Roadmap of Famous Algorithms in Computer Science

1. Fundamentals

- **Big O Notation:** Understanding time and space complexity.
- **Recursion:** Solving problems by breaking them down into smaller, self-similar subproblems.
- **Iteration:** Repeating a block of code until a condition is met.

2. Data Structures

- **Arrays:** Ordered collection of elements.
    - Algorithms: Linear Search, Binary Search (on sorted arrays).
- **Linked Lists:** Linear collection of elements with nodes containing data and pointers to the next node.
    - Algorithms: Singly Linked List, Doubly Linked List.
- **Stacks:** LIFO (Last-In, First-Out) data structure.
    - Algorithms: Stack operations (push, pop, peek).
- **Queues:** FIFO (First-In, First-Out) data structure.
    - Algorithms: Queue operations (enqueue, dequeue).
- **Trees:** Non-linear hierarchical data structure.
    - Algorithms: Binary Tree Traversal (In-order, Pre-order, Post-order), Binary Search Tree (BST) operations.
- **Graphs:** Collection of nodes (vertices) connected by edges.
    - Algorithms: Graph Traversal (BFS, DFS), Shortest Path Algorithms (Dijkstra's, Bellman-Ford).
- **Hashing:** Mapping keys to values using a hash function.
    - Algorithms: Hash Table implementation, collision handling.

3. Sorting Algorithms

- **Bubble Sort:** Simple but inefficient, compares and swaps adjacent elements.
- **Selection Sort:** Finds the minimum element and places it in the correct position.
- **Insertion Sort:** Builds the sorted array one element at a time.
- **Merge Sort:** Divide and Conquer, recursively divides the array and merges sorted subarrays.
- **Quick Sort:** Divide and Conquer, partitions the array around a pivot and recursively sorts subarrays.
- **Heap Sort:** Uses a binary heap to efficiently sort elements.

4. Searching Algorithms

- **Linear Search:** Searches an array sequentially.
- **Binary Search:** Efficiently searches a sorted array by repeatedly dividing the search interval in half.

5. Dynamic Programming

- **Overlapping Subproblems:** Breaking down a problem into smaller subproblems that are reused multiple times.
- **Optimal Substructure:** The optimal solution to a problem can be constructed from the optimal solutions to its subproblems.
    - Algorithms: Fibonacci sequence, Knapsack problem, Longest Common Subsequence (LCS).

6. Greedy Algorithms

- **Making locally optimal choices at each step in the hope of finding a global optimum.**
    - Algorithms: Activity Selection, Huffman Coding.

7. Advanced Topics

- **String Algorithms:** KMP algorithm, Rabin-Karp algorithm.
- **Computational Geometry:** Convex Hull, Closest Pair of Points.
- **Backtracking:** Systematic search algorithm that explores all possible paths.
- **Approximation Algorithms:** Algorithms that find near-optimal solutions for NP-hard problems.

Note: This roadmap provides a basic overview. Each algorithm has its own nuances and variations. It's essential to practice and understand the underlying concepts to master these algorithms.

- https://www.geeksforgeeks.org/learn-data-structures-and-algorithms-dsa-tutorial/