**[First Missing Positive](https://leetcode.com/problems/first-missing-positive/)**

Given an unsorted integer array nums. Return the *smallest positive integer* that is *not present* in nums.

You must implement an algorithm that runs in O(n) time and uses O(1) auxiliary space.

**Example 1:**

**Input:** nums = [1,2,0]

**Output:** 3

**Explanation:** The numbers in the range [1,2] are all in the array.

**Example 2:**

**Input:** nums = [3,4,-1,1]

**Output:** 2

**Explanation:** 1 is in the array but 2 is missing.

**Example 3:**

**Input:** nums = [7,8,9,11,12]

**Output:** 1

**Explanation:** The smallest positive integer 1 is missing.

**Constraints:**

* 1 <= nums.length <= 105
* -231 <= nums[i] <= 231 - 1

class Solution {

public:

    int firstMissingPositive(vector<int>& nums) {

        int n= size(nums);

        for(int i=0;i<n;i++){

            int x=nums[i]; // x = current element

        // x>=1 && x<=n : to check if x is in range[1, n]

        // x != i+1 : skip if at index i correct element is present.

        // nums[x-1]!=x: skip if at index x-1 correct element is present

            while(x>=1 && x<=n && x!=i+1 && nums[x-1]!=x){

                swap(nums[x-1],nums[i]);

                x=nums[i];

            }

        }

        for(int i=0;i<n;i++){

            if(nums[i] == i+1)continue;

                return i+1;

        }

        return n+1;

    }

};

Link : <https://leetcode.com/problems/first-missing-positive/?envType=daily-question&envId=2024-03-26>