Mexican Standoff (`dessert`)

Edoardo is organising the traditional New Year's Eve dinner of the *OIS* staff: after a round of well-stuffed pizzas, it will be desserts time! As everybody knows, this is a very delicate moment: everybody likes to order a dessert, but feels ashamed to do it. Thus, it is very easy to create a "Mexican standoff", where everyone is waiting for the others to take their move.

Figure 1: To dessert or not to dessert? That is the question.

More precisely, N staff members are attending the dinner. Member i (for $i = 0 \ldots N-1$) has M_i esteemed friends among the others,[1] which we call $F_{i,j}$ for $j = 0 \ldots M_i - 1$. Member i will feel allowed to take the dessert if and only if at least L_i of his esteemed friends have already done the same.

After the waiter asks for desserts, members that are willing to order one will do so, possibly making other members order one for themself, until everybody that has not ordered a dessert does not feel allowed to do so. Help Edoardo plan the dinner budget by determining how many staff members will order a dessert!

> ☞ Among the attachments of this task you may find a template file `dessert.*` with a sample incomplete implementation.

Input

The first line contains the only integer N. The description of the N members follows, each consisting of two lines: the first containing the two integers M_i, L_i, the second containing the M_i integers $F_{i,j}$.

Output

You need to write a single line with an integer: the number of staff members ordering a dessert.

---

[1]There are numerous rivalries in the OIS staff... their are not all friends with each other!

Constraints

- $2 \leq N \leq 300\,000$.
- $1 \leq M = \sum_{i=0}^{N-1} M_i \leq 300\,000$.
- $0 \leq L_i \leq M_i < N$ and $M_i \geq 1$ for each $i = 0 \ldots N - 1$.
- $0 \leq F_{i,j} \leq N - 1$, $F_{i,j} \neq i$ for each $i = 0 \ldots N - 1$, $j = 0 \ldots M_i - 1$.
- There are no repeated friends: $F_{i,j} \neq F_{j,k}$ for $j \neq k$.
- Friendship is not symmetric: if i is a friend of j, j doesn't need to be a friend of i.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points)   Examples.

– **Subtask 2** (20 points) $M_i = 1$ for each $i = 0 \ldots N - 1$.

– **Subtask 3** (10 points) $L_i \leq 1$ for each $i = 0 \ldots N - 1$.

– **Subtask 4** (40 points) $N, M \leq 1000$.

– **Subtask 5** (30 points)   No additional limitations.

Examples

input	output
2 1 1 1 1 0 0	2
3 2 0 2 1 1 1 2 2 2 0 1	1

Explanation

In the **first sample case**, staff members 0 and 1 are friends with each other. Staff member 1 immediately orders a dessert, since he doesn't need any friends to have already ordered a dessert in order to feel allowed to do the same. After that, staff member 0 feels allowed to order a dessert as well.

In the **second sample case**, staff member 0 doesn't wait for anybody and immediately orders a dessert. Then, staff members 1 and 2 are both blocked waiting for each other to order a dessert: none of them will order one.

A Fair Rock Game (`fairgame`)

Alice and Bob are playing a game: they have a pile of N rocks, and in every turn the current player takes at least 1 rock and at most K rocks. Alice makes the first move, then Bob does the second, and so on as they alternate turns.

Figure 1: Playing with rocks.

If one player takes an odd number of rocks, he has to pay M euros. When the pile is empty, the player who made the last move gets P euros and the other one gets Q euros. After the game ends, Alice will end up with an amount of euros, let's call it X, and Bob will end up with another amount of euros, let's call it Y.

They both play optimally, which means Alice wants to maximize the value $X - Y$ and Bob wants to minimize it. Find out the value of $X - Y$.

> ☞ Among the attachments of this task you may find a template file `fairgame.*` with a sample incomplete implementation.

Input

The first line contains the integers N, K, M, P, Q which describe the game.

Output

You need to write a single line with an integer: the final difference in euros $X - Y$.

Constraints

- $1 \le K \le N \le 5\,000\,000$.
- $1 \le M, P, Q \le 5\,000\,000$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points)   Examples.

– **Subtask 2** (10 points) $K = 1$.

– **Subtask 3** (15 points) $N \le 5$.

– **Subtask 4** (25 points) $N \le 1000$.

– **Subtask 5** (10 points) $N \le 100\,000$, $K \le 900$.

– **Subtask 6** (20 points) $N \le 100\,000$.

– **Subtask 7** (20 points)   No additional limitations.

Examples

input	output
6 3 5 4 2	2

In the **first sample case** one possible optimal strategy is:

- Alice removes two rocks, paying nothing. Four rocks are left on the stack.

- Bob removes three rocks, paying five euros. One rock is left on the stack.

- Alice must remove the last rock, paying five euros. She gets four euros, Bob gets two.

In the end, Alice ends up with -1 euros (she lost 1 euro), while Bob finishes with -3 euros (he lost 3 euros). In total, the difference between what Alice and Bob ended up with is 2 euros.

Halloween Candies (`halloween`)

Marco prepared a big basket with M candies for the usual "trick-or-treat" that is going to take place in his district during Halloween. He placed this basket on his porch, in front of the door, so that kids can take candies without having to ring the doorbell.

In this district, things are very regular and the N kids who live here are numbered from 0 to $N-1$. Some kids are completely selfless and will not take any candies (to leave more candies for other kids to enjoy) while some other kids are greedy and will take one or more candies whenever they approach a house.

Figure 1: Kids approaching a house to get candies.

Marco knows that the kids will approach his house in order, from the 0-th kid to the $(N-1)$-th kid, and that they will repeatedly come back to his house (after visiting the other houses in the district) in the same order, until there are no candies left.

Help Marco calculate which kid is going to take the last candy in the basket!

> ☞ Among the attachments of this task you may find a template file `halloween.*` with a sample incomplete implementation.

Input

The first line contains two integers N and M. The second line contains N integers L_i, the number of candies that each kid will take during his/her turn.

Output

You need to write a single line with an integer: the index (from 0 to $N-1$) of the kid that is going to take the very last candy from the basket.

Constraints

- $1 \leq N \leq 100\,000$.
- $1 \leq M \leq 10^{18}$.
- $0 \leq L_i \leq 10^9$ for each $i = 0 \ldots N-1$.
- There is at least one greedy kid i (with $L_i > 0$).

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points)   Examples.

- **Subtask 2** (30 points) $N, M \leq 1000$.

- **Subtask 3** (20 points) $M \leq 1\,000\,000$.

- **Subtask 4** (10 points)   Every kid takes exactly 1 candy.

- **Subtask 5** (15 points)   Every kid takes at least 1 candy.

- **Subtask 6** (25 points)   No additional limitations.

Examples

input	output
5 9 1 0 2 1 1	3
2 1000000000000000000 1 1	1

Explanation

In the **first sample case**, the kid with index 3 will get the last candy.

In the **second sample case** at each turn the two kids take 1 candy each, which means that the last candy will be taken by the second kid (index 1) since M is an even number.

Lightweight Ladder (`ladder`)

In the recent festivities, Luca observed some kids playing in a gym. They seemed to enjoy very much moving around with all these colorful blocks, but sometimes they were unable to climb on them as they were too high when stacked!

Consider this simplified representation of a typical situation: each block is a cube with all sides of one meter. Some cubes can optionally be stacked to form a heap and the heaps are aligned one next to each other, as in picture 2.

Figure 1: Kids at work.

When two heaps have the same number of cubes stacked, a kid can just walk to proceed to the next one. Similarly, in case of a descent when the next heap has fewer cubes than the current one, the kid exploits the gravity and falls gently on the soft cube.

As gravity does not work in the reverse direction, kids cannot climb to higher heaps without the help of a ladder. In the situation presented in figure 2, two ladders can be strategically placed between the start and the first heap and before the last heap.
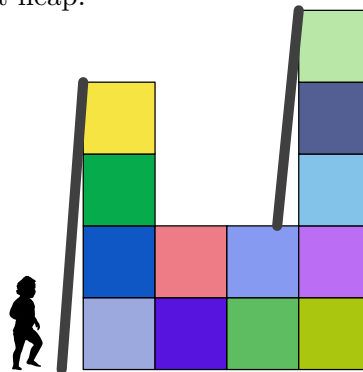

Figure 2: A sequence of four heaps with two ladders of length 4 and 3 meters.

Placing these ladders all over the path can be quite expensive, especially when it is really long. Luca had an idea: kids could carry themselves a single lightweight ladder to reach the end of the path; thus, they need a ladder which covers all their needs, optionally being longer in some cases, but never shorter (otherwise they cannot make it!).

What is the shortest ladder that one can carry to complete the path from the ground at the beginning until the last of N heaps?

> ☞ Among the attachments of this task you may find a template file `ladder.*` with a sample incomplete implementation.

Input

The first line contains the number N of heaps. The second line contains N integers C_i, the number of cubes stacked in the i-th heap.

Output

You need to write a single line with an integer: the minimum length of a ladder that can be carried to complete the path.

Constraints

- $1 \le N \le 1\,000\,000$.
- $0 \le C_i \le 10^9$ for each $i = 0 \dots N-1$.
- In case no ladder is needed at all, you must output the value 0.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points)   Examples.

– **Subtask 2** (10 points) $N \le 2$.

– **Subtask 3** (45 points) $N, C_i \le 1000$ for each $i = 0 \dots N-1$.

– **Subtask 4** (45 points)   No additional limitations.

Examples

input	output
4 4 2 2 5	4
10 2 6 3 7 0 0 7 3 6 4	7

Explanation

In the **first sample case**, already described in this statement, the shortest possible length for the ladder is 4 meters (note that in the picture they are shown as two different ladders, but in reality it is just one carried by the kid as he proceeds):
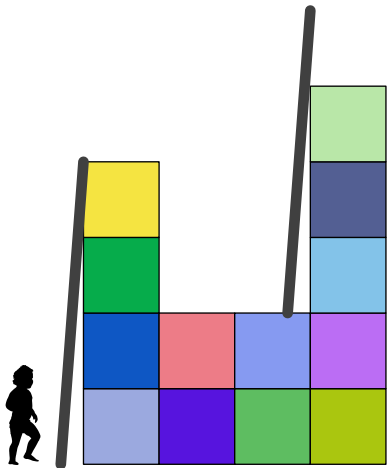
In the **second sample case** we need at least a ladder of length 7 meters (any higher length would work as well, but we are required to find the minimum solution):
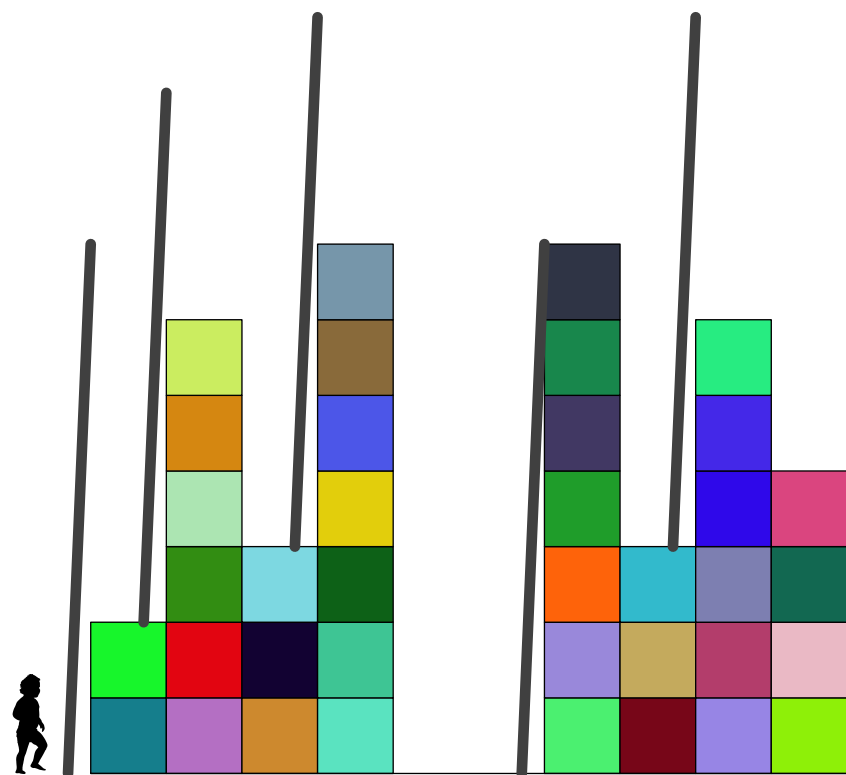
Board Game (`marcel`)

Marcel's son is eager to try out the new board game he just received for Christmas!
The initial setup is a simple square grid of $N \times N$ cards. Every card has an associated value, which can be either positive or negative.

Figure 1: Cards being aligned for the start of the game.

At each step of the game, you can perform one of these five actions:

1. collect all the cards from the first row;

2. collect all the cards from the last row;

3. collect all the cards from the first column;

4. collect all the cards from the last column;

5. terminate the match.

When the cards are collected, they are effectively removed from the grid and the game goes on without them. However, you are only allowed to collect cards if the sum of their values is at least X. Should none of the first four actions be allowed, you are forced to terminate the game (which is the only action you can *always* do).

Marcel wonders whether his son will enjoy many different matches or he will bore immediately. Help him: how many *different* matches can be played?

> ☞ Among the attachments of this task you may find a template file `marcel.*` with a sample incomplete implementation.

Input

The first line contains the integers N and X. The following N lines contain N integers each, the values of the cards in the i-th row of the grid.

Output

You need to write a single line with an integer: the number of possible matches (modulo $1\,000\,000\,007$).

Constraints

- $1 \le N \le 120$.
- $-10^9 \le X \le 10^9$.
- The value associated with a card ranges between -10^6 and 10^6 (inclusive).
- Two matches are considered different if there exists at least a moment when Marcel's son decided to take a different action out of the five possible ones.
- As the result can be big, you must output it modulo $1\,000\,000\,007$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points)   Examples.

– **Subtask 2** (10 points) $N \le 2$.

– **Subtask 3** (30 points) $N \le 5$.

– **Subtask 4** (20 points) $X = 0$ and all the cards have a strictly positive value.

– **Subtask 5** (25 points) $N \le 50$.

– **Subtask 6** (15 points)   No additional limitations.

Examples

input	output
1 10 23	5
1 23 10	1
2 10 23 23 23 23	53
3 4 1 4 2 3 9 -6 7 8 5	62

Explanation

In the **first sample case**, Marcel's son can choose four different ways (actions 1 to 4 in the statement) to collect the only card. Moreover, he can also opt to immediately terminate (which is always a valid action).

In the **second sample case**, collecting the card of value 10 would not satisfy the goal: thus, the only option left is terminate immediately.

In the **third sample case**, you may terminate immediately (1). If you don't, you can choose any of the other 4 actions and reach a configuration with two cards. Then, you can either terminate (1), remove both cards and then terminate (2×1), remove one card and then perform any of the 5 actions with the only remaining card (2×5). Overall, there are $1 + 4 \times (1 + 2 \times 1 + 2 \times 5) = 53$ ways to play the game.

In the **fourth sample case**, you cannot start with action 4 as the sum of the last column is $2 - 6 + 5 = 1 < 4$. If for instance you choose action 1 (remove the first row), then you will not be able to choose action 1 again since the sum of the remaining first row would be $3 + 9 - 6 = 0 < 4$. Overall, there are 62 possible games.

Nice Enough Permutations (`permutation`)

The Romanian Scientific Committee of the IIOT loves playing around with numbers and thinking about puzzles. This time they have come up with an interesting property for permutations.
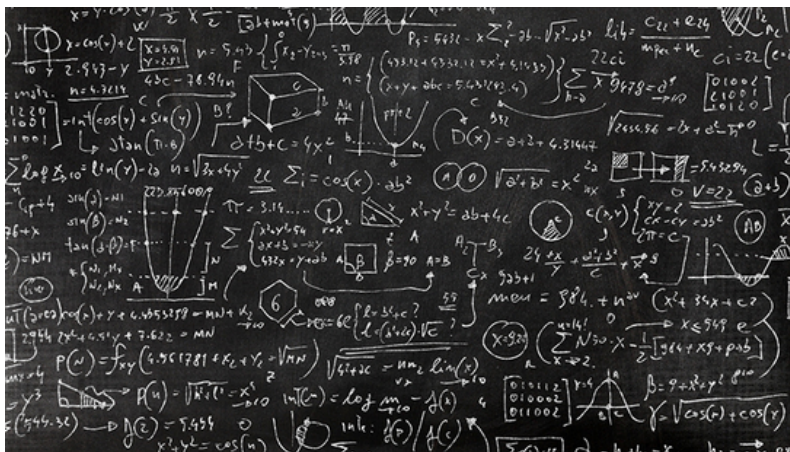

Figure 1: The blackboard just a few minutes after the beginning of a committee's meeting.

When you look at the N integer numbers from 1 to N in some order, you may notice that some arrangments "look nicer" than others. After a long debate, the committee has eventually reached an agreement on what constitutes the "niceness". Consider two adjacent positions i and $i + 1$ in a permutation P: if the number at position $i + 1$ is exactly the successor of the number at position i, that pair is considered "nice" (in other words, $P[i + 1] - P[i]$ should be equal to 1).

A permutation is *nice enough* iff at most one pair of adjacent positions does not respect the aforementioned property. Again, this means that the property should hold at least for $N - 2$ pairs out of the $N - 1$ (total number of pairs of adjacent positions).

You want to make the committee happy by making subtle moves: exchanging values in adjacent positions to make the permutation nice enough and earn the favour of the judges. What is the **minimum** number of moves that you have to make?

☞ Among the attachments of this task you may find a template file `permutation.*` with a sample incomplete implementation.

Input

The first line contains the number of tests T. Each test is described by two lines: the first contains the integer N and the second contains N integers P_i which describe the permutation.

Output

For each test, you need to write a single line with an integer: the minimum number of moves (possibly zero) that you have to do in order to make the permutation nice enough.

Constraints

- $1 \leq N \leq 500\,000$.
- Every P represents a valid permutation (i.e., it contains all the numbers from 1 to N once).
- $1 \leq T \leq 5$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points) Examples.

– **Subtask 2** (10 points) $N \leq 7$.

– **Subtask 3** (20 points) $N \leq 50$.

– **Subtask 4** (20 points) Every permutation can be made nice enough with at most 2 moves.

– **Subtask 5** (20 points) Exchanging values to reach the identity permutation $e = (1, \ldots, N)$ is optimal.

– **Subtask 6** (30 points) No additional limitations.

Examples

input	output
5 4 1 2 3 4 4 4 1 2 3 4 2 3 4 1 5 3 2 1 5 4 6 1 2 4 3 5 6	0 0 0 4 1

Explanation

In the **sample case**, there are five tests. In the first three tests, the permutations are already nice enough, so no move is needed.

The fourth permutation can be transformed with 4 swaps into the permutations `2 3 4 5 1` or `1 2 3 4 5`, both of them being nice enough.

The fifth permutation can be transformed into the permutation `1 2 3 4 5 6` in one move by swapping 3 and 4.

Police Investigation (`police`)

Dark days here in Gotham: criminality is at its highest, and the fearsome William is still free. The police is non-stop searching for him, and the last clues point towards the long and creepy Terror Street.

Figure 1: Terror Street.

Terror Street is composed of N houses, numbered from 1 to N. It is conjectured that William hides in one of them... which one though?

The police starts searching from the first, and stops only when William is found. The agents, if they don't find the fugitive in a house, will interrogate the inhabitant until he gives them the number of the next house to check. So the agents go, house by house, following the information told by the citizens, until William is found, hoping he'll stay still!

Plot twist! You are William! And you don't like to get caught.

Knowing in which house you are, and what the citizens will tell to the police, count after how many wrong attempts you'll be found by the police... so that you can plan your last-minute escape accordingly!

> ☞ Among the attachments of this task you may find a template file `police.*` with a sample incomplete implementation.

Input

The first line contains the integers N and K, the total number of houses and the number of William's house. The second line contains N integers V_i, the answer of the citizen in the i-th house.

Output

You need to write a single line with an integer: the number of failed attempts of the police (zero if William is found in the first house, a positive number x if he's found after x attempts, -1 if he cannot be found by the police).

Constraints

- $1 \le N, K \le 100\,000$.
- $1 \le V_i \le N$ for each $i = 1 \ldots N$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points)   Examples.

– **Subtask 2** (20 points) $N \le 10$.

– **Subtask 3** (20 points) $N \le 1000$.

– **Subtask 4** (40 points)   William is always locatable by the police.

– **Subtask 5** (20 points)   No additional limitations.

Examples

input	output
5 3 2 4 1 3 3	3
5 1 3 2 1 2 5	0
5 5 2 3 1 1 5	-1

Explanation

In the **first sample case** the police starts from 1, then goes to 2, then to 4, and finally finds William at 3. It took 3 attempts before finding him.

In the **second sample case** William is in the first house, so there are no failing attempts.

In the **third sample case** the police starts from 1, then goes to 2, then to 3, and then back to 1, looping forever: William is safe and doesn't have to escape at all.

Pulsating Radars (`pulse`)

The two famous bug-countries *United Bug Land* and *Al-Bugida* are at war. Tzutzu, the best spy from *United Bug Land*, has to infiltrate in *Al-Bugida* to end the war once and for all.

Figure 1: A boring old-fashioned radar, scanning by increasing angle instead of radius.

The war zone between the two countries is an $N \times N$ matrix, so that *United Bug Land* is situated at coordinates $(1, 1)$ and *Al-Bugida* is situated at coordinates (N, N). Tzutzu starts his mission from coordinates $(1, 1)$ and each day he can choose to move one unit up, down, left or right or he can stay still without ever leaving the world map.

Unfortunately, the war zone is monitored by P pulsating radars from *Al-Bugida*, that Tzutzu needs to avoid at all costs. The radars are positioned at coordinates (X_i, Y_i), and scan the surrounding area up to a radius of R_i (for $i = 0 \ldots P-1$). More precisely, every R_i days the radars scan their surroundings starting with day 0, the start of Tzutzu's mission. In the first day, every radar is scanning the immediate vicinities, detecting presence of spies in the exact coordinates (X_i, Y_i). In the following days, the detection radius increases by 1, expanding the monitored area by 1 in the four cardinal directions from any previously detected zone. At day R_i, the radar resets, starting again from scanning the immediate vicinities.

Help Tzutzu by computing the minimum number of days he has to spend in the war zone, in order to accomplish its mission while avoiding the radars!

> ☞ Among the attachments of this task you may find a template file `pulse.*` with a sample incomplete implementation.

Input

The first line contains two integers N, P. The following P lines contains integers X_i, Y_i, R_i each.

Output

You need to write a single line with an integer: the minimum number of days Tzutzu has to spend in the war zone.

Constraints

- $1 \leq N \leq 500$.
- $1 \leq P \leq 150\,000$.
- $1 \leq X_i, Y_i \leq N$ for each $i = 0 \ldots P - 1$.
- $1 \leq R_i \leq 6$.
- There may be more than one radar in the same coordinates.
- Tzutzu can always reach his destination.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.
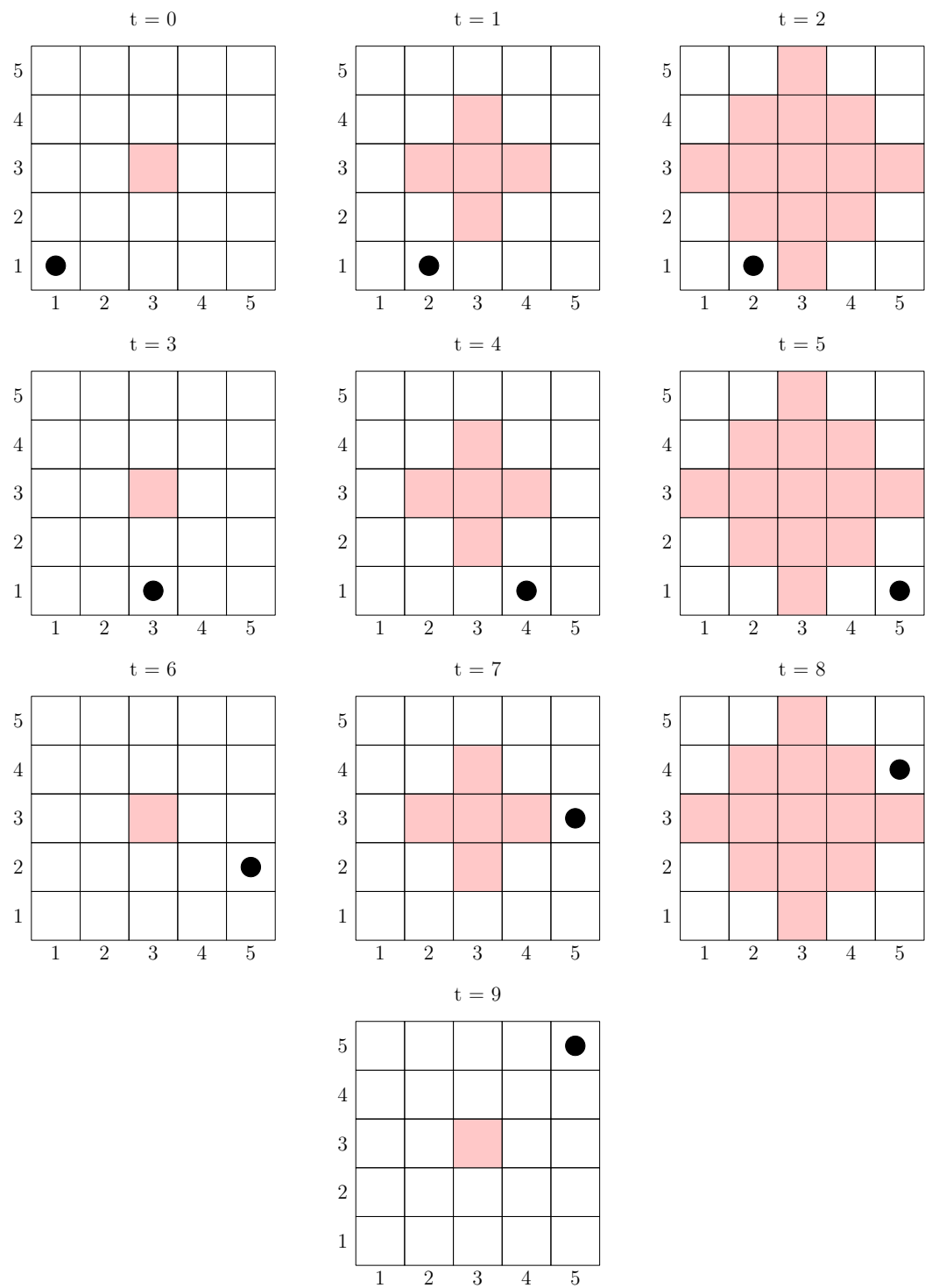
– **Subtask 1** (0 points)   Examples.

– **Subtask 2** (10 points) $N \leq 10$.

– **Subtask 3** (20 points) $R_i = 1$ for all $i = 0 \ldots P - 1$.

– **Subtask 4** (20 points) $N \leq 50$.

– **Subtask 5** (50 points)   No additional limitations.
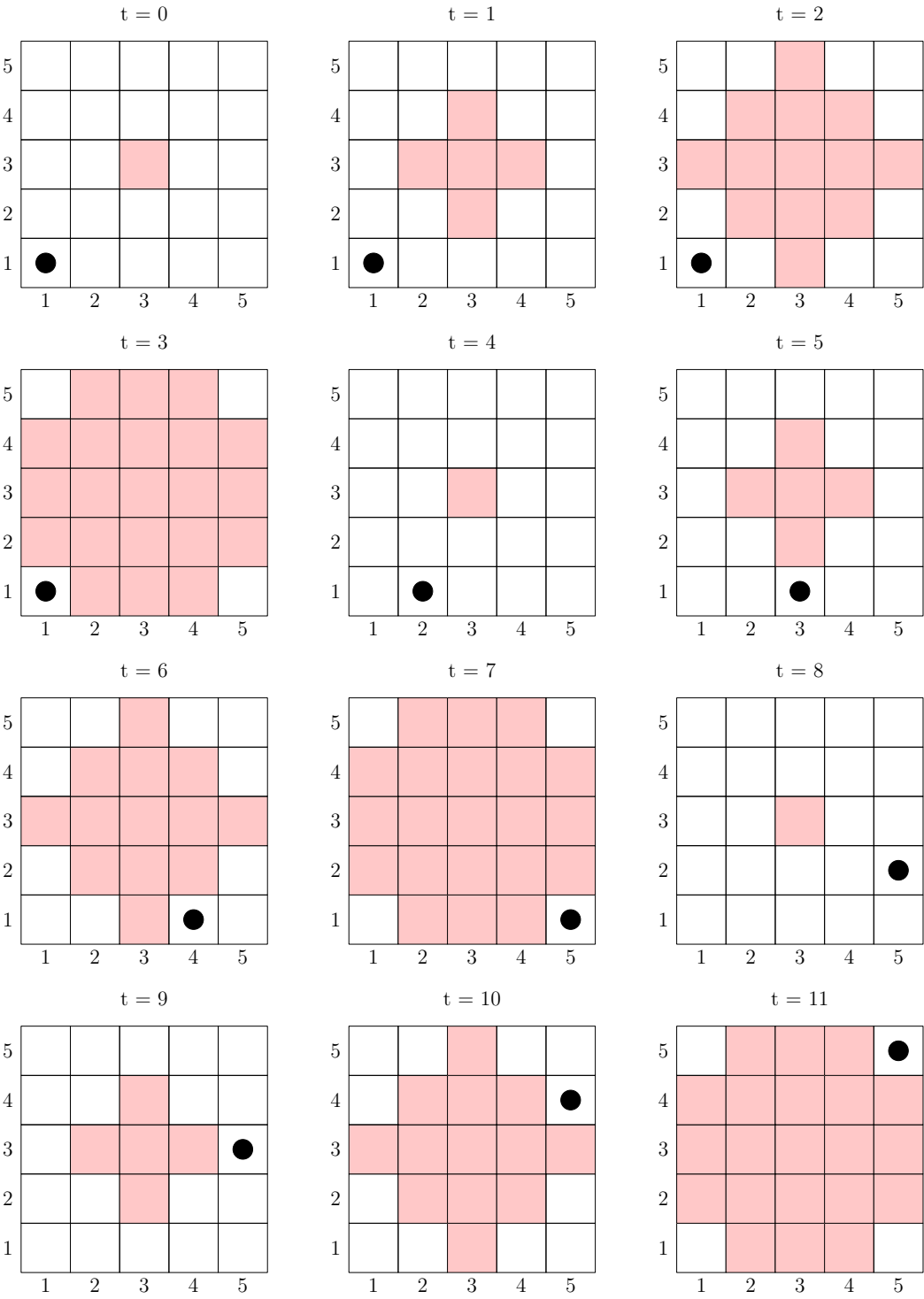
Examples

input	output
5 1 3 3 3	9
5 1 3 3 4	11

Explanation

In the **first sample case**, a valid path for Tzutzu is:

In the **second sample case**, a valid path for Tzutzu is:

Musical Notes (scmax)

Giorgio has composed a new song... but he's not quite satisfied of the result, and plans to improve it. The song is a sequence of N musical notes and the i-th note has a pitch P_i, represented as an integer from 1 to N for simplicity.

To improve the song with the lowest effort, Giorgio has decided to proceed by removing some of the musical notes in the original song. First of all, he knows that for every note pitch a, there is a complementary note pitch C_a. Let S_j be the pitch of the j-th musical note of the new song. He wants the final song to be such that, for every pair of consecutive notes j and $j + 1$, either $S_j < S_{j+1}$ or $S_{j+1} = C_{S_j}$.

Of course, Giorgio would like to remove as little notes as possible, otherwise the new song could end up being too short. Help Giorgio find the length of the longest song he can make!

> ☞ Among the attachments of this task you may find a template file scmax.* with a sample incomplete implementation.

Input

The first line contains one integer N, the number of musical notes in the original song. The second line contains N integers P_i, the pitch of the i-th musical note. The third line contains N integers C_a, the pitch complementary to note pitch a, from pitch 1 to pitch N.

Output

You need to write a single line with an integer: the length of the longest song Giorgio can make.

Constraints

- $1 \leq N \leq 100\,000$.
- $1 \leq P_i \leq N$, for each $i = 0 \ldots N - 1$.
- $1 \leq C_a \leq N$, for each $a = 1 \ldots N$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points)   Examples.

- **Subtask 2** (15 points) $N \leq 10$.

- **Subtask 3** (20 points) $N \leq 1000$.

- **Subtask 4** (20 points) $P_i \leq 10$.

- **Subtask 5** (25 points) $C_a = a$, for each $a = 1 \ldots N$

- **Subtask 6** (20 points)   No additional limitations.

Examples

input	output
6 1 2 3 1 2 3 6 5 1 3 2 1	6
6 5 1 3 2 4 6 6 5 2 3 2 1	5

Explanation

In the **first sample case**, no musical note needs to be removed. Note that the fourth note has a pitch complementary to the pitch of the previous note (the note pitch 3 has complementary note pitch 1).

In the **second sample case**, one solution would be to remove the first note, obtaining $(1, 3, 2, 4, 6)$. Note that the complementary of note pitch 3 is the note pitch 2.