The-CODE-Plus-Plus-Community /
**Competitive-Programming-and-DSA**

Code Issues Pull requests Actions Projects Security Insigh

**Competitive-Programming-and-DSA / ROADMAP.md**

**Subhampreet** 2 years ago

239 lines (218 loc) · 7 KB

**Preview** Code Blame

1. Introduction to Git
2. Introduction to Programming

- Types of languages
- Flowcharts & Pseudocode
- Flow of the program
- Time & Space Complexity

3. Basics of Java

- Array
  - Introduction
  - Memory management
  - Input and Output

  - ArrayList Introduction
  - Sorting
  - Insertion Sort
  - Selection Sort
  - Bubble Sort
  - Count Sort
  - Radix Sort
  - Searching
  - Linear Search
  - Binary Search
  - Modified Binary Search
  - Two Pointer
  - Subarray Questions

- Strings

  - Introduction
  - How Strings work
  - Comparison of methods
  - Operations in Strings
  - StringBuilder in java

- Maths for DSA

  - Introduction
  - Complete Bitwise Operators
  - Prime numbers
  - HCF / LCM
  - Sieve of Eratosthenes
  - Newton's Square Root Method
  - Number Theory
  - Euclidean algorithm
  - Advanced Concepts for CP (later in the course)
  - Bitwise + DP
  - Extended Euclidean algorithm
  - Modulo Properties
  - Modulo Multiplicative Inverse
  - Linear Diophantine Equations
  - Fermat’s Theorem

  - Wilson's Theorem
  - Lucas Theorem
  - Chinese Remainder Theorem

- Functions

  - Introduction
  - Solving the above math problems in code
  - Scoping in Java
  - Shadowing
  - Variable Length Arguments

- Recursion

  - Introduction
  - Why recursion?
  - Flow of recursive programs - stacks
  - Convert recursion to iteration
  - Tree building of function calls
  - Tail recursion

- Sorting:

  - Merge Sort
  - Quick Sort
  - Cyclic Sort

- Backtracking

  - Sudoku Solver
  - N-Queens
  - N-Knights
  - Maze problems
  - Recursion String Problems
  - Recursion Array Problems
  - Recursion Pattern Problems
  - Subset Questions

- Space and Time Complexity Analysis

  - Introduction
  - Comparisons of various cases
  - Solving Linear Recurrence Relations

  - Solving Divide and Conquer Recurrence Relations
  - Big-O, Big-Omega, Big-Theta Notations
  - Get equation of any relation easily - best and easiest approach
  - Complexity discussion of all the problems we do
  - Space Complexity
  - Memory Allocation of various languages
  - NP-Completeness and Hardness

- Object Oriented Programming
  - Introduction
  - Classes & its instances
  - this keyword in Java
  - Properties
  - Inheritance
  - Abstraction
  - Polymorphism
  - Encapsulation
  - Overloading & Overriding
  - Static & Non-Static
  - Access Control
  - Interfaces
  - Abstract Classes
  - Singleton Class
  - final, finalize, finally
  - Exception Handling

- Stacks & Queues
  - Introduction
  - Interview problems
  - Push efficient
  - Pop efficient
  - Queue using Stack and Vice versa
  - Circular Queue

- Linked List
  - Introduction
  - Fast and slow pointer

    - Cycle Detection
    - Single and Doubly LinkedList
    - Reversal of LinkedList

- Dynamic Programming

    - Introduction
    - Recursion + Recursion DP + Iteration + Iteration Space Optimized
    - Complexity Analysis
    - 0/1 Knapsack
    - Subset Questions
    - Unbounded Knapsack
    - Subsequence questions
    - String DP

- Trees

    - Introduction
    - Binary Trees
    - Binary Search Trees
    - DFS
    - BFS
    - AVL Trees
    - Segment Tree
    - Fenwick Tree / Binary Indexed Tree
    - Square Root Decomposition

- Heaps

    - Introduction
    - Theory
    - Priority Queue
    - Two Heaps Method
    - k-way merge
    - top k elements
    - interval problems

- HashMap

    - Introduction
    - Theory - how it works
    - Comparisons of various forms

  - Limitations and how to solve
  - Map using LinkedList
  - Map using Hash
  - Chaining
  - Probing
  - Huffman-Encoder
  - Tries

- Graphs

  - Introduction
  - BFS
  - DFS
  - Working with graph components
  - Minimum Spanning Trees
  - Kruskal Algorithm
  - Prims Algorithm
  - Dijkstra’s shortest path algorithm
  - Topological Sort
  - Bellman ford
  - A* pathfinding Algorithm

## What basic data structures and algorithms should one learn before starting competitive programming?

1. Basic data sturctures (arrays, queues, linked lists, etc.).
2. Bit manipulation.
3. Advanced data structures:

- Union-Find Disjoint Sets.
- Segment Tree.
- Binary Indexed Tree (a.k.a Fenwik Tree).
- Graph.
- Tree
- Skip Lists.
- Some self balanced Binary Search trees (e.g. Red Black Trees).

4. Brute force and it's tricks and advanced techniques (such as, pruning, bitmasks, meet in the middle, iterative deepining etc.)
5. Binary Search (not only the basic code).

6. Greedy.
7. Dynamic programming and it's tricks and optimisations (Knuth optimisation, convex hull optimisation, bitmasks, etc.).
8. Graph algorithms:

- Traversal (DFS & BFS) algorithms and how to use them.
- Finding Connected Components.
- Flood Fill.
- Topological Sorting (the famous algorithm uses DFS but you should also know Kahn's algorithm that uses BFS as it has much applications).
- Bipartite Check.
- Finding Strongly Connected Components.
- Kruskal's and Prim's algorithms for finding the Minimum Spanning Tree of a graph and the variants of the problem.
- Dijkstra's algorithm for solving the Single Source Shortest Path (SSSP) Problem with out negaitive cycles.
- Bellman-Ford's algorithm for solving the SSSP problem with negative sycles.
- Floyd-Warshall's algorithm for solving the All Pairs Shortest Path (APSP) problem and it's variants.
- Network Flow problem (all it's algorithms, variants and the problems reducable to it). 9 Mathematics:
- You should be familiar with the BigInteger class in Java (maybe write your own if you are in love with C++).
- Some Combinatorics.
- Number Theory (all what you can learn about it).
- Probability Theory.
- Floyd-Cycle detection algorithm.
- Game Theory (especially impartial games and Sprague-Grundy Theorem).

10. Strings:

- Basic Manipulation.
- Z-Algorithm for finding a pattern in a text.
- Knuth-Morris-Pratt Algorithm for finding a pattern in a text.
- Hashing and Rabin-Karp Algorithm for finding a pattern in a text.
- Trie data structure.
- Aho-Corasick Algorithm for finding multiple patterns in a text.
- Suffix Array data structure.
- Suffix Automaton data structure.

11. Computational Geometry Algorithms.

## Resources

- Codeforces Candidate Master RoadMap
- DSA Cracker Sheet
- Striver's CP List
- SDE-Problems
- A2OJ
- Competitive Programming Algorithms