

Grokking the Coding Interview: Patterns for Coding Questions

69% 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

Solution Review: Problem Challenge 1

We'll cover the following

- Search Bitonic Array (medium)
- Solution
- Code
 - Time complexity
 - Space complexity

Search Bitonic Array (medium)

Given a Bitonic array, find if a given 'key' is present in it. An array is considered bitonic if it is monotonically increasing and then monotonically decreasing. Monotonically increasing or decreasing means that for any index `i` in the array `arr[i] != arr[i+1]`.

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

Example 1:

```
Input: [1, 3, 8, 4, 3], key=4
Output: 3
```

Example 2:

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

Example 3:

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

Example 4:

```
Input: [10, 9, 8], key=10
Output: 0
```

Solution

The problem follows the **Binary Search** pattern. Since Binary Search helps us efficiently find a number in a sorted array we can use a modified version of the Binary Search to find the 'key' in the bitonic array.

Here is how we can search in a bitonic array:

- First, we can find the index of the maximum value of the bitonic array, similar to [Bitonic Array Maximum](#). Let's call the index of the maximum number `maxIndex`.
- Now, we can break the array into two sub-arrays:
 - Array from index '0' to `maxIndex`, sorted in ascending order.
 - Array from index `maxIndex+1` to `array_length-1`, sorted in descending order.
- We can then call **Binary Search** separately in these two arrays to search the 'key'. We can use the same [Order-agnostic Binary Search](#) for searching.

Code

Here is what our algorithm will look like:

```
Java Python3 C++ JS

1 function search_bitonic_array(arr, key) {
2   const maxIndex = find_max(arr);
3   const keyIndex = binary_search(arr, key, 0, maxIndex);
4   if (keyIndex !== -1) {
5     return keyIndex;
6   }
7   return binary_search(arr, key, maxIndex + 1, arr.length - 1);
8 }
9
10 // find index of the maximum value in a bitonic array
11 function find_max(arr) {
12   let start = 0,
13   end = arr.length - 1;
14   while (start < end) {
15     const mid = Math.floor(start + (end - start) / 2);
16     if (arr[mid] > arr[mid + 1]) {
17       end = mid;
18     } else {
19       start = mid + 1;
20     }
21   }
22   // at the end of the while loop, 'start === end'
```

Explore

Tracks

My Courses

Edpresso

Refer a Friend

Create

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)

23 return start;

24 }

25

26

27 // order-agnostic binary search

28 function binary_search(arr, key, start, end) {

RUN

SAVE

RESET

Close

Output

2.343s

3

1

3

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 $O(1)$.

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

← Back

Next →

Problem Challenge 1

Problem Challenge 2

✓ Mark as Completed

Report an Issue

Ask a Question