977. Squares of a Sorted Array

Easy 8.3K 201

Companies

Given an integer array `nums` sorted in **non-decreasing** order, return *an array of **the squares of each number** sorted in non-decreasing order.*

Example 1:

```
Input: nums = [-4,-1,0,3,10]
Output: [0,1,9,16,100]
Explanation: After squaring, the array becomes [16,1,0,9,100].
After sorting, it becomes [0,1,9,16,100].
```

Example 2:

```
Input: nums = [-7,-3,2,3,11]
Output: [4,9,9,49,121]
```

Constraints:

- `1 <= nums.length <= 10^4`
- `-10^4 <= nums[i] <= 10^4`
- `nums` is sorted in **non-decreasing** order.

Follow up: Squaring each element and sorting the new array is very trivial, could you find an `O(n)` solution using a different approach?

Accepted 1.5M | Submissions 2.1M | Acceptance Rate 71.8%

Approach 1: just square each element and then sort the array

TC: O(nlogn)

$S(n)$: sorting space

Approach 2:

① find the first positive element and put pointer on that first element.

② keep one pointer on the last negative element.

Positive pointer moves forward and negative pointer moves backward.

③ Compare the absolute values pointed by p and n and then add the minimum b/w them to the ans vector by squaring it and move the pointer on minimum value side.

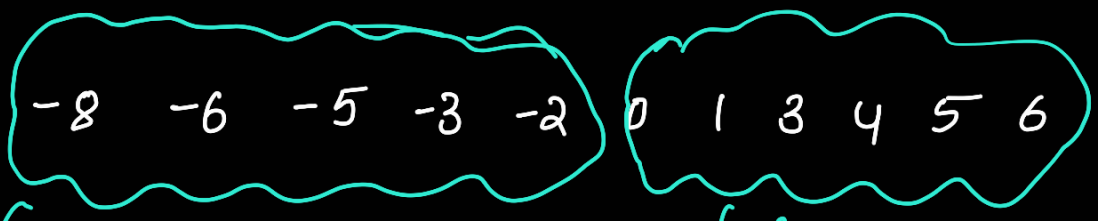
```
class Solution {
public:
    vector<int> sortedSquares(vector<int>& nums) {
        int i=0;
        int j;
        vector<int> ans;
        while(i<nums.size() && nums[i]<0) i++;

        j=i-1;
        while(j>-1 && i<nums.size())
            {
            if(abs(nums[j]) <= abs(nums[i]))        ans.push_back(nums[j]*nums[j]),j--;
            else ans.push_back(nums[i]*nums[i]),i++;
            }

      while(j>-1) ans.push_back(nums[j]*nums[j]),j--;

      while(i<nums.size()) ans.push_back(nums[i]*nums[i]),i++;


       return ans;
    }
};
```

To avoid error of accessing out of range element here

Idea used in merge operation of merge sort.

$T(n) : O(n)$
$S(n) : O(n)$

Approach 3: Using Two pointers approach



↳ The absolute values are in increasing order from right to left.

↳ in increasing order from left to right

So lets think of them like two different sorted arrays and we will just do the merge operation of those two arrays by comparing squares of elements.

here we start comparing from the max elements and fill the answer array from right to left..

```
i=0 , j = n-1 , k=n-1

while(i ≤ j)
{
   if (a[i]^2 ≥ a[j]^2)
        ans[k--] = a[i++]^2
   else
        ans[k--] = a[j--]^2
}
```

T(n) : O(n)

ง (ก) : บ (ก)