Given an unsorted array, find the smallest missing positive integer

input: [3,4,-1,1] | input: [1,2,0] | input: [7,8,9,11,12]
output: 2 | output: 3 | output: 1

* smallest positive integer → the smallest positive integer is 1 → This also tells us that 1 is our lower bound
∴ we ignore all negative numbers and zero. Our upper bound becomes len(array)+1 ... Why? B/c if we have an input like [1,2,3,4] the next smallest missing integer is 5 ... len(array)+1

```
def firstMissingPositive(array):
    if 1 not in nums:
        return 1
```
base case

```
    for index, value in enumerate(array):
        if value <= 0:
            array[index] = len(array)+1
```
Here we identify all the negative numbers and zeros and replace them with len(array)+1

```
    for element in array:
        num = abs(num)
        if num <= len(array):
            array[n-1] = -1 * abs(array[n-i])
```
Here we place our markers, identifying all relevant elements in the input array
Treating the input array as an array of indicies and make elements present at the given index negative
For example [3,1,4,0,-1] → [-3,1,-4,-6,6] *remember all zeros and negative numbers were replaced with len(array)+1
but we made the first element (3) negative b/c the input array contains a 1
the third element (4) negative b/c the input array contains a 3
the fourth element (6) negative b/c the input array contains a 4
} this is the repeative action we do to every element in the array

```
    for index in range(len(array)):
        if array[i] > 0:
            return i+1
```
Return the first index that has a positive element
Working from the previous example: [-3,1,-4,-6,6]
The first index with a positive element is 1 → index: 1 + 1 = 2
The negative numbers are our markers

* remember arrays are 0 based so we need to adjust and "+1" when dealing with arrays

Brute force Solution

```
def firstMissingInteger(nums):
    for index in range(1, len(nums)+2):
        if index not in nums:
            return index
```

This solution runs in $O(n^{n-1})$ time
Checks the array at every iteration

Walk through

input array: [3,2,1,-1,6]

1st iteration

```
for index,value in enumerate(nums):
    if value <= 0:
        nums[index] = len(nums)+1
```

3 <= 0 → #F

2 <= 0 → #F

1 <= 0 → #F

-1 <= 0 → #T

replace with len(nums)+1 = 6

6 <= 0 → #F

nums = [3,2,1,6,6]

```
for num in nums:
    num = abs(num)
    if num <= len(nums):
        nums[num-1] = -1 * abs(nums[num-1])
```

3 ≤ 6 → #True

nums[3-1] = -1 * abs(nums[3-1])

nums[2] = -1 * abs(nums[2])

nums[2] = -1 * 1

nums[2] = -1

2 ≤ 6 → #True

nums[2-1] = -1 * abs(nums[2-1])

nums[1] = -1 * abs(nums[1])

nums[1] = -1 * 2

nums[1] = -2

1 ≤ 6 → #true

nums[1-1] = -1 * abs(nums[1-1])

nums[0] = -1 * abs(nums[0])

nums[0] = -1 * 3

nums[0] = -3

6 ≤ 6 → #True

nums[6-1] = -1 * abs(nums[6-1])

nums[5] = -1 * abs(nums[5])

nums[5] = -1 * 6

nums[5] = -6

nums = [-3,-2,-1,6,-6]

```
for i in range(len(nums)):
    if nums[i] > 0:
        return i + 1
```

if nums[0] > 0:

-3 > 0

#F

if nums[1] > 0:

-2 > 0

#F

if nums[2] > 0:

-1 > 0

#F

if nums[3] > 0:

6 > 0

#T

return 3 + 1

return 4