$$\begin{tikzcd}
& \text{Space} \arrow[dl, dashed] \arrow[dr, dashed] & \\
\text{In-Place} & & \text{Out-Place}
\end{tikzcd}$$

In-Place Algo:
Space Complexity : O(1)

```
int[] arr = {1,2,3,4};


square :

for(int i = 0 ; i < arr.length ; i++)
{
    arr[i] = arr[i] * arr[i];
}

int[] arr = {2,4,8,16};
```

Out-Place Algo:
Space Complexity : O(n)

```
int[] arr = {1,2,3,4};
int[] square = new int[arr.length];

for(int i = 0 ; i < arr.length ; i++)
{
    square[i] = arr[i] * arr[i];
}
```

287. Find the Duplicate Number

Medium 12229 1366 Add to List Share

Given an array of integers `nums` containing `n + 1` integers where each integer is in the range `[1, n]` inclusive.

There is only **one repeated number** in `nums`, return *this repeated number*.

You must solve the problem **without** modifying the array `nums` and uses only constant extra space.

Example 1:

```
Input: nums = [1,3,4,2,2]
Output: 2
```

Example 2:

```
Input: nums = [3,1,3,4,2]
Output: 3
```

Constraints:

- `1 <= n <= `10^5
- `nums.length == n + 1`
- `1 <= nums[i] <= n`
- All the integers in `nums` appear only **once** except for **precisely one integer** which appears **two or more** times.

Follow up:

- How can we prove that at least one duplicate number must exist in `nums`?
- Can you solve the problem in linear runtime complexity?

OutPlace Algo

0	1	2	3	4
1	3	4	2	2

Time Complexity : O(n)

Excepted Output : 2

OutPlace Algo

Space Complexity : O(n)

-> Take a boolean[] array with size n. Has the default values are False.

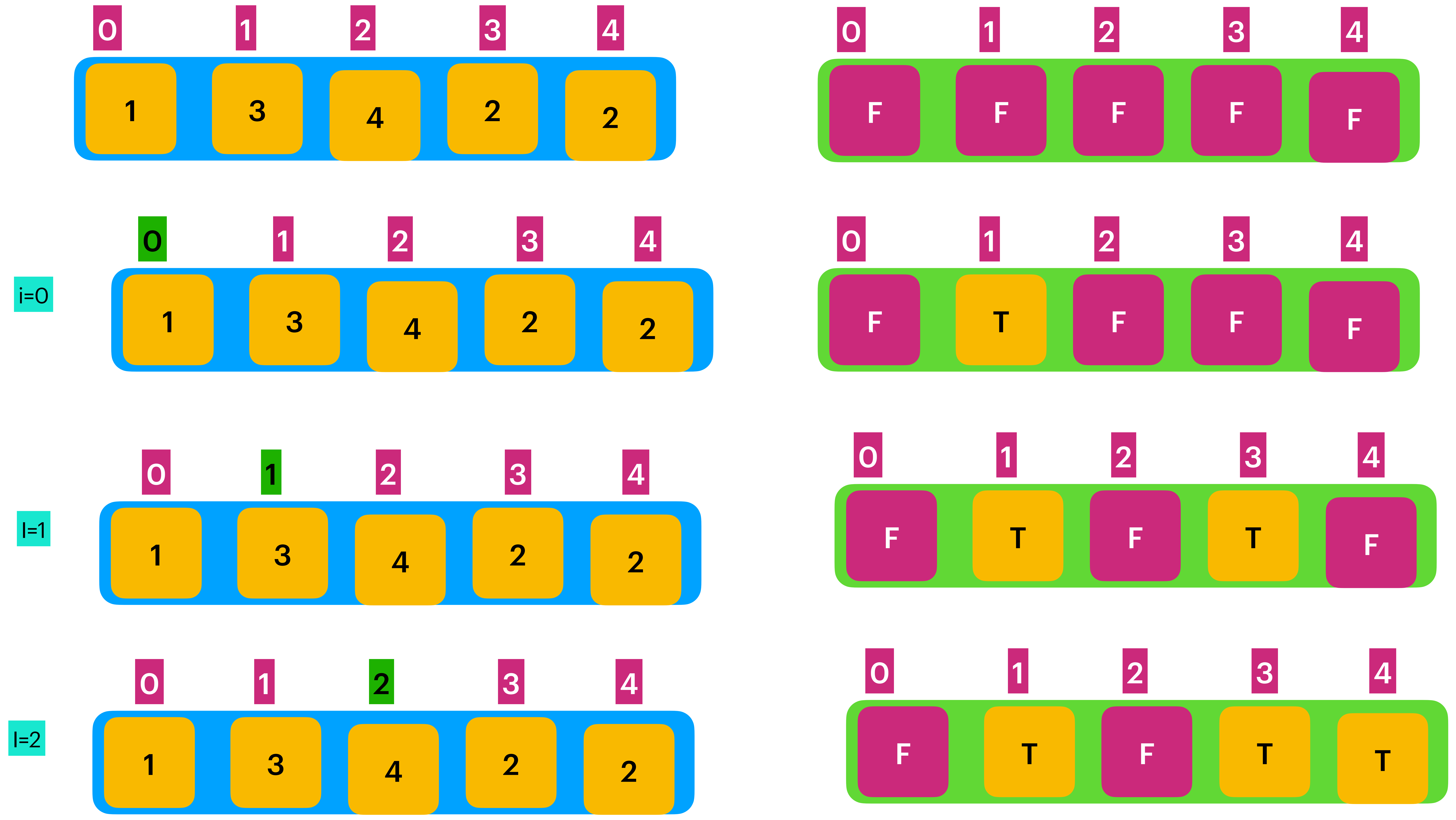
-> For each iteration of input array, update the respective boolean[] array index value with True .

-> If the value is repeated for the second visit you already find true in boolean[] array so return the value.

Clue

Hit is values are in the range of [1,n] and
the length of the array is : n+1
So that each value can be uniquely identified by an array index.

OutPlace Algo



OutPlace Algo

I:3

0	1	2	3	4
1	3	4	2	2

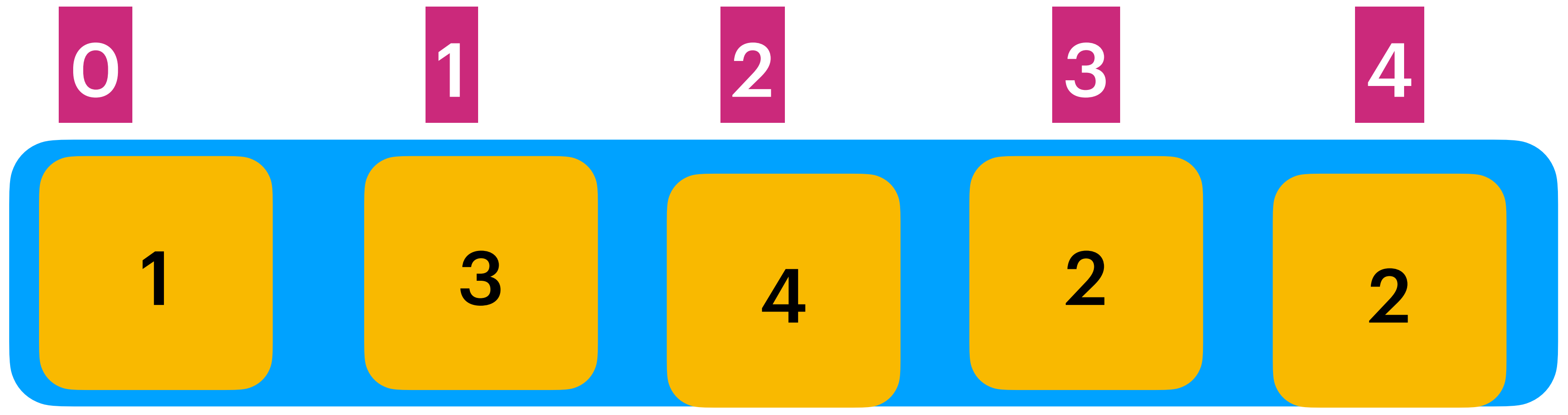
0	1	2	3	4
F	T	T	T	T

I:4

0	1	2	3	4
1	3	4	2	2

0	1	2	3	4
F	T	T	T	T

InPlace Algo



Time Complexity : O(n)

Excepted Output : 2

InPlace Algo

Space Complexity : O(1)

-> Here we replace the sign of index value mapping to current iterated value with in the same input array.

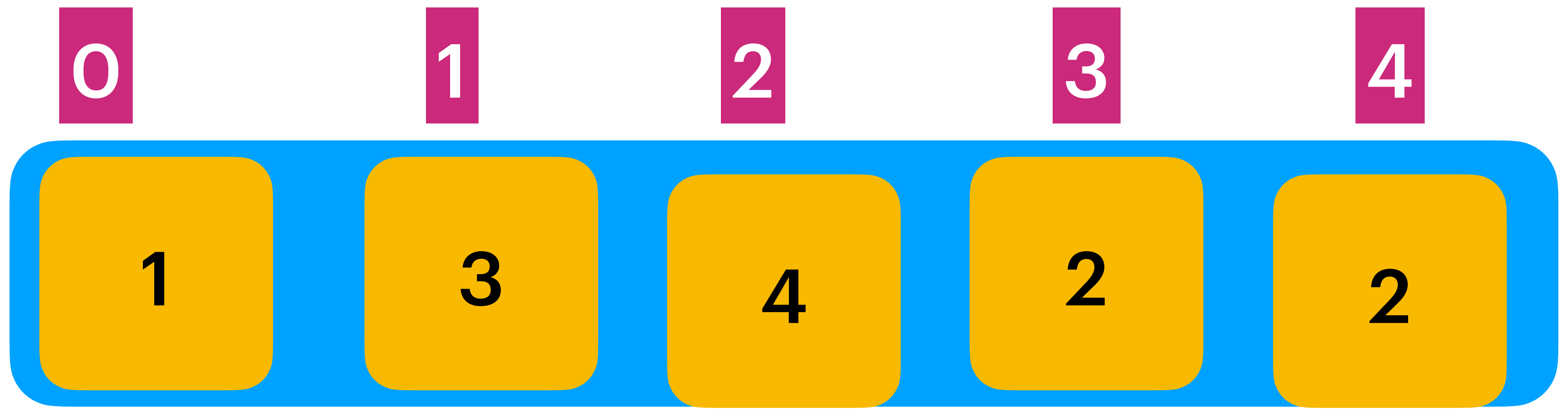
-> If the value is repeated then the sign would have already been updated so that it is the duplicate.

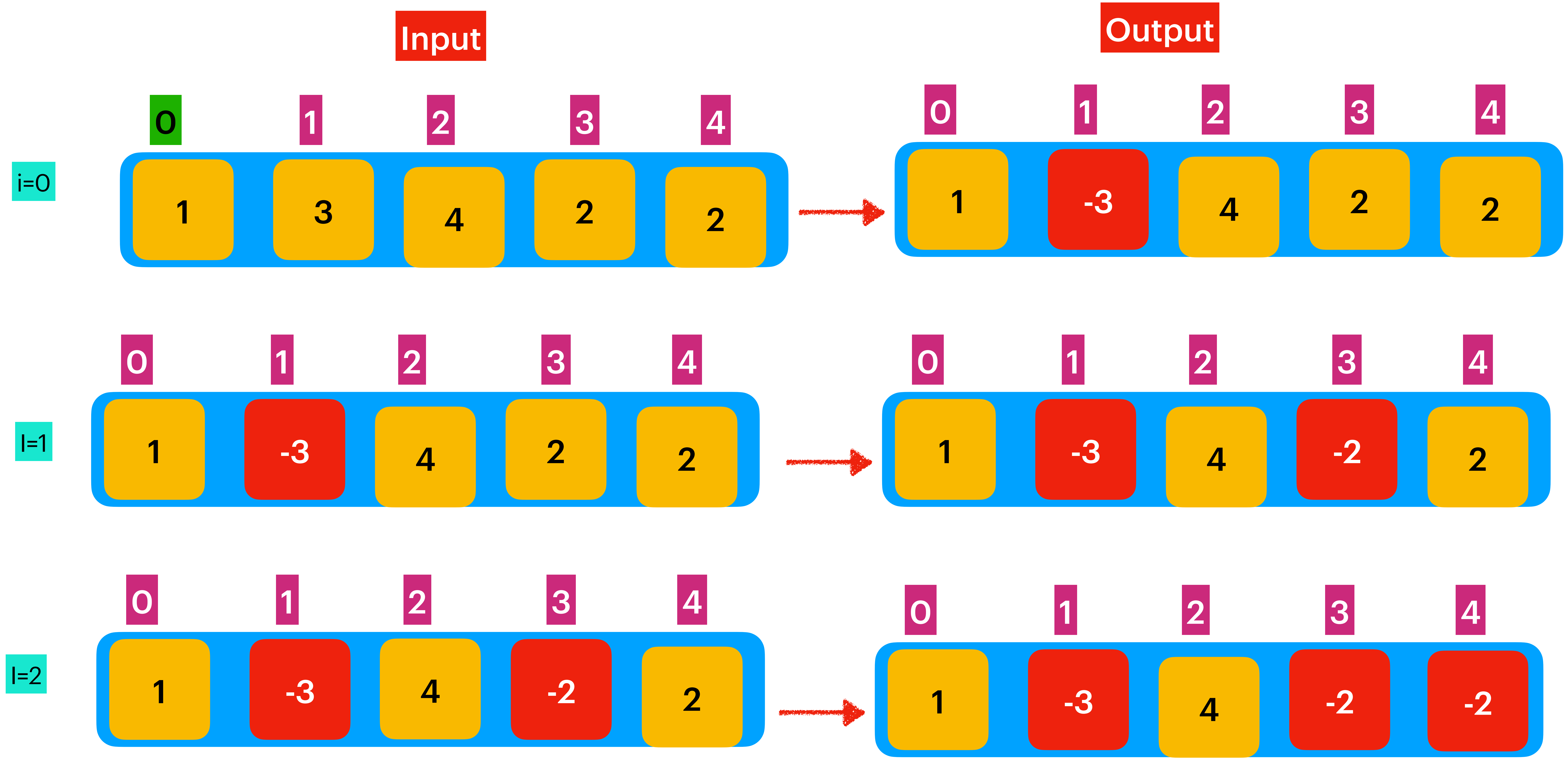
Note : As we are updating given input array, always take the absolute value for each iteration.

Clue

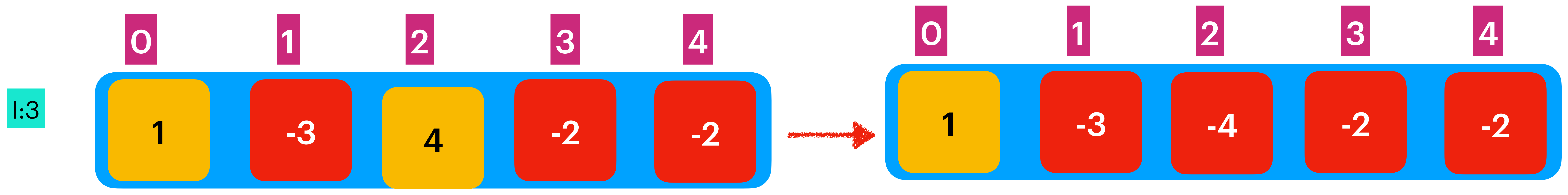
Hit is values are in the range of [1,n] and
the length of the array is : n+1
So that each value can be uniquely identified by an array index.

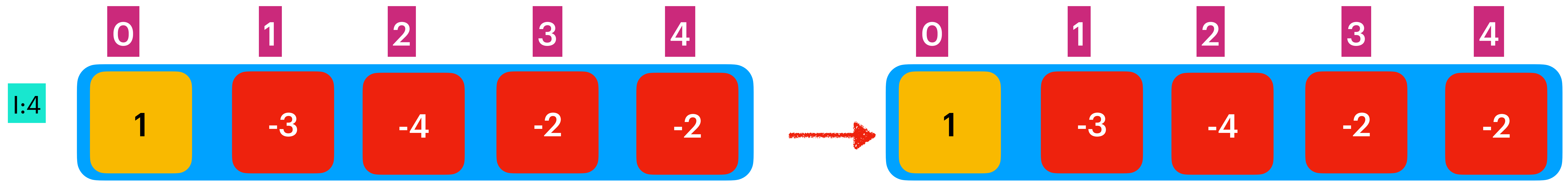
InPlace Algo





In-Place Algo

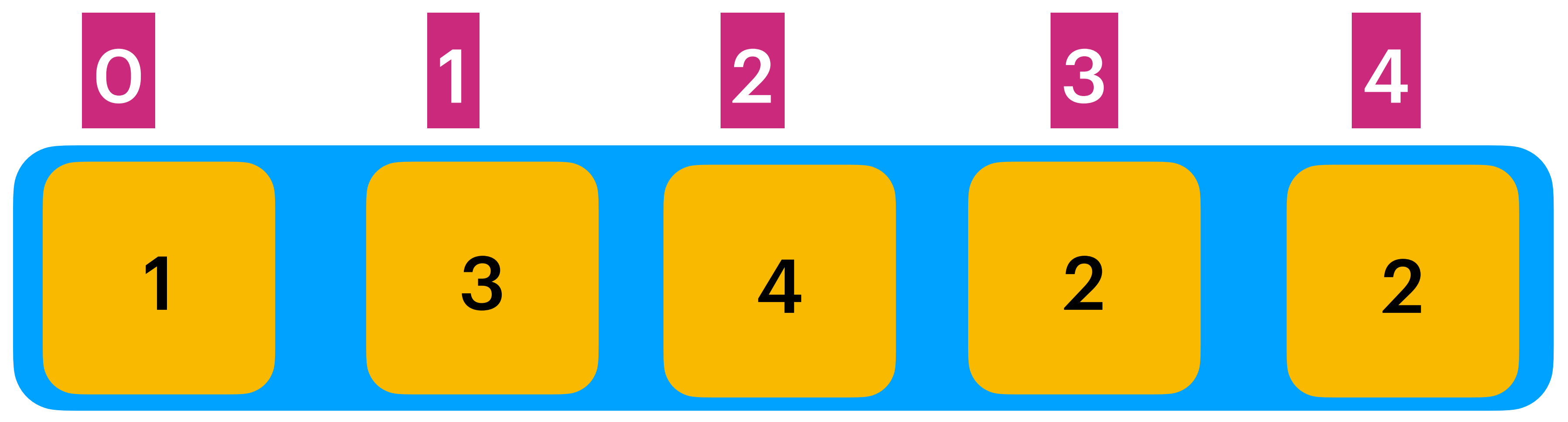




Time Complexity : O(n)

Excepted Output : 2

Space Complexity : O(1)



41. First Missing Positive

Hard 8998 1283 Add to List Share

Given an unsorted integer array `nums`, return the smallest missing positive integer.

You must implement an algorithm that runs in `O(n)` time and uses constant extra space.

Example 1:

```
Input: nums = [1,2,0]
Output: 3
```

Example 2:

```
Input: nums = [3,4,-1,1]
Output: 2
```

Example 3:

```
Input: nums = [7,8,9,11,12]
Output: 1
```

Constraints:

- $1 <= \texttt{nums.length} <= 5 * 10^5$
- $-2^{31} <= \texttt{nums[i]} <= 2^{31} - 1$