

Grokking the Coding Interview: Patterns for Coding Questions

45% completed



Search Course

- Solution Review: Problem Challenge 2
- Problem Challenge 3
- Solution Review: Problem Challenge 3

Pattern: In-place

Reversal of a

LinkedList

- Introduction
- Reverse a LinkedList (easy)
- Reverse a Sub-list (medium)
- Reverse every K-element Sub-list (medium)
- Problem Challenge 1
- Solution Review: Problem Challenge 1
- Problem Challenge 2
- Solution Review: Problem Challenge 2

Pattern: Tree Breadth

First Search

- Introduction
- Binary Tree Level Order Traversal (easy)
- Reverse Level Order Traversal (easy)
- Zigzag Traversal (medium)
- Level Averages in a Binary Tree (easy)
- Minimum Depth of a Binary Tree (easy)
- Level Order Successor (easy)
- Connect Level Order Siblings (medium)
- Problem Challenge 1
- Solution Review: Problem Challenge 1
- Problem Challenge 2
- Solution Review: Problem Challenge 2

Pattern: Tree Depth

First Search

- Introduction
- Binary Tree Path Sum (easy)
- All Paths for a Sum (medium)
- Sum of Path Numbers (medium)
- Path With Given Sequence (medium)
- Count Paths for a Sum (medium)
- Problem Challenge 1
- Solution Review: Problem Challenge 1
- Problem Challenge 2
- Solution Review: Problem Challenge 2

Pattern: Two Heaps

- Introduction
- Find the Median of a Number Stream (medium)

Ceiling of a Number (medium)

We'll cover the following

- Problem Statement
- Try it yourself
- Solution
- Code
 - Time complexity
 - Space complexity
- Similar Problems
- Problem 1
- Code

Problem Statement

Given an array of numbers sorted in an ascending order, find the ceiling of a given number 'key'. The ceiling of the 'key' will be the smallest element in the given array greater than or equal to the 'key'.

Write a function to return the index of the ceiling of the 'key'. If there isn't any ceiling return -1.

Example 1:

```
Input: [4, 6, 10], key = 6
Output: 1
Explanation: The smallest number greater than or equal to '6' is '6' having index '1'.
```

Example 2:

```
Input: [1, 3, 8, 10, 15], key = 12
Output: 4
Explanation: The smallest number greater than or equal to '12' is '15' having index '4'.
```

Example 3:

```
Input: [4, 6, 10], key = 17
Output: -1
Explanation: There is no number greater than or equal to '17' in the given array.
```

Example 4:

```
Input: [4, 6, 10], key = -1
Output: 0
Explanation: The smallest number greater than or equal to '-1' is '4' having index '0'.
```

Try it yourself

Try solving this question here:

 Java
 Python
 JS
 C++

```

1 class CeilingOfANumber {
2
3     public static int searchCeilingOfANumber(int[] arr, int key) {
4         // TODO: Write your code here
5         return -1;
6     }
7
8     public static void main(String[] args) {
9         System.out.println(CeilingOfANumber.searchCeilingOfANumber(new int[] { 4, 6, 10 }, 6));
10        System.out.println(CeilingOfANumber.searchCeilingOfANumber(new int[] { 1, 3, 8, 10, 15 }, 12));
11        System.out.println(CeilingOfANumber.searchCeilingOfANumber(new int[] { 4, 6, 10 }, 17));
12        System.out.println(CeilingOfANumber.searchCeilingOfANumber(new int[] { 4, 6, 10 }, -1));
13    }
14 }
```

RUN
SAVE
RESET

Solution

This 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 ceiling 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 we find the 'key', we return its index as the ceiling. If we can't find the 'key', the next big number will be pointed out by the index `start`. Consider Example-2 mentioned above:

- Sliding Window Median (hard)
- Maximize Capital (hard)
- Problem Challenge 1
- Solution Review: Problem Challenge 1

Pattern: Subsets

- Introduction
- Subsets (easy)
- Subsets With Duplicates (easy)
- Permutations (medium)
- String Permutations by changing case (medium)
- Balanced Parentheses (hard)
- Unique Generalized Abbreviations (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

Pattern: Modified Binary Search

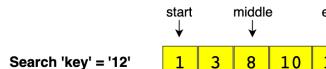
- Introduction
- Order-agnostic Binary Search (easy)
- Ceiling of a Number (medium)
- Next Letter (medium)
- Number Range (medium)
- Search in a Sorted Infinite Array (medium)
- Minimum Difference Element (medium)
- Bitonic Array Maximum (easy)
- Problem Challenge 1
- Solution Review: Problem Challenge 1
- Problem Challenge 2
- Solution Review: Problem Challenge 2
- Problem Challenge 3
- Solution Review: Problem Challenge 3

Pattern: Bitwise XOR

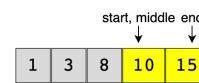
- Introduction
- Single Number (easy)
- Two Single Numbers (medium)
- Complement of Base 10 Number (medium)
- Problem Challenge 1
- Solution Review: Problem Challenge 1

Pattern: Top 'K' Elements

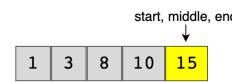
- Introduction
- Top 'K' Numbers (easy)
- Kth Smallest Number (easy)
- 'K' Closest Points to the Origin (easy)
- Connect Ropes (easy)
- Top 'K' Frequent Numbers (medium)
- Frequency Sort (medium)
- Kth Largest Number in a Stream (medium)
- 'K' Closest Numbers (medium)



As key > arr[middle], therefore start = middle + 1



As key > arr[middle], therefore start = middle + 1



As key < arr[middle], therefore end = middle - 1, and the loop will break as end has become less than start

Since we are always adjusting our range to find the 'key', when we exit the loop, the start of our range will point to the smallest number greater than the 'key' as shown in the above picture.

We can add a check in the beginning to see if the 'key' is bigger than the biggest number in the input array. If so, we can return '-1'.

Code

Here is what our algorithm will look like:

Java
Python
C++
JS

```

1 class CeilingOfANumber {
2
3     public static int searchCeilingOfANumber(int[] arr, int key) {
4         if (key > arr[arr.length - 1]) // if the 'key' is bigger than the biggest element
5             return -1;
6
7         int start = 0, end = arr.length - 1;
8         while (start <= end) {
9             int mid = start + (end - start) / 2;
10            if (key < arr[mid]) {
11                end = mid - 1;
12            } else if (key > arr[mid]) {
13                start = mid + 1;
14            } else { // found the key
15                return mid;
16            }
17        }
18        // since the loop is running until 'start <= end', so at the end of the while loop, 'start == end+1'
19        // we are not able to find the element in the given array, so the next big number will be arr[start]
20        return start;
21    }
22
23    public static void main(String[] args) {
24        System.out.println(CeilingOfANumber.searchCeilingOfANumber(new int[] { 4, 6, 10 }, 6));
25        System.out.println(CeilingOfANumber.searchCeilingOfANumber(new int[] { 1, 3, 8, 10, 15 }, 12));
26        System.out.println(CeilingOfANumber.searchCeilingOfANumber(new int[] { 4, 6, 10 }, 17));
27        System.out.println(CeilingOfANumber.searchCeilingOfANumber(new int[] { 4, 6, 10 }, -1));
28    }
}

```

RUN
SAVE
RESET

Close

Output
1.484s

```
1
4
-1
0
```

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 $O(\log N)$ where 'N' is the total elements in the given array.

Space complexity

The algorithm runs in constant space $O(1)$.

Similar Problems

Problem 1

Given an array of numbers sorted in ascending order, find the floor of a given number 'key'. The floor of the 'key' will be the biggest element in the given array smaller than or equal to the 'key'

Write a function to return the index of the floor of the 'key'. If there isn't a floor, return -1.

Example 1:

```
Input: [4, 6, 10], key = 6
Output: 1
```

- 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

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



Tracks

My Courses



Refer a Friend

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



[Mark Course as Completed](#)

Explanation: The biggest number smaller than or equal to '6' is '6' having index '1'.

Example 2:

Input: [1, 3, 8, 10, 15], key = 12
Output: 3
Explanation: The biggest number smaller than or equal to '12' is '10' having index '3'.

Example 3:

Input: [4, 6, 10], key = 17
Output: 2
Explanation: The biggest number smaller than or equal to '17' is '10' having index '2'.

Example 4:

Input: [4, 6, 10], key = -1
Output: -1
Explanation: There is no number smaller than or equal to '-1' in the given array.

Code #

The code is quite similar to the above solution; only the highlighted lines have changed:

Java
Python3
C++
JS JS

```

3  public static int searchFloorOfANumber(int[] arr, int key) {
4      if (key < arr[0]) // if the 'key' is smaller than the smallest element
5          return -1;
6
7      int start = 0, end = arr.length - 1;
8      while (start <= end) {
9          int mid = start + (end - start) / 2;
10         if (key < arr[mid]) {
11             end = mid - 1;
12         } else if (key > arr[mid]) {
13             start = mid + 1;
14         } else { // found the key
15             return mid;
16         }
17     }
18     // since the loop is running until 'start <= end', so at the end of the while loop, 'start == end+1'
19     // we are not able to find the element in the given array, so the next smaller number will be arr[end]
20     return end;
21 }
22
23 public static void main(String[] args) {
24     System.out.println(FloorOfANumber.searchFloorOfANumber(new int[] { 4, 6, 10 }, 6));
25     System.out.println(FloorOfANumber.searchFloorOfANumber(new int[] { 1, 3, 8, 10, 15 }, 12));
26     System.out.println(FloorOfANumber.searchFloorOfANumber(new int[] { 4, 6, 10 }, 17));
27     System.out.println(FloorOfANumber.searchFloorOfANumber(new int[] { 4, 6, 10 }, -1));
28 }
29 }
```

RUN
SAVE
RESET
Close

Output 1.490s

```
1
3
2
-1
```

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

[Back](#)

[Next](#)

Order-agnostic Binary Search (easy)

Next Letter (medium)

[Mark as Completed](#)

Report an Issue Ask a Question