Beginner Projects

Basic Data Structures

1. **Array Sorting** - Implement bubble, insertion, and selection sorts.
2. **Stack** - Create a stack with basic operations (push, pop).
3. **Queue** - Build a queue with enqueue and dequeue.
4. **Linked List** - Implement a singly linked list with add and delete functions.
5. **Doubly Linked List** - Create a doubly linked list with forward and backward traversal.
6. **Circular Linked List** - Build a circular linked list with basic operations.
7. **Hash Table** - Make a hash table with basic functions like insert and search.
8. **Dynamic Array** - Implement a resizable array.

Basic Algorithm Problems

9. **Fibonacci Sequence** - Generate Fibonacci numbers using recursion and iteration.
10. **Factorial Calculation** - Compute factorials recursively and iteratively.
11. **Prime Checker** - Check if a number is prime.
12. **Palindrome Checker** - Determine if a string or number is a palindrome.
13. **Binary Search** - Find an element in a sorted array.
14. **GCD Calculation** - Compute the greatest common divisor.
15. **Merge Sorted Arrays** - Combine two sorted arrays into one.
16. **Counting Sort** - Implement counting sort for integers.
17. **Radix Sort** - Sort numbers using radix sort.
18. **Linear Search** - Find an element in an unsorted array.

Intermediate Projects

Advanced Data Structures

19. **Binary Search Tree (BST)** - Build a BST with insert and search functions.
20. **AVL Tree** - Implement a self-balancing AVL tree.
21. **Red-Black Tree** - Create a red-black tree with insert and delete functions.
22. **Heap** - Implement a min-heap and max-heap.
23. **Trie** - Build a trie for storing words.
24. **Graph Representation** - Represent graphs using adjacency lists and matrices.
25. **Disjoint Set (Union-Find)** - Manage a set of disjoint sets with union and find operations.
26. **Segment Tree** - Build a segment tree for range queries.
27. **Fenwick Tree** - Implement a Fenwick tree (binary indexed tree).
28. **Suffix Tree** - Create a suffix tree for substring queries.

Intermediate Algorithm Problems

29. **DFS** - Implement depth-first search for graph traversal.

30. **BFS** - Use breadth-first search to find the shortest path.
31. **Dijkstra's Algorithm** - Find the shortest path in a weighted graph.
32. **Kruskal's Algorithm** - Compute the minimum spanning tree.
33. **Prim's Algorithm** - Another method for finding the minimum spanning tree.
34. **Bellman-Ford Algorithm** - Handle shortest paths with negative weights.
35. **Floyd-Warshall Algorithm** - Find shortest paths between all pairs of nodes.
36. **Topological Sort** - Sort nodes in a directed acyclic graph (DAG).
37. **Knapsack Problem** - Solve the knapsack problem with dynamic programming.
38. **Longest Common Subsequence (LCS)** - Find the longest common subsequence of two strings.

Advanced Projects

Complex Data Structures

39. **B-Tree** - Implement a B-tree for balanced searching.
40. **B+ Tree** - Create a B+ tree for range queries.
41. **R-Tree** - Build an R-tree for spatial indexing.
42. **Interval Tree** - Manage intervals with efficient querying.
43. **K-D Tree** - Implement a k-dimensional tree for multidimensional search.
44. **Splay Tree** - Create a self-adjusting splay tree.
45. **Bloom Filter** - Implement a probabilistic data structure for set membership.
46. **Skip List** - Build a skip list for efficient search operations.
47. **Octree** - Create an octree for 3D spatial partitioning.
48. **Van Emde Boas Tree** - Implement a van Emde Boas tree for fast operations.

Advanced Algorithmic Challenges

49. *A Search Algorithm** - Use A* for pathfinding with heuristics.
50. **Traveling Salesman Problem (TSP)** - Solve TSP with dynamic programming and heuristics.
51. **KMP Algorithm** - Implement Knuth-Morris-Pratt for substring matching.
52. **Suffix Array** - Build a suffix array for substring queries.
53. **Sieve of Eratosthenes** - Find all prime numbers up to a given limit.
54. **Maximum Flow** - Implement algorithms for finding the maximum flow in a network.
55. **Minimum Cut** - Find the minimum cut in a flow network.
56. **Hungarian Algorithm** - Solve the assignment problem.
57. **Convex Hull** - Find the convex hull of a set of points.
58. **Segment Tree with Lazy Propagation** - Implement advanced segment tree operations.

Real-World Applications

Practical Projects

59. **Social Network Analysis** - Analyze connections in a social network using graph algorithms.
60. **Recommendation System** - Build a system to recommend items based on user preferences.
61. **Text Editor** - Create a text editor with undo and redo features using stacks.
62. **File Compression** - Implement Huffman coding for file compression.
63. **Database Indexing** - Use B-trees for indexing in a database.
64. **Scheduling System** - Develop a system to manage and schedule tasks using interval trees.
65. **Game Pathfinding** - Implement pathfinding algorithms for game AI.
66. **Real-Time Data Processing** - Handle and process streaming data efficiently.
67. **Event Calendar** - Build a calendar app with date and event management.
68. **Digital Forensics** - Create tools for file recovery and analysis.

Competitive Programming

69. **LeetCode Problems** - Solve various LeetCode problems and learn from solutions.
70. **Codeforces Contests** - Participate in contests and solve problems on Codeforces.
71. **Hackerrank Challenges** - Complete algorithm challenges on HackerRank.
72. **TopCoder Problems** - Solve problems from TopCoder contests.
73. **AtCoder Contests** - Compete in AtCoder contests and improve your skills.
74. **Project Euler** - Work on mathematical and algorithmic problems on Project Euler.
75. **Google Code Jam** - Prepare for Google Code Jam contests by solving problems.
76. **Facebook Hacker Cup** - Participate in Facebook Hacker Cup and solve challenges.
77. **SPOJ Challenges** - Solve problems from Sphere Online Judge (SPOJ).
78. **CodeChef Contests** - Compete in CodeChef contests and practice problem-solving.

Data Handling & Algorithms

Data Processing

79. **Data Analysis with Trees** - Use trees to analyze hierarchical data.
80. **Data Aggregation** - Combine and process large datasets.
81. **Text Parsing** - Implement algorithms for parsing and processing text.
82. **Web Crawling** - Use graph algorithms to crawl and analyze web pages.
83. **Streaming Data Analysis** - Analyze real-time data streams.
84. **Data Deduplication** - Remove duplicate entries from datasets.
85. **Data Compression** - Build algorithms to compress data efficiently.
86. **Data Encryption** - Implement encryption algorithms for secure data.
87. **Data Serialization** - Serialize and deserialize data formats.
88. **Big Data Processing** - Handle and process large-scale data.

Algorithm Optimization

89. **Algorithm Complexity** - Analyze and optimize algorithm performance.

90. **Greedy Algorithms** - Solve optimization problems using greedy methods.
91. **Dynamic Programming** - Use dynamic programming to solve complex problems.
92. **Backtracking** - Solve problems using backtracking techniques.
93. **Divide and Conquer** - Apply divide and conquer strategies for problem-solving.
94. **Branch and Bound** - Use branch and bound for optimization problems.
95. **Approximation Algorithms** - Develop algorithms to approximate solutions for hard problems.
96. **Heuristic Algorithms** - Apply heuristics to solve complex problems.
97. **Randomized Algorithms** - Use randomness to improve algorithm efficiency.
98. **Parallel Algorithms** - Develop algorithms that can run in parallel.

Algorithmic Challenges

Graph Algorithms

99. **Shortest Path** - Implement algorithms to find the shortest path in graphs.
100. **Strongly Connected Components** - Find strongly connected components in a graph.
101. **Eulerian Path/Cycle** - Detect Eulerian paths and cycles.
102. **Hamiltonian Path/Cycle** - Find Hamiltonian paths and cycles.
103. **Graph Coloring** - Solve graph coloring problems.
104. **Maximum Bipartite Matching** - Find maximum matchings in bipartite graphs.
105. **Cycle Detection** - Detect cycles in directed and undirected graphs.
106. **Network Flow** - Solve network flow problems using algorithms like Ford-Fulkerson.
107. **Topological Sorting** - Sort nodes in a directed acyclic graph (DAG).
108. **Minimum Spanning Tree** - Build minimum spanning trees using Kruskal’s or Prim’s algorithms.

String Algorithms

109. **Pattern Matching** - Implement algorithms like Rabin-Karp for pattern matching.
110. **String Compression** - Build algorithms to compress strings.
111. **Edit Distance** - Compute edit distance between two strings.
112. **Longest Palindromic Substring** - Find the longest palindromic substring.
113. **Substring Search** - Implement algorithms like Boyer-Moore for searching substrings.
114. **Anagram Detection** - Check for anagrams and permutations.
115. **Regular Expressions** - Implement basic regex matching algorithms.
116. **String Permutations** - Generate all permutations of a string.
117. **Text Justification** - Create algorithms for text alignment.
118. **Substring Count** - Count occurrences of substrings in a string.

Advanced Projects

Machine Learning Algorithms

119. **Decision Trees** - Implement decision trees for classification.
120. **K-Means Clustering** - Use k-means clustering for data segmentation.
121. **Naive Bayes Classifier** - Build a Naive Bayes classifier for text classification.
122. **Linear Regression** - Develop linear regression models for predictions.
123. **Support Vector Machines (SVM)** - Implement SVM for classification tasks.
124. **Neural Networks** - Create basic neural networks for pattern recognition.
125. **Random Forest** - Build random forest algorithms for robust classification.
126. **Dimensionality Reduction** - Use PCA for reducing data dimensions.
127. **Gradient Boosting** - Develop gradient boosting algorithms for better predictions.
128. **Reinforcement Learning** - Implement basic reinforcement learning algorithms.

Cryptography & Security

129. **Symmetric Encryption** - Implement symmetric encryption like AES.
130. **Asymmetric Encryption** - Build asymmetric encryption like RSA.
131. **Hash Functions** - Create hash functions for data integrity.
132. **Digital Signatures** - Implement digital signatures for authentication.
133. **Public Key Infrastructure (PKI)** - Develop a basic PKI system.
134. **Cryptographic Protocols** - Implement SSL/TLS for secure communication.
135. **Secure Hash Algorithms** - Use SHA algorithms for data verification.
136. **Key Exchange Algorithms** - Implement algorithms like Diffie-Hellman.
137. **Message Authentication Codes (MAC)** - Create MAC algorithms for data authenticity.
138. **Cryptographic Attacks** - Explore and analyze common cryptographic attacks.

Data Science & Visualization

Data Analysis

139. **Exploratory Data Analysis (EDA)** - Analyze data distributions and patterns.
140. **Data Cleaning** - Implement algorithms to clean and preprocess data.
141. **Statistical Analysis** - Conduct statistical analysis and hypothesis testing.
142. **Time Series Analysis** - Analyze and forecast time series data.
143. **Dimensionality Reduction** - Apply techniques like PCA for data reduction.
144. **Clustering Analysis** - Use clustering algorithms to group data.
145. **Outlier Detection** - Identify outliers in datasets.
146. **Data Normalization** - Normalize data for consistency.
147. **Feature Selection** - Select important features for model improvement.
148. **Data Transformation** - Transform data into useful formats.

Data Visualization

149. **Interactive Dashboards** - Create interactive dashboards for data visualization.
150. **Graph Visualization** - Visualize graphs with network diagrams.
151. **Heatmaps** - Develop heatmaps to show data density.

152. **Histograms** - Use histograms to display data distributions.
153. **Pie Charts** - Create pie charts for categorical data.
154. **Scatter Plots** - Implement scatter plots to analyze variable relationships.
155. **Box Plots** - Use box plots to show data spread and outliers.
156. **Line Charts** - Develop line charts for trend analysis.
157. **Geospatial Maps** - Create maps for location-based data visualization.
158. **3D Visualization** - Implement 3D visualizations for complex data.

Miscellaneous

Algorithm Implementation

159. **Polynomial Arithmetic** - Perform arithmetic operations on polynomials.
160. **Fraction Arithmetic** - Handle arithmetic with fractions.
161. **Matrix Operations** - Implement matrix addition, multiplication, and inversion.
162. **Graph Routing** - Use algorithms for network routing.
163. **Data Serialization** - Serialize and deserialize data formats.
164. **Recursive Algorithms** - Solve problems using recursion.
165. **Dynamic Programming for Games** - Apply dynamic programming to game strategies.
166. **Approximation Algorithms** - Develop algorithms to approximate solutions.
167. **Numerical Methods** - Use numerical methods for solving equations.
168. **Monte Carlo Simulations** - Implement Monte Carlo methods for simulations.

Advanced Concepts

169. **Quantum Algorithms** - Explore basic quantum algorithms.
170. **Genetic Algorithms** - Use genetic algorithms for optimization.
171. **Swarm Intelligence** - Implement algorithms inspired by swarm behavior.
172. **Simulated Annealing** - Apply simulated annealing for optimization.
173. **Neural Network Optimization** - Optimize neural network models.
174. **Federated Learning** - Implement federated learning for decentralized training.
175. **Blockchain Algorithms** - Explore blockchain algorithms for secure transactions.
176. **Graph Theory Applications** - Apply graph theory to real-world problems.
177. **Computational Geometry** - Solve geometric problems with algorithms.
178. **Algorithmic Trading** - Develop algorithms for trading strategies.

Challenges for Skill Improvement

Coding Competitions

179. **Algorithmic Challenges on CodeChef** - Solve problems on CodeChef.
180. **Competitive Programming on Codeforces** - Participate in Codeforces contests.
181. **HackerRank Challenges** - Work on challenges on HackerRank.
182. **TopCoder Problems** - Solve TopCoder problems.

183. **AtCoder Contests** - Compete in AtCoder contests.
184. **Project Euler** - Solve problems on Project Euler.
185. **Google Kick Start** - Participate in Google Kick Start contests.
186. **Facebook Hacker Cup** - Compete in Facebook Hacker Cup.
187. **SPOJ Problems** - Work on problems from SPOJ.
188. **CodeChef Contests** - Compete in CodeChef contests.

Personal Projects

189. **Custom DSA Library** - Build a library of data structures and algorithms.
190. **Algorithm Visualizer** - Create a tool to visualize algorithms.
191. **Coding Bootcamp** - Develop a curriculum with DSA problems.
192. **Automated Test Suite** - Build a test suite for DSA implementations.
193. **Open Source Contributions** - Contribute to open-source DSA projects.
194. **Algorithmic Art** - Use algorithms to create art.
195. **Game Development** - Apply DSA concepts in game development.
196. **AI Integration** - Integrate algorithms into AI projects.
197. **Algorithmic Blogging** - Write about DSA concepts and solutions.
198. **Educational Tools** - Create tools for teaching DSA.