2161. Partition Array According to Given Pivot

You are given a **0-indexed** integer array `nums` and an integer `pivot`. Rearrange `nums` such that the following conditions are satisfied:

- Every element less than `pivot` appears **before** every element greater than `pivot`.
- Every element equal to `pivot` appears **in between** the elements less than and greater than `pivot`.
- The **relative order** of the elements less than `pivot` and the elements greater than `pivot` is maintained.
    - More formally, consider every `pi`, `pj` where `pi` is the new position of the `ith` element and `pj` is the new position of the `jth` element. If `i < j` and **both** elements are smaller (*or larger*) than `pivot`, then `pi < pj`.

Return `nums` *after the rearrangement.*

Example 1:

```
Input: nums = [9,12,5,10,14,3,10], pivot = 10
Output: [9,5,3,10,10,12,14]
Explanation:
The elements 9, 5, and 3 are less than the pivot so they are on the left side of the array.
The elements 12 and 14 are greater than the pivot so they are on the right side of the array.
The relative ordering of the elements less than and greater than pivot is also maintained. [9, 5, 3] and [12, 14] are the respective orderings.
```

Example 2:

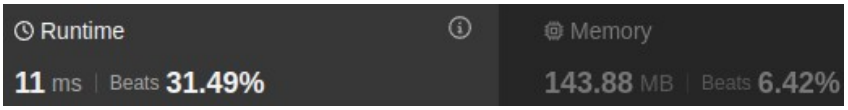
```
Input: nums = [-3,4,3,2], pivot = 2
Output: [-3,2,4,3]
Explanation:
The element -3 is less than the pivot so it is on the left side of the array.
The elements 4 and 3 are greater than the pivot so they are on the right side of the array.
The relative ordering of the elements less than and greater than pivot is also maintained. [-3] and [4, 3] are the respective orderings.
```

Constraints:

- $1 \leq nums.length \leq 10^5$
- $-10^6 \leq nums[i] \leq 10^6$
- `pivot` equals to an element of `nums`.

2161. Partition Array According to Given Pivot

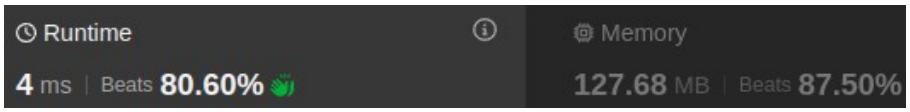
```
/*
   Two passes: Mapping with array+fill answer
   Time complexity: O(2n)
   Space complexity: O(n)
*/
```




```
class Solution {
   public:
      std::vector<int> pivotArray(std::vector<int>& nums, int pivot) {
         // Pass #1: Mapping
         // mapping[0]: contains all elements less than pivot
         // mapping[1]: contains all elements equal pivot
         // mapping[2]; contains all elements greater than pivot
         std::vector<std::vector<int>> mapping(3);
         for(auto& e: nums){
            if(e<pivot) mapping[0].push_back(e);
            else if(e==pivot) mapping[1].push_back(e);
            else mapping[2].push_back(e);
         }

         // Pass #2: Fill the answer
         std::vector<int> ans;
         for(int i=0;i<=2;++i){
            for(auto& e: mapping[i]) ans.push_back(e);
         }

         return ans;
      }
};
```

2161. Partition Array According to Given Pivot

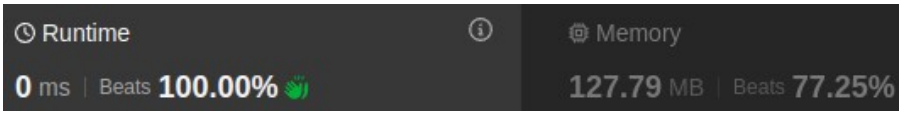
```
/*
   Two passes: left to right+right to left
   Time complexity: O(2n)
   Space complexity: O(1)
*/
```




```
class Solution {
   public:
      std::vector<int> pivotArray(std::vector<int>& nums, int pivot) {
         int n=nums.size();

         // Create answer array initialized with pivot, in order to
         // focus on elements lesser and greater that pivot
         std::vector<int> ans(n,pivot);

         // Pass#1: Left to right

         int wl=0; // Pointer to track the next smaller element's position
         // Iterate on elements from left to right, to maintain the relative order of all lesser elements
         for(int cur_index=0;cur_index<n;++cur_index){
            // If the current element at the current index is lesser than pivot:
            if(nums[cur_index]<pivot){
               ans[wl]=nums[cur_index]; // place at its correct position in answer array, pointed by wl
               wl++; // Prepare the position for the next smaller in answer array
            }
         }

         // Pass#2: Left to right

         int wg=n-1; // Pointer to track the next greater element's position
         // Iterate on elements from right to left, to maintain the relative order of all greater elements
         for(int cur_index=n-1;cur_index>=0;--cur_index){
            // If the current element at the current index is greater than pivot:
            if(nums[cur_index]>pivot){
               ans[wg]=nums[cur_index]; // place at its correct position in answer array, pointed by wg
               wg--; // Prepare the position for the next greater in answer array
            }
         }

         return ans;
      }
};
```

2161. Partition Array According to Given Pivot

```
/*
   One passes: two pointers
   Time complexity: O(n)
   Space complexity: O(1)
*/
```




```
class Solution {
   public:
      std::vector<int> pivotArray(std::vector<int>& nums, int pivot) {
         int n=nums.size();

         // Create answer array initialized with pivot, in order to
         // focus on elements lesser and greater that pivot
         std::vector<int> ans(n,pivot);

         // Pointer wl to track the next smaller element's position
         // Pointer wg to track the next greater element's position
         int wl=0,wg=n-1;

         // Iterate over all elements from left to right
         for(int cur_index=0;cur_index<n;++cur_index){
            // If the leftmost current element at the current index is lesser than pivot:
            // place at its correct position in answer array, pointed by wl
            // and prepare the position for the next smaller in answer array
            if(nums[cur_index]<pivot) ans[wl]=nums[cur_index],wl++;

            // If the rightmost current element at the current index is greater than pivot:
            // place at its correct position in answer array, pointed by wg
            // and prepare the position for the next greater in answer array
            if(nums[n-1-cur_index]>pivot) ans[wg]=nums[n-1-cur_index],wg--;
         }

         return ans;
      }
};
```