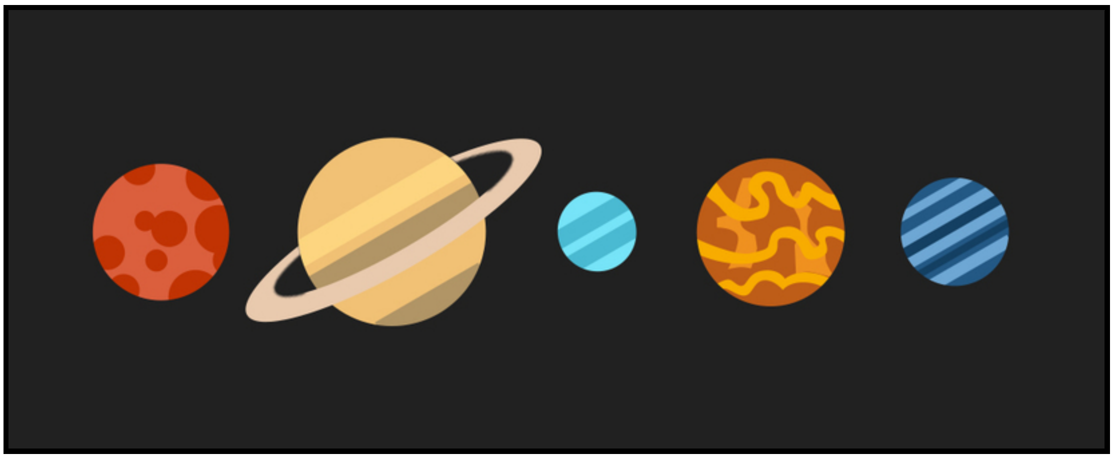
Sorting the Planets



As part of his cosmic maintenance, Kevin is sorting the planets in the solar system.

The planets are arranged in a row, and no two planets have the same size. Kevin has an unlimited number of sorting machines. A machine is assigned to several consecutive planets; each planet is assigned to exactly one machine, and each machine is assigned to at least one planet. When the machines are activated, they sort the planets they are assigned to.

For example, suppose the planets have sizes $2, 3, 1, 6, 5$, in that order. Kevin can assign the first three planets to one machine, and the last two planets to another. When he activates the machines, the first three planets are sorted to $1, 2, 3$, and the last two planets are sorted to $5, 6$. The order of the planets is now $1, 2, 3, 5, 6$, which is sorted.

Of course, it is possible for Kevin to use only a single machine to sort the planets. But as Kevin loves distributed algorithms, he wants to use as many machines as he can. For a list of planets s, let $f(s)$ be the maximum number of machines Kevin can use. For example, $f([2, 3, 1, 6, 5]) = 2$: as shown above, 2 machines can sort it, yet it can be proven that 3 machines cannot.

Kevin is not only sorting the planets in our universe, but he is simultaneously sorting the planets in *all* the universes. Given a list of planets, a **universe** s is defined as a sublist of *consecutive* planets of this list. To properly allocate resources, Kevin then computes the sum of $f(s)$ over all possible universes s. Your task is to follow his example and compute this sum as well.

Input Format

The first line of input contains a single integer t, denoting the number of test cases.

The first line of each test case contains an integer n, the number of planets. The second line contains n distinct space-separated integers $a_1, a_2, \dots, a_n$, representing the sizes of the planets as arranged in the row.

Output Format

For each test case, output one line containing a single integer denoting the answer for that test case.

Constraints and Subtasks

For all subtasks
$1 \leq t \leq 5$
$1 \leq a_i \leq 10^6$
$1 \leq n \leq 2 \cdot 10^5$
The a_is are distinct.

Subtask	Points	Constraints
1	**10**	$1 \leq n \leq 12$
2	**19**	$1 \leq n \leq 100$
3	**11**	$1 \leq n \leq 500$
4	**11**	$1 \leq n \leq 5000$
5	**16**	$\lvert a_i - a_{i-1} \rvert < 3$
6	**33**	No additional constraints.

Sample I/O

Input	Output

`2`
`3`
`1 3 2`
`4`
`1 2 5 3`
	`8`
	`17`

Explanation

For the first test case, there are six possible universes. We compute the maximum number of machines needed for each universe:

- $f([1]) = 1$. Similarly, $f([3]) = 1$ and $f([2]) = 1$.
- $f([3, 2]) = 1$, as both need to be assigned to the same machine.
- $f([1, 3, 2]) = 2$: we can assign the first planet to one machine, and the other two planets to a different machine.
- $f([1, 3]) = 2$, as the two planets can be assigned to different machines.

Thus, the answer for this case is 8.

The second test case has ten possible universes. One of them is $[1, 2, 5, 3]$, which can be sorted with 3 machines. Another is $[2, 5, 3]$, which can be sorted with 2 machines. The other eight universes can be analyzed similarly, for a total of 17.