Grokking the Coding Interview: Patterns for Coding Questions

68% completed

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 (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

Bitonic Array Maximum (easy)

We'll cover the following

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

Problem Statement

Find the maximum value in a given Bitonic array. 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]`.

Example 1:

```
Input: [1, 3, 8, 12, 4, 2]
Output: 12
Explanation: The maximum number in the input bitonic array is '12'.
```

Example 2:

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

Example 3:

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

Example 4:

```
Input: [10, 9, 8]
Output: 10
```

Try it yourself

Try solving this question here:

Java | Python3 | JS | C++

```
const find_max_in_bitonic_array = function(arr) {
  // TODO: Write your code here
  return -1;
};


console.log(find_max_in_bitonic_array([1, 3, 8, 12, 4, 2]))
console.log(find_max_in_bitonic_array([3, 8, 3, 1]))
console.log(find_max_in_bitonic_array([1, 3, 8, 12]))
console.log(find_max_in_bitonic_array([10, 9, 8]))
```

RUN SAVE RESET

Solution

A bitonic array is a sorted array; the only difference is that its first part is sorted in ascending order and the second part is sorted in descending order. We can use a similar approach as discussed in Order-agnostic Binary Search. Since no two consecutive numbers are same (as the array is monotonically increasing or decreasing), whenever we calculate the `middle`, we can compare the numbers pointed out by the index `middle` and `middle+1` to find if we are in the ascending or the descending part. So:

1. If `arr[middle] > arr[middle + 1]`, we are in the second (descending) part of the bitonic array. Therefore, our required number could either be pointed out by `middle` or will be before `middle`. This means we will be doing: `end = middle`.
2. If `arr[middle] < arr[middle + 1]`, we are in the first (ascending) part of the bitonic array. Therefore, the required number will be after `middle`. This means we will be doing: `start = middle + 1`.

We can break when `start == end`. Due to the two points mentioned above, both `start` and `end` will be pointing at the maximum number of the bitonic array.

Code

Here is what our algorithm will look like:

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

Mark Course as Completed

Explore
Tracks
My Courses
Edpresso
Refer a Friend
Create

Here is what our algorithm will look like:

Java | Python3 | C++ | JS

```javascript
function find_max_in_bitonic_array(arr) {
  let start = 0;
  let end = arr.length - 1;
  while (start < end) {
    mid = Math.floor(start + (end - start) / 2);
    if (arr[mid] > arr[mid + 1]) {
      end = mid;
    } else {
      start = mid + 1;
    }
  }

  // at the end of the while loop, 'start === end'
  return arr[start];
}


console.log(find_max_in_bitonic_array([1, 3, 8, 12, 4, 2]));
console.log(find_max_in_bitonic_array([3, 8, 3, 1]));
console.log(find_max_in_bitonic_array([1, 3, 8, 12]));
console.log(find_max_in_bitonic_array([10, 9, 8]));
```

RUN SAVE RESET

Close

Output 2.062s

```
12
8
12
10
```

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 $O(1)$.