

Grokking the Coding Interview: Patterns for Coding Questions

67% completed



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)
- 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

Minimum Difference Element (medium)

We'll cover the following

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

Problem Statement

Given an array of numbers sorted in ascending order, find the element in the array that has the minimum difference with the given 'key'.

Example 1:

```
Input: [4, 6, 10], key = 7
Output: 6
Explanation: The difference between the key '7' and '6' is minimum than any other number in the array
```

Example 2:

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

Example 3:

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

Example 4:

```
Input: [4, 6, 10], key = 17
Output: 10
```

Try it yourself

Try solving this question here:

JavaPython3JS C++

```
1 const search_min_diff_element = function(arr, key) {
2   // TODO: Write your code here
3   return -1;
4 };
5
6
7 console.log(search_min_diff_element([4, 6, 10], 7))
8 console.log(search_min_diff_element([4, 6, 10], 4))
9 console.log(search_min_diff_element([1, 3, 8, 10, 15], 12))
10 console.log(search_min_diff_element([4, 6, 10], 17))
11
```

RUNSAVERESET

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 number that has the minimum difference with the given 'key'.

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 will return it as the minimum difference number. If we can't find the 'key', (at the end of the loop) we can find the differences between the 'key' and the numbers pointed out by indices **start** and **end**, as these two numbers will be closest to the 'key'. The number that gives minimum difference will be our required number.

Code

Here is what our algorithm will look like:

JavaPython3C++JS

```
1 function search_min_diff_element(arr, key) {
2   if (key < arr[0]) {
3     return arr[0];
4   }
5 }
```

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

Explore

Tracks

My Courses

Edpresso

Refer a Friend

Create

```
4 }
5 const n = arr.length;
6 if (key > arr[n - 1]) {
7   return arr[n - 1];
8 }
9
10 let start = 0;
11 let end = n - 1;
12 while (start <= end) {
13   mid = Math.floor(start + (end - start) / 2);
14   if (key < arr[mid]) {
15     end = mid - 1;
16   } else if (key > arr[mid]) {
17     start = mid + 1;
18   } else {
19     return arr[mid];
20   }
21 }
22
23 // at the end of the while loop, 'start === end+1'
24 // we are not able to find the element in the given array
25 // return the element which is closest to the 'key'
26 if ((arr[start] - key) < (key - arr[end])) {
27   return arr[start];
28 }
```

RUN

SAVE

RESET

Close

2.000s

Output

6
4
10
10

Time complexity

Since, we are reducing the search range by half at every step, this means 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 $O(1)$.

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

← Back

Next →

Search in a Sorted Infinite Array (medi...

Bitonic Array Maximum (easy)

Mark as Completed

Report an Issue

Ask a Question