sorted.

In the second testcase, $a = [69\,420, 69, 420]$ satisfies $n = 3, k = 1$:

The three cyclic shifts of a are $[a_1, a_2, a_3]$, $[a_2, a_3, a_1]$, $[a_3, a_1, a_2]$, which are $[69\,420, 69, 420]$, $[69, 420, 69\,420]$, and $[420, 69\,420, 69]$, respectively.

Only $[69, 420, 69\,420]$ is sorted.

,

B. Bessie and MEX

```
Input file:     standard input
Output file:    standard output
Time limit:     1.5 seconds
Memory limit:   256 megabytes
```

MOOO! - Doja Cat

Farmer John has a permutation $p_1, p_2, \ldots, p_n$, where every integer from 0 to $n-1$ occurs exactly once. He gives Bessie an array a of length n and challenges her to construct p based on a.

The array a is constructed so that a_i = $\texttt{MEX}(p_1, p_2, \ldots, p_i) - p_i$, where the $\texttt{MEX}$ of an array is the minimum non-negative integer that does not appear in that array. For example, $\texttt{MEX}(1, 2, 3) = 0$ and $\texttt{MEX}(3, 1, 0) = 2$.

Help Bessie construct any valid permutation p that satisfies a. The input is given in such a way that at least one valid p exists. If there are multiple possible p, it is enough to print one of them.

Input

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

The first line of each test case contains an integer n ($1 \le n \le 2 \cdot 10^5$) — the lengths of p and a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($-n \le a_i \le n$) — the elements of array a.

It is guaranteed that there is at least one valid p for the given data.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output n integers on a new line, the elements of p.

If there are multiple solutions, print any of them.

Standard Input	Standard Output

3
5
1 1 -2 1 2
5
1 1 1 1 1
3
-2 1 2
	0 1 4 2 3
	0 1 2 3 4
	2 0 1

Note

In the first case, $p = [0, 1, 4, 2, 3]$ is one possible output.

a will then be calculated as $a_1 = \texttt{MEX}(0) - 0 = 1$, $a_2 = \texttt{MEX}(0, 1) - 1 = 1$, $a_3 = \texttt{MEX}(0, 1, 4) - 4 = -2$, $a_4 = \texttt{MEX}(0, 1, 4, 2) - 2 = 1$, $a_5 = \texttt{MEX}(0, 1, 4, 2, 3) - 3 = 2$.

So, as required, a will be $[1, 1, -2, 1, 2]$.

,

C1. Bessie's Birthday Cake (Easy Version)

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

Proof Geometric Construction Can Solve All Love Affairs - manbo-p

This is the easy version of the problem. The only difference between the two versions is the constraint on y. In this version $y = 0$. You can make hacks only if both versions are solved.

Bessie has received a birthday cake from her best friend Elsie, and it came in the form of a regular polygon with n sides. The vertices of the cake are numbered from 1 to n clockwise. You and Bessie are going to choose some of those vertices to cut **non-intersecting** diagonals into the cake. In other words, the endpoints of the diagonals must be part of the chosen vertices.

Bessie would only like to give out pieces of cake which result in a triangle to keep consistency. The size of the pieces doesn't matter, and the whole cake does not have to be separated into all triangles (other shapes are allowed in the cake, but those will not be counted).

Bessie has already chosen x of those vertices that can be used to form diagonals. She wants you to choose **no more than** y other vertices such that the number of triangular pieces of cake she can give out is maximized.

What is the maximum number of triangular pieces of cake Bessie can give out?

Input

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

The first line of each test case consists of three integers, n, x, and y ($4 \leq n \leq 10^9$, $2 \leq x \leq \min(n, 2 \cdot 10^5)$, $y = 0$) — the number of sides of the polygon, number of vertices Bessie has chosen, and the maximum number of other vertices you can choose.

The second line consists of x distinct integers from 1 to n, representing the vertices Bessie has chosen.

It is guaranteed the sum of x over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single integer: the maximum number of non-intersecting triangular pieces of cake she can give out.

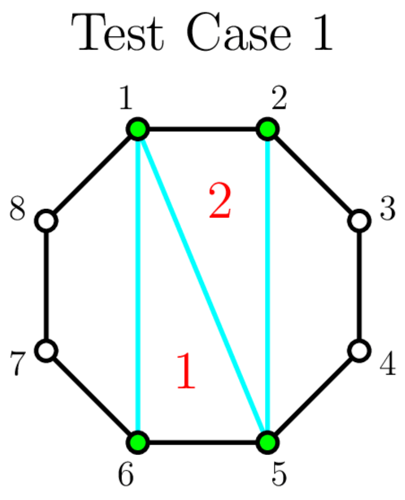
Standard Input	Standard Output

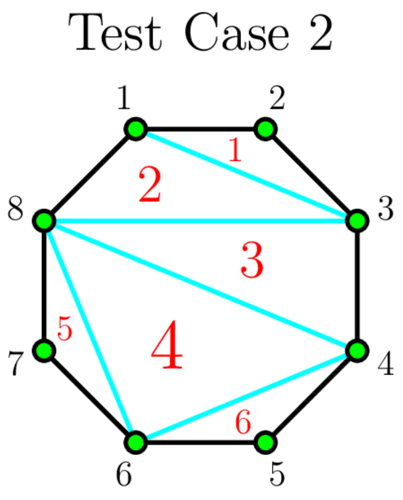
3
8 4 0
1 6 2 5
8 8 0
1 3 2 5 4 6 7 8
4 2 0
1 3
	2
	6
	2

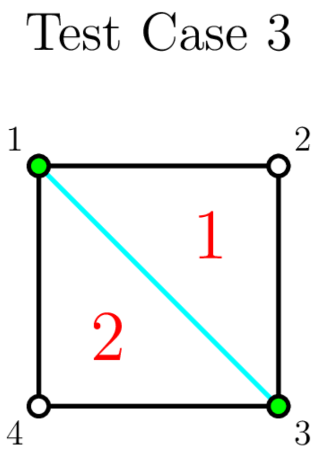
Note

In test cases 1, 2 and 3, you can get 2, 6 and 2 non-intersecting triangular pieces of cake, respectively. A possible construction is shown in the following pictures:

The green dots represent vertices that can be used, the blue lines represent diagonals that are drawn, and the red numbers represent triangles that are counted.







,

C2. Bessie's Birthday Cake (Hard Version)

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

Proof Geometric Construction Can Solve All Love Affairs - manbo-p

---

This is the hard version of the problem. The only difference between the two versions is the constraint on y. In this version $0 \le y \le n - x$. You can make hacks only if both versions are solved.

Bessie has received a birthday cake from her best friend Elsie, and it came in the form of a regular polygon with n sides. The vertices of the cake are numbered from 1 to n clockwise. You and Bessie are going to choose some of those vertices to cut **non-intersecting** diagonals into the cake. In other words, the endpoints of the diagonals must be part of the chosen vertices.

Bessie would only like to give out pieces of cake which result in a triangle to keep consistency. The size of the pieces doesn't matter, and the whole cake does not have to be separated into all triangles (other shapes are allowed in the cake, but those will not be counted).

Bessie has already chosen x of those vertices that can be used to form diagonals. She wants you to choose **no more than** y other vertices such that the number of triangular pieces of cake she can give out is maximized.

What is the maximum number of triangular pieces of cake Bessie can give out?

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 consists of three integers, n, x, and y ($4 \le n \le 10^9$, $2 \le x \le \min(n, 2 \cdot 10^5)$, $0 \le y \le n - x$) — the number of sides of the polygon, number of vertices Bessie has chosen, and the maximum number of other vertices you can choose.

The second line consists of x distinct integers from 1 to n, representing the vertices Bessie has chosen.

It is guaranteed the sum of x over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single integer: the maximum number of non-intersecting triangular pieces of cake she can give out.

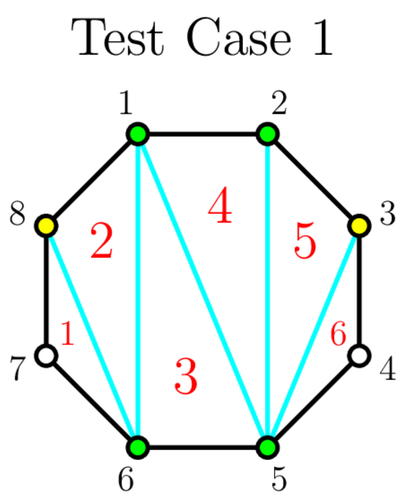
Standard Input	Standard Output

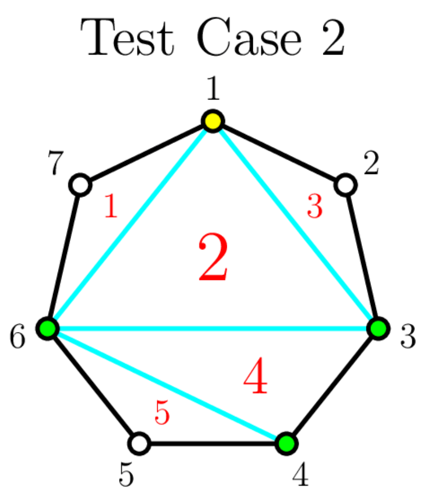
3
8 4 2
1 6 2 5
7 3 1
6 4 3
4 2 2
1 3
	6
	5
	2

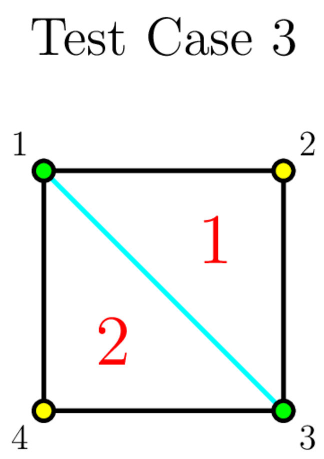
Note

In test cases 1, 2 and 3, you can get 6, 5 and 2 non-intersecting triangular pieces of cake, respectively. A possible construction is shown in the following pictures:

The green dots represent vertices that Bessie chose, the yellow dots represent vertices that you chose, the blue lines represent diagonals that are drawn, and the red numbers represent triangles that are counted.





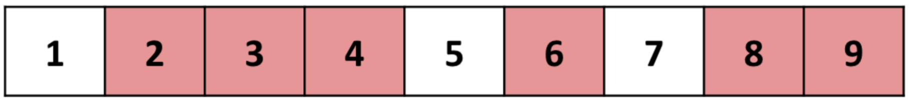


D. Learning to Paint

```
Input file:     standard input
Output file:    standard output
Time limit:     4.5 seconds
Memory limit:   512 megabytes
```

Pristine Beat - Touhou

Elsie is learning how to paint. She has a canvas of n cells numbered from 1 to n and can paint any (potentially empty) subset of cells.



Elsie has a 2D array a which she will use to evaluate paintings. Let the maximal contiguous intervals of painted cells in a painting be $[l_1, r_1], [l_2, r_2], \ldots, [l_x, r_x]$. The *beauty* of the painting is the sum of a_{l_i, r_i} over all $1 \le i \le x$. In the image above, the maximal contiguous intervals of painted cells are $[2, 4], [6, 6], [8, 9]$ and the *beauty* of the painting is $a_{2,4} + a_{6,6} + a_{8,9}$.

There are 2^n ways to paint the strip. Help Elsie find the k largest possible values of the *beauty* of a painting she can obtain, among all these ways. Note that these k values do not necessarily have to be distinct. It is guaranteed that there are at least k different ways to paint the canvas.

Input

The first line contains a single integer t ($1 \le t \le 10^3$) — the number of test cases.

The first line of each test case contains 2 integers n and k ($1 \le n \le 10^3, 1 \le k \le \min(2^n, 5 \cdot 10^3)$) — the number of cells and the number of largest values of the *beauty* of a painting you must find.

The next n lines of each test case describe a where the i-th of which contains $n - i + 1$ integers $a_{i,i}, a_{i,i+1}, \ldots, a_{i,n}$ ($-10^6 \le a_{i,j} \le 10^6$).

It is guaranteed the sum of n over all test cases does not exceed 10^3 and the sum of k over all test cases does not exceed $5 \cdot 10^3$.

Output

For each test case, output k integers in one line: the i-th of them must represent the i-th largest value of the *beauty* of a painting Elsie can obtain.

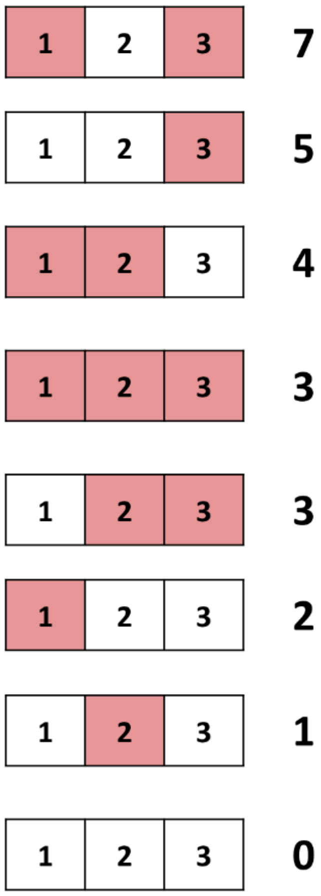
Standard Input	Standard Output
4	0 -5
1 2	2 0 -1 -3
-5	7 5 4 3 3 2 1 0
2 4	8 8 7 7 5 5 2 2 1 1 1 1 1 1 0 0 0 0 0 0 -1
2 -3	
-1	
3 8	
2 4 3	
1 3	
5	
6 20	

```
0 -6 -3 0 -6 -2
-7 -5 -2 -3 -4
7 0 -9 -4
2 -1 1
1 -2
-6
```

Note

In the first test case, Elsie can either paint the only cell or not paint it. If she paints the only cell, the beauty of the painting is -5. If she chooses not to paint it, the beauty of the painting is 0. Thus, the largest beauty she can obtain is 0 and the second largest beauty she can obtain is -5.

Below is an illustration of the third test case.



,

E. Farm Game

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

Lunatic Princess - Touhou

Farmer Nhoj has brought his cows over to Farmer John's farm to play a game! FJ's farm can be modeled by a number line with walls at points 0 and $l+1$. On the farm, there are $2n$ cows, with n of the cows belonging to FJ and the other n belonging to FN. They place each of their cows at a distinct point, and no two FJ's cows nor FN's cows are adjacent. Two cows are adjacent if there are no other cows between them.

Formally, if $a_1, a_2, \ldots, a_n$ represents the positions of FJ's cows and $b_1, b_2, \ldots, b_n$ represents the positions of FN's cows, then either $0 < a_1 < b_1 < a_2 < b_2 < \ldots < a_n < b_n < l+1$ or $0 < b_1 < a_1 < b_2 < a_2 < \ldots < b_n < a_n < l+1$.

In one move, a farmer chooses a number k $(1 \le k \le n)$ and a direction (left or right). Then, that farmer chooses k of his cows and moves them one position towards the chosen direction. A farmer cannot move any of his cows onto the walls or onto another farmer's cow. If a farmer cannot move any cows, then that farmer loses. FJ starts the game, making the first turn.

Given l and n, find the number of possible game configurations for Farmer John to win if both farmers play optimally. It may be the case that the game will continue indefinitely, in which no farmer wins. A configuration is different from another if there is any i such that a_i or b_i is different. Output the answer modulo $998\,244\,353$.

Input

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

Each test case contains two integers l and n $(2 \le l \le 10^6, 1 \le n \le \lfloor \frac{l}{2} \rfloor)$ — the length of the number line and the number of cows each farmer will place.

It is guaranteed the sum of l over all test cases does not exceed 10^6.

Output

For each test case output an integer: the number of game configurations where Farmer John wins if both farmers play optimally, modulo $998\,244\,353$.

Standard Input	Standard Output

3
2 1
3 1
420 69
	0
	2
	870279412

Note

Let J denote FJ's cow, N denote FN's cow, and _ denote an empty space.

For the first test case, the two possible configurations are JN or NJ. In both cases, since FJ makes the first turn and cannot make any moves, he cannot win.

For the second case there are two possible configurations for FJ to win: `N_J` and `J_N`.

,

F. Farmer John's Favorite Function

Input file:	standard input
Output file:	standard output
Time limit:	5 seconds
Memory limit:	256 megabytes

ΩΩPARTS - Camellia

Farmer John has an array a of length n. He also has a function f with the following recurrence:

- $f(1) = \sqrt{a_1}$;
- For all $i > 1$, $f(i) = \sqrt{f(i-1) + a_i}$.

Note that $f(i)$ is not necessarily an integer.

He plans to do q updates to the array. Each update, he gives you two integers k and x and he wants you to set $a_k = x$. After each update, he wants to know $\lfloor f(n) \rfloor$, where $\lfloor t \rfloor$ denotes the value of t rounded down to the nearest integer.

Input

The first line contains n and q ($1 \le n, q \le 2 \cdot 10^5$), the length of a and the number of updates he will perform.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^{18}$).

The next q lines each contain two integers k and x ($1 \le k \le n$, $0 \le x \le 10^{18}$), the index of the update and the element he will replace a_k with.

Output

For each update, output an integer, $\lfloor f(n) \rfloor$, on a new line.

Standard Input	Standard Output

5 6
0 14 0 7 6
1 4
1 3
2 15
4 1
5 2
5 8
	3
	2
	3
	2
	1
	3

15 10
3364 1623 5435 7 6232 245 7903 3880 9738 577
4598 1868 1112 8066 199
14 4284
14 8066
6 92
6 245
2 925
2 1623
5 176
	16
	17
	16
	17
	16
	17
	16
	17
	16
	17

5 6232
3 1157
3 5435	

2 2
386056082462833225 923951085408043421
1 386056082462833225
1 386056082462833224
	961223744
	961223743

13 10
31487697732100 446330174221392699
283918145228010533 619870471872432389
11918456891794188 247842810542459080
140542974216802552 698742782599365547
533363381213535498 92488084424940128
401887157851719898 128798321287952855
137376848358184069
3 283918145228010532
3 283918145228010533
1 2183728930312
13 1000000000000000000
10 1000000000000000000
9 1000000000000000000
8 1000000000000000000
7 1000000000000000000
6 1000000000000000000
5 1000000000000000000
	370643829
	370643830
	370643829
	1000000000
	1000000000
	1000000000
	1000000000
	1000000000
	1000000000
	1000000000

Note

In the first test case, the array after the first update is $[4, 14, 0, 7, 6]$. The values of f are:

- $f(1) = 2$;
- $f(2) = 4$;
- $f(3) = 2$;
- $f(4) = 3$;
- $f(5) = 3$.

Since $\lfloor f(5) \rfloor = 3$, we output 3.

The array after the second update is $[3, 14, 0, 7, 6]$. The values of f, rounded to 6 decimal places, are:

- $f(1) \approx 1.732051$;
- $f(2) \approx 3.966365$;
- $f(3) \approx 1.991573$;
- $f(4) \approx 2.998595$;
- $f(5) \approx 2.999766$.

Since $\lfloor f(5) \rfloor = 2$, we output 2.

,

G. Bessie and Cards

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

Second Dark Matter Battle - Pokemon Super Mystery Dungeon

Bessie has recently started playing a famous card game. In the game, there is only one deck of cards, consisting of a "draw 0" cards, b "draw 1" cards, c "draw 2" cards, and 5 special cards. At the start of the game, all cards are in the randomly shuffled deck.

Bessie starts the game by drawing the top 5 cards of the deck. She may then play "draw x" cards from the hand to draw the next x cards from the top of the deck. Note that every card can only be played once, special cards cannot be played, and if Bessie uses a "draw 2" card when there is only 1 card remaining in the deck, then she simply draws that remaining card. Bessie wins if she draws all 5 special cards.

Since Bessie is not very good at math problems, she wants you to find the probability that she wins, given that the deck is shuffled randomly over all $(a+b+c+5)!$ possible orderings. It can be shown that this answer can always be expressed as a fraction $\frac{p}{q}$ where p and q are coprime integers. Output $p \cdot q^{-1}$ modulo $998\,244\,353$.

Input

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

Each test case contains three integers a, b, and c ($0 \le a, b, c \le 2 \cdot 10^5$) – the number of draw 0 cards, draw 1 cards, and draw 2 cards, respectively.

It is guaranteed that the sum of a over all test cases does not exceed $2 \cdot 10^5$, the sum of b over all test cases does not exceed $2 \cdot 10^5$, and the sum of c over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single integer — the probability that Bessie wins, modulo $998\,244\,353$.

Standard Input	Standard Output

4
1 1 1
0 0 0
5 3 7
3366 1434 1234
	903173463
	1
	35118742
	398952013

Note

In the first case, we have 1 of each type of "draw" card and 5 special cards. There are $30\,720$ starting decks where Bessie will win by drawing the top 5 cards and $40\,320$ starting decks in total. Thus, the probability of Bessie winning is $\frac{30\,720}{40\,320} = \frac{16}{21}$.

One example of a winning starting deck is, top to bottom,

1. "Special",

2. "Draw 1",
3. "Special",
4. "Special",
5. "Draw 0",
6. "Draw 2",
7. "Special",
8. "Special".

One example of a losing starting deck is:

1. "Special",
2. "Draw 1",
3. "Special",
4. "Special",
5. "Draw 0",
6. "Special",
7. "Special",
8. "Draw 2".

,

H. Farmer John's Favorite Intern

```
Input file:     standard input
Output file:    standard output
Time limit:     6 seconds
Memory limit:   512 megabytes
```

Peaches...

Ruby just won an internship position at Farmer John's farm by winning a coding competition! As the newly recruited intern, Ruby is tasked with maintaining Farmer John's peach tree, a tree consisting of n nodes rooted at node 1. Each node initially contains $a_i = 0$ peaches, and there are two types of events that can happen:

1. Growth event at some node x: Ruby must choose **either** the parent of x **or** any node in the subtree of x and increase the amount of peaches it contains by one.
2. Harvest event at some node x: Ruby must choose a single node that is in the subtree of x and decrease the amount of peaches it contains by one. Note that this is **not** the same set of nodes as the growth event.

Note that the subtree of x includes the node x as well.
Ruby is also given an array b of length n. The peach tree is deemed healthy if $a_i \geq b_i$ for every node i.

Ruby is asked to perform q operations of two types:

- `1 x v` — Perform v growth events on node x. Ruby does **not** have to choose the same node to increase in every growth event.
- `2 x v` — Perform v harvest events on node x. Ruby does **not** have to choose the same node to decrease in every harvest event.

For every prefix of operations, Ruby asks you to find if she can perform these operations **in some order** such that the resulting peach tree (at the end of these operations) is healthy. Note that Ruby can't perform a harvest event that makes any a_i negative.

Every prefix is independent, meaning that for a given operation, Ruby may choose different nodes to perform events on for every prefix that contains that operation.

Input

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

The first line of each test case contains two integers n and q $(1 \leq n, q \leq 2 \cdot 10^5)$ — the size of the tree and the number of operations.

The second line contains $n - 1$ integers $p_2, p_3, \ldots, p_n$ $(1 \leq p_i < i)$ — the parent of each node.

The third line contains n integers $b_1, b_2, \ldots, b_n$ $(0 \leq b_i \leq 10^6)$ — the minimum number of peaches each node needs for the peach tree to be considered healthy.

The next q lines describe the operations Ruby will perform. Each line contains three integers t, x, and v ($1 \leq t \leq 2$, $1 \leq x \leq n$, $1 \leq v \leq 10^6$). If $t = 1$, this denotes that Ruby must perform v growth events on node x. If $t = 2$, this denotes that Ruby must perform v harvest events on node x.

It is guaranteed that the sum of n does not exceed $2 \cdot 10^5$ and the sum of q does not exceed $2 \cdot 10^5$

Output

For each test case, output q lines. The i-th line should contain "YES" if Ruby can make the peach tree healthy after performing operations $1, 2, \ldots, i$ in some order. Otherwise, it should contain "NO".

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

Standard Input	Standard Output
`2`	`NO`
`8 8`	`NO`
`1 1 1 4 3 6 6`	`YES`
`5 6 2 9 8 4 1 3`	`NO`
`1 3 14`	`YES`
`1 4 17`	`NO`
`1 2 7`	`NO`
`2 2 1`	`YES`
`1 6 1`	`NO`
`2 1 1000000`	`NO`
`1 4 999999`	`NO`
`1 3 1`	`YES`
`10 20`	`YES`
`1 1 1 2 5 2 4 7 2`	`NO`
`311353 270334 74853 385085 315501 183346`	`NO`
`234819 417314 103862 429437`	`YES`
`1 1 837541`	`YES`
`1 10 933876`	`NO`
`1 1 565958`	`YES`
`1 4 791455`	`YES`
`2 3 85054`	`NO`
`2 3 440978`	`YES`
`1 4 981040`	`NO`
`1 5 68522`	`NO`
`2 1 858305`	`YES`
`2 4 184308`	`YES`
`1 4 905081`	`NO`
`2 8 519626`	`NO`
`2 2 269090`	
`1 1 43016`	
`2 2 517644`	
`1 5 355792`	
`1 9 319241`	
`2 10 125447`	
`2 10 523890`	
`1 10 241045`	

Note

For the prefix containing operations $1, 2, \ldots, 5$ in the first test case, Ruby may perform the operations in the following order:

1. Ruby performs operation 2 and chooses to increase a_4 by 9 and a_5 by 8.

2. Ruby performs operation 1 and chooses to increase a_1 by 5, a_3 by 2, a_6 by 4, and a_8 by 3.
3. Ruby performs operation 3 and chooses to increase a_2 by 7.
4. Ruby performs operation 4 and chooses to decrease a_2 by 1.
5. Ruby performs operation 5 and chooses to increase a_7 by 1.