Problem A. Adobe Flash Games

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

The year 2021 marks the death of *Adobe Flash*, a multimedia software for animations, applications, games, and so on. Together with it, the era of Flash games (especially from 2005 to 2012) has started to fade away into history. Some good news on this front is that people have already been working on a way to preserve Flash content. Major efforts include *Ruffle*: an open source Flash player written in Rust, and *Flashpoint*: an archive of over 78,000 Flash applications.

In memory of the golden era of Flash games, we present a simplified variant of a famous Flash game called *Factory Balls*. You are tasked with painting a ball in a given pattern. The surface of a ball can be divided into N distinct regions, enumerated with indices from 1 to N. You have K paint cans full of paint, where the i-th paint can has the color i. You are also given M pieces of equipment. Each piece of equipment is specified by a set of regions, and it precisely covers that subset of the regions of the ball.

In the beginning, all regions have color 1 and all pieces of equipment are unequipped. You may perform one of the following actions any number of times:

1. Immerse the ball into the i-th paint can. Every region not covered by any of the pieces of equipment on the ball will be painted with color i.
2. Pick one piece of equipment currently not equipped, and equip it. You can equip multiple pieces of equipment.
3. Pick one piece of equipment currently equipped, and unequip it.

In the end, each region of the ball should have a specific color, and all pieces of equipment should be unequipped. Find the minimum number of actions required to paint the ball, or report that it is impossible.

Input

The first line contains three integers: N, K, and M.

The next line contains N integers. The i-th integer c_i is the desired color of the i-th region.

Each of the next M lines describes a piece of equipment. The first number of each line, r_j, denotes the number of regions the piece of equipment covers, followed by r_j distinct positive integers, denoting the indices of the regions the piece of equipment covers.

- $1 \le N, K \le 10$
- $0 \le M \le 10$
- $1 \le c_i \le K$
- $1 \le r_j \le N$, and for each piece of equipment, all indices of the regions are distinct.

Output

If it's not possible to paint each region of the ball to a given color, output `-1`. Otherwise, output the minimum number of actions required.

Examples

standard input	standard output
3 5 3 1 3 5 1 1 1 2 1 3	6
4 3 2 3 3 2 1 2 1 2 2 2 3	7
4 2 2 1 2 2 1 2 1 2 2 3 4	-1
2 10 0 1 1	0

Note

In the second example, this is the fastest method:

- Immerse the ball into the third paint can.
- Equip the first piece of equipment.
- Immerse it into the second paint can.
- Equip the second piece of equipment.
- Immerse it into the first paint can.
- Unequip both pieces of equipment.

Problem B. Distance Optimizing Triangulation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Consider a planar graph in the shape of a convex polygon with $2N$ vertices. Each vertex is numbered clockwise from 1 to $2N$. Currently, there are $2N$ bidirectional edges along the perimeter of the convex polygon. In other words, for every $1 \le i \le 2N$, vertex i and vertex $(i \pmod{2N}) + 1$ are connected by an edge.

Each vertex is colored with one of N colors numbered from 1 to N. For each color i $(1 \le i \le N)$, there are exactly two vertices with color i. The indices of vertices with color i are x_i and y_i.

We want to add exactly $2N - 3$ bidirectional edges to this graph. After doing so, the following conditions must be satisfied:

- Each edge must connect two different vertices via a straight line segment.
- For any two distinct vertices, there can be at most one edge directly connecting the two vertices.
- The graph must still be planar.

Let $dist(a, b)$ be the length of the shortest path between two vertices a and b. Among all possible edge additions satisfying the above conditions, find one that minimizes $\sum_{i=1}^{N} dist(x_i, y_i)$.

Input

The first line contains a single integer N $(2 \le N \le 200\,000)$.

Then, N lines are given. The i-th line contains two integers x_i, y_i denoting the indices of the two vertices with color i. $(1 \le x_i, y_i \le 2N)$.

Output

On the first line, output the minimum possible value of $\sum_{i=1}^{N} dist(x_i, y_i)$.

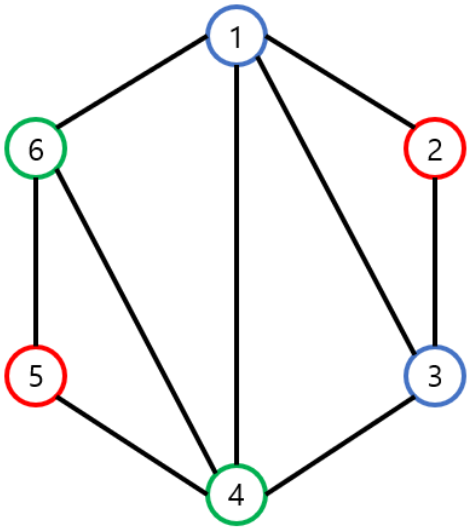
On the next $2N - 3$ lines, output two integers denoting the endpoints of each newly added edge.

If there are multiple answers, any of them will be accepted.

Example

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

Note


1
2
3
4
5
6

Problem C. UCP-Clustering

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	1024 megabytes

Clustering algorithms are algorithms that partition data into subsets named *clusters*, intending to place similar data into the same cluster and different data into different clusters.

Professor Jihoon Ko discovered the revolutionary clustering algorithm *UCP-Clustering*. Given N distinct points in a 2-dimensional space, the algorithm partitions the points into K clusters with the following algorithm.

- Each cluster is assigned a *representative coordinate* (RC) which will be used later. Initially, K points from the given N points are chosen, and each cluster's RC corresponds to one of the chosen points. Even though the coordinates are chosen from the points, all clusters are empty at this point.
- Repeat the following algorithm `recompute`:
  - For each point, insert it into the cluster which minimizes the distance to its RC. If there is more than one such cluster, pick the one with the smallest RC per their lexicographic order.
  - For each cluster, recompute its RC. The x-coordinate of the new RC is the median of the x-coordinates of points in the cluster. Similarly, the y-coordinate is the median of the y-coordinates.
  - If none of the clusters had their RC change, the algorithm terminates. Otherwise, empty all the clusters and run the `recompute` algorithm again (emptying does not reinitialize the RC).

Here, the following definitions are used:

- Point (x_1, y_1) is lexicographically smaller than (x_2, y_2) if and only if $x_1 < x_2$ or $x_1 = x_2$ and $y_1 < y_2$.
- For a sequence of length $n > 0$, the median is defined as the $(n+1)/2$-th largest value if n is odd, and as the **average** of the $n/2$-th and $n/2+1$-th largest values if n is even.

For example, consider the case where $N = 3$ points $\{(1,2),(3,4),(5,6)\}$ are clustered into $K = 2$ sets. Suppose that the points $(1,2)$ and $(3,4)$ are selected as RCs. The first iteration of the `recompute` algorithm clusters the points into $\{(1,2)\}$ and $\{(3,4),(5,6)\}$ with their RCs being $(1,2)$ and $(4,5)$. The second iteration does not change the RCs, thus the `recompute` algorithm is executed twice. Suppose that the points $(1,2)$ and $(5,6)$ are selected as RCs. Then the first iteration of the `recompute` algorithm clusters the point into $\{(1,2),(3,4)\}$ and $\{(5,6)\}$ with their RCs being $(2,3)$ and $(5,6)$. The second iteration does not change the RCs, thus the `recompute` algorithm is executed twice. From the above example, we can see that the initial choice of the RC may change the result of the clustering.

In this problem, let's focus on the case $K = 2$. Suppose that all $N(N-1)/2$ possible initial representative coordinates are chosen for each cluster equiprobably. Please find all the possible sets of representative coordinates at the end of the algorithm, and the expected value of execution count for *recompute* if the algorithm reaches this set of points.

Input

The first line contains a single integer N $(2 \le N \le 512)$.

The next N lines contain two integers x_i and y_i denoting point (x_i, y_i). All points are distinct $(-10^6 \le x_i, y_i \le 10^6)$.

Output

Output all the possible sets of representative coordinates, one per line, in the following format.

Let $(x_1, y_1), (x_2, y_2)$ be the final representative coordinates. Output five real numbers x_1, y_1, x_2, y_2, E where (x_1, y_1) is lexicographically smaller than (x_2, y_2), and E is the expected value of execution count for *compute* if the algorithm reaches this set of points.

The sets must be printed in lexicographic order by (x_1, y_1), with sets that have the same representative coordinate for (x_1, y_1) printed in lexicographic order by (x_2, y_2).

All values must be within an absolute or relative error of 10^{-6}.

The input data are designed such that any choice of initial RC will not make the algorithm reach some state where any of the following conditions is true:

- Some cluster becomes empty.
- The *recompute* algorithm is invoked infinitely many times.
- Two or more clusters have the same RC.

Examples

standard input
4 0 0 0 3 3 0 3 3
standard output
0.000000 0.000000 3.000000 3.000000 1.000000000000 0.000000 1.500000 3.000000 1.500000 2.000000000000 0.000000 3.000000 3.000000 0.000000 1.000000000000 1.500000 0.000000 1.500000 3.000000 2.000000000000
standard input
3 1 2 3 4 5 6
standard output
1.000000 2.000000 4.000000 5.000000 2.000000000000 2.000000 3.000000 5.000000 6.000000 2.000000000000

Problem D. Triple Sword Strike

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	1024 megabytes

There are N monsters on a two-dimensional plane. Each monster has a value associated with it.

A *sword strike* is an action that slays all monsters along a line. If you slay a monster, the monster disappears. The line must be parallel to one of the coordinate axes.

Compute the maximum sum of values of monsters that you can slay given that you can perform up to three sword strikes.

Input

In the first line, a single integer N is given ($1 \le N \le 300\,000$).

Each of the next N lines contains three integers x, y, v, indicating there is a monster located at (x, y) with value v ($0 \le x, y \le 1\,000\,000$, $1 \le v \le 7\,000$).

All monsters are at distinct locations.

Output

Output the maximum sum of values of monsters that you can slay given that you can perform up to three sword strikes.

Examples

standard input	standard output
10 1 1 8 1 4 1 1 5 9 2 3 2 2 4 1 3 1 9 3 2 9 3 4 4 4 3 3 5 4 7	48
8 1 0 1 1 1000000 1 2 1 1 2 999999 1 3 2 1 3 999998 1 4 3 1 4 999997 1	6
1 1 1 3	3

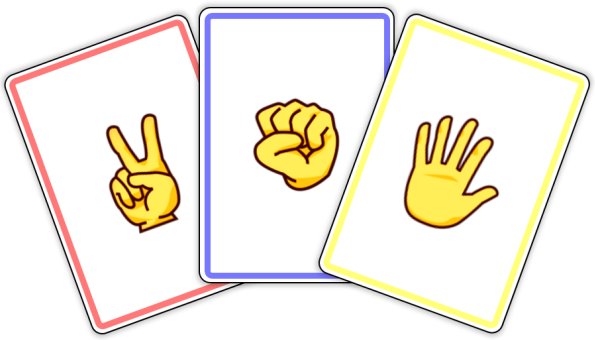
Problem E. RPS Bubble Sort

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

Yihwan is a 5-year-old genius who enjoys rock-paper-scissors. He wanted to play rock-paper-scissors alone at his house, so he showed his creativity and made a game using some cards.

Yihwan places N cards, each with one of scissors, rock, or paper, on positions $1, 2, \ldots, N$. Then, for each of $i = 1, 2, \ldots, N-1$ in increasing order, Yihwan swaps the card at positions i and $i+1$ if the card in the i-th position beats the card in the $i+1$-th position. As the process of the game is similar to bubble sort, Yihwan named this game 'Rock-paper-scissors Bubble Sort'. The win-lose relationship of the cards is as follows:

- A card with scissors beats a card with paper.
- A card with paper beats a card with rock.
- A card with rock beats a card with scissors.



Yihwan is very good at calculating, so he played this game T times and went to sleep. However, while Yihwan was sleeping, the cards were accidentally shuffled. Fortunately, since Yihwan is a genius with excellent memory, he remembered the arrangement of the initial cards before the game. However, he couldn't remember the final arrangement of cards after T games and now needs your help.

Please help Yihwan by restoring the final arrangement of cards after playing the game T times.

Input

The first line contains two integers denoting the number of cards N ($2 \le N \le 500\,000$) and the number of times T ($1 \le T \le 1\,000\,000\,000$) Yihwan played the game.

The second line contains a string of length N. The i-th character indicates the type of card placed at the i-th position. A card with rock is `R`, a card with scissors is `S`, and a card with paper is `P`.

Output

Output the string of length N after playing the game T times.

Examples

standard input	standard output
5 1 RSPSP	SRPPS
10 3 RSRRRRRRSR	SRRRRSRRRR

Problem F. Stones 1

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

N stones labeled from 1 to N are arranged in a row in increasing order. Each stone is colored either white or black. The weight of the i-th stone is A_i.

You will remove the stones one at a time until all the stones are removed.

When removing a stone, if the stone is not the leftmost or rightmost of all remaining stones, and neither stone adjacent to the stone being removed matches it in color, your score increases by the weight of the stone being removed. Two stones are adjacent if there are no stones in between those two.

Find a way to remove the stones to maximize your score.

Input

The first line contains a single integer N ($1 \le N \le 300\,000$).

The second line contains a string S of length N where each character is either `B` or `W`. The ith character of S, S_i, is `B` if the i-th stone is black, otherwise, S_i is `W` and the i-th stone is white.

The third line contains N integers $A_1, A_2, \cdots, A_N$ ($1 \le A_i \le 10^9$). A_i represents the weight of stone i.

Output

Output the maximum score that can be obtained if you take the stones optimally.

Examples

standard input	standard output
4 WBWB 6 4 5 3	5
8 WBBWBWBB 6 4 8 2 5 3 1 5	13

Problem G. Stones 2

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	1024 megabytes

N stones labeled from 1 to N are arranged in a row in increasing order. Each stone is colored either white or black. The weight of the i-th stone is A_i.

You will remove the stones one at a time until all the stones are removed.

When removing a stone, if the stone is not the leftmost or rightmost of all remaining stones, and neither stone adjacent to the stone being removed matches it in color, your score increases by the weight of the stone being removed. Two stones are adjacent if there are no stones in between those two.

There are $N!$ possible ways to remove all the stones. Compute the sum of scores from each possible way, modulo 998 244 353.

Input

The first line contains a single integer N ($1 \le N \le 300\,000$).

The second line contains a string S of length N where each character is either `B` or `W`. The ith character of S, S_i, is `B` if the i-th stone is black, otherwise, S_i is `W` and the i-th stone is white.

The third line contains N integers $A_1, A_2, \cdots, A_N$ ($1 \le A_i \le 10^9$). A_i represents the weight of stone i.

Output

Output the sum of the scores over all possible ways to take the stones, modulo 998 244 353.

Examples

standard input	standard output
4 WBWB 6 4 5 3	72
8 WBBWBWBB 6 4 8 2 5 3 1 5	218304

Problem H. Beacon Towers

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

There are N villages in a row, numbered from 1 to N. Village 1 is the leftmost village and village N is the rightmost village. The heights of the villages are distinct integers ranging from 1 to N.

You would like to divide the villages into several segments. Each segment must contain at least one village and every village should belong to exactly one segment. If two villages are in the same segment, then all villages in between them must also be in that segment.

In each segment, a beacon will be installed in the village with the highest height. For efficient mutual communication among beacon towers, the heights of the villages where the beacon towers are installed must form an increasing sequence from left to right.

Please count the number of possible segment divisions that will cause the beacon towers to satisfy the given constraints.

Input

The first line contains an integer N representing the number of villages ($1 \le N \le 500\,000$).

The second line contains N integers $h_1, h_2, \cdots, h_N$ representing the heights of villages $1, 2, \ldots, N$. ($1 \le h_i \le N$). All h_i's are distinct.

Output

Count the number of possible segment divisions that will cause the beacon towers to satisfy the given constraints. As the answer could be large, please output the answer modulo $10^9 + 7$.

Examples

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

Problem I. Marbles

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	32 megabytes

As a child, Jeyeon had N marbles numbered from 1 to N and infinitely many bags. Each marble is either red or blue. Each marble belonged to its own bag by itself originally.

Jeyeon kept a diary of every change he made to the marbles. The diary consists of M entries. There are three different types of entries:

- `1 i j` : Combine the bag containing marble i and the bag containing marble j. $(1 \le i, j \le N)$
- `2 i` : Lose marble i forever. $(1 \le i \le N)$
- `3 i l h` : Observe that the bag containing marble i contained at least l and at most h red marbles. $(1 \le i \le N, 0 \le l \le h \le N)$

10 years later, Jeyeon, looked at his diary and tried to restore the color of each of his marbles. Please construct a sequence of colors that satisfies all entries of his diary, or state that no such sequence exists.

Input

The first line contains two integers N and M $(2 \le N \le 2\,000,\ 1 \le M \le 4\,000)$.

The next M lines contain the diary entries per the format described above.

The input data are designed such that the marbles i, j in any type-1 entry were in different bags prior to the entry, and the marble lost in the type-2 entry will never be referenced in any subsequent entry.

Output

On the first line, output `YES` if there is a sequence of colors of the N marbles that satisfies all M entries. Otherwise, output `NO`.

If the answer is `YES`, output a string of length N on the next line. The ith character of the string must be `R` if marble i is red, or `B` if marble i is blue. If there are multiple answers, any of them will be accepted.

Examples

standard input	standard output
5 9 1 2 4 1 1 5 3 4 1 2 3 5 0 1 1 4 1 3 4 2 5 2 1 3 4 0 1 3 3 1 1	YES RBRRB
3 4 1 1 2 3 2 1 2 1 2 3 3 3 0 0	NO

Problem J. Exam

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

Professor Deokin lectures on algorithms. For the upcoming midterm exam, he decided to add the maximum subarray sum problem taught in his lecture, which is the following problem:

- Given the array $A = [a_1, a_2, \ldots, a_n]$, compute the value $MSS(A) = \max_{1 \le i \le j \le n}(\sum_{k=i}^{j} a_k)$

Deokin wants to force the students to solve the maximum subarray sum problem by hand. For this, he uses the following procedure.

- Prepare a two-dimensional array of integers of size $N \times N$.
- Start from the cell $(1, 1)$, and repeatedly move either right or down to reach the cell (N, N). If the current cell is (i, j), he can move right to the cell $(i, j + 1)$ if $j < N$, or move down to the cell $(i + 1, j)$ if $i < N$.
- Write a sequence of length $2N - 1$ which contains the elements of the cells in the order that the cells were visited.

Professor Deokin wants to add an inside joke to the problem so that the students who paid attention to his lectures can have confidence in their answers. He wants to create a sequence where the value of $MSS(A)$ is exactly K. Given a two-dimensional array of size $N \times N$, you need to find the number of different paths in the grid which yield sequences that have a maximum subarray sum value of exactly K.

Input

The first line contains two integers N denoting the size of the two-dimensional array, and K denoting the desired value ($1 \le N \le 20, -4 \times 10^{10} \le K \le 4 \times 10^{10}$).

The next N lines contain N integers denoting the values in the two-dimensional array. The j-th integer in the i-th line denotes the value $A_{i,j}$. All indices used in the array are 1-indexed ($-10^9 \le A_{i,j} \le 10^9$).

Output

Output a single integer denoting the number of different paths in a grid which yield a sequence with a maximum subarray sum value of exactly K.

Examples

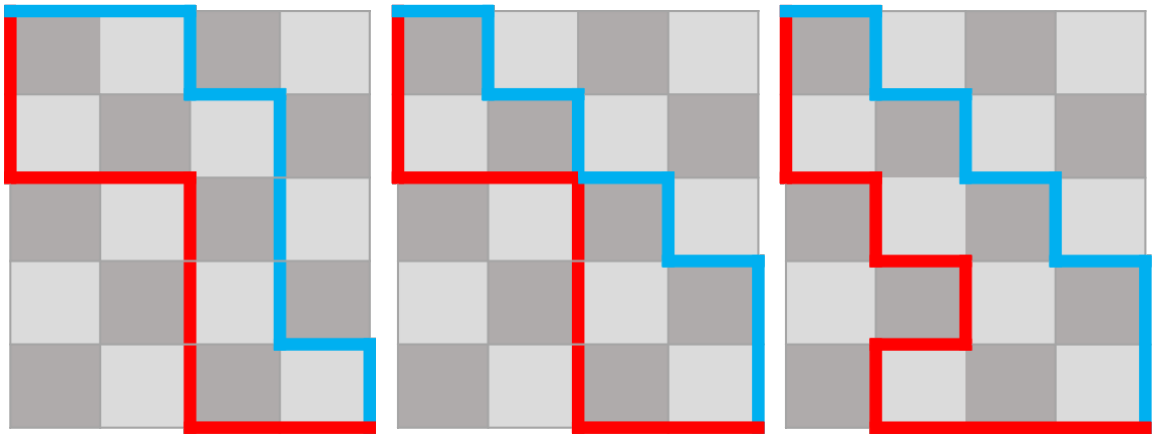
standard input	standard output
3 3 1 2 -5 -2 3 0 -1 -1 1	2
4 3 1 -1 1 1 1 1 1 1 1 1 1 1 1 1 -1 1	4

Problem K. Board Game

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

Jeyeon and Deokin are addicted to problem-solving and they never want to retire. They developed a new board game, which they decided to not retire before they fully analyze.

The game consists of K boards and K pieces. The $i(i = 1, \cdots, K)$th board is a grid of size $N_i \times M_i$. For each board, there are two *fences* that form a boundary. Fences are paths of length $N_i + M_i$ that start from the upper-left **vertex** and ends at the lower-right **vertex**, and can be traced by moving only right (`R`) or down (`D`) in steps of length 1. The two fences do not meet except at the upper-left and lower-right vertices.



For example, consider the case where $N_i = 5, M_i = 4$. In the case of (a), the two paths `RRDRDDDRD`, `DDRRDDDRR` do not meet except at the upper-left and lower-right vertices, so this can be a valid input. On the other hand, in the case of (b), the selected two fences `RDRDRDRDD`, `DDRRDDDRR` meet at a vertex other than the upper-left and lower-right vertices, so it is an invalid input. In the case of (c), one of the fences cannot be traced by moving only right and down, so it is also an invalid input.

The game starts with K boards, and there are two fences on every board. Each board has a single piece that is placed on the upper left **square**. Jeyeon starts first, and then they alternate turns. In Jeyeon's turn, Jeyeon must select one of the K pieces and move it to the square directly to the **right** of the current square; In Deokin's turn, Deokin must select one of the K pieces and move it to the square directly **below** the current square; It is not allowed to move a piece across the fence. The player who is unable to make a move on their turn loses the game.

You are an avid problem solver and have a desperate desire to win. However, since Jeyeon and Deokin are very strong, you can never beat them. However, if you perfectly analyze this game, maybe they can consider retiring?

Given the K boards, please determine the winner of the game if both players play optimally.

Input

The first line contains a single integer K $(1 \le K)$.

The next $3K$ lines contains the information of the $i = 1, 2, \ldots, K$-th grids in order.

Each grid is represented by three lines. The first line contains two integers N_i, M_i. The next two lines contain the direction sequence of each of the fences, which is a string of length $N_i + M_i$ consisting of only `R` or `D` $(1 \le N_i,\ 1 \le M_i)$.

The sum of $N_i + M_i$ does not exceed 500 000.

Output

Output `First` if Jeyeon (the first player) wins. Output `Second` if Deokin (the second player) wins.

Examples

standard input	standard output
2 3 2 RRDDD DRDDR 1 2 DRR RRD	First
2 2 2 DRDR RRDD 4 5 RDRDDRRRD DDDRDRRRR	Second

Note

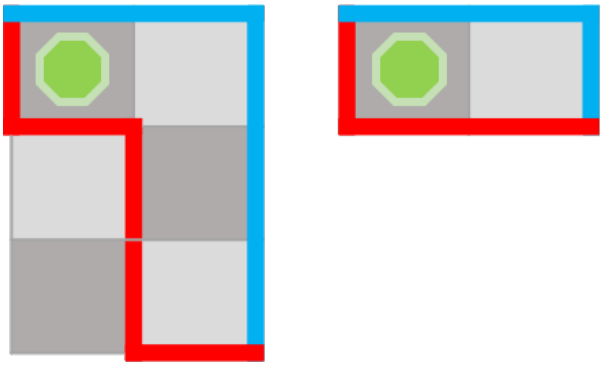
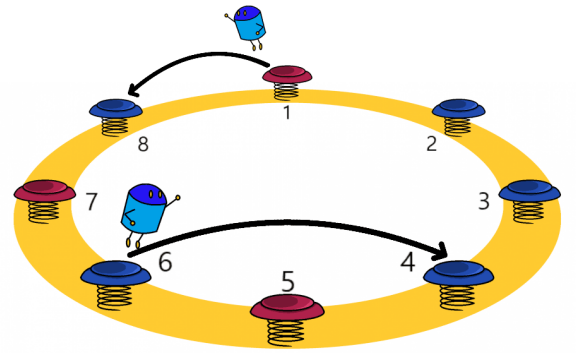


Illustration of sample test 1.

Problem L. Make Different

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

There are N springs on a circular game board. There are two types of springs: red springs and blue springs. When the robot steps on a red spring, it can jump one spring left or right. When the robot steps on a blue spring, it can jump two springs left or right.



The case when the two robots are on springs 1 and 6 respectively, and a counterclockwise jump command is given.

You will play a game using two robots. In the beginning, you will place the two robots on different springs on the board. You can then issue a direction - either clockwise or counterclockwise. Both robots will jump simultaneously in the direction you command. The game ends when the robots step on springs of different colors.

You are given Q queries. Each query contains the starting springs of the two robots. For each query, find the minimum number of commands needed to finish the game.

Input

The first line contains two integers N and Q ($3 \le N \le 100\,000$, $1 \le Q \le 100\,000$).

The next line contains N integers, the i-th integer denoting the type of spring i. Red springs are denoted by a 1 and blue springs are denoted by a 2.

The next Q lines each contain two integers p_1 and p_2 denoting the positions at which the two robots will start the game ($1 \le p_1, p_2 \le N$, $p_1 \ne p_2$).

Output

Output Q lines. On the i-th line, output a single integer denoting the minimum number of commands required to get the two robots to land on different colored springs for the i-th query. If it is impossible to get the robots to land on different colored springs, output `-1`.

Example

standard input	standard output
8 3 1 2 2 2 1 2 1 2 1 2 1 5 3 6	0 -1 1

Problem M. Short Question

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	1024 megabytes

I have a question, could you please answer it?

You are given two sequences $(p_1, p_2, \ldots, p_N)$, $(q_1, q_2, \ldots, q_N)$ of length N. What is the value of this sum?

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \min(|p_i - p_j|, |q_i - q_j|)$$

Input

In the first line, a single integer N is given ($1 \le N \le 1\,000\,000$).

In the second line, N integers $p_1, p_2, \ldots, p_N$ are given ($1 \le p_i \le 1\,000\,000$).

In the third line, N integers $q_1, q_2, \ldots, q_N$ are given ($1 \le q_i \le 1\,000\,000$).

Output

Output a single integer, the answer to the question.

Examples

standard input	standard output
3 1 3 2 1 2 3	6
4 1 1 1000000 1000000 1000000 1000000 1 1	7999992