Square and Sort

# Question

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]

# Pseudo Code

\*returnSize <- numsSize

Declare and Initialize tempVar to zero

Dynamically Allocate the Memory to an Array

Run the first for loop

Multiply the value of the array by itself

Then store it in the same location

Run the second(Outer) for loop

Run the Inner for loop

tempVar <- arr[i]

Arr[i] <- arr[i+1]

Arr[i+1] <- tempVar

Run the Third for loop

Assign the values of arr to the array in Heap

Reference: https://discuss.codechef.com/t/how-to-tackle-returnsize-in-leetcode/28713/6

# Source Code

## V 1.0

1. /\*\*
2. \* Note: The returned array must be malloced, assume caller calls free().
3. \*/
4. int\* sortedSquares(int\* nums, int numsSize, int\* returnSize){
5. \*returnSize = numsSize;
6. int tempVar = 0;
7. int\* ptr = (int\*)malloc(numsSize \* sizeof(int));
9. for(int i=0 ; i<numsSize ; i++) {
10. nums[i] \*= nums[i];
11. }
13. for(int i=0 ; i<(numsSize - 1) ; i++) {
15. for(int j=0 ; j<(numsSize - i - 1) ; j++) {
17. if(nums[j] > nums[j+1]) {
18. tempVar = nums[j];
19. nums[j] = nums[j+1];
20. nums[j+1] = tempVar;
21. }
22. }
23. }
25. for(int i=0 ; i<numsSize ; i++)
26. ptr[i] = nums[i];
28. return ptr;
29. }