977. Squares of a sorted array:

I/P: nums = [-4, -1, 0, 3, 10]

O/P: [0, 1, 9, 16, 100]

Explanation: After squaring, the array becomes [16, 1, 0, 9, 100]. After sorting

Brute force approach:

1. create a new array
2. square each element
3. sort new array
4. return it.

```
class Solution {
    public int[] sortedSquares(int[] nums) {
        int n = nums.length;
        int[] result = new int[n];
        for(int i = 0; i < n; i++) {
            result[i] = nums[i] * nums[i];
        }
        Arrays.sort(result);
        return result;
    }
}
```

Time → squaring → o(n) sorting → o(n log n)

space → O(n).

optimized approach (two pointer):

```
class Solution {
public int[] sortedSquares(int[] nums){
    int n = nums.length;
    int left = 0;
    int right = n-1;
    int[] result = new int[n];
    int pos = n-1;
while (left <= right) {
    int leftSq = nums[left] * nums[left];
    int rightSq = nums[right] * nums[right];
if (leftSq > rightSq){
    result[pos] = leftSq;
    left++;
}else {
    result[pos] = rightSq;
    right--;
    }
    pos--;
    }
    return result;
}
}.
```

Time → O(n) space → O(n).