Delivery Time (`delivery`)

Marco has decided that time has come to buy a new laptop. As in 2020 nobody is able to go to a physical store to make the purchase, the manufacturer's website is the only alternative. As everyone knows, though, when you buy something online you can never be certain about *when* it will be delivered: delivery times vary in rapid and unpredictable ways.
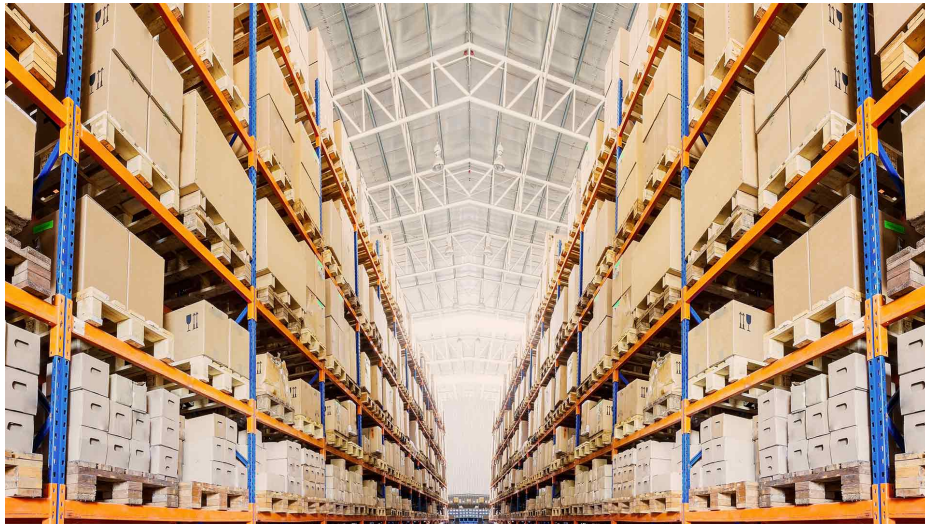

Figure 1: The warehouse containing lots of packages: unsurprisingly, deliveries from here take time.

After buying the laptop at day 0 with an expected delivery time of $T[0]$ days from that moment, Marco monitored the website for the next $N-1$ days (counted from 1 to $N-1$), writing down the delivery time $T[i]$ for laptops bought on that day (the i-th).

When looking at all this data, Marco thought that he could have (ab)used the free cancellation policy of the website to cancel an order and buy the laptop again, if the delivery time was more favorable on a certain day. If Marco could travel back in time and every morning cancel and reissue the purchase if the delivery time was more favorable (i.e., the delivery would happen at least one day earlier), how many times would have he reissued the purchase and when would the delivery occur (measured in days since the first purchase on day 0)?

> ☞ Among the attachments of this task you may find a template file `delivery.*` with a sample incomplete implementation.

Input

The first line contains the only integer N. The second line contains N integers T_i.

Output

You need to write a single line with two integers: respectively, the day of the delivery (measured since day 0) and how many re-orders Marco would have done (excluding the initial order).

Constraints

- $1 \le N \le 10\,000$.
- $1 \le T_i \le 100\,000$ for each $i = 0 \dots N - 1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points) Examples.

– **Subtask 2** (10 points) $N = 2$.

– **Subtask 3** (40 points) $N \le 100$.

– **Subtask 4** (50 points) No additional limitations.

Examples

input	output
2 7 4	5 1
5 20 19 17 15 25	18 2

Explanation

In the **first sample case** Marco buys the laptop and expects the delivery in 7 days (as $T[0] = 7$). The following day he cancels his order and reissues it to receive the laptop 4 days after (or equivalently, 5 days since the initial purchase). Overall, he cancelled and reissued the order only once.

In the **second sample case** Marco buys the laptop and expects the delivery in 20 days (as $T[0] = 20$). Two days after, he cancels his order and reissues it to receive the laptop 17 days after that moment (or equivalently, 19 days since the initial purchase). The next day, he cancels again his order and reissues it to receive the laptop 15 days after that moment (or equivalently, 18 days since the initial purchase). Overall, he cancelled and reissued the order twice.

The Enigma of the Dungeon Cave (dungeon)

Everybody knows that Giorgio is a great fan of pen-and-paper role-playing games, and is in fact the game master for a heroic decade-long campaign. For the next game session, he has to prepare an enigma protecting the entrance of the final cave of his dungeon.
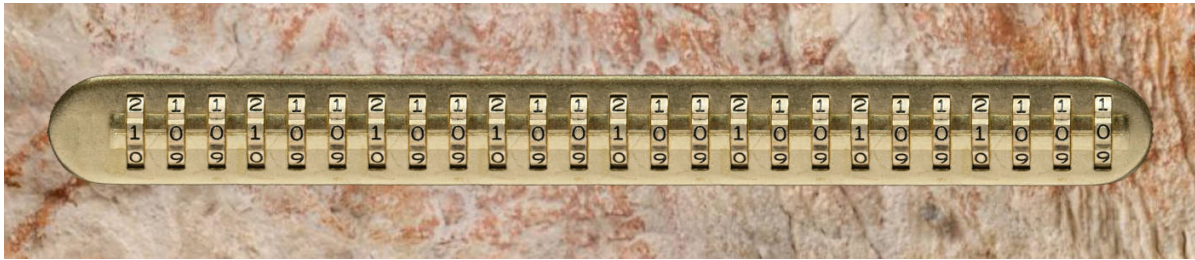

Figure 1: What could be the correct numeric code?

In particular, the entrance will be locked by a numeric code of N digits, with a rhymed poetry hinting at the fact that the sum of its digits is equal to the product of them. However, Giorgio is not sure if N is the right length for the numeric code, in order to give exactly the right amount of challenge to his players. Help him compute how many N-digit codes exist with the same sum and product of their digits!

> ☞ Among the attachments of this task you may find a template file `dungeon.*` with a sample incomplete implementation.

Input

The first and only line contains the only integer N.

Output

You need to write a single line with an integer: the number of N-digit codes with the same sum and product of their digits, **modulo** $10^9 + 7$.

> ✎ The *modulo* operation ($a \bmod m$) can be written in C/C++/Python as (`a % m`) and in Pascal as (`a mod m`). To avoid the *integer overflow* error, remember to reduce all partial results through the modulus, and not just the final result!
>
> *Notice that if $x < 10^9 + 7$, then $2x$ fits into a* C/C++ `int` *and* Pascal `longint`.

Constraints

- $1 \leq N \leq 100\,000$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points)   Examples.

– **Subtask 2** (10 points) $N \leq 6$.

– **Subtask 3** (30 points) $N \leq 1000$.

– **Subtask 4** (60 points)   No additional limitations.

Examples

input	output
1	10
2	2

Explanation

In the **first sample case**, there are 10 codes: every 1-digit integer is equal to both the sum and product of its only digit.

In the **second sample case**, there are only two codes: numbers 00 and 22.

Tax Fraud Detection (`fraud`)

Edoardo has been hired by the Inland Revenue Agency with the task of detecting fraudulent tax forms. After months of thorough analysis, he has finally found a measure of *fraudulence*!

Figure 1: Fraudulent tax forms.

You are given a tax form, represented as an array of N elements V_i for $i = 0 \ldots N-1$. The *fraudulence* of the sub-array $[V_i, \ldots V_j]$ (for $0 \le i \le j \le N-1$) is defined as the product of the frequency of the rarest element (the smallest number of times an element appears in the sub-array) with the frequency of the most common element (the highest number of times an element appears in the sub-array). For example,

$$[1, 3, 2, 3, 1, 2, 3]$$

has fraudulence $6 = 2 \times 3$ since the rarest elements (1 and 2) appear twice, while the most common element (3) appears 3 times. Compute the maximum fraudulence for a sub-array of the given array!

> ☞ Among the attachments of this task you may find a template file `fraud.*` with a sample incomplete implementation.

Input

The first line contains the only integer N. The second line contains N integers V_i.

Output

You need to write a single line with an integer: the maximum fraudulence for a sub-array of the given array.

Constraints

- $1 \le N \le 100\,000$.
- $1 \le V_i \le 100\,000$ for each $i = 0 \ldots N-1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points)　　　　Examples.

– **Subtask 2** (15 points)　　　　$N \leq 100$.

– **Subtask 3** (20 points)　　　　$N \leq 1000$, $V_i \leq 200$ for each $i = 0 \ldots N - 1$.

– **Subtask 4** (30 points)　　　　$N \leq 2000$.

– **Subtask 5** (35 points)　　　　No additional limitations.

Examples

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

Explanation

In the **first sample case**, you may choose either $[1, 2, 1]$ or $[1, 2, 1, 3, 2]$ or $[2, 1, 3, 2]$: all of them have the lowest frequency equal to one (attained by numbers 2, 3, 1 and 3 respectively), and the highest frequency equal to two (attained by 1, 1 and 2, 2 respectively). Altogether, the fraudulence of any of those sub-arrays is $1 \times 2 = 2$.

In the **second sample case**, the highest fraudulence is attained by sub-array $[4, 2, 2, 4]$ which has both lowest and highest frequencies equal to 2.

Electrical Power Line (`powerline`)

Luca needs a lot of electricity to run its homelab, composed of lots of very powerful servers. For this reason, some years ago he built a very complex *overhead power line* composed of N towers. The N towers are connected in a line and the i-th tower is H_i meters high.

Figure 1: Some of the towers in Luca's power line.

In order to increase their efficiency, a new law mandates that every tower in a power line must be at least as high as the following one. Luca needs to change the height of some towers, in order to meet the new requirements. Every day, starting from the first tower, Luca will increase the height of every tower to the same height of the next one. Of course, if one tower is already taller than the following one it will not be modified. How many days does Luca need to finish the job?

> ☞ Among the attachments of this task you may find a template file `powerline.*` with a sample incomplete implementation.

Input

The first line contains the only integer N, the number of towers in the power line. The second line contains N integers H_i, the height of the i-th tower, in the same order as they are connected.

Output

You need to write a single line with an integer: how many days Luca needs to fix the power line.

Constraints

- $1 \leq N \leq 100\,000$.
- $1 \leq H_i \leq 10^9$ for each $i = 0 \ldots N-1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points)   Examples.

– **Subtask 2** (10 points) $H_i \leq H_{i+1}$ for each $i = 0 \ldots N-2$.

– **Subtask 3** (30 points) $N \leq 5\,000$.

– **Subtask 4** (60 points)   No additional limitations.

Examples

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

Explanation

In the **first sample case**, Luca needs two days to fix the powerline. After the first day, the towers' heights are 4, 2, 3, 3 and 1. After the second day, the towers' heights are 4, 3, 3, 3 and 1 and Luca has finished its job.

In the **second sample case**, the power line is already compliant with the new law, so Luca doesn't need to do anything.

Pordenone Hill Sign (`stringstreak`)

Edoardo is trying to build a "customized" version of the Hollywood sign: same massive size but different text! The sign is going to be used on the hill of his small town in Italy, Pordenone.

He managed to buy a sign S, but he would like to modify its letters in an optimal way so that the sign ends up having a substring repeating a same letter which is as long as possible.

Figure 1: The famous "Hollywood" sign.

In order to change such a large wooden sign, Edoardo is asking for his *Falegname Di Fiducia*'s help. The rate charged by the *FDF* is quite peculiar: he will charge 2^{j-i+1} euro to change the letters of each contiguous substring $S[i \dots j]$ in the sign. For example, if the sign was `aaxyaa` and Edoardo wanted to change the substring $S[3 \dots 4]$ from `xy` to `aa`, he would have to pay $2^2 = 4$ euro.

Asking the *FDF* to change one character at a time is not allowed: Edoardo must choose which characters he wants to change, and then the *FDF* will charge him according to the contiguous substrings selected.

After spending his money to buy the sign, Edoardo is left with B euro in his budget. Help him find an optimal way to change the sign so that the budget is not exceeded and **the length of the longest substring** which can be formed by the same character repeated is maximal.

> ☞ Among the attachments of this task you may find a template file `stringstreak.*` with a sample incomplete implementation.

Input

The first line contains a string S, the sign. The second line contains a number B, the budget.

Output

You need to write a single line with an integer: the unique integer that solves this task.

Constraints

- $1 \leq |S| \leq 100\,000$, where $|S|$ is the length of the string.
- $1 \leq B \leq 10^9$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points)   Examples.

– **Subtask 2** (30 points) $|S| \leq 100$.

– **Subtask 3** (70 points)   No additional limitations.

Examples

input	output
xabaabxab 10	9

Explanation

In the **first sample case** we can modify the letters in positions 1, 3, 6, 7 and 9 to get nine `a` letters in a row, with a cost of 10 euro.

Subset Fight (`subsetfight`)

Marco is playing a game using a deck of cards. The deck consists of K different types of cards. For each type $i = 1 \ldots K$ there are V_i cards of that type, which are distinguishable by their seed, but they all share the same *value i*. Notice the total number of cards in the deck is $N = V_1 + V_2 + \cdots + V_K$. For example, let's say that $K = 3$ and $V = [1, 2, 3]$. This means that the values of the cards in the deck are as follows:

$$1, 2, 2, 3, 3, 3$$

Marco must choose anyone of the possible 2^N subsets of cards, including the empty subset with zero cards. If the sum of the values in the subset is a multiple of K, Marco wins! In the example above, there are 24 different ways of choosing a winning subset of cards.

Figure 1: A possible subset of cards chosen by Marco.

Since the game is quite easy, Marco is now wondering: **how many ways** are there to win the game?

> ☞ Among the attachments of this task you may find a template file `subsetfight.*` with a sample incomplete implementation.

Input

The first line contains the only integer K. The second line contains K integers V_i.

Output

You need to write a single line with an integer: the number of different winning subsets of cards **modulo** $10^9 + 7$ (cards with the same value but different seed are considered different).

> ✎ The *modulo* operation $(a \bmod m)$ can be written in C/C++/Python as `(a % m)` and in Pascal as `(a mod m)`. To avoid the *integer overflow* error, remember to reduce all partial results through the modulus, and not just the final result!
> *Notice that if $x < 10^9 + 7$, then $2x$ fits into a C/C++ `int` and Pascal `longint`.*

Constraints

- $1 \le K \le 100$.
- $1 \le V_i \le 200\,000$ for each $i = 1 \ldots K$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points)   Examples.

- **Subtask 2** (20 points)   The total number of cards in the deck is at most 20.

- **Subtask 3** (40 points)   The total number of cards in the deck is at most $100\,000$.

- **Subtask 4** (40 points)   No additional limitations.

Examples

input	output
3 1 1 1	4
5 69 420 1017 128 953	985611225

Explanation

In the **first sample case** the correct subsets are:

- The empty set $\{\}$,

- The two cards of values $\{1, 2\}$,

- The only card of value $\{3\}$,

- The three cards of values $\{1, 2, 3\}$.

Swimming Pool (`swimmingpool`)

To keep in shape after the first lockdown, Luca decided to sign up at a local swimming pool. Being a novice, he opted for a course where a professional trainer supervises a number of athletes to give advice while they do their lengths.
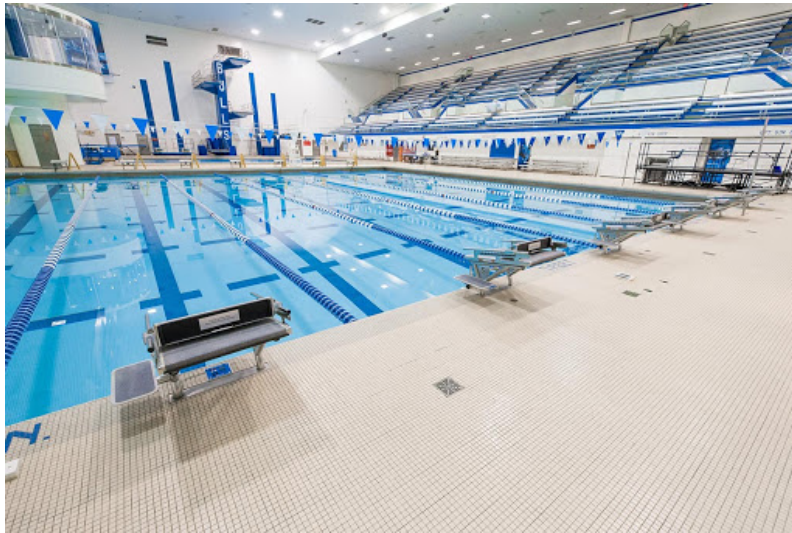

Figure 1: Starting blocks at a swimming pool.

Luca, however, couldn't help to notice that the position of the trainer is suboptimal and she has to continuously shout to be heard by everybody, even before the first dive.

The N athletes are in fact starting at different positions P_i (from 0 to $N-1$), seen from left to right and measured in centimeters from the left wall of the pool. The trainer **always** chooses to stay exactly behind one of them, i.e. she selects a position P_t that coincides with that of one of the athletes. In this way, she is exactly 0 centimeters away from one athlete, but potentially very distant from some of them.

We call D the distance to the furthest athlete: assuming that the trainer chooses wisely the position in which to stay in order to minimize the distance and save her voice, what is the best value of D she can achieve?

> ☞ Among the attachments of this task you may find a template file `swimmingpool.*` with a sample incomplete implementation.

Input

The first line contains the only integer N. The second line contains N integers P_i.

Output

You need to write a single line with an integer D: the minimum distance to the furthest athlete the trainer can achieve staying in the best position.

Constraints

- $2 \leq N \leq 10\,000$.

- $0 \leq P_i \leq 1\,000\,000$ for each $i = 0 \ldots N - 1$.

- $P_i < P_{i+1}$ for each $i = 0 \ldots N - 2$: positions are given in order and each athlete occupies a different position.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points)   Examples.

– **Subtask 2** (20 points) $N \leq 3$.

– **Subtask 3** (30 points) $N \leq 500$.

– **Subtask 4** (50 points)   No additional limitations.

Examples

input	output
4 10 12 20 60	40
7 100 200 300 400 500 600 700	300

Explanation

In the **first sample case**, the best position for the trainer is at $P_2 = 20$: in that case, the furthest athlete is at $P_3 = 60$.

In the **second sample case**, the best position for the trainer is at $P_3 = 400$: in that case, she has two atheltes at distance 300.

Christmas Tree (`xmastree`)

Each year the Christmas decorations are prepared earlier and earlier. This year is no exception, and William wants to be prepared! William is planning to put a really big Christmas tree in his garden, bigger and brighter than everyone else!

Figure 1: A Christmas tree for sure nicer that William's one.

The tree is built starting from a big pine tree, where many lights are installed on the branches. Starting from the root one or more cables are installed, and when there's a branch a light is placed there and one or more new cables come out of it. When a leaf is reached a new light is placed together with a switch. Pressing the switch will toggle all the lights in the path from that leaf to the root (turning on the switched off lights and vice-versa).

After all this work William turns on all the N lights but realized that his tree is pretty boring. To avoid re-routing all the lights he decides to assign a niceness level A_i to all the N lights (A_i might be negative as well). If a light is turned off its niceness level goes to zero. Now he's questioning what is the maximum level of niceness achievable acting only on the switches.

> ☞ Among the attachments of this task you may find a template file `xmastree.*` with a sample incomplete implementation.

Input

The first line contains the only integer N. The following $N-1$ lines contain two integers a and b, meaning that the light a is connected with the light b. The next line contains N integers, the values of A_i.

Output

You need to write a single line with an integer: the maximum possible sum of the A_i of the turned on lights.

Constraints

- $2 \leq N \leq 200\,000$.
- $-10^9 \leq A_i \leq 10^9$ for each $i = 0 \ldots N - 1$.
- The root light has index 0.
- There cannot be a switch on the root.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points) Examples.

– **Subtask 2** (10 points) There is only one switch and $N < 1000$.

– **Subtask 3** (20 points) There are exactly $N - 1$ switches.

– **Subtask 4** (15 points) $N \leq 20$.

– **Subtask 5** (20 points) $N \leq 1000$.

– **Subtask 6** (35 points) No additional limitations.

Examples

input	output
4 0 3 0 2 1 3 -10 8 3 5	13
5 0 1 1 3 1 4 0 2 1 2 4 -4 -5	7

Explanation

In the **first sample case** by pressing the switch on the light 2 the root and the light 2 get tuned off, keeping a total niceness of $5 + 8 = 13$.

In the **second sample case** you can proceed as follows:

- Switch light 3 (that turns off also lights 0 and 1)

- Switch light 4 (that turns on again lights 0 and 1)

In the end the lights 0, 1 and 2 are powered, with a total niceness of $1 + 2 + 4 = 7$.

Xorstanta (`xorstanta`)

Giorgio is a big fan of board games and he has asked William to play with him. This time they are going to play a cooperative game, which means they need to play together in order to maximize their score. They have a lot of free time, so they are going to play T matches.

Figure 1: The numbers used in the game.

The rules of the game are very simple. First of all, in the i-th match they choose a number N_i. Then, for each number from 1 to N_i they need to decide whether Giorgio or William keeps it. Let's call A the set of numbers given to Giorgio and B the set of numbers given to William. Let X_A be the *bitwise xor* of all the numbers in A and X_B the *bitwise xor* of all the numbers in B. The score of the match is given by the sum of X_A and X_B. What is the maximum score they can achieve in each match?

☞ Among the attachments of this task you may find a template file `xorstanta.*` with a sample incomplete implementation.

Input

The first line contains the only integer T, the number of matches. Each one of the next T lines contains one integer N_i, the number chosen in the i-th match.

Output

For each match, you need to write a single line with an integer: the maximum score that can be achieved.

Constraints

- $1 \leq T \leq 100\,000$.
- $1 \leq N_i \leq 10^9$ for each $i = 0 \ldots T - 1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points)   Examples.

– **Subtask 2** (10 points) $T = 1$ and $N_0 \leq 20$.

– **Subtask 3** (20 points) $T = 1$ and $N_0 \leq 100\,000$.

– **Subtask 4** (70 points)   No additional limitations.

Examples

input	output
2 6 15	7 30

Explanation

In the **first sample case** there are two matches.

In the first match, $N = 6$ and the best score is 7, which can be achieved by giving all the numbers from one to six to Giorgio. In this case $A = \{1, 2, 3, 4, 5, 6\}$, B is empty, $X_A = 7$ and $X_B = 0$.

In the second match, $N = 15$ and the best score is 30, which can be achieved by giving the numbers 1, 5 and 11 to Giorgio and the rest to William. In this case $A = \{1, 5, 11\}$, $B = \{2, 3, 4, 6, 7, 8, 9, 10, 12, 13, 14, 15\}$, $X_A = 15$ and $X_B = 15$.