

# A. Max Plus Size

Input file: standard input  
Output file: standard output  
Time limit: 1 second  
Memory limit: 256 megabytes

[EnV - Dynasty](#)

You are given an array  $a_1, a_2, \dots, a_n$  of positive integers.

You can color some elements of the array red, but there cannot be two adjacent red elements (i.e., for  $1 \leq i \leq n - 1$ , at least one of  $a_i$  and  $a_{i+1}$  must not be red).

Your score is the maximum value of a red element plus the number of red elements. Find the maximum score you can get.

## Input

Each test contains multiple test cases. The first line contains the number of test cases  $t$  ( $1 \leq t \leq 500$ ). The description of the test cases follows.

The first line of each test case contains a single integer  $n$  ( $1 \leq n \leq 100$ ) — the length of the array.

The second line of each test case contains  $n$  integers  $a_1, a_2, \dots, a_n$  ( $1 \leq a_i \leq 1000$ ) — the given array.

## Output

For each test case, output a single integer: the maximum possible score you can get after coloring some elements red according to the statement.

Standard Input	Standard Output
4	7
3	6
5 4 5	10
3	97
4 5 4	
10	
3 3 3 3 4 1 2 3 4 5	
9	
17 89 92 42 29 92 14 70 45	

## Note

In the first test case, you can color the array as follows:  $[5, 4, 5]$ . Your score is  $\max([5, 5]) + \text{size}([5, 5]) = 5 + 2 = 7$ . This is the maximum score you can get.

In the second test case, you can color the array as follows:  $[4, 5, 4]$ . Your score is  $\max([4, 4]) + \text{size}([4, 4]) = 4 + 2 = 6$ . This is the maximum score you can get.

In the third test case, you can color the array as follows:  $[3, 3, 3, 3, 4, 1, 2, 3, 4, 5]$ . Your score is  $\max([3, 3, 4, 3, 5]) + \text{size}([3, 3, 4, 3, 5]) = 5 + 5 = 10$ . This is the maximum score you can get.

## B. All Pairs Segments

Input file: standard input  
Output file: standard output  
Time limit: 1.5 seconds  
Memory limit: 256 megabytes

[Shirobon - FOX](#)

You are given  $n$  points on the  $x$  axis, at increasing positive integer coordinates  $x_1 < x_2 < \dots < x_n$ .

For each pair  $(i, j)$  with  $1 \leq i < j \leq n$ , you draw the segment  $[x_i, x_j]$ . The segments are closed, i.e., a segment  $[a, b]$  contains the points  $a, a + 1, \dots, b$ .

You are given  $q$  queries. In the  $i$ -th query, you are given a positive integer  $k_i$ , and you have to determine how many points with integer coordinates are contained in exactly  $k_i$  segments.

### Input

Each test contains multiple test cases. The first line contains the number of test cases  $t$  ( $1 \leq t \leq 10^4$ ). The description of the test cases follows.

The first line of each test case contains two integers  $n, q$  ( $2 \leq n \leq 10^5, 1 \leq q \leq 10^5$ ) — the number of points and the number of queries.

The second line of each test case contains  $n$  integers  $x_1, x_2, \dots, x_n$  ( $1 \leq x_1 < x_2 < \dots < x_n \leq 10^9$ ) — the coordinates of the  $n$  points.

The third line of each test case contains  $q$  integers  $k_1, k_2, \dots, k_q$  ( $1 \leq k_i \leq 10^{18}$ ) — the parameters of the  $q$  queries.

It is guaranteed that the sum of  $n$  over all test cases does not exceed  $10^5$ , and the sum of  $q$  over all test cases does not exceed  $10^5$ .

### Output

For each test case, output a single line with  $q$  integers: the  $i$ -th integer is the answer to the  $i$ -th query.

Standard Input	Standard Output
3 2 2 101 200 2 1 6 15 1 2 3 5 6 7 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 5 8 254618033 265675151 461318786 557391198 848083778 6 9 15 10 6 9 4 4294967300	0 100 0 0 0 0 2 0 0 0 3 0 2 0 0 0 0 291716045 0 0 0 291716045 0 301749698 0

### Note

In the first example, you only draw the segment  $[101, 200]$ . No point is contained in exactly 2 segments, and the 100 points  $101, 102, \dots, 200$  are contained in exactly 1 segment.

In the second example, you draw 15 segments:

$[1, 2], [1, 3], [1, 5], [1, 6], [1, 7], [2, 3], [2, 5], [2, 6], [2, 7], [3, 5], [3, 6], [3, 7], [5, 6], [5, 7], [6, 7]$ . Points 1, 7 are contained in exactly 5 segments; points 2, 4, 6 are contained in exactly 9 segments; points 3, 5 are contained in exactly 11 segments.

# C. Cards Partition

Input file: standard input  
Output file: standard output  
Time limit: 2 seconds  
Memory limit: 256 megabytes

[DJ Genki vs Gram - Einherjar Joker](#)

You have some cards. An integer between 1 and  $n$  is written on each card: specifically, for each  $i$  from 1 to  $n$ , you have  $a_i$  cards which have the number  $i$  written on them.

There is also a shop which contains unlimited cards of each type. You have  $k$  coins, so you can buy **at most**  $k$  new cards in total, and the cards you buy can contain any integer **between 1 and  $n$** , inclusive.

After buying the new cards, you must partition **all** your cards into decks, according to the following rules:

- all the decks must have the same size;
- there are no pairs of cards with the same value in the same deck.

Find the maximum possible size of a deck after buying cards and partitioning them optimally.

## Input

Each test contains multiple test cases. The first line contains the number of test cases  $t$  ( $1 \leq t \leq 10^4$ ). The description of the test cases follows.

The first line of each test case contains two integers  $n, k$  ( $1 \leq n \leq 2 \cdot 10^5, 0 \leq k \leq 10^{16}$ ) — the number of distinct types of cards and the number of coins.

The second line of each test case contains  $n$  integers  $a_1, a_2, \dots, a_n$  ( $0 \leq a_i \leq 10^{10}, \sum a_i \geq 1$ ) — the number of cards of type  $i$  you have at the beginning, for each  $1 \leq i \leq n$ .

It is guaranteed that the sum of  $n$  over all test cases does not exceed  $2 \cdot 10^5$ .

## Output

For each test case, output a single integer: the maximum possible size of a deck if you operate optimally.

Standard Input	Standard Output
9	2
3 1	3
3 2 2	1
5 4	7
2 6 1 2 4	2
2 100	2
1410065408 10000000000	1
10 8	1
7 4 6 6 9 3 10 2 8 7	2
2 12	
2 2	
2 70	
0 1	
1 0	

1	
3 0	
2 1 2	
3 1	
0 3 3	

## Note

In the first test case, you can buy one card with the number 1, and your cards become [1, 1, 1, 1, 2, 2, 3, 3]. You can partition them into the decks [1, 2], [1, 2], [1, 3], [1, 3]: they all have size 2, and they all contain distinct values. You can show that you cannot get a partition with decks of size greater than 2, so the answer is 2.

In the second test case, you can buy two cards with the number 1 and one card with the number 3, and your cards become [1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 4, 4, 5, 5, 5, 5], which can be partitioned into [1, 2, 3], [1, 2, 4], [1, 2, 5], [1, 2, 5], [2, 3, 5], [2, 4, 5]. You can show that you cannot get a partition with decks of size greater than 3, so the answer is 3.

## D. Speedbreaker

Input file: standard input  
Output file: standard output  
Time limit: 2 seconds  
Memory limit: 256 megabytes

[Dijaner - Speedbreaker](#)

There are  $n$  cities in a row, numbered  $1, 2, \dots, n$  left to right.

- At time 1, you conquer exactly one city, called the *starting city*.
- At time  $2, 3, \dots, n$ , you can choose a city adjacent to the ones conquered so far and conquer it.

You win if, for each  $i$ , you conquer city  $i$  at a time no later than  $a_i$ . A winning strategy may or may not exist, also depending on the starting city. How many starting cities allow you to win?

### Input

Each test contains multiple test cases. The first line contains the number of test cases  $t$  ( $1 \leq t \leq 10^4$ ). The description of the test cases follows.

The first line of each test case contains a single integer  $n$  ( $1 \leq n \leq 2 \cdot 10^5$ ) — the number of cities.

The second line of each test case contains  $n$  integers  $a_1, a_2, \dots, a_n$  ( $1 \leq a_i \leq n$ ) — the deadlines for conquering the cities.

It is guaranteed that the sum of  $n$  over all test cases does not exceed  $2 \cdot 10^5$ .

### Output

For each test case, output a single integer: the number of starting cities that allow you to win.

Standard Input	Standard Output
3 6 6 3 3 3 5 5 6 5 6 4 1 4 5 9 8 6 4 2 1 3 5 7 9	3 0 1

### Note

In the first test case, cities 2, 3, and 4 are good starting cities.

In the second test case, there are no good starting cities.

In the third test case, the only good starting city is city 5.

## E. Tree Pruning

Input file: standard input  
Output file: standard output  
Time limit: 3 seconds  
Memory limit: 256 megabytes

[t+pazolite, ginkiha, Hommarju - Paved Garden](#)

You are given a tree with  $n$  nodes, rooted at node 1. In this problem, a leaf is a non-root node with degree 1.

In one operation, you can remove a leaf and the edge adjacent to it (possibly, new leaves appear). What is the minimum number of operations that you have to perform to get a tree, also rooted at node 1, where all the leaves are at the same distance from the root?

### Input

Each test contains multiple test cases. The first line contains the number of test cases  $t$  ( $1 \leq t \leq 10^4$ ). The description of the test cases follows.

The first line of each test case contains a single integer  $n$  ( $3 \leq n \leq 5 \cdot 10^5$ ) — the number of nodes.

Each of the next  $n - 1$  lines contains two integers  $u, v$  ( $1 \leq u, v \leq n, u \neq v$ ), describing an edge that connects  $u$  and  $v$ . It is guaranteed that the given edges form a tree.

It is guaranteed that the sum of  $n$  over all test cases does not exceed  $5 \cdot 10^5$ .

### Output

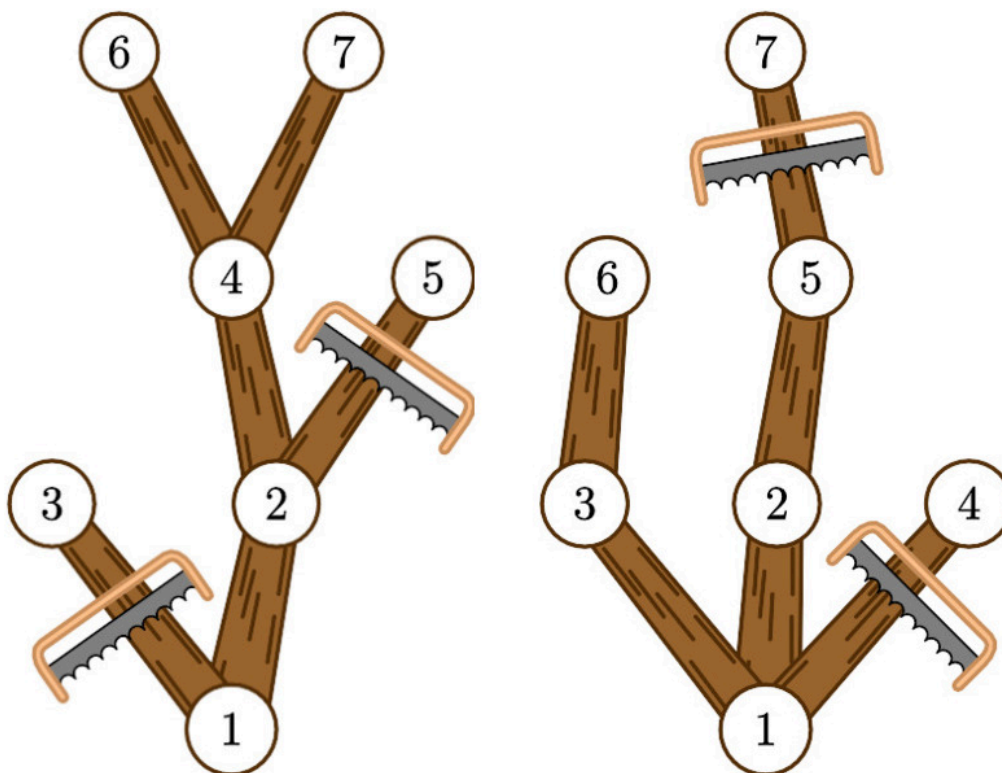
For each test case, output a single integer: the minimum number of operations needed to achieve your goal.

Standard Input	Standard Output
3	2
7	2
1 2	5
1 3	
2 4	
2 5	
4 6	
4 7	
7	
1 2	
1 3	
1 4	
2 5	
3 6	
5 7	
15	
12 9	
1 6	
6 14	
9 11	

8 7	
3 5	
13 5	
6 10	
13 15	
13 6	
14 12	
7 2	
8 1	
1 4	

### Note

In the first two examples, the tree is as follows:



In the first example, by removing edges  $(1, 3)$  and  $(2, 5)$ , the resulting tree has all leaves (nodes 6 and 7) at the same distance from the root (node 1), which is 3. The answer is 2, as it is the minimum number of edges that need to be removed to achieve the goal.

In the second example, removing edges  $(1, 4)$  and  $(5, 7)$  results in a tree where all leaves (nodes 4 and 5) are at the same distance from the root (node 1), which is 2.



## F. Max Plus Min Plus Size

Input file: standard input  
Output file: standard output  
Time limit: 2 seconds  
Memory limit: 256 megabytes

[EnV - The Dusty Dragon Tavern](#)

You are given an array  $a_1, a_2, \dots, a_n$  of positive integers.

You can color some elements of the array red, but there cannot be two adjacent red elements (i.e., for  $1 \leq i \leq n - 1$ , at least one of  $a_i$  and  $a_{i+1}$  must not be red).

Your score is the maximum value of a red element, plus the minimum value of a red element, plus the number of red elements. Find the maximum score you can get.

### Input

Each test contains multiple test cases. The first line contains the number of test cases  $t$  ( $1 \leq t \leq 10^4$ ). The description of the test cases follows.

The first line of each test case contains a single integer  $n$  ( $1 \leq n \leq 2 \cdot 10^5$ ) — the length of the array.

The second line of each test case contains  $n$  integers  $a_1, a_2, \dots, a_n$  ( $1 \leq a_i \leq 10^9$ ) — the given array.

It is guaranteed that the sum of  $n$  over all test cases does not exceed  $2 \cdot 10^5$ .

### Output

For each test case, output a single integer: the maximum possible score you can get after coloring some elements red according to the statement.

Standard Input	Standard Output
4	12
3	11
5 4 5	12
3	186
4 5 4	
10	
3 3 3 3 4 1 2 3 5 4	
10	
17 89 92 42 29 41 92 14 70 45	

### Note

In the first test case, you can color the array as follows:  $[5, 4, 5]$ . Your score is  $\max([5, 5]) + \min([5, 5]) + \text{size}([5, 5]) = 5 + 5 + 2 = 12$ . This is the maximum score you can get.

In the second test case, you can color the array as follows:  $[4, 5, 4]$ . Your score is  $\max([5]) + \min([5]) + \text{size}([5]) = 5 + 5 + 1 = 11$ . This is the maximum score you can get.

In the third test case, you can color the array as follows:  $[3, 3, 3, 3, 4, 1, 2, 3, 5, 4]$ . Your score is  $\max([3, 3, 4, 3, 4]) + \min([3, 3, 4, 3, 4]) + \text{size}([3, 3, 4, 3, 4]) = 4 + 3 + 5 = 12$ . This is the maximum

score you can get.