

Grokking the Coding Interview: Patterns for Coding Questions

75% completed



Pattern: Two Pointers ▾

Pattern: Fast & Slow pointers ▾

Pattern: Merge Intervals ▾

Pattern: Cyclic Sort ▾

Pattern: In-place Reversal of a LinkedList ▾

Pattern: Tree Breadth First Search ▾

Pattern: Tree Depth First Search ▾

Pattern: Two Heaps ▾

Pattern: Subsets ▾

Pattern: Modified Binary Search ▾

Pattern: Bitwise XOR ▾

Pattern: Top 'K' Elements ▴

- Introduction
- Top 'K' Numbers (easy)
- Kth Smallest Numbers (easy)
- 'K' Closest Points to the Origin (easy)
- Connect Ropes (easy)
- Top 'K' Frequent Numbers (medium)
- Frequency Sort (medium)
- Kth Largest Number in a Stream (medium)
- 'K' Closest Numbers (medium)
- Maximum Distinct Elements (medium)
- Sum of Elements (medium)
- Rearrange String (hard)
- Problem Challenge 1
- Solution Review: Problem Challenge 1
- Problem Challenge 2
- Solution Review: Problem Challenge 2
- Problem Challenge 3
- Solution Review: Problem Challenge 3

'K' Closest Points to the Origin (easy)

We'll cover the following ▴

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

Problem Statement

Given an array of points in the a $2D$ plane, find 'K' closest points to the origin.

Example 1:


```
Input: points = [[1,2],[1,3]], K = 1
Output: [[1,2]]
Explanation: The Euclidean distance between (1, 2) and the origin is sqrt(5).
The Euclidean distance between (1, 3) and the origin is sqrt(10).
Since sqrt(5) < sqrt(10), therefore (1, 2) is closer to the origin.
```


Example 2:


```
Input: point = [[1, 3], [3, 4], [2, -1]], K = 2
Output: [[1, 3], [2, -1]]
```


Try it yourself

Try solving this question here:

 Java

 Python3

 JS


 C++

```
1 class Point {
2
3   constructor(x, y) {
4     this.x = x;
5     this.y = y;
6   }
7
8   get_point() {
9     return "[" + this.x + ", " + this.y + "] ";
10  }
11 };
12
13 const find_closest_points = function(points, k) {
14   result = [];
15   // TODO: Write your code here
16   return result;
17 };
18
19
20 points = find_closest_points([new Point(1, 3), new Point(3, 4), new Point(2, -1)], 2)
21 result = "Here are the k points closest the origin: ";
22 for (i=0; i < points.length; i++)
23   result += points[i].get_point();
24 console.log(result);
25
```

RUN

SAVE

RESET



Close

Output

3.7178

Here are the k points closest the origin:

Solution

The **Euclidean distance** of a point $P(x,y)$ from the origin can be calculated through the following formula:

$$\sqrt{x^2 + y^2}$$

This problem follows the **Top 'K' Numbers** pattern. The only difference in this problem is that we need to find the closest point (to the origin) as compared to finding the largest numbers.

Following a similar approach, we can use a **Max Heap** to find 'K' points closest to the origin. While iterating through all points, if a point (say 'P') is closer to the origin than the top point of the max-heap, we will remove that top point from the heap and add 'P' to always keep the closest points in the heap.

Pattern: K-way merge

Introduction

Merge K Sorted Lists (medium)

Kth Smallest Number in M Sorted Lists (Medium)

Kth Smallest Number in a Sorted Matrix (Hard)

Smallest Number Range (Hard)

Problem Challenge 1

Solution Review: Problem Challenge 1

Pattern : 0/1 Knapsack (Dynamic Programming)

Introduction

0/1 Knapsack (medium)

Equal Subset Sum Partition (medium)

Subset Sum (medium)

Minimum Subset Sum Difference (hard)

Problem Challenge 1

Solution Review: Problem Challenge 1

Problem Challenge 2

Solution Review: Problem Challenge 2

Pattern: Topological Sort (Graph)

Introduction

Topological Sort (medium)

Tasks Scheduling (medium)

Tasks Scheduling Order (medium)

All Tasks Scheduling Orders (hard)

Alien Dictionary (hard)

Problem Challenge 1

Solution Review: Problem Challenge 1

Problem Challenge 2

Solution Review: Problem Challenge 2

Miscellaneous

Kth Smallest Number (hard)

Conclusions

Where to Go from Here

+

Create

Mark Course as Completed

Code #

Here is what our algorithm will look like:

JavaPython3C++JS

```
1  const Heap = require('./collections/heap'); //http://www.collectionsjs.com
2
3
4  class Point {
5    constructor(x, y) {
6      this.x = x;
7      this.y = y;
8    }
9
10   // used for max-heap
11   compare(other) {
12     return this.distance_from_origin() - other.distance_from_origin();
13   }
14
15   distance_from_origin() {
16     // ignoring sqrt to calculate the distance
17     return (this.x * this.x) + (this.y * this.y);
18   }
19
20   print_point() {
21     process.stdout.write(`${this.x}, ${this.y} `);
22   }
23 }
24
25 function find_closest_points(points, k) {
26   const maxHeap = new Heap([], null, ((a, b) => a.compare(b)));
27   // put first 'k' points in the max heap
28   for (i = 0; i < k; i++) {
```

RUN

SAVE

RESET

Close

Output

5.187s

Here are the k points closest the origin: [1, 3] [2, -1]

Time complexity #

The time complexity of this algorithm is $(N * \log K)$ as we iterating all points and pushing them into the heap.

Space complexity #

The space complexity will be $O(K)$ because we need to store 'K' point in the heap.

Interviewing soon? We've partnered with [Hired](#) so that companies apply to you instead of you applying to them. [See how](#)

← Back

Next →

Kth Smallest Number (easy)

Connect Ropes (easy)

✓ Mark as Completed

[Report an Issue](#) [Ask a Question](#)