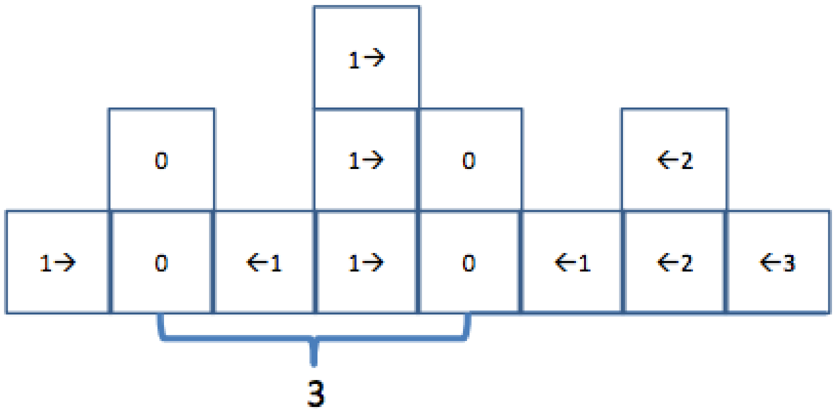
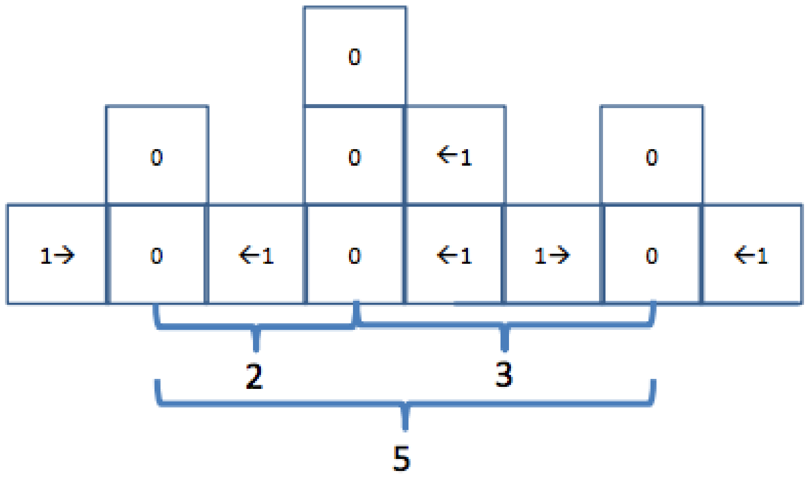
Beautiful Mountains

Paco loves playing with stacks of blocks. He likes to pretend that they are mountains, and he loves making his own terrain. Lately, he's been restricting himself to a particular way of rearranging the blocks. He puts all of his stacks of blocks into a straight line. Then, he only changes the arrangement one block at a time. Paco does this by finding two adjacent stacks of blocks and moving one block from one stack to the other.

Paco has made all sorts of arrangements of his 'mountains' using this technique. Now he has decided to make his most beautiful arrangements yet. Paco finds a mountain range beautiful if, for every pair of mountains, the distance between the two mountains is a prime number (that's every pair, not just every adjacent pair). A mountain range with a single stack is beautiful by default. Paco considers a stack of blocks to be a mountain if it has at least one block.



This diagram shows an initial configuration of the blocks, and a way to make two stacks, at a distance of three apart, with 13 moves. However, with only 6 moves, Paco can make a beautiful arrangement with 3 stacks:



Given a current arrangement of blocks, what is the minimum amount of effort needed for Paco to make it beautiful?

Input

There will be several test cases in the input. Each test case will begin with a line with one integer **n** (1≤**n**≤30,000), which is the number of stacks of blocks. On the next line will be **n** integers **b**

(0≤**b**≤1,000), indicating the number of blocks in that stack. The integers will be separated by a single space, with no leading or trailing spaces. Note that every stack will be listed, even those with zero blocks. End of input will be marked by a line with a single **0**.

Output

For each test case, output a single integer indicating the least number of moves required to make Paco's stacks of blocks 'beautiful'. Output no spaces, and do not separate answers with blank lines.

Example

Input:

```
5
1 2 1 2 1
8
1 2 1 3 2 1 2 1
0
```

Output:

```
3
6
```