,

Problem A. Beach volleyball

Input file:	`volley.in`
Output file:	`volley.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Vera has worked hard this year, being the example for all of her colleagues. So her manager rewarded her with a vacation on Hawaii, where she is now, lying on the beach near the ocean.

Vera's favorite game is beach volleyball and she is eagerly awaiting for her first match, but she can't decide which team is good enough to join.

Each of the n captains dreams about getting Vera into his team, so they are trying so show her who is the best. There is only one field and everybody wants to play, so all teams formed a queue. First match will be played between first two teams in the queue. Winner is staying on the field, while loser goes to the end of the queue. Each of the next matches is played between the winner of the previous match and the next team in the queue. Winner always stay to play again, while loser always goes to the end of the queue. Each team has its *power*, and different teams have different values of their powers. Winner is always the team with greater power. There could be infinite number of matches.

Vera decided that she will join one of the teams in the k-th match. But she has q such requests, because she can change her mind at any time. For each of such these values k_i you need to determine who will be playing in the match number k_i.

Input

First line of the input contains one integer n — number of teams ($2 \leqslant n \leqslant 100\,000$). Second line contains n different numbers from 1 to n — powers of the teams. First number corresponds to the team at the beginning of initial queue, and the last number to the last team in the queue.

Third line contains one integer q ($1 \leqslant q \leqslant 100\,000$) — number of queries. Each of the next q lines contains one number k_i ($1 \leqslant k_i \leqslant 10^{18}$) — number of the match that Vera is interested in. Note that k_i could be greater than n.

Output

Output q lines. For each query output two numbers — powers of the teams in this match. You could output this two numbers in any order.

Examples

volley.in	volley.out
4 1 3 2 4 1 3	3 4
4 2 1 4 3 3 1 5 2	2 1 4 2 2 4

Note

First sample test:

,

	Who is playing	Queue	Winner	Loser
Match №1	**1 3**	2 4	3	1
Match №2	**3 2**	4 1	3	2
Match №3	**3 4**	1 2	4	3

So the answer will be 4 and 3.

Tests for this problem are divided into 4 groups.

0. Tests 1–2. Sample test cases, 0 points.
1. Tests 3–18. $N \leqslant 2\,000$, $Q = 1$, $K_i \leqslant 2\,000$. 30 points. You will get points for this group only if your program will work on all tests from this group.
2. Tests 19–25. $N \leqslant 100\,000$, $1 \leqslant Q \leqslant 10$, $K_i \leqslant 100\,000$. 30 points. You will get points for this group only if your program will work on all tests from this and from the previous groups.
3. No additional constraints. 40 points. This is **offline** group, i.e. your solution will be tested on this tests after the end of the contest and only if you passed all previous groups. You will get points for the tests of this group *independently*.

Problem B. Tourist Petr

Input file:	`tourist.in`
Output file:	`tourist.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

During the trip to Warm Country Vera became familiar with a handsome tractor driver named Petr. Petr is an experienced tourist, and right after Warm Country he intends to make a small eurotrip. As you know, Europe has an advanced transportation system: there are v interesting (for Petr) cities connected by e railway routes. Each route connects exactly two distinct cities, and it takes exactly one night to get from one city to another using any of these routes. Note, that all routes are **bidirectional**.

The main goal of Petr's trip is sightseeing. As he is a very busy man, his trip cannot last longer than **four** days. Being an experienced tourist, Petr needs exactly one day to a visit all remarkable places in one particular city. He wants his trip to be as saturated as possible: every day he will do sightseeing in some city and every night he will use one of railway routes to travel somewhere. Of course, Petr has no interest in visiting same city twice.

However, Petr's need for optimizations doesn't end here. As a truly experienced tourist, he read through many guidebook, and for each city i determined some value p_i — the expected happiness he will obtain from spending a day in this city. Now he wants to create a travel plan with maximum total the sum of values p_i for all visited cities. As Petr is a tractor driver and not a programmer, he asks for your help.

Input

First line of the input contains two integers v and e ($1 \le v, e \le 300\,000$) — the number of cities Petr is interested in and the number of railway routes between these cities, respectively.

Next line contains v integers p_i ($1 \le p_i \le 10^8$), i-th of these values stands for the value of happiness from visiting city number i.

Then follow e lines, each containing two integers a_i and b_i ($1 \le a_i, b_i \le v, a_i \ne b_i$) — two indices of cities connected by the corresponding route. It's guaranteed that no pair of cities is connected by more than one route.

Output

First line of the output must contain a single integer k ($1 \le k \le 4$) — the number of cities in any optimal trip plan. Second line must contain exactly k indices of cities in the order Petr should visit them. If there is more than one optimal trip, print any.

Examples

tourist.in	tourist.out
5 4 4 2 3 1 5 1 2 2 3 3 4 4 5	4 2 3 4 5
4 3 1 2 3 4 1 2 1 3 1 4	3 4 1 3

,

Note

Tests for this problem are divided into five groups. For each group you earn points only if your solutions manages to pass all tests in this group and all tests in all **previous** groups.

0. Tests 1–2. Tests from the problem statement. This group costs 0 points.
1. Tests 3–16. For all tests in this group condition $v, e \leq 100$ holds. This group costs 20 points.
2. Tests 17–32. For all tests in this group condition $v, e \leq 1000$ holds. This group costs 20 points.
3. Tests 33–53. For all tests in this group conditions $v \leq 3000$ and $e \leq 60\,000$ hold. This group costs 30 points.
4. There are no additional limitations in this group. This group costs 30 point and is **offline**, that is, you will know your score on it only after the end of the contest.

Problem C. Turtle Sprint

Input file:	`turtle.in`
Output file:	`turtle.out`
Time limit:	1 second
Memory limit:	256 megabytes

Pafnuty and his friends are huge fans of board games. They especially like games that require the skill of performing non-trivial calculations in mind. Recently they got a "Turtle Sprint" board game that immediately became the most popular evening activity in their university. The game package consists of the following items:

- Rectangular board containing a grid of size $n \times m$. Each cell of the grid is either free or blocked.
- q constraint cards. Each card contains descriptions of three sets: set A of possible start cells, set B of possible blocked cells and set C of possible finish cells. Sets A, B, C are non-empty, they consist only of free cells of a grid and intersection of any two of these three sets is empty.
- Small piece in form of a turtle.

Rules of the game are pretty simple. On each turn, a constraint card is shown to all players. After that they have to calculate the number of *good* cell triples (a_i, b_j, c_k) where $a_i \in A$, $b_j \in B$ and $c_k \in C$. The triple (a_i, b_j, c_k) is called *good* if it is possible to move turtle from the start cell a_i to the finish cell c_k without visiting the extra blocked cell b_j. The turtle path should satisfy three following constraints:

- Turtle can move only right or down within the board.
- It is prohibited to move through the blocked cells of the board.
- It is also prohibited to move through the cell b_j.

Game package doesn't contain correct answers for all constraint cards and sometimes it is hard to verify the answer given by a player. Help your friends to create a cheat-sheet containing the correct answers for all constraint cards.

Input

First line of the input contains two integers n, m ($1 \le n, m \le 150$), the number of rows and the number of columns of the grid.

Each of the following n lines contains the string of length m with the description of a game board grid in order from top to bottom and from left to right. Each character of the description is either '`.`' denoting the free cell, or '`#`' denoting the blocked cell. Rows are numbered from 1 to n, columns are numbered from 1 to m.

Next line contains an integer q ($1 \le q \le 100\,000$), the number of constraint cards.

In following lines there are q data blocks containing descriptions of each constraint card. Each block consists of three lines corresponding to each of sets A, B and C. Each of these three lines starts with the size s ($s \ge 1$) of the corresponding set followed by the list of $2s$ integers specifying the s cells belonging to the corresponding set. Each cell is specified by two integers, its row number and column number.

It is guaranteed that each set is non-empty, that each cell of each set is a free cell and that each cell of the grid belongs to no more than one set on the card.

It is guaranteed that the total size of all sets on all constraint cards is no more than 300 000: $\sum_{i=1}^{q} (|A_i| + |B_i| + |C_i|) \le 300\,000$.

Also it is guaranteed that the total number of all triples (both good and bad) over all cards doesn't exceed $2 \cdot 10^7$: $2 \cdot 10^7$: $\sum_{i=1}^{q} |A_i| \cdot |B_i| \cdot |C_i| = Q_{total} \leqslant 2 \cdot 10^7$..

Output

For each constraint card output the correct answer for it on a separate line.

Examples

turtle.in	turtle.out
5 6 ..##.. ....#. .#.#.. .#...# ..#... 2 1 1 1 3 2 1 2 3 4 3 1 5 6 2 1 2 2 1 2 3 1 3 3 2 5 1 5 6	1 3

Note

In the sample test there are two constraint cards:

In all triples defined by the first card the turtle starts in the top left corner and finishes in the bottom right corner. It's easy to see that it is possible to find a valid path only if the first cell of the second row is chosen as the blocked cell, i. e. $(1, 1) - (2, 1) - (5, 6)$ is a good triple.

For a second card, the following triples are *good*: $(1, 2) - (3, 1) - (5, 6)$, $(2, 1) - (3, 1) - (5, 6)$, $(2, 1) - (3, 3) - (5, 1)$.

Tests are divided into four groups.

0. Test 1. The sample test, it costs 0 points.
1. Tests 2–18. In tests of this group $n, m \leq 100$, $q_{total} \leq 1000$. Tjos group costs 30 points. You earn points for this group only if you pass all tests of this group.
2. Tests 19–32. In tests of this group $n, m \leq 100$, $q_{total} \leq 1\,000\,000$. This group costs 30 points. You earn points for this group only if you pass all tests of this group and the first group.
3. In tests of this group there are no extra constraints. We guarantee that there will be different values of n and q_{total}, the small ones as well as the large ones, depending on the number of the test. Solution will be graded **offline,** i. e. after the end of the round. You get the points for this group only if you pass all tests of all previous groups. Tests are graded **independently**.

Problem D. Tree "Chapayev" game

Input file:	game.in
Output file:	game.out
Time limit:	2 seconds
Memory limit:	512 megabytes

Vladimir and Marina like to play games and especially they love creating their own rules for existing games. Recently they found out an interesting board game named "Chapayev" in which players should knock opponents pieces off the chessboard by flicking their own pieces. After spending many hours playing this game, they decided to modify its rules making it more complex and formal.

Now they play "Chapayev" but not on the chessboard, they use a board in form of a tree! Their tree consists of n nodes indexed from 1 to n. Node 1 is a *Root* of a tree. From each of the nodes from 2 to n there is an edge leading to its parent which is a node with a strictly smaller index.

In this game they use pieces of the same color, initially located in some of the tree nodes. On his turn, player chooses some piece and then, by flicking it, he sends it towards the root of the tree, knocking out all the pieces on its way. When the chosen piece reaches the root of the tree, it also gets off the table.

Players take alternate moves. If at the beginning of a player's move there are no pieces remaining on the board, he loses.

The described game is notable as it allows many initial piece positions for a same board. Children have found out that the game is most interesting when they put pieces on the board as follows: they choose some node *Root*, and then they put pieces in every node v that is a descendant of *Root*, that is, in every node $v \neq Root$ such that the path between v and 1 in the tree contains *Root* node. Note that they **do not** put a piece in *Root* itself.

Children decided to play n rounds, choosing each possible node as a *Root*. If a node they choose as *Root* doesn't have any descendants, i. e. there are no pieces initially on the board, they do not play a round. In each round Marina starts the game.

Vladimir asks you, in how many rounds Marina will be the winner, if both players play optimally.

Input

The first line contains n ($1 \le n \le 500\,000$), the number of nodes in the tree.

The second line contains $n - 2$ space-separated numbers, $p_2, p_3, \ldots, p_n$, where p_i is an index of a node that is a parent of a node i ($1 \le p_i < i$). There will be an empty line if $n = 1$.

Output

Output the number of rounds in which Marina wins.

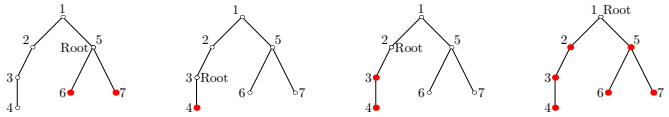
Examples

game.in	game.out
7 1 2 3 1 5 5	3

Note

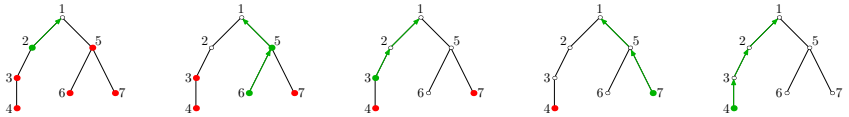
Consider a sample test. The board is shown on pictures below. Children will play four rounds, choosing nodes $1, 2, 3$ and 5 as *Root* node. Choosing any of the three remaining nodes as *Root* leads to the situation when there are no pieces initially on the board, so those round are discarded.

If the node 5 is chosen as *Root* node, pieces are initially located in nodes 6 and 7. In such round Vladimir has a winning strategy. After Marina knocks off any of the two initial pieces, he knocks off the remaining one and Marina loses.



If the node 2 or 3 is chosen as *Root* node, Marina wins by knocking off the piece in the node 4. Note that, if $Root = 2$, the flicked piece will also knock off the piece in the node 3 according to the game rules.

It can be shown that if node 1 is chosen as *Root* node, Marina also has a winning strategy. To win, she must knock off the piece form the node 2 on her first turn. One of the possible rounds with such initial position is shown below.



So, Marina wins in three rounds.

Scoring

Tests are divided into five groups. You earn points for each of the groups only if you pass **all** previous groups.

0. Test 1. The sample test, it is costs zero points.
1. Tests 2–17. In tests of this group $n \leq 20$. This group costs 20 points.
2. Tests 18–38. In tests of this group $n \leq 200$. This group costs 20 points.
3. Tests 39–59. In tests of this group $n \leq 5000$. This group costs 20 points.
4. In tests of this group there are no additional constraints. This group costs 40 points. Your solution will be graded in **offline**, i. e. you will find out your result after the end of the contest.