Problem A. xuanquang1999 and the Problem Selection Process

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

VNOI Cup is a competitive programming contest that requires participants to use their knowledge of algorithms and data structures to solve problems presented in each round. Although it has only been organized for two seasons, the VNOI Cup has attracted a large number of participants. One of the reasons for this is the high quality, uniqueness, and challenging nature of the problems set by the problem setters team. However, selecting problems for a round is not an easy task. The problem setters team has created too many good problems, which has left the coordinator, *Kuroni*, struggling to choose the problems for the final round!

The members of the team have proposed n problems with different topics. The originality of the problems is represented by the values $a_1, a_2, \ldots, a_n$.

To help *Kuroni* reduce the number of problems to consider while maintaining the originality of the problems, *xuanquang1999* suggests that some problems can be replaced by merging them to create a more original problem. Specifically, *xuanquang1999* can perform the following transformations:

- Choose an index i such that $1 \le i < |a|$ and $a_i < a_{i+1}$.
- Create a new problem with originality value of $x = a_i + a_{i+1}$.
- Remove the i-th and $(i+1)$-th problems from the list and insert the new problem with originality value x at position i.
- After removal, the indices of problems starting from $i+2, i+3, \ldots$ will be decreased by 1.

By using these transformations optimally (zero or more times), help *xuanquang1999* determine the minimum number of problems achievable so that *Kuroni* can more easily select the problems for the final round!

Input

The first line contains an integer t ($1 \le t \le 10\,000$) – the number of test cases. The description of each test case follows.

The first line of each test case contains an integer n ($2 \le n \le 200\,000$) – the number of problems proposed by the problem selection committee.

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 uniqueness values of the problems proposed by the problem setters team.

The sum of n over all test cases does not exceed 200 000.

Output

For each test case, print an integer representing the minimum number of problems that *xuanquang1999* can achieve using the given transformations (zero or more times).

Scoring

The score for this problem is 1250 points.

Example

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

Note

In the first example, an optimal way to transform the originality values of the problems is as follows:

Selected index	Array a before transformation		Array a after transformation
Choose $i = 3$:	$[2, 3, \underline{1}, \underline{2}, 4, 3]$	$\rightarrow$	$[2, 3, \underline{3}, 4, 3]$
Choose $i = 1$:	$[\underline{2}, \underline{3}, 3, 4, 3]$	$\rightarrow$	$[\underline{5}, 3, 4, 3]$
Choose $i = 2$:	$[5, \underline{3}, \underline{4}, 3]$	$\rightarrow$	$[5, \underline{7}, 3]$
Choose $i = 1$:	$[\underline{5}, \underline{7}, 3]$	$\rightarrow$	$[\underline{12}, 3]$

In the second example, an optimal way to transform the originality values of the problems is as follows:

Selected index	Array a before transformation		Array a after transformation
Choose $i = 3$:	$[1, 1, \underline{1}, \underline{2}]$	$\rightarrow$	$[1, 1, \underline{3}]$
Choose $i = 2$:	$[1, \underline{1}, \underline{3}]$	$\rightarrow$	$[1, \underline{4}]$
Choose $i = 1$:	$[\underline{1}, \underline{4}]$	$\rightarrow$	$[\underline{5}]$

In the third example, it is not possible to perform any transformations to reduce the number of problems.

Problem B. TrungNotChung and the Competition Preparation

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

To provide the best experience for the contestants, not only does VNOI Cup invest in the quality of the problems, but it also pays attention to the overall competition experience. The organizers of VNOI Cup have worked tirelessly to create the best competition environment for the contestants, making them feel like they are competing at home. In particular, in the XX-th season of VNOI Cup, the organizers allow the contestants to choose their preferred type of desk and monitor to participate in the competition!

In the XX-th season of VNOI Cup, there are n contestants, numbered from 1 to n. Coincidentally, in the city of Ha Long, where VNOI Cup takes place, there are n desk brands and n monitor brands. The i-th contestant wants to use a desk from brand p_i and a monitor from brand q_i ($1 \le p_i, q_i \le n$).

After gathering all the contestants' requests for desks and monitors, the organizers decided that desks purchased from the same brand i must have the same positive integer height t_i. However, to ensure that there is no confusion between desks, desks purchased from **different** brands must have **different** heights. In other words, $t_i \neq t_j$ for $i \neq j$.

Similarly, monitors purchased from the same brand i must have the same positive integer height m_i, and no two monitors purchased from different brands can have the same height ($m_i \neq m_j$ for $i \neq j$).

To ensure fairness in the competition, the organizers want the height from the ground to the top of the monitor, when placed on the desk, to be the same for all contestants. Therefore, for each contestant, the sum of the height of their desk and monitor must be a constant c, i.e., $t_{p_i} + m_{q_i} = c$.

TrungNotChung has been assigned the task of purchasing desks and monitors for the contestants. Help *TrungNotChung* find the heights $t_1, t_2, \ldots, t_n$ of the desks to be purchased and the heights $m_1, m_2, \ldots, m_n$ of the monitors to be purchased, such that the sum of the height of the desk and monitor for each contestant is equal to each other and equal to a constant c. If there are multiple solutions, choose the heights of the desks and monitors such that the constant c is **minimized** to save on the cost of purchase and transportation. If there is still more than one answer, print any of them.

If there is no solution that satisfies the requirements, determine that.

Input

The first line contains an integer n ($1 \le n \le 10^5$) – the number of contestants in the final round of VNOI Cup season XX.

The second line contains n positive integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$) – the desk brands chosen by the contestants.

The third line contains n positive integers $q_1, q_2, \ldots, q_n$ ($1 \le q_i \le n$) – the monitor brands chosen by the contestants.

Output

If there is no way to buy a table that satisfies the requirements of the organizers, print `NO`.

Otherwise,

- On the first line, print `YES`.
- The second line contains n positive integers $t_1, t_2, \ldots, t_n$ ($1 \le t_i \le 10^9$, $t_i \neq t_j$ for $i \neq j$) — where t_i is the height of the table from the i-th brand that the organizers will buy.
- The third line contains n positive integers $m_1, m_2, \ldots, m_n$ ($1 \le m_i \le 10^9$, $m_i \neq m_j$ for $i \neq j$) — where m_i is the height of the screen from the i-th brand that the organizers will buy.

The answer must satisfy the given height condition of the organizers, and the sum of the height of the table and the screen for each participant must be equal to the minimum possible constant c. If there are multiple answers that satisfy the condition, print any of them.

Scoring

The score for this problem is 1250 points.

Examples

standard input	standard output
3 2 1 3 2 3 1	YES 1 2 3 1 2 3
2 1 1 2 1	NO
4 1 2 1 4 2 3 2 1	YES 1 2 4 3 1 3 2 4

Problem C. ngfam the Navigator

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

This is an interactive problem

After purchasing tables and monitors, *TrungNotChung* needs to bring them to the location of the VNOI Cup. Ha Long City, where the VNOI Cup takes place, has n intersections numbered from 1 to n, and there are $n-1$ roads. From one intersection, it is possible to reach any other intersection using some roads in the city. Thus, the roads in Ha Long City form a *tree* structure.

Currently, *TrungNotChung* is standing at intersection 1. Unfortunately, *TrungNotChung*'s phone cannot receive any internet signal, so *TrungNotChung* cannot use the GPS feature. *TrungNotChung* does not know what the map of Ha Long city looks like, but only remembers that the VNOI Cup is held at Ha Long High School for Gifted Students – a location with a prime position as a **centroid** of the city.

Not knowing the destination, *TrungNotChung* immediately calls *ngfam* and asks for directions. *ngfam* also does not know the place where *TrungNotChung* is at looks like, so *ngfam* does not know how to give straight directions to Ha Long High School for Gifted Students. But after some thinking, the two of them agreed to communicate with each other according to the following description.

Let the position where *TrungNotChung* is standing be r, initially $r = 1$. Considering r as the root of the tree, *TrungNotChung* will use the following commands to ask *ngfam*.

Command		Explanation
`adj`	v	Find an intersection adjacent to intersection r on the path from r to v.
`subtree`	v	Find the number of vertices in the subtree rooted at intersection v.
`move`	v	*TrungNotChung* moves to position v with the condition that v is a **neighbor** of r. Then assign $r \leftarrow v$.

Of course, time is precious, so *TrungNotChung* needs to move to Ha Long Specialized High School as soon as possible to install the equipment. With only `15 555` commands, help *TrungNotChung* find a **centroid** of the city.

The city may have multiple centroids, but *TrungNotChung* only needs to find one centroid. When *TrungNotChung* arrives at a centroid that is not the competition venue, at that time *ngfam* also knows where *TrungNotChung* is and will go to pick him up.

An intersection is called a **centroid** of a city with a tree structure consisting of n vertices if, when removing this intersection (and the edges connected to this intersection) from the city, the resulting connected components all have a number of vertices not exceeding $\frac{n}{2}$.

Interaction Protocol

In this problem, the jury program will play the role of *ngfam* – executing the commands of *TrungNotChung*, and your program will play the role of *TrungNotChung* – giving commands to find the centroid of the city.

First, you need to read an integer n ($1 \le n \le 10\,000$) - the number of vertices in the tree.

Next, you need to interact with the jury program using the operations mentioned in the problem statement as follows:

Command	Interaction	
`adj`	Output	`adj` v $(1 \le v \le n)$
	Input	• Integer t when $v \neq r$ – the vertex adjacent to vertex r on the path from r to v. • -1 when $v = r$. You should terminate the program immediately in this case.
`subtree`	Output	`subtree` v $(1 \le v \le n)$
	Input	Integer s - the number of vertices in the subtree rooted at v.
`move`	Output	`move` v $(1 \le v \le n)$
	Input	• 1 when v is adjacent to r. Then r will be assigned to v. • -1 when v is not adjacent to r. You should terminate the program immediately in this case.

When you are certain that vertex r is a centroid of the city, output `found` and terminate the program. This command will not count towards the given operation limit.

If the tree has multiple centroids, move r to any centroid.

The tree is **fixed** before the interaction process and will **not change** during the interaction.

After outputting a command, remember to go to a new line and flush the standard output, otherwise you may receive a `Time limit exceeded` verdict. To do this, you can use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- consult the documentation for other languages.

Scoring

Subtask	Score	Constraints
1	500	$n \le 100$
2	750	$n \le 1000$
3	1000	No additional constraints
Total	2250	

Examples

standard input	standard output
2 1	move 2 found
3 1 3 1	subtree 2 adj 2 move 3 found

Note

In the first example, the city of Ha Long consists of only 2 intersections connected to each other. Both of these intersections are centroid of the city. Therefore, besides moving to intersection 2, it is possible to immediately respond to the jury while being at intersection 1.

In the second example, the city consists of 3 intersections and 2 connecting roads 1–3 and 2–3.

- With the first query, we can conclude that 2 is a leaf intersction.
- With the second query, we can conclude that intersection 1 is not adjacent to intersection 2.

From here, it can be concluded that 3 is the centroid of the city.

Problem D. darkkcyan and Contestant Positions Planning

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

The final round of the VNOI Cup season XX will be held in a room of length n. In the room, there are n evenly spaced positions numbered from 1 to n. At position i, we can place a desk or a network connection device.

For each contestant, we need to choose a pair of positions (u, v) $(1 \leq u < v \leq n)$, one position for placing his desk and one position for placing the network connection device. To ensure the best possible connection for each contestant, the computer on the contestant's desk will be wired to their network connection device. Therefore, when arranging positions for the contestants, the following conditions must be satisfied:

- No two contestants can have desks at the same position.
- No two network connection devices can be placed at the same position.
- The positions of the desks and network connection devices must be different to ensure safety.
- When arranging positions for the contestants, there cannot exist two pairs of positions (u, v) and (u', v') such that $u < u' < v < v'$. This is because if these two pairs of positions exist, the two network cables connecting the pairs of positions will *intersect*, making it difficult for the technical staff to wire them.

darkkcyan is assigned the task of arranging positions for the contestants and network connection devices. Of course, *darkkcyan* needs to do this optimally so that as many contestants as possible can sit in the contest room. During the process of arranging positions for the contestants, there are m events that occur. At the i-th event, a new contestant enters the room and is assigned the pair of positions (u_i, v_i). It is guaranteed that at any given time, the positions of the contestants in the room satisfy the aforementioned requirements.

After each event, help *darkkcyan* calculate the **maximum** number of contestants that can be added to the contest room while still satisfying the requirements.

Input

The first line contains two integers n and m $(2 \leq n \leq 10^9,\ 1 \leq m \leq \min\left\{\frac{n}{2}, 500\,000\right\})$ – the number of positions in the contest room and the number of events.

The i-th line of the next m lines contains two integers u_i and v_i $(1 \leq u_i < v_i \leq n)$ – the position pair of the contestant entering the room in the i-th event.

It is guaranteed that at any given time, the positions of the contestants in the room satisfy the requirements mentioned in the statement.

Output

Print m lines. The i-th line should contain a single integer, the maximum number of contestants that can be added to the contest room while still satisfying the requirements.

Scoring

Subtask	Score	Constraints
1	500	$m \leq 5000$
2	500	$u_i > u_{i+1}$ for all $1 \leq i < m$
3	1250	No additional constraints
Total	2250	

Examples

standard input	standard output
7 3 3 4 5 6 1 7	2 1 0
8 2 5 7 2 4	2 1
100 10 84 95 75 76 71 73 70 78 59 66 58 69 6 11 4 22 3 51 1 79	49 48 46 45 44 43 42 41 40 38

Problem E. lanhf and the VNOI Cup T-Shirt Distribution Process

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

In the XX season, VNOI Cup attracts n participants. The competition consists of m rounds, and to increase the chances of receiving a shirt, all participants participate in all m rounds. In each round, the participants are ranked from 1 to n, and no two participants have the same rank. The *ranking table* for a round is defined as a list of participants in order of their rankings from 1 to n.

After the end of all m rounds, the organizers will choose a number k and distribute shirts to all participants with a rank no greater than k in each round. The organizers want to choose the largest possible value of k, such that the total number of participants receiving shirts does not exceed a given limit s.

For example,

- If $n = 3$, $m = 3$, $s = 2$, the ranking tables for the 3 rounds are
  - $[3, 1, 2]$ (participant 3 has rank 1, participant 1 has rank 2, and participant 2 has rank 3),
  - $[2, 1, 3]$,
  - $[2, 3, 1]$.

  The organizers will choose $k = 1$. In this case, the number of participants who receive shirts is 2, which are participants with indices 3 and 2. The organizers cannot choose $k = 2$ or $k = 3$, as all 3 participants would receive shirts.
- If $n = 2$, $m = 2$, $s = 1$, the ranking tables for the 2 rounds are
  - $[1, 2]$,
  - $[2, 1]$.

  The organizers will choose $k = 0$. In this case, the number of shirts distributed is 0. The organizers cannot choose $k = 1$ or $k = 2$, as both participants would receive shirts.

lanhf is responsible for distributing shirts to the participants. Currently, the competition has gone through $m-1$ rounds. Assuming the result of the last round is completely random, meaning each of the $n!$ different ranking tables has a probability of $\frac{1}{n!}$ to occur. Help *lanhf* calculate the probability for each participant to be on the list of shirt recipients so that *lanhf* can better prepare the shirts for packaging!

Input

The first line contains three positive integers n, m, s ($1 \le n, m \le 2000$, $1 \le s \le n$) — the number of participants, the number of rounds in the competition, and the maximum number of participants who can receive shirts.

The i-th line among the next $m-1$ lines contains n distinct positive integers $r_{i,1}, r_{i,2}, \ldots, r_{i,n}$ ($1 \le r_{i,j} \le n$) — the ranking table for the i-th round.

Output

Print n integers $p_1, p_2, \ldots, p_n$ — where p_i is the probability for the i-th participant to receive a shirt, modulo 998 244 353.

Let $M = 998\,244\,353$. We can represent the answer as a simplified fraction $\frac{p}{q}$, where p and q are integers and $q \not\equiv 0 \pmod{M}$. Print the number corresponding to $p \cdot q^{-1} \bmod M$, or in other words, print the corresponding integer x such that $0 \le x < M$ and $x \cdot q \equiv p \pmod{M}$.

Scoring

Subtask	Score	Constraints
1	500	$n, m \leq 8$
2	1250	$n, m \leq 500$
3	1000	No additional constraints
Total	2750	

Examples

standard input	standard output
3 3 2 3 1 2 2 1 3	0 665496236 665496236
2 2 1 1 2	499122177 0
3 3 3 1 2 3 2 1 3	1 1 1
8 4 6 2 7 1 4 8 6 5 3 2 7 1 3 8 6 5 4 7 2 1 8 3 6 4 5	1 1 516947969 516947969 623902721 623902721 1 516947969
3 1 2	665496236 665496236 665496236

Note

In the first sample test, consider all possible rankings that can occur in the final round:

- $[1, 2, 3]$: the organizers will choose $k = 0$, and no one will receive a shirt.
- $[1, 3, 2]$: the organizers will choose $k = 0$, and no one will receive a shirt.
- $[2, 1, 3]$: the organizers will choose $k = 1$, and the people who receive shirts are 2 and 3.
- $[2, 3, 1]$: the organizers will choose $k = 1$, and the people who receive shirts are 2 and 3.
- $[3, 1, 2]$: the organizers will choose $k = 1$, and the people who receive shirts are 2 and 3.
- $[3, 2, 1]$: the organizers will choose $k = 1$, and the people who receive shirts are 2 and 3.

Therefore, the probability of person 1 receiving a shirt is 0, while the probability of person 2 and 3 receiving shirts is both $\frac{4}{6} = \frac{2}{3}$. Since $665\,496\,236 \cdot 3 \equiv 2 \pmod{998\,244\,353}$, we will print 665 496 236.

In the second sample test, consider all possible rankings that can occur in the final round:

- $[1, 2]$: the organizers will choose $k = 1$, and the person who receives a shirt is 1.
- $[2, 1]$: the organizers will choose $k = 0$, and no one will receive a shirt.

Therefore, the probability of person 1 receiving a shirt is $\frac{1}{2}$, while the probability of person 2 receiving a shirt is 0. Since $499\,122\,177 \cdot 2 \equiv 1 \pmod{998\,244\,353}$, we will print 499 122 177.

In the third sample test, since the maximum number of shirts distributed is equal to the number of participants, everyone will definitely receive a shirt.

In the fourth sample test, the probabilities for each participant are $[1, 1, \frac{27}{56}, \frac{27}{56}, \frac{3}{8}, \frac{3}{8}, 1, \frac{27}{56}]$.

In the fifth sample test, since there is only one round, all participants have an equal probability of receiving a shirt, which is $\frac{2}{3}$.

Problem F1. FireGhost and Perfect Slice 1

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

*In this version, you need to find a cut that **minimizes** the difference in area between the two cake pieces.*

Seeing *lanhf* packing clothes to distribute to the winners of the VNOI Cup in the XX season, *tahp* and his sister *FireGhost* decided to help. Touched by their actions, *lanhf* decided to treat them to a special treat. *lanhf* took *tahp* and *FireGhost* to the most famous bakery in Ha Long city, where the chef *MofK* creates delicious and interesting-shaped cakes. *lanhf* allowed the two siblings to choose any cake they wanted!

After some consideration, the two siblings finally chose a cake in the shape of a convex polygon with n vertices. They decided to share the cake by cutting it into two parts with a single straight cut. Since they are very cooperative, the two cake pieces after the cut must have **equal perimeter**.

FireGhost is fair and equal, so *FireGhost* wants the two cake pieces after the cut to have the **smallest possible difference in area**.

After hearing *FireGhost*'s request, the chef *MofK* is at a loss on how to proceed. Given the coordinates of the n vertices of the cake, help the chef find the corresponding cut that satisfies *FireGhost*'s request.

Input

The first line contains an integer n ($3 \le n \le 20\,000$) – the number of vertices of the cake.

The i-th line among the n following lines contains two integers x_i and y_i ($|x_i|, |y_i| \le 10^6$) – the coordinates of the i-th vertex of the polygon.

The input is guaranteed to satisfy:

- the list of points is given in counter-clockwise order,
- no three consecutive points are collinear,
- the polygon is convex,
- no edge has length exceeding 49.99% the perimeter of the polygon.

Output

Print two pairs of real numbers (x_{f1}, y_{f1}) and (x_{f2}, y_{f2}) – the coordinates of two points on the edge of the cake representing the cut line that satisfies the requirements of *FireGhost* – the difference in area between the two cut pieces of cake should be **as small as possible**.

Definitions:

- x and y are the perimeters of the two cake pieces cut by the line you printed,
- S is the initial area of the cake,
- A is the difference in area between the two cake pieces created by the line you printed,
- B is the difference in area between the two cake pieces created by the corresponding line provided by the jury.

Your answer will be considered **correct** if:

- The two endpoints of the printed line should have a distance to the cake boundary of no more than 10^{-6},
- The absolute error between the perimeters of the two cake pieces should be no more than 10^{-6}, i.e., $|x - y| \leq 10^{-6}$,
- The relative error between A and B compared to the area of the cake should be no more than 10^{-12}, i.e., $\frac{|A - B|}{S} \leq 10^{-12}$.

Scoring

Subtask	Score	Constraints
1	250	$n = 3$
2	750	$3 \leq n \leq 500$
3	750	No additional constraints
Total	1750	

Example

standard input	standard output
3 3 5 0 7 0 3	3 5 0 5

Note

In the example test, since the cake is an isosceles triangle, we can easily divide the cake into 2 parts with equal perimeter and area.

Problem F2. Tahp and Perfect Slice 2

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

*In this version, you need to find a cut that **maximizes** the difference in area between the two cake pieces.*

Seeing *lanhf* packing clothes to distribute to the winners of the VNOI Cup in the XX season, *tahp* and his sister *FireGhost* decided to help. Touched by their actions, *lanhf* decided to treat them to a special treat. *lanhf* took *tahp* and *FireGhost* to the most famous bakery in Ha Long city, where the chef *MofK* creates delicious and interesting-shaped cakes. *lanhf* allowed the two siblings to choose any cake they wanted!

After some consideration, the two siblings finally chose a cake in the shape of a convex polygon with n vertices. They decided to share the cake by cutting it into two parts with a single straight cut. Since they are very cooperative, the two cake pieces after the cut must have **equal perimeter**.

As a really doting brother, *tahp* wants the two pieces of cake after being cut to have the **largest possible difference** in area so that he can give the larger piece to his sister.

After hearing *tahp*'s request, the head chef *MofK* was very confused and didn't know how to handle it. Given the coordinates of the n vertices of the cake, help the head chef find the corresponding cut line that satisfies *tahp*'s request.

Input

The first line contains an integer n ($3 \le n \le 20\,000$) – the number of vertices of the cake.

The i-th line among the n following lines contains two integers x_i and y_i ($|x_i|, |y_i| \le 10^6$) – the coordinates of the i-th vertex of the polygon.

The input is guaranteed to satisfy:

- the list of points is given in counter-clockwise order,
- no three consecutive points are collinear,
- the polygon is convex,
- no edge has length exceeding 49.99% the perimeter of the polygon.

Output

Print two pairs of real numbers (x_{f1}, y_{f1}) and (x_{f2}, y_{f2}) – the coordinates of two points on the edge of the cake representing the cut line that satisfies the requirements of *tahp* – the difference in area between the two cut pieces of cake should be **as large as possible**.

Definitions:

- x and y are the perimeters of the two cake pieces cut by the line you printed,
- S is the initial area of the cake,
- A is the difference in area between the two cake pieces created by the line you printed,
- B is the difference in area between the two cake pieces created by the corresponding line provided by the jury.

Your answer will be considered **correct** if:

- The two endpoints of the printed line should have a distance to the cake boundary of no more than 10^{-6},
- The absolute error between the perimeters of the two cake pieces should be no more than 10^{-6}, i.e., $|x - y| \leq 10^{-6}$,
- The relative error between A and B compared to the area of the cake should be no more than 10^{-12}, i.e., $\frac{|A - B|}{S} \leq 10^{-12}$.

Scoring

Subtask	Score	Constraints
1	250	$n = 3$
2	750	$3 \leq n \leq 500$
3	750	No additional constraints
Total	1750	

Example

standard input	standard output
3 7 0 -7 6 -1 -3	7 0 -5.8547905591911690 4.2821858387867535

Problem G. MofK and Equipment Installation

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

TrungNotChung has moved the equipments to the VNOI Cup competition location. At the same time, *darkkcyan* has planned the positions for the contestants, and *lanhf*, *tahp*, and *FireGhost* have just finished packing the clothes. So, the young team together unloads the equipment from the vehicle, moves them into the competition room, and starts installing them!

There are n pieces of equipment on the vehicle, numbered from 1 to n. The i-th piece has a weight of a_i. To move and install the equipment in a scientific way, the team will move the pieces that need to be installed first into the competition room, and then the other pieces. After observing for a while, the team realizes that for each piece u ($2 \le u \le n$), there exists a piece p_u ($1 \le p_u < u$) that needs to be installed **after** piece u is installed. Therefore, the piece 1 is always the last piece to be installed.

Due to the narrow vehicle body, only one piece can be taken out of the vehicle at a time. If the i-th piece moved out of the vehicle is u, the time to move this piece out of the vehicle is $i \cdot a_u$ time units, as the team gets tired and moves slower the longer they work.

MofK is assigned to supervise the installation of the pieces into the competition room. To ensure the progress of the work, *MofK* needs to arrange and assign the team members to unload and move the pieces into the competition room as quickly as possible. Given the list of piece weights and the list of dependencies between them, help *MofK* find the minimum total time for the team to unload all the items from the vehicle.

Input

The first line contains an integer n ($1 \le n \le 150\,000$) — the number of pieces to be installed in the competition room.

The second line contains n integers separated by spaces $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^8$) — the weights of the pieces.

The third line contains $n - 1$ integers separated by spaces $p_2, p_3, \ldots, p_n$ ($1 \le p_i < i$) — describing the dependencies between the pieces.

Output

Print a single integer — the minimum total time for the team to unload all the pieces from the vehicle.

Scoring

Subtask	Score	Constraints
1	250	$a_{p_i} \le a_i$ for all $2 \le i \le n$
2	1000	$n \le 2000$
3	1000	There exists k such that $3 \le k \le n$ and $p_k = 1$; otherwise, $p_i = i - 1$ for all $2 \le i \le n$, $i \ne k$
4	1000	No additional constraints
Total	3250	

Example

standard input	standard output
8 1 4 15 9 11 5 5 2 1 2 3 3 2 5 5	210

Note

For the first example, the optimal unloading plan for the team is as follows:

$$4 \to 7 \to 8 \to 5 \to 3 \to 6 \to 2 \to 1$$

The total time for this unloading plan is:

$$\begin{aligned} & 1 \cdot a_4 + 2 \cdot a_7 + 3 \cdot a_8 + 4 \cdot a_5 + 5 \cdot a_3 + 6 \cdot a_6 + 7 \cdot a_2 + 8 \cdot a_1 \\ = \; & 1 \cdot 9 + 2 \cdot 5 + 3 \cdot 2 + 4 \cdot 11 + 5 \cdot 15 + 6 \cdot 5 + 7 \cdot 4 + 8 \cdot 1 \\ = \; & 210 \end{aligned}$$

It can be shown that there is no other plan that yields a shorter time.

Problem H. Kuroni and the Sharing of the VNOI Cup Problem Setting Process

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

Appearing in today's live stream session of the VNOI Cup are coordinator *Kuroni* and *RR* – former competitive programmer, who have reached the ICPC World Finals twice and is one of the two Vietnamese contestants who made it to the finals of the Facebook Hacker Cup. Still passionate about competitive programming, *RR* is also the owner of the Vietnamese fan page *Code cùng RR*, which has nearly 10 000 followers and features discussions on competitive programming.

Apart from monitoring and commenting on the solutions of the contestants, *RR* and *Kuroni* also discuss and share their experiences in the problem-setting process. *Kuroni* has many interesting stories about the contest. For example, one member of the problem setters team accidentally reinvented the "Slope trick" technique at some point. Or the story of *Kuroni* having to sadly reject some very good problems simply because they had similar ideas to problems from competitions many years ago. To ensure high-quality and consistent problem statements, all the problems are authored by a single member of the problem setters team...

After a conversation, *RR* has a question:

– With the investment in problem quality to achieve such high standards, can you share with me and the audience the process of problem selection?

– Sure! Since the problem setters team consists of many members, there are a large number of proposed problems. So the challenge for us is not creating problems but rather selecting which ones to include. For example, for the first problem in the set, we proposed m problems, numbered from 1 to m. It is worth noting that the number of proposed problems m is an **odd** number. Each problem i has a difficulty level of a_i. Since we can only choose one problem, we need to eliminate some of the problems. To do this, we perform the following steps until only one problem remains:

- Choose an index i ($1 < i < m$).
- Consider the three consecutive problems with difficulties a_{i-1}, a_i, and a_{i+1}.
- Remove the **easiest** and the **hardest** problems among the three. If there is more than one problem with the same minimum or maximum difficulty, remove any one of them.
- Re-number the problems from 1 to $m - 2$ according to their order. The number of problems m decreases by 2.

With this approach, we ensure that the problems that are too easy or too hard are eliminated, and ultimately only one problem remains. Since this is the finals, to ensure a challenging competition for the contestants, we try to perform the above steps to find the **maximum possible** difficulty level for the problem.

– Oh, it sounds very interesting! But it seems quite complicated. Now, for example, if you have some problems with difficulty levels $a = [7, 8, 3, 1]$, how would you proceed?

– Oh, the number of problems m needs to be **odd**!

– Ah, right! So let's take $a = [7, 8, 3, 1, 5]$. How would you find the suitable problem using the elimination process?

– Well. First, I would choose $i = 4$ and consider the three problems with difficulties $[3, 1, 5]$. The problems with difficulties 1 and 5 would be removed, so we would have the list $a = [7, 8, 3]$. I would then perform

this operation with the remaining 3 elements, resulting in $a = [7]$. Therefore, the problem we choose would have a difficulty level of 7. It can be seen that this is the **maximum** difficulty level that can be found, as there is certainly no way to filter out a problem with a difficulty level of 8.

– Wow, that's amazing!

Fascinated by the problem selection process of this year's VNOI Cup, *RR* wants to continue asking similar questions. *RR* has taken n problems from his personal computer with difficulty levels $b_1, b_2, \ldots, b_n$. Instead of asking about the difficulty level of all the problems, *RR* will ask q questions. The i-th question consists of two numbers (l_i, r_i) $(1 \le l_i \le r_i \le n,\ l_i \equiv r_i \pmod 2)$. *RR* wants to know the difficulty level of the problem that *Kuroni* will choose in the case of $a = [b_{l_i}, b_{l_i+1}, \ldots, b_{r_i}]$.

Knowing that *RR* wants to test the speed and accuracy of the problem setters team, *Kuroni* is ready to answer the questions. However, due to the large number of questions and *Kuroni* being in the live stream session, the use of a computer is limited. Given the list of difficulty levels of *RR*'s problems and the list of questions, help *Kuroni* find the problem with the *maximum* difficulty level using the elimination process described above.

Input

The first line contains two integers n and q $(1 \le n, q \le 100\,000)$ — the number of problems and questions of *RR*.

The next line contains n integers $b_1, b_2, \ldots, b_n$ $(0 \le b_i \le 10^9)$ — the difficulty levels of RR's problems.

The i-th line among the q following lines contains two integers l_i and r_i $(1 \le l_i \le r_i \le n,\ r_i \equiv l_i \pmod 2)$ — the parameters for the i-th question of RR.

Output

Print q lines. The i-th line is the *maximum* difficulty level of the problem that *Kuroni* will choose when using the elimination operation described in the problem statement.

Scoring

Subtask	Score	Constraints
1	500	$n \le 20$, $q \le 1000$
2	1500	$n, q \le 1000$
3	1500	No additional constraints
Total	3500	

Example

standard input	standard output
7 5 0 7 8 3 1 5 9 1 7 2 6 1 3 3 7 3 3	7 7 7 8 8

Note

In the first query, we need to find the answer for the problem sequence $[0, 7, 8, 3, 1, 5, 9]$. We can obtain the problem with difficulty level 7 as follows:

- Choose index $i = 5$ and remove the problems with difficulty levels 1 and 5. The sequence becomes

$$[0, 7, 8, \underline{3, 1, 5}, 9] \to [0, 7, 8, \underline{3}, 9]$$

- Choose index $i = 2$ and remove the problems with difficulty levels 0 and 8. The sequence becomes

$$[\underline{0, 7, 8}, 3, 9] \to [\underline{7}, 3, 9]$$

- Choose index $i = 2$ and remove the problems with difficulty levels 3 and 9. The sequence becomes

$$[\underline{7, 3, 9}] \to [7]$$

Since there is no sequence of operations that can result in a problem with difficulty level greater than 7, 7 is the maximum difficulty level of the problem that *Kuroni* will choose.

In the second query, we need to find the answer for the problem sequence $[7, 8, 3, 1, 5]$. The appropriate operations to find the problem have been explained by *Kuroni* in the problem statement.