

Grokking the Coding Interview: Patterns for Coding Questions

49% completed



Pattern: Cyclic Sort

- Introduction
- Cyclic Sort (easy)
- Find the Missing Number (easy)
- Find all Missing Numbers (easy)
- Find the Duplicate Number (easy)
- Find all Duplicate Numbers (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: 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)

Find all Duplicate Numbers (easy)

We'll cover the following

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

Problem Statement

We are given an unsorted array containing 'n' numbers taken from the range 1 to 'n'. The array has some duplicates, **find all the duplicate numbers without using any extra space**.

Example 1:

Input: [3, 4, 4, 5, 5]
Output: [4, 5]

Example 2:

Input: [5, 4, 7, 2, 3, 5, 3]
Output: [3, 5]

Try it yourself

Try solving this question here:

JavaPython3JS C++

```
1 using namespace std;
2
3 #include <iostream>
4 #include <vector>
5
6 class FindAllDuplicate {
7 public:
8     static vector<int> findNumbers(vector<int> &nums) {
9         vector<int> duplicateNumbers;
10        // TODO: Write your code here
11        return duplicateNumbers;
12    }
13 };
14
```

TESTSAVERESET

Solution

This problem follows the **Cyclic Sort** pattern and shares similarities with [Find the Duplicate Number](#). Following a similar approach, we will place each number at its correct index. After that, we will iterate through the array to find all numbers that are not at the correct indices. All these numbers are duplicates.

Code

Here is what our algorithm will look like:

JavaPython3C++JS

```
1 using namespace std;
2
3 #include <iostream>
4 #include <vector>
5
6 class FindAllDuplicate {
7 public:
8     static vector<int> findNumbers(vector<int> &nums) {
9         int i = 0;
10        while (i < nums.size()) {
11            if (nums[i] != nums[nums[i] - 1]) {
12                swap(nums, i, nums[i] - 1);
13            } else {
14                i++;
15            }
16        }
17
18        vector<int> duplicateNumbers;
19        for (i = 0; i < nums.size(); i++) {
20            if (nums[i] != i + 1) {
21                duplicateNumbers.push_back(nums[i]);
22            }
23        }
24    }
25 }
```

Explore

Tracks

My Courses

Edpresso

Refer a Friend

Create

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)

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

24

25

26

27

28

private:

return duplicateNumbers;

}

RUN

SAVE

RESET

Close

Output

0.833s

Duplicates are: 5 4

Duplicates are: 3 5

Time complexity

The time complexity of the above algorithm is $O(n)$.

Space complexity

Ignoring the space required for storing the duplicates, 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

Find the Duplicate Number (easy)

Problem Challenge 1

Mark as Completed

Report an Issue

Ask a Question