Codeforces Round #174 (Div. 1)

A. Cows and Sequence

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Bessie and the cows are playing with sequences and need your help. They start with a sequence, initially containing just the number 0, and perform n operations. Each operation is one of the following:

1. Add the integer x_i to the first a_i elements of the sequence.
2. Append an integer k_i to the end of the sequence. (And hence the size of the sequence increases by 1)
3. Remove the last element of the sequence. So, the size of the sequence decreases by one. Note, that this operation can only be done if there are at least two elements in the sequence.

After each operation, the cows would like to know the average of all the numbers in the sequence. Help them!

Input

The first line contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the number of operations. The next n lines describe the operations. Each line will start with an integer t_i ($1 \le t_i \le 3$), denoting the type of the operation (see above). If $t_i = 1$, it will be followed by two integers a_i, x_i ($|x_i| \le 10^3$; $1 \le a_i$). If $t_i = 2$, it will be followed by a single integer k_i ($|k_i| \le 10^3$). If $t_i = 3$, it will not be followed by anything.

It is guaranteed that all operations are correct (don't touch nonexistent elements) and that there will always be at least one element in the sequence.

Output

Output n lines each containing the average of the numbers in the sequence after the corresponding operation.

The answer will be considered correct if its absolute or relative error doesn't exceed 10^{-6}.

Examples

input
5 2 1 3 2 3 2 1 3

output
0.500000 0.000000 1.500000 1.333333 1.500000

input
6 2 1 1 2 20 2 2 1 2 -3 3 3

output
0.500000 20.500000 14.333333 12.333333 17.500000 17.000000

Note

In the second sample, the sequence becomes $\{0\} \to \{0, 1\} \to \{20, 21\} \to \{20, 21, 2\} \to \{17, 18, 2\} \to \{17, 18\} \to \{17\}$.

B. Cow Program

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Farmer John has just given the cows a program to play with! The program contains two integer variables, x and y, and performs the following operations on a sequence $a_1, a_2, \ldots, a_n$ of positive integers:

1. Initially, $x = 1$ and $y = 0$. If, after any step, $x \leq 0$ or $x > n$, the program immediately terminates.
2. The program increases both x and y by a value equal to a_x simultaneously.
3. The program now increases y by a_x while decreasing x by a_x.
4. The program executes steps 2 and 3 (first step 2, then step 3) repeatedly until it terminates (it may never terminate). So, the sequence of executed steps may start with: step 2, step 3, step 2, step 3, step 2 and so on.

The cows are not very good at arithmetic though, and they want to see how the program works. Please help them!

You are given the sequence $a_2, a_3, \ldots, a_n$. Suppose for each i $(1 \leq i \leq n - 1)$ we run the program on the sequence $i, a_2, a_3, \ldots, a_n$. For each such run output the final value of y if the program terminates or -1 if it does not terminate.

Input

The first line contains a single integer, n $(2 \leq n \leq 2 \cdot 10^5)$. The next line contains $n - 1$ space separated integers, $a_2, a_3, \ldots, a_n$ $(1 \leq a_i \leq 10^9)$.

Output

Output $n - 1$ lines. On the i-th line, print the requested value when the program is run on the sequence $i, a_2, a_3, \ldots a_n$.

Please do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Examples

input
4 2 4 1

output
3 6 8

input
3 1 2

output
-1 -1

Note

In the first sample

1. For $i = 1$, x becomes $1 \rightarrow 2 \rightarrow 0$ and y becomes $1 + 2 = 3$.
2. For $i = 2$, x becomes $1 \rightarrow 3 \rightarrow -1$ and y becomes $2 + 4 = 6$.
3. For $i = 3$, x becomes $1 \rightarrow 4 \rightarrow 3 \rightarrow 7$ and y becomes $3 + 1 + 4 = 8$.

C. Coin Troubles

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

In the Isle of Guernsey there are n different types of coins. For each i ($1 \le i \le n$), coin of type i is worth a_i cents. It is possible that $a_i = a_j$ for some i and j ($i \ne j$).

Bessie has some set of these coins totaling t cents. She tells Jessie q pairs of integers. For each i ($1 \le i \le q$), the pair b_i, c_i tells Jessie that Bessie has a strictly greater number of coins of type b_i than coins of type c_i. It is known that all b_i are distinct and all c_i are distinct.

Help Jessie find the number of possible combinations of coins Bessie could have. Two combinations are considered different if there is some i ($1 \le i \le n$), such that the number of coins Bessie has of type i is different in the two combinations. Since the answer can be very large, output it modulo 1000000007 ($10^9 + 7$).

If there are no possible combinations of coins totaling t cents that satisfy Bessie's conditions, output 0.

Input

The first line contains three space-separated integers, n, q and t ($1 \le n \le 300$; $0 \le q \le n$; $1 \le t \le 10^5$). The second line contains n space separated integers, $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^5$). The next q lines each contain two distinct space-separated integers, b_i and c_i ($1 \le b_i, c_i \le n$; $b_i \ne c_i$).

It's guaranteed that all b_i are distinct and all c_i are distinct.

Output

A single integer, the number of valid coin combinations that Bessie could have, modulo 1000000007 ($10^9 + 7$).

Examples

input
4 2 17 3 1 2 5 4 2 3 4
output
3

input
3 2 6 3 1 1 1 2 2 3
output
0

input
3 2 10 1 2 3 1 2 2 1
output
0

Note

For the first sample, the following 3 combinations give a total of 17 cents and satisfy the given conditions:
$\{0 \textit{ of type } 1, 1 \textit{ of type } 2, 3 \textit{ of type } 3, 2 \textit{ of type } 4\}, \{0, 0, 6, 1\}, \{2, 0, 3, 1\}$.

No other combinations exist. Note that even though 4 occurs in both b_i and c_i, the problem conditions are still satisfied because all b_i are distinct and all c_i are distinct.

D. Cows and Cool Sequences

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Bessie and the cows have recently been playing with "cool" sequences and are trying to construct some. Unfortunately they are bad at arithmetic, so they need your help!

A pair (x, y) of positive integers is "cool" if x can be expressed as the sum of y consecutive integers (not necessarily positive). A sequence $(a_1, a_2, \ldots, a_n)$ is "cool" if the pairs $(a_1, a_2), (a_2, a_3), \ldots, (a_{n-1}, a_n)$ are all cool.

The cows have a sequence of n positive integers, $a_1, a_2, \ldots, a_n$. In one move, they may replace some a_i with any other positive integer (there are no other limits on the new value of a_i). Determine the smallest number of moves needed to make the resulting sequence cool.

Input

The first line contains a single integer, n ($2 \le n \le 5000$). The next line contains n space-separated integers, $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^{15}$).

Please do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Output

A single integer, the minimum number of a_i that must be changed to make the sequence cool.

Examples

input
3 6 4 1
output
0

input
4 20 6 3 4
output
2

Note

In the first sample, the sequence is already cool, so we don't need to change any elements. In the second sample, we can change a_2 to 5 and a_3 to 10 to make (20, 5, 10, 4) which is cool. This changes 2 elements.

E. Cow Tennis Tournament

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Farmer John is hosting a tennis tournament with his n cows. Each cow has a skill level s_i, and no two cows having the same skill level. Every cow plays every other cow exactly once in the tournament, and each cow beats every cow with skill level lower than its own.

However, Farmer John thinks the tournament will be demoralizing for the weakest cows who lose most or all of their matches, so he wants to flip some of the results. In particular, at k different instances, he will take two integers a_i, b_i $(a_i < b_i)$ and flip all the results between cows with skill level between a_i and b_i inclusive. That is, for any pair x, y $(x \neq y;\ s_x, s_y \in [a_i, b_i])$ he will change the result of the match on the final scoreboard (so if x won the match, the scoreboard will now display that y won the match, and vice versa). It is possible that Farmer John will change the result of a match multiple times. It is not guaranteed that a_i and b_i are equal to some cow's skill level.

Farmer John wants to determine how balanced he made the tournament results look. In particular, he wants to count the number of triples of cows (p, q, r) for which the final leaderboard shows that cow p beats cow q, cow q beats cow r, and cow r beats cow p. Help him determine this number.

Note that two triples are considered different if they do not contain the same set of cows (i.e. if there is a cow in one triple that is not in the other).

Input

On the first line are two space-separated integers, n and k $(3 \le n \le 10^5;\ 0 \le k \le 10^5)$. On the next line are n space-separated distinct integers, $s_1, s_2, \ldots, s_n$ $(1 \le s_i \le 10^9)$, denoting the skill levels of the cows. On the next k lines are two space separated integers, a_i and b_i $(1 \le a_i < b_i \le 10^9)$ representing the changes Farmer John made to the scoreboard in the order he makes it.

Output

A single integer, containing the number of triples of cows (p, q, r) for which the final leaderboard shows that cow p beats cow q, cow q beats cow r, and cow r beats cow p.

Please do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Examples

input
3 2 1 2 3 1 2 2 3

output
1

input
5 3 5 9 4 1 7 1 7 2 8 3 9

output
3

Note

In the first sample, cow 3 > cow 1, cow 3 > cow 2, and cow 2 > cow 1. However, the results between cows 1 and 2 and cows 2 and 3 are flipped, so now FJ's results show that cow 1 > cow 2, cow 2 > cow 3, and cow 3 > cow 1, so cows 1, 2, and 3 form a balanced triple.