

[← Back To Course Home](#)

Grokking the Coding Interview: Patterns for Coding Questions

6% completed



 Search Course

grokking-the-coding-interview/N8vB7OVYo2D)

Solution Review: Problem Challenge 1

(/courses/grokking-the-coding-interview/N0o9QnPLKNv)

Problem Challenge 2

(/courses/grokking-the-coding-interview/YQ8N2OZq0VM)

Solution Review: Problem Challenge 2

(/courses/grokking-the-coding-interview/xl2g3vvrMq3)

Problem Challenge 3

(/courses/grokking-the-coding-interview/3wDJAYG2pAR)

Solution Review: Problem

Squaring a Sorted Array (easy)

We'll cover the following ^

- Problem Statement
- Try it yourself
- Solution
 - Code
 - Time complexity
 - Space complexity

Problem Statement

Given a sorted array, create a new array containing **squares of all the number of the input array** in the sorted order.

Example 1:

Input: [-2, -1, 0, 2, 3]

Output: [0, 1, 4, 4, 9]

Example 2:

Input: [-3, -1, 0, 1, 2]

Output: [0 1 1 4 9]



[← Back To Course Home](#)

Grokking the Coding Interview: Patterns for Coding Questions

6% completed



grokking-the-coding-interview/N8vB7OVYo2D)

Solution Review: Problem Challenge 1

(/courses/grokking-the-coding-interview/N0o9QnPLKNv)

Problem Challenge 2

(/courses/grokking-the-coding-interview/YQ8N2OZq0VM)

Solution Review: Problem Challenge 2

(/courses/grokking-the-coding-interview/xl2g3vvrMq3)

Problem Challenge 3

(/courses/grokking-the-coding-interview/3wDJAYG2pAR)

Solution Review: Problem

Try it yourself

Try solving this question here:



Java



Python3



JS



C++

```
1 class SortedArraySquares {
2
3     public static int[] makeSquares(int[] arr) {
4         int[] squares = new int[arr.length];
5         // TODO: Write your code here
6         int n = arr.length;
7         int left = 0, right = n-1;
8         int index = n-1;
9         while(left<=right)
10            {
11                if(arr[left]*arr[left]>arr[right]*arr[right]){
12                    squares[index] = arr[left]*arr[left];
13                    left++;
14                }
15                else{
16                    squares[index] = arr[right]*arr[right];
17                    right--;
18                }
19                index--;
20            }
21        return squares;
22    }
23 }
24
```



[← Back To Course Home](#)

Grokking the Coding Interview: Patterns for Coding Questions


6% completed 




- [Grokking the Coding Interview/N8vB7OVYo2D](#)
- [Solution Review: Problem Challenge 1 \(/courses/grokking-the-coding-interview/N0o9QnPLKNv\)](#)
- [Problem Challenge 2 \(/courses/grokking-the-coding-interview/YQ8N2OZq0VM\)](#)
- [Solution Review: Problem Challenge 2 \(/courses/grokking-the-coding-interview/xl2g3vxrMq3\)](#)
- [Problem Challenge 3 \(/courses/grokking-the-coding-interview/3wDJAYG2pAR\)](#)
- [Solution Review: Problem](#)

Test

Save *

Reset 



Show Results

Show Console

2 of 2 Tests Passed

Result	Input	Expected Output	Actual Output	Reason
✓	makeSquares([-2, -1, 0, 2, 3])	[0, 1, 4, 4, 9]	[0, 1, 4, 4, 9]	Succeeded
✓	makeSquares([-3, -1, 0, 1, 2])	[0, 1, 1, 4, 9]	[0, 1, 1, 4, 9]	Succeeded

5.280s

Solution

This is a straightforward question. The only trick is that we can have negative numbers in the input array, which will make it a bit difficult to generate the output array with squares in sorted order.

An easier approach could be to first find the index of the first non-negative number in the array. After that, we can use **Two Pointers** to iterate the array. One pointer will move forward to iterate the non-negative numbers and the other pointer will move backward to iterate the negative numbers. At any step, whichever number gives us a bigger square will be added to the output array. For the above-mentioned Example-1, we will do something like this:

[← Back To Course Home](#)

Grokking the Coding Interview: Patterns for Coding Questions

6% completed



[Grokking the Coding Interview/N8vB7OVYo2D](#)

[Solution Review: Problem Challenge 1](#)

[\(/courses/grokking-the-coding-interview/N0o9QnPLKNv\)](#)

[Problem Challenge 2](#)

[\(/courses/grokking-the-coding-interview/YQ8N2OZq0VM\)](#)

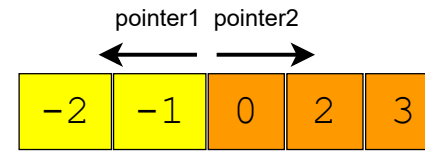
[Solution Review: Problem Challenge 2](#)

[\(/courses/grokking-the-coding-interview/xl2g3vvrMq3\)](#)

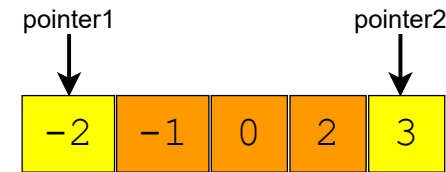
[Problem Challenge 3](#)

[\(/courses/grokking-the-coding-interview/3wDJAYG2pAR\)](#)

[Solution Review: Problem](#)



Since the numbers at both the ends can give us the largest square, an alternate approach could be to use two pointers starting at both the ends of the input array. At any step, whichever pointer gives us the bigger square we add it to the result array and move to the next/previous number according to the pointer. For the above-mentioned Example-1, we will do something like this:



Code #

Here is what our algorithm will look like:

Java

Python3

C++

JS

```
1 class SortedArraySquares {
2
3     public static int[] makeSquares(int[] arr) {
4         int n = arr.length;
5         int[] squares = new int[n];
6         int highestSquareIdx = n - 1;
7         int left = 0, right = arr.length - 1;
8         while (left <= right) {
9             int leftSquare = arr[left] * arr[left];
```



[← Back To Course Home](#)

Grokking the Coding Interview: Patterns for Coding Questions

6% completed



grokking-the-coding-interview/N8vB7OVYo2D)

Solution Review: Problem Challenge 1

(/courses/grokking-the-coding-interview/N0o9QnPLKNv)

Problem Challenge 2

(/courses/grokking-the-coding-interview/YQ8N2OZq0VM)

Solution Review: Problem Challenge 2

(/courses/grokking-the-coding-interview/xl2g3vvrMq3)

Problem Challenge 3

(/courses/grokking-the-coding-interview/3wDJAYG2pAR)

Solution Review: Problem

```
10     int rightSquare = arr[right] * arr[right];
11     if (leftSquare > rightSquare) {
12         squares[highestSquareIdx--] = leftSquare;
13         left++;
14     } else {
15         squares[highestSquareIdx--] = rightSquare;
16         right--;
17     }
18 }
19 return squares;
20 }
21
22 public static void main(String[] args) {
23
24     int[] result = SortedArraySquares.makeSquares(r
25     for (int num : result)
26         System.out.print(num + " ");
27     System.out.println();
28
29     result = SortedArraySquares.makeSquares(new int
30     for (int num : result)
31         System.out.print(num + " ");
```

Run

Save

Reset



Time complexity

The time complexity of the above algorithm will be $O(N)$ as we are iterating the input array only once.

Space complexity

The space complexity of the above algorithm will also be $O(N)$; this space will be used for the output array.

[← Back](#)

[Next →](#)

Remove Duplicates (easy)

Triplet Sum to Zero (medium)

☒ Mark as Completed

[← Back To Course Home](#)

Grokking the Coding Interview: Patterns for Coding Questions

6% completed



6% completed, meet the [criteria](#) and claim your course certificate!

[Claim Certificate](#)

Report an Issue

Ask a Question

(https://discuss.educative.io/tag/squaring-a-sorted-array-easy__pattern-two-pointers__grokking-the-coding-interview-patterns-for-coding-questions)

Search Course

[Grokking the Coding Interview/N8vB7OVYo2D](#)

Solution Review: Problem Challenge 1

(/courses/grokking-the-coding-interview/N0o9QnPLKNv)

Problem Challenge 2

(/courses/grokking-the-coding-interview/YQ8N2OZq0VM)

Solution Review: Problem Challenge 2

(/courses/grokking-the-coding-interview/xl2g3vvrMq3)

Problem Challenge 3

(/courses/grokking-the-coding-interview/3wDJAYG2pAR)

Solution Review: Problem