

Grokking the Coding Interview: Patterns for **Coding Questions** C 66% completed Q Search Course Pattern: Modified Binary Search Introduction Order-agnostic Binary Search (easy) Ceiling of a Number (medium) Next Letter (medium) Number Range (medium) Search in a Sorted Infinite Array (medium) Minimum Difference Element Bitonic Array Maximum (easy) Problem Challenge 1 Solution Review: Problem Challenge 1 Problem Challenge 2 Solution Review: Problem Challenge 2 Problem Challenge 3 Solution Review: Problem Challenge 3 Pattern: Bitwise XOR Introduction Single Number (easy) Two Single Numbers (medium) Complement of Base 10 Number (medium) Problem Challenge 1 Solution Review: Problem Challenge 1 Pattern: Top 'K' Elements Introduction Top 'K' Numbers (easy) Kth Smallest Number (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

Problem Challenge 2

Problem Challenge 3

Solution Review: Problem

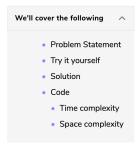
Pattern: K-way merge

Solution Review: Problem Challenge 2

Challenge 1

Challenge 3

Number Range (medium)



Problem Statement

Given an array of numbers sorted in ascending order, find the range of a given number 'key'. The range of the 'key' will be the first and last position of the 'key' in the array.

Write a function to return the range of the 'key'. If the 'key' is not present return [-1,-1].

Example 1:

```
Input: [4, 6, 6, 6, 9], key = 6
Output: [1, 3]
```

Example 2:

```
Input: [1, 3, 8, 10, 15], key = 10
Output: [3, 3]
```

Example 3:

```
Input: [1, 3, 8, 10, 15], key = 12
Output: [-1, -1]
```

Try it yourself

Try solving this question here:



Solution

The problem follows the **Binary Search** pattern. Since Binary Search helps us find a number in a sorted array efficiently, we can use a modified version of the Binary Search to find the first and the last position of a number.

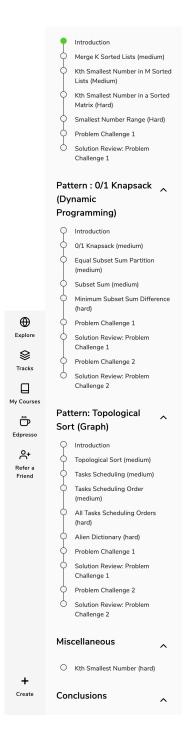
We can use a similar approach as discussed in Order-agnostic Binary Search. We will try to search for the 'key' in the given array; if the 'key' is found (i.e. key == arr[middle) we have two options:

- When trying to find the first position of the 'key', we can update end = middle 1 to see if the key is present before middle.
- When trying to find the last position of the 'key', we can update start = middle + 1 to see if the key is present after middle.

In both cases, we will keep track of the last position where we found the 'key'. These positions will be the required range.

Code

Here is what our algorithm will look like:



```
result[1] = binary_search(arr, key, true);
     function binary_search(arr, key, findMaxIndex) {
       let keyIndex = −1;
       let start = 0;
       let end = arr.length - 1;
       while (start <= end) {
    mid = Math.floor(start + (end - start) / 2);</pre>
         if (key < arr[mid]) {</pre>
         } else if (key > arr[mid]) {
  start = mid + 1;
            keyIndex = mid;
            if (findMaxIndex) {
              end = mid - 1; // search behind to find the first index of 'key'
                                                                                               SAVE
                                                                                                                    03
                                                                                                            RESET
                                                                                                                 Close
                                                                                                                 1.801s
Output
 [ 1, 3 ]
 [ 3, 3 ]
 [ -1, -1 ]
```

Time complexity

Since, we are reducing the search range by half at every step, this means that the time complexity of our algorithm will be O(logN) where 'N' is the total elements in the given array.

Space complexity

The algorithm runs in constant space O(1).

Interviewing soon? We've partnered with Hired so that companies apply to you instead of you applying to them. See how \odot

