You must hurry to find Professor Boolean's secret lab. You fear it may be too late.

Rumor has it that one of the pirates in this tavern knows where it is. There's a whole row of them seated at the bar. You start by talking to the pirate seated on the left-most barstool. However, the pirate redirects you to another pirate. Fine... you go to talk to that one. To your great annoyance, that pirate redirects you to yet another pirate! And on and on it goes. Is there no end to this madness? You notice each pirate has a number tattooed on his arm and decide to ID each man by his number. Each pirate redirects to a different pirate, other than himself. Because of this, it is guaranteed that you will start going in loops talking to them.

Write a function, answer(numbers), which returns the number of pirates which form a loop, given that you start by talking to the left-most pirate, 0. numbers will be an array of non-negative integers such that number[m] is the number of the pirate to whom pirate m redirects. No pirate redirects to himself. The left-most pirate is number 0, the next is number 1, and so on. Each element in the numbers list will be in the range [0, n-1] where n is the length of the numbers list.

For example, suppose the numbers list were [1, 2, 3, 4, 3].  Pirate 0 redirects to pirate 1, who redirects to pirate 2, who redirects to pirate 3, who redirects to pirate 4, who redirects back to pirate 3. Then, we are stuck going between pirate 3 and pirate 4 forever.  There are two pirates in this loop, thus the answer would be 2. Note that even though you visited pirates 0, 1, and 2, they are not part of the loop and don’t factor into the answer.

The number of pirates will be at least 2 and no more than 5000.

**Test cases:**

Inputs:  numbers = [1,0]

Output: 2

Inputs:  numbers = [1,2,1]

Output: 2

Inputs:  numbers = [2,0,4,3,5,2,1]

Outputs: 3

**Method Signature (Java):**

public static int answer(int [] pirates)