**1. Implementation**

* Start with basic implementation problems to build your coding skills and problem-solving strategies.

**2. Sorting**

* Learn and practice basic sorting algorithms as they are fundamental to many other topics.

**3. Binary Search**

* A key algorithm used in many areas, including arrays and optimization problems.

**4. Greedy**

* Simple but powerful; many problems can be solved with a greedy approach.

**5. Data Structures**

* Focus on basic data structures (arrays, stacks, queues) and then move to more advanced ones.

**6. DFS and Similar**

* Explore depth-first search and related algorithms like BFS, which are essential for graph problems.

**7. Graphs**

* Study graph algorithms, including traversal, shortest path, and other fundamental techniques.

**8. Trees**

* Learn tree-specific algorithms, including traversals, binary search trees, and segment trees.

**9. Two Pointers**

* A useful technique for solving problems on arrays and strings.

**10. Math**

* Cover general mathematical concepts that are common in competitive programming.

**11. Number Theory**

* Dive into GCD, LCM, prime numbers, and other number theory concepts.

**12. Combinatorics**

* Work on problems related to counting, permutations, and combinations.

**13. Probabilities**

* Explore problems involving probability theory.

**14. Constructive Algorithms**

* Practice problems where you have to construct a solution based on given constraints.

**15. Divide and Conquer**

* Learn to break problems into smaller subproblems that can be solved independently.

**16. Dynamic Programming (DP)**

* A crucial topic; start with simple problems and gradually move to more complex ones.

**17. Bitmasks**

* Work on problems involving bit manipulation and bitmask DP.

**18. DSU (Disjoint Set Union)**

* A data structure used in many graph problems, especially those involving connectivity.

**19. Shortest Path**

* Focus on algorithms like Dijkstra, Bellman-Ford, and Floyd-Warshall for finding shortest paths in graphs.

**20. String Algorithms**

* Cover basic string manipulations, pattern matching, and other string-related problems.

**21. Hashing**

* Learn about hash functions and their use in competitive programming.

**22. Ternary Search**

* A specialized form of binary search used in optimization problems.

**23. Expression Parsing**

* Work on parsing and evaluating expressions, which often appear in compilers and interpreters.

**24. Meet-in-the-Middle**

* A powerful technique used to solve problems with large input sizes.

**25. Games**

* Focus on game theory and problems involving strategies.

**26. Trees**

* Delve deeper into advanced tree structures like Fenwick Tree, Segment Tree, etc.

**27. Chinese Remainder Theorem**

* A mathematical theorem used in number theory problems.

**28. 2-SAT**

* A special case of the Boolean satisfiability problem, solvable in polynomial time.

**29. FFT (Fast Fourier Transform)**

* An advanced algorithm used in polynomial multiplication and related problems.

**30. Flows**

* Study maximum flow, minimum cut, and other flow-related problems in graphs.

**31. Graph Matching**

* Explore problems related to matching in bipartite and general graphs.

**32. Interactive**

* Practice problems that require interaction with a grader, often seen in competitions.

**33. String Suffix Structures**

* Work on suffix arrays, suffix trees, and related structures for advanced string processing.

**34. Matrices**

* Focus on matrix manipulations and problems involving linear algebra.

**35. Schedules**

* Solve problems related to scheduling, often involving dynamic programming and greedy approaches.