Problem A. Letters Swap

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

You are given a non-empty sequence of letters "a", "b", and "c". Find the number of ways to swap two **distinct** letters (distinct by value, you are **not allowed** to pick two equal letters at distinct positions) in this sequence so that the resulting sequence is *correct*.

A correct sequence of letters is defined as follows:

- An empty sequence is correct.
- If A is a correct sequence, then aAa, bAb, and cAc are correct as well.
- If A and B are correct sequences, then AB (concatenation of A and B) is correct as well.
- Sequences that cannot be obtained by rules above are not correct.

Input

The only line contains a non-empty sequence of characters "a", "b", and "c". The length of the sequence does not exceed 100 000 characters.

Output

Print one number — the number of ways to swap two distinct letters so that the resulting sequence is correct.

Examples

standard input	standard output
abba	2
abcabc	6
aaba	0

Note

In the first sample we can obtain correct sequences "aabb" and «bbaa» by swapping letters. Note that the initial sequence may be correct, but it is not allowed to leave it unchanged.

In the second sample we can obtain correct sequences "abbacc", "abccba", "accabb", "aacbbc", "bbcaac", and "cbaabc".

In the third sample the number of "a" letters is odd, hence making a correct sequence is impossible.

Problem B. Big Data

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

You are given an array a of length n, and an array b of length m. All elements of both arrays are digits from 0 to 9. Using these arrays, construct a table c of size $n \times m$, where the element in the i-th row and the j-th column is determined by $c_{i,j} = a_i \cdot 10^9 + b_j$.

Consider all paths from the cell $c_{1,1}$ to the cell $c_{n,m}$ that consist exclusively of moving down and to the right (that is, from the cell $c_{i,j}$ you can only move to cells $c_{i+1,j}$ and $c_{i,j+1}$, provided that these cells are inside the table boundaries). Out of all these paths, find one with the largest possible sum of numbers in the visited cells, and print this sum.

Input

The first line contains two integers n and m — sizes of arrays a and b respectively ($1 \le n, m \le 100\,000$).

The second line contains n integers $a_1, \ldots, a_n$ ($0 \le a_i \le 9$).

The third line contains m integers $b_1, \ldots, b_m$ ($0 \le b_j \le 9$).

Output

Print one integer — the largest sum in visited cells among all paths satisfying rules above in the table c.

Examples

standard input	standard output
3 4 1 1 1 1 1 1 1	6000000006
5 3 1 2 3 4 5 6 7 8	25000000045
7 4 0 7 1 7 6 7 6 4 1 9 7	55000000068

Note

In the second sample, the optimal path first goes down from cell $(1, 1)$ to cell $(5, 1)$, then goes to the right till the end at the cell $(5, 3)$.

In the third sample the optimum path is: $(1, 1) \to (2, 1) \to (2, 2) \to (2, 3) \to (3, 3) \to (4, 3) \to (5, 3) \to$ $\to (6, 3) \to (6, 4) \to (7, 4)$.

Problem C. World of Darkraft 3

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Roma plays a new version of the "World of Darkraft" game as a wizard character. Roma values his time and tries to complete in-game tasks as efficient as possible.

Roma is now fighting n monsters. The i-th of these monsters has h_i health points. A monster dies when his health points become zero of negative (at that moment Roma's character gains experience).

Roma's character has a pool of m mana points. The character possesses two spell abilities: "Lightning strike" and "Ring of ice".

- "Lightning strike" spell deals d_s damage to a chosen monster (that is, descreases its health points by d_s), and one cast of this spell costs c_s mana points.
- "Ring of ice" spell deals d_a damage to **all** monsters still alive, and one cast of this spell costs c_a mana points.

The character can not use a spell if the number of mana points left is strictly less than the cost of this spell. The battle finishes as soon as all monsters are dead, or if it is not possible to cast any of two spells (then the character needs to retreat and rest).

In this battle Roma wants to gain as much experience as possible. Help Roma determine the maximum possible number of monsters he can kill.

Input

The first line contains six integers n, m, d_s, c_s, d_a and c_a ($1 \le n, m, d_s, c_s, d_a, c_a \le 100$) — the number of monsters, the amount of characters' mana, and also damage and mana cost of "Lightning strike" and "Ring of ice" spells respectively.

The second line contains n integers $h_1, h_2, \ldots, h_n$ — monsters' health points ($1 \le h_i \le 100$).

Output

Print one number — the maximum number of monsters that can be defeated.

Examples

standard input	standard output
3 5 1 1 3 3 4 1 5	2
5 70 10 10 1 20 11 11 11 11 11	5

Note

In the first sample, the character can cast "Ring of ice" once (as a result, the first monster will have one health point, the third monster will have two health points, and the second monster will die). After that, the character can apply "Lightning strike" twice to kill another monster.

In the second sample, the only way to defeat all monsters is to cast "Ring of ice" once, and "Lightning strike" once for each monster.

Problem D. Sports Analytics

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	512 megabytes

n athletes are going to take part in the Berlandian multisport championship. The competition will consist of k stages, and in each stage an athlete can score any real number of points in range from 0 to 1.

Little boy Oleg loves sports analytics, and he is going to highlight the championship in his sports blog. Oleg doesn't know anything about the competitors, or how the competition is arranged. Though, Oleg knows probability theory very well, so he is going to assume that each athlete scores random number of points equidistributed on the segment $[0, 1]$ in each stage. Oleg also assumes that scores of all athletes on all stages are mutually independent.

The most important part of analytics is to comprise a final rating list, that is, order all athletes from the best to the worst. After studying hundreds of different rating systems, Oleg decided to upload all possible *fair* rating lists. Oleg thinks that in a *fair* rating list any athlete (except for the last) scored strictly greater number of points than the **subsequent** athlete in at least one stage of the competition.

Note that in a fair rating list one athlete can still be higher than another athlete even if he lost him in all stages. Consider a situation where in a two-stage competition the scores of the athlete 1 are $(0.2, 0.2)$, the scores of the athlete 2 are $(0.8, 0.8)$, and the scores of the athlete 3 are $(0, 1)$. Then, the rating table $(1, 3, 2)$ is fair, since the athlete 1 beat the athlete 3 in the first stage, and the athlete 3 beat the athlete 2 in the second stage, however, the athlete 1 lost to the athlete 2 in both.

The championship has not started yet, but Oleg wants to reserve enough space for the rating tables on his hosting. To do this, he asked you to find the average number of ways to comprise a fair rating list at the end of the competition. Oleg also wants to train his number theory skills, so he wants the answer modulo $P = 998\,244\,353$, that is, if the answer is an irreducible fraction A/B, you have to output such an integer x that $0 \le x < P$ and $A \equiv B \cdot x \pmod{P}$ (it is guaranteed that such x exists and is unique).

The average value of an integer-valued random variable is defined as $\sum_x xp_x$, where the sum is over all integers, and p_x is the probability of the variable begin equal to x.

Input

The only line contains two integers n and k — the number of athletes and stages in the competition respectively $(1 \le n, k \le 1000)$.

Output

Print one number — the average number of fair rating lists modulo 998 244 353.

Examples

standard input	standard output
10 1	1
1 10	1
2 2	499122178
5 3	778699985

Note

In the first sample, there is only one fair rating list for every outcome, that is, sorted by points in the only stage in decreasing order (except for the case when several athletes share the same score, but the probability of this event is zero).

In the second sample, there is only one fair list that consists of one athlete.

In the third sample the answer is $\frac{3}{2}$, and in the fourth sample the answer is $\frac{163}{36}$.

Problem E. Bonsai

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Vasily likes bonsasi — the art of growing decorative trees. He has two beatiful trees, but his aesthetical sense tells him that these trees should be made as similar as possible.

Consider the task formally. Let us represent each tree with a connected graph without cycles with a specified *root* vertex. Vertices of both trees are numbered starting from one, and the root of each tree has number 1 among the vertices of this tree. Let us direct all edges away from the root of the tree. All edges starting at the same vertex are ordered (from left to right when looking at the tree from the front) by increasing of the index of vertices where the edges are directed to.

Vasily can perform two kinds of operation with either of the trees:

- Cut a *leaf* vertex from any tree, that is, a vertex without outgoing edges. The root of the tree can not be cut.
- Take two **neighboring** edges leaving the same vertex, and carefully *tie* them together with a piece of string. The ends of these edges effectively become one vertex, and all edges previously leaving the ends of the tied edges now leave this vertex instead. The order of the edges leaving the new vertex is defined as follows: all edges leaving the end of the left edge, followed by all edges leaving the end of the right edge. The relative order of edges in both lists is unchanged. Two edges that were tied are now considered as a single edge and two vertices are considered as a single vertex, thus they can be a part of a leaf cut or tie operation during next steps.

Vasily thinks that the trees have become equal if it is possible to put vertices, as well as edges of both trees in one-to-one correspondence so that:

- For any pair of corresponding vertices the number of edges leaving these vertices is the same, and all this edges are pairwise corresponding with respect to their order.
- The ends of corresponding edges are corresponding as well.

In particular, it follows from the rules that the roots of the trees must be corresponding.

Of course, Vasily doesn't want to ruin his trees too much. What is the smallest number of operations needed to make the trees equal?

Input

The first line contains an integer n — the number of vertices of the first tree ($1 \le n \le 5000$).

The second line contains integers $p_2, \ldots, p_n$ ($1 \le p_i < i$). Here, the number p_i is the number of the *parent* vertex of a non-root vertex i — the start vertex of the unique edge entering i.

The third line contains an integer m — the number of vertices of the second tree ($1 \le m \le 5000$).

The fourth line describes the parents of vertices $2, \ldots, m$ of the second tree in the same format as described above.

Output

Print one number — the smallest number of operations needed to make the trees equal.

Examples

standard input	standard output
3 1 1 3 1 2	2
7 1 1 2 2 3 3 7 1 1 2 3 2 3	0
1 5 1 2 1 4	4

Note

In the first sample it is possible to reach the goal in two operations: for example, we can cut the leaf vertex 3 of the first tree, and tie the edges $(1, 2)$ and $(1, 3)$ in the second tree.

In the second sample the trees are already equal regardless of the fact that the leaves are numbered differently.

In the third sample we have to successively cut all vertices of the second tree except for the root. Note that if a tree has only one vertex, the second line of its description is empty.

Problem F. Alfred and Georg

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	512 megabytes

Mathematicians Alfred and Georg play a game. They take a white rectangular sheet of paper of size $n \times m$ divided into unit square cells. First, Alfred paints some of the cells black, and then Georg has to paint all the cells white again. To do that, Georg can perform the following operation an arbitrary number of times:

- Put a pencil to the bottom left corner of the sheet (let us assume that this corner has coordinates $(0, 0)$), and draw a path to the top right corner of the sheet (that is, to the point with coordinates (n, m)). The path should go along the borders of the cells, and it is only allowed to move the pencil up or to the right. That is, if the pencil is located at coordinates (x, y) Georg is only allowed to move it to positions $(x + 1, y)$ and $(x, y + 1)$ and only if it will remain inside the sheet of paper or on its border.
- Change the color of all cells that are located under the path (that is, repaint black all white cells under the path, and repaint white all black cells). After that, erase the drawn path, and (if necessary), proceed to the next operation.

While Georg is working, Alfred mentally tries to count the smallest number of operations needed to finish the game starting from the initial configuration. He asked you to write a program that would answer this question.

Input

The first line contains three integers n, m and k ($1 \le n, m \le 500\,000$, $0 \le k \le 500\,000$) — the dimensions of the sheet, and the number of black cells respectively.

The subsequent k lines describe the black cells. The i-th of these lines contains two integers x_i, y_i — coordinates of the bottom left corner of the i-the black cell in the coordinate system described above ($0 \le x_i < n$, $0 \le y_i < m$). It is guaranteed that all cells in the input are distinct.

Output

Print one number — the smallest number of operations needed to finish the game. If it is impossible to paint the whole sheet white, print -1.

Examples

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

Note

In first sample one path is enough: $(0, 0) \to (0, 1) \to (1, 1) \to (1, 2) \to (2, 2)$.

In the second sample two paths are enough: $(0, 0) \to (1, 0) \to (1, 1) \to (2, 1) \to (2, 2)$ and $(0, 0) \to (0, 2) \to (2, 2)$.