Codeforces Round #198 (Div. 2)

A. The Wall

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Iahub and his friend Floyd have started painting a wall. Iahub is painting the wall red and Floyd is painting it pink. You can consider the wall being made of a very large number of bricks, numbered 1, 2, 3 and so on.

Iahub has the following scheme of painting: he skips $x - 1$ consecutive bricks, then he paints the x-th one. That is, he'll paint bricks x, $2 \cdot x$, $3 \cdot x$ and so on red. Similarly, Floyd skips $y - 1$ consecutive bricks, then he paints the y-th one. Hence he'll paint bricks y, $2 \cdot y$, $3 \cdot y$ and so on pink.

After painting the wall all day, the boys observed that some bricks are painted both red and pink. Iahub has a lucky number a and Floyd has a lucky number b. Boys wonder how many bricks numbered no less than a and no greater than b are painted both red and pink. This is exactly your task: compute and print the answer to the question.

Input

The input will have a single line containing four integers in this order: x, y, a, b. ($1 \leq x, y \leq 1000$, $1 \leq a, b \leq 2 \cdot 10^9$, $a \leq b$).

Output

Output a single integer — the number of bricks numbered no less than a and no greater than b that are painted both red and pink.

Sample test(s)

input
2 3 6 18
output
3

Note

Let's look at the bricks from a to b ($a = 6$, $b = 18$). The bricks colored in red are numbered 6, 8, 10, 12, 14, 16, 18. The bricks colored in pink are numbered 6, 9, 12, 15, 18. The bricks colored in both red and pink are numbered with 6, 12 and 18.

B. Maximal Area Quadrilateral

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Iahub has drawn a set of n points in the cartesian plane which he calls "special points". A quadrilateral is a simple polygon without self-intersections with four sides (also called edges) and four vertices (also called corners). Please note that a quadrilateral doesn't have to be convex. A special quadrilateral is one which has all four vertices in the set of special points. Given the set of special points, please calculate the maximal area of a special quadrilateral.

Input

The first line contains integer n ($4 \leq n \leq 300$). Each of the next n lines contains two integers: x_i, y_i ($-1000 \leq x_i, y_i \leq 1000$) — the cartesian coordinates of ith special point. It is guaranteed that no three points are on the same line. It is guaranteed that no two points coincide.

Output

Output a single real number — the maximal area of a special quadrilateral. The answer will be considered correct if its absolute or relative error does't exceed 10^{-9}.

Sample test(s)

input
5 0 0 0 4 4 0 4 4 2 3
output
16.000000

Note

In the test example we can choose first 4 points to be the vertices of the quadrilateral. They form a square by side 4, so the area is $4 \cdot 4 = 16$.

C. Tourist Problem

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Iahub is a big fan of tourists. He wants to become a tourist himself, so he planned a trip. There are n destinations on a straight road that Iahub wants to visit. Iahub starts the excursion from kilometer 0. The n destinations are described by a non-negative integers sequence $a_1, a_2, ..., a_n$. The number a_k represents that the kth destination is at distance a_k kilometers from the starting point. No two destinations are located in the same place.

Iahub wants to visit each destination only once. Note that, crossing through a destination is not considered visiting, unless Iahub explicitly wants to visit it at that point. Also, after Iahub visits his last destination, he doesn't come back to kilometer 0, as he stops his trip at the last destination.

The distance between destination located at kilometer x and next destination, located at kilometer y, is $|x - y|$ kilometers. We call a "route" an order of visiting the destinations. Iahub can visit destinations in any order he wants, as long as he visits all n destinations and he doesn't visit a destination more than once.

Iahub starts writing out on a paper all possible routes and for each of them, he notes the total distance he would walk. He's interested in the average number of kilometers he would walk by choosing a route. As he got bored of writing out all the routes, he asks you to help him.

Input

The first line contains integer n ($2 \le n \le 10^5$). Next line contains n distinct integers $a_1, a_2, ..., a_n$ ($1 \le a_i \le 10^7$).

Output

Output two integers — the numerator and denominator of a fraction which is equal to the wanted average number. The fraction must be irreducible.

Sample test(s)

input
3 2 3 5
output
22 3

Note

Consider 6 possible routes:

- [2, 3, 5]: total distance traveled: |2 – 0| + |3 – 2| + |5 – 3| = 5;
- [2, 5, 3]: |2 – 0| + |5 – 2| + |3 – 5| = 7;
- [3, 2, 5]: |3 – 0| + |2 – 3| + |5 – 2| = 7;
- [3, 5, 2]: |3 – 0| + |5 – 3| + |2 – 5| = 8;
- [5, 2, 3]: |5 – 0| + |2 – 5| + |3 – 2| = 9;
- [5, 3, 2]: |5 – 0| + |3 – 5| + |2 – 3| = 8.

The average travel distance is $\frac{1}{6} \cdot (5 + 7 + 7 + 8 + 9 + 8) = \frac{44}{6} = \frac{22}{3}$.

D. Bubble Sort Graph

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Iahub recently has learned Bubble Sort, an algorithm that is used to sort a permutation with n elements $a_1, a_2, ..., a_n$ in ascending order. He is bored of this so simple algorithm, so he invents his own graph. The graph (let's call it G) initially has n vertices and 0 edges. During Bubble Sort execution, edges appear as described in the following algorithm (pseudocode).

```
procedure bubbleSortGraph()
    build a graph G with n vertices and 0 edges
    repeat
        swapped = false
        for i = 1 to n - 1 inclusive do:
            if a[i] > a[i + 1] then
                add an undirected edge in G between a[i] and a[i + 1]
                swap( a[i], a[i + 1] )
                swapped = true
            end if
        end for
    until not swapped
    /* repeat the algorithm as long as swapped value is true. */
end procedure
```

For a graph, an independent set is a set of vertices in a graph, no two of which are adjacent (so there are no edges between vertices of an independent set). A maximum independent set is an independent set which has maximum cardinality. Given the permutation, find the size of the maximum independent set of graph G, if we use such permutation as the premutation a in procedure bubbleSortGraph.

Input

The first line of the input contains an integer n $(2 \le n \le 10^5)$. The next line contains n distinct integers $a_1, a_2, ..., a_n$ $(1 \le a_i \le n)$.

Output

Output a single integer — the answer to the problem.

Sample test(s)

input
3 3 1 2
output
2

Note

Consider the first example. Bubble sort swaps elements 3 and 1. We add edge (1, 3). Permutation is now [1, 3, 2]. Then bubble sort swaps elements 3 and 2. We add edge (2, 3). Permutation is now sorted. We have a graph with 3 vertices and 2 edges (1, 3) and (2, 3). Its maximal independent set is [1, 2].

E. Iahub and Permutations

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Iahub is so happy about inventing bubble sort graphs that he's staying all day long at the office and writing permutations. Iahubina is angry that she is no more important for Iahub. When Iahub goes away, Iahubina comes to his office and sabotage his research work.

The girl finds an important permutation for the research. The permutation contains n distinct integers $a_1, a_2, ..., a_n$ $(1 \le a_i \le n)$. She replaces some of permutation elements with -1 value as a revenge.

When Iahub finds out his important permutation is broken, he tries to recover it. The only thing he remembers about the permutation is it didn't have any fixed point. A fixed point for a permutation is an element a_k which has value equal to k $(a_k = k)$. Your job is to proof to Iahub that trying to recover it is not a good idea. Output the number of permutations which could be originally Iahub's important permutation, modulo 1000000007 $(10^9 + 7)$.

Input

The first line contains integer n $(2 \le n \le 2000)$. On the second line, there are n integers, representing Iahub's important permutation after Iahubina replaces some values with -1.

It's guaranteed that there are no fixed points in the given permutation. Also, the given sequence contains at least two numbers -1 and each positive number occurs in the sequence at most once. It's guaranteed that there is at least one suitable permutation.

Output

Output a single integer, the number of ways Iahub could recover his permutation, modulo 1000000007 $(10^9 + 7)$.

Sample test(s)

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

Note

For the first test example there are two permutations with no fixed points are [2, 5, 4, 3, 1] and [5, 1, 4, 3, 2]. Any other permutation would have at least one fixed point.