E. Special Elements

time limit per test: 1 second
memory limit per test: 64 megabytes

Pay attention to the non-standard memory limit in this problem.

In order to cut off efficient solutions from inefficient ones in this problem, the time limit is rather strict. Prefer to use compiled statically typed languages (e.g. C++). If you use Python, then submit solutions on PyPy. Try to write an efficient solution.

The array $a = [a_1, a_2, \ldots, a_n]$ $(1 \le a_i \le n)$ is given. Its element a_i is called special if there exists a pair of indices l and r $(1 \le l < r \le n)$ such that $a_i = a_l + a_{l+1} + \ldots + a_r$. In other words, an element is called special if it can be represented as the sum of **two or more consecutive elements** of an array (no matter if they are special or not).

Print the number of special elements of the given array a.

For example, if $n = 9$ and $a = [3, 1, 4, 1, 5, 9, 2, 6, 5]$, then the answer is 5:

- $a_3 = 4$ is a special element, since $a_3 = 4 = a_1 + a_2 = 3 + 1$;
- $a_5 = 5$ is a special element, since $a_5 = 5 = a_2 + a_3 = 1 + 4$;
- $a_6 = 9$ is a special element, since $a_6 = 9 = a_1 + a_2 + a_3 + a_4 = 3 + 1 + 4 + 1$;
- $a_8 = 6$ is a special element, since $a_8 = 6 = a_2 + a_3 + a_4 = 1 + 4 + 1$;
- $a_9 = 5$ is a special element, since $a_9 = 5 = a_2 + a_3 = 1 + 4$.

Please note that some of the elements of the array a may be equal — if several elements are equal and special, then all of them should be counted in the answer.

Input

The first line contains an integer t $(1 \le t \le 1000)$ — the number of test cases in the input. Then t test cases follow.

Each test case is given in two lines. The first line contains an integer n $(1 \le n \le 8000)$ — the length of the array a. The second line contains integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le n)$.

It is guaranteed that the sum of the values of n for all test cases in the input does not exceed 8000.

Output

Print t numbers — the number of special elements for each of the given arrays.

Example

input Copy

```
5
9
3 1 4 1 5 9 2 6 5
3
1 1 2
5
1 1 1 1 1
8
8 7 6 5 4 3 2 1
1
1
```

output Copy

```
5
1
0
4
0
```