Top LeetCode Patterns for FAANG Coding Interviews

Preparing for coding interviews can be made easier by focusing on coding patterns.

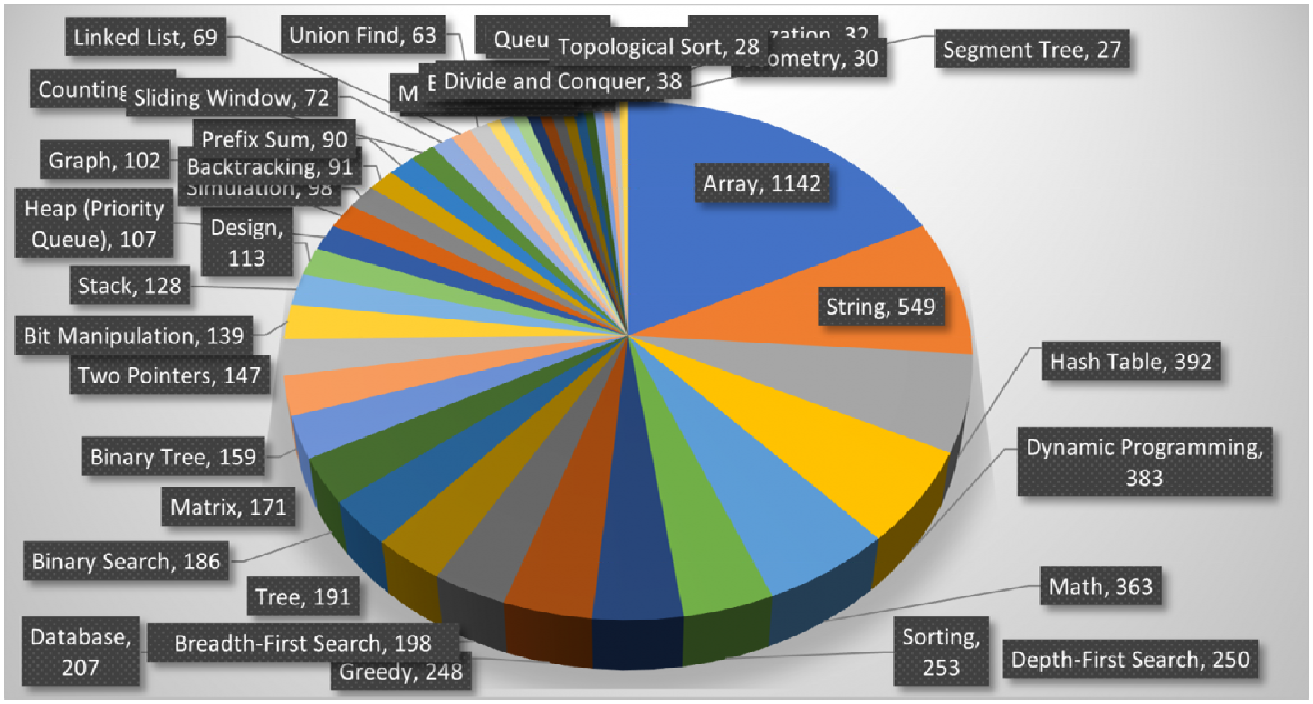
Every software engineer should learn coding patterns such as **Sliding Window**, **Two Pointers**, **Two Heaps**, etc. By doing so, software engineers will be able to develop the skill of "mapping a new problem to an existing one." In this post, we will learn which coding patterns have the highest return on investment for software engineers.

Grokking the Coding Interview came up with a list of 18 patterns for coding questions based on the similarities in the techniques needed to solve them. The course's idea is to teach famous coding patterns so that once someone is familiar with a pattern, they will be able to solve dozens of problems with it.

LeetCode Problems Distribution

LeetCode (LC), being the largest repository of coding problems, contains more than 2k+ questions. Each question on LC can be tagged with one or more topics. These topics are either data structures like Array, HashTable, Tree, etc., or algorithmic techniques like Greedy, Divide and Conquer, Sorting, etc., or coding patterns like Sliding Window, Depth First Search, Topological Sort, etc.

Here is the topic distribution for LC questions:



The top topic is Array with 1142 problems, followed by String with 549 problems, and so on. Let's take a closer look at each category of **topics**, namely **Data Structures**, **Algorithms**, and **Coding Patterns**.

Top Data Structures with Best ROI

Here are the top Data Structures with the highest return on investment:

1. Array (1142 problems)
2. String (549)
3. Hash Table (392)
4. Tree (191)
5. Matrix (171)
6. Stack (128)
7. Heap or Priority Queue (107)
8. Graph (102)
9. Linked List (69)
10. Trie (44)

Top Algorithmic Techniques with Best ROI

Here are the top algorithmic techniques with the highest return on investment:

1. Dynamic Programming (383)

2. Sorting (253)
3. Greedy (248)
4. Binary Search (186)
5. Backtracking (91)
6. Recursion (44)
7. Divide and Conquer (38)

Top Coding Patterns with Best ROI

Here are the top coding patterns with the highest return on investment:

1. Depth First Search (250)
2. Breadth First Search(198)
3. Binary Search (186)
4. Two Pointers (147)
5. Sliding Window (72)
6. Monotonic Stack (44)
7. Union Find (63)
8. Memoization (32)
9. Topological Sort (28)
10. Segment Tree (27)

Best Coding Patterns with Highest ROI

Combining all categories from the above data, here is the list of best coding patterns/techniques with the highest ROI:

1. Two Pointers (Arrays, Strings, Fast & Slow Pointer)
This pattern covers a huge set of questions related to Arrays and Strings, which are the highest tagged data structures. Fast & Slow Pointer can be easily understood as a variation of the Two Pointers pattern.

2. Sliding Window (Arrays, Strings, Hash Tables)
Sliding Window covers most of the problems related to top data structures like Arrays, Strings, and HashTables.

3. Tree and Graph Depth First Search (Matrix Traversal)
Most Trees and Graphs problems can be solved using Depth First Search (DFS). Matrix Traversal, which is also DFS based pattern, covers most of the matrix-related problems.

4. **Tree and Graph Breadth First Search** (Queue, Subsets, Matrix Traversal, Topological Sort)
   Breadth First Search (BFS) is a very handy pattern. BFS's patterns like Subsets, Matrix Traversal, and Topological Sort cover a good number of problems.

5. **Binary Search** (Arrays)
   Binary Search and its variants are used to solve a huge number of coding questions.

6. **Interval Merge**
   Although there are not many problems related to Interval Merge, these problems frequently appear in coding interviews.

7. **Recursion/Backtracking**
   Backtracking and recursion are used to solve a wide range of problems. Mastering these techniques is highly recommended.

Conclusion

Most programming interviews include LeetCode-type questions. Software engineers practice such coding problems before interviews. The highest return on investment is achieved by preparing smartly and focusing on the problem patterns. You can learn more about these patterns and related problems in Grokking the Coding Interview and Grokking Dynamic Programming for Coding Interviews.

Read more on Coding Interviews:

- System Design Interview Survival Guide (2023)
- The Complete Guide to Ace the System Design Interview
- System Design Interviews: What distinguishes you from others?
- Grokking LeetCode: A Smarter Way to Prepare for Coding Interviews
- 14 Most Popular Amazon Coding Interview Questions

https://www.youtube.com/watch?v=OgX75yqQvC0

LeetCode101: 20 Coding Patterns to Master MAANG Interviews

Coding patterns enhance our “ability to map a new problem to an already known problem.”

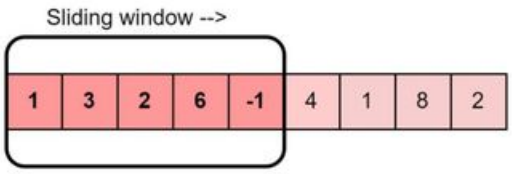
Here are the 20 coding patterns:

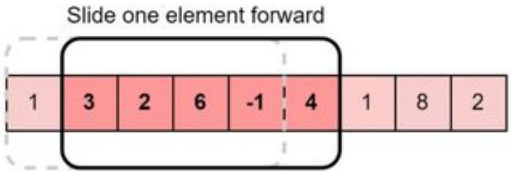
1. Sliding Window
2. Islands (Matrix Traversal)
3. Two Pointers
4. Fast & Slow Pointers
5. Merge Intervals
6. Cyclic Sort
7. In-place Reversal of a LinkedList
8. Tree Breadth-First Search
9. Tree Depth First Search
10. Two Heaps
11. Subsets
12. Modified Binary Search
13. Bitwise XOR
14. Top ‘K’ Elements
15. K-way Merge
16. Topological Sort
17. Unbounded Knapsack
18. Fibonacci Numbers
19. Palindromic Subsequence
20. Longest Common Substring

1. Sliding Window

Usage: This algorithmic technique is used when we need to handle the input data in a specific window size.

DS Involved: Array, String, HashTable





Sample Problems:

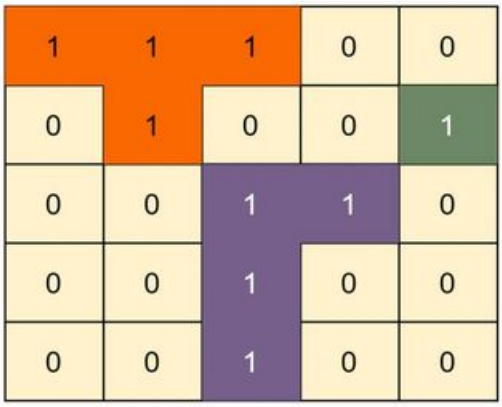
- Longest Substring with 'K' Distinct Characters
- Fruits into Baskets

DesignGurus.org

2. Islands (Matrix Traversal)

Usage: This pattern describes all the efficient ways of traversing a matrix (or 2D array).

DS Involved: Matrix, Queue



Sample Problems:

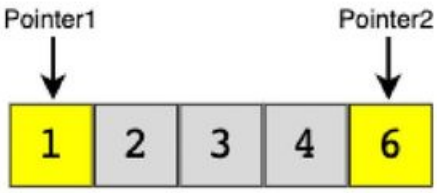
- Number of Islands
- Flood Fill
- Cycle in a Matrix

DesignGurus.org

3. Two Pointers

Usage: This technique uses two pointers to iterate input data. Generally, both pointers move in the opposite direction at a constant interval.

DS Involved: Array, String, LinkedList



Sample Problems:

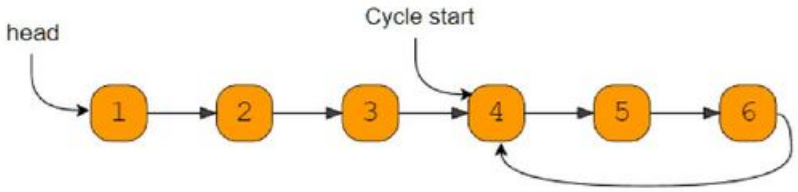
- Squaring a Sorted Array
- Dutch National Flag Problem
- Minimum Window Sort

DesignGurus.org

4. Fast & Slow Pointers

Usage: Also known as Hare & Tortoise algorithm. This technique uses two pointers that traverse the input data at different speeds.

DS Involved: Array, String, LinkedList



Sample Problems:

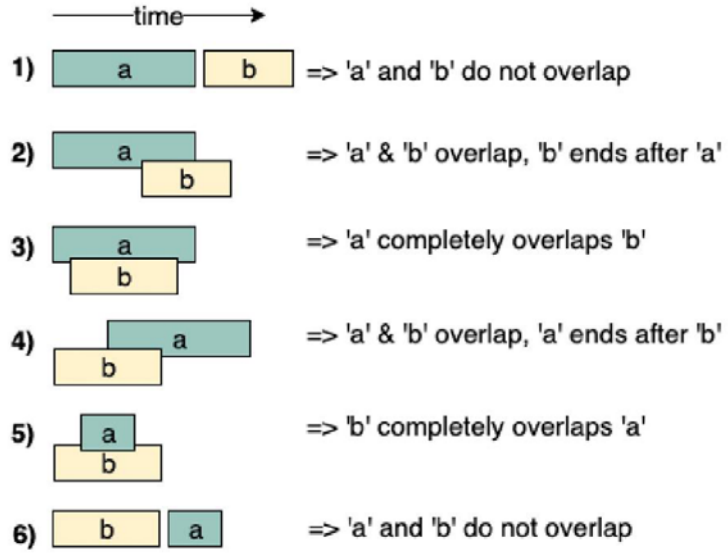
- Middle of the LinkedList
- Happy Number
- Cycle in a Circular Array

DesignGurus.org

5. Merge Intervals

Usage: This technique is used to deal with overlapping intervals.

DS Involved: Array, Heap



Sample Problems:

- Conflicting Appointments
- Minimum Meeting Rooms

DesignGurus.org

6. Cyclic Sort

Usage: Use this technique to solve array problems where the input data lies within a fixed range.

DS Involved: Array

Sample Problems:

- Find all Missing Numbers
- Find all Duplicate Numbers
- Find the First K Missing Positive Numbers

DesignGurus.org

7. In-place Reversal of a LinkedList

Usage: This technique describes an efficient way to reverse the links between a set of nodes of a LinkedList. Often, the constraint is that we need to do this in-place, i.e., using the existing node objects and without using extra memory.

DS Involved: LinkedList

Sample Problems:

- Reverse every K-element Sub-list
- Rotate a LinkedList

DesignGurus.org

8. Breadth-First Search

Usage: This technique is used to solve problems involving traversing trees or graphs in a breadth-first search manner.

DS Involved: Tree, Graph, Matrix, Queue

Sample Problems:

- Binary Tree Level Order Traversal
- Minimum Depth of a Binary Tree
- Connect Level Order Siblings

9. Depth First Search

Usage: This technique is used to solve problems involving traversing trees or graphs in a depth-first search manner.

DS Involved: Tree, Graph, Matrix

Sample Problems:

- Path With Given Sequence
- Count Paths for a Sum

10. Two Heaps

Usage: In many problems, we are given a set of elements that can be divided into two parts. We are interested in knowing the smallest element in one part and the biggest element in the other part. As the name suggests, this technique uses a Min-Heap to find the smallest element and a Max-Heap to find the biggest element.

DS Involved: Heap, Array

Sample Problems:

- Find the Median of a Number Stream
- Next Interval

11. Subsets

Usage: Use this technique when the problem asks to deal with permutations or combinations of a set of elements.

DS Involved: Queue, Array, String

Sample Problems:

- String Permutations by changing case
- Unique Generalized Abbreviations

DesignGurus.org

12. Modified Binary Search

Usage: Use this technique to search a sorted set of elements efficiently.

DS Involved: Array

Sample Problems:

- Ceiling of a Number
- Bitonic Array Maximum

DesignGurus.org

13. Bitwise XOR

Usage: This technique uses the XOR operator to manipulate bits to solve problems.

DS Involved: Array, Bits

Sample Problems:

- Two Single Numbers
- Flip and Invert an Image

14. Top ‘K’ Elements

Usage: This technique is used to find top/smallest/frequently occurring ‘K’ elements in a set.

DS Involved: Array, Heap, Queue

Sample Problems:

- ‘K’ Closest Points to the Origin
- Maximum Distinct Elements

DesignGurus.org

15. K-way Merge

Usage: This technique helps us solve problems that involve a list of sorted arrays.

DS Involved: Array, Queue, Heap

Sample Problems:

- Kth Smallest Number in M Sorted Lists
- Kth Smallest Number in a Sorted Matrix

DesignGurus.org

16. Topological Sort

Usage: Use this technique to find a linear ordering of elements that have dependencies on each other.

DS Involved: Array, HashTable, Queue, Graph

Sample Problems:

- Tasks Scheduling
- Alien Dictionary

DesignGurus.org

17. 0/1 Knapsack

Usage: This technique is used to solve optimization problems. Use this technique to select elements that give maximum profit from a given set with a limitation on capacity and that each element can only be picked once.

DS Involved: Array, HashTable

Sample Problems:

- Equal Subset Sum Partition
- Minimum Subset Sum Difference

18. Fibonacci Numbers

Usage: Use this technique to solve problems that follow the Fibonacci numbers sequence, i.e., every subsequent number is calculated from the last few numbers.

DS Involved: Array, HashTable

Sample Problems:

- Staircase
- House Thief

DesignGurus.org

19. Palindromic Subsequence

Usage: This technique is used to solve optimization problems related to palindromic sequences or strings.

DS Involved: Array, HashTable

Sample Problems:

- Longest Palindromic Subsequence
- Minimum Deletions in a String to make it a Palindrome

20. Longest Common Substring

Usage: Use this technique to find the optimal part of a string/sequence or set of strings/sequences.

DS Involved: Array, HashTable

Sample Problems:

- Maximum Sum Increasing Subsequence
- Edit Distance