

Algorithm and Problem Solving Quick Guide in C++

Data Structures, Algorithms and Coding Interview Problem Patterns in C++

rfdavid, 2025

rfdavid.com

INTRODUCTION

Motivation

The tech industry hiring standard is based on algorithm and data structure.

There are plenty of free resources available around algorithms and data structures. The purpose of this project is to be a quick guide where you can learn and review learned algorithms and data structures.

Some of the intended **key features:**

- Non-verbose, short-structured, and easy to follow descriptions
- Slide-based, practical for reviewing
- Free and open-source

★ If you like, please add a star at [**github.com/rfdavid/cpp-algo-cheatsheet**](https://github.com/rfdavid/cpp-algo-cheatsheet)

Some Useful Links

Tech Interview Handbook

<https://www.techinterviewhandbook.org>

A very well-structured resource for interview preparation

Blind 75 Leetcode Questions

<https://leetcode.com/discuss/general-discussion/460599/blind-75-leetcode-questions>

Blind 75

- Blind 75 is a popular list of algorithm problems that intends to cover the main data structures and patterns.
- It is a curated list of 75 popular coding questions created by an ex-Meta Staff Engineer

Array

- ✓ [Two Sum](#)
- ✓ [Best Time to Buy and Sell Stock](#)
- ✓ [Contains Duplicate](#)
- ✓ [Product of Array Except Self](#)
- ✓ [Maximum Subarray](#)
- ✓ [Maximum Product Subarray](#)
- ✓ [Find Minimum in Rotated Sorted Array](#)
- [Search in Rotated Sorted Array](#)
- [3 Sum](#)
- ✓ [Container With Most Water](#)

Binary

- ✓ [Sum of Two Integers](#)
- ✓ [Number of 1 Bits](#)
- ✓ [Counting Bits](#)
- ✓ [Missing Number](#)
- ✓ [Reverse Bits](#)

Dynamic Programming

- ✓ [Climbing Stairs](#)
- [Coin Change](#)
- [Longest Increasing Subsequence](#)
- ✓ [Longest Common Subsequence](#)
- [Word Break](#)
- [Combination Sum](#)
- [House Robber](#)
- [House Robber II](#)
- [Decode Ways](#)
- ✓ [Unique Paths](#)
- [Jump Game](#)

Matrix

- ✓ [Set Matrix Zeroes](#)
- ✓ [Spiral Matrix](#)
- ✓ [Rotate Image](#)
- [Word Search](#)

Blind 75

Tree

- ✓ [Maximum Depth of Binary Tree](#)
- ✓ [Same Tree](#)
- ✓ [Invert/Flip Binary Tree](#)
- ✓ [Binary Tree Maximum Path Sum](#)
- ✓ [Binary Tree Level Order Traversal](#)
- ✓ [Serialize and Deserialize Binary Tree](#)
- ✓ [Subtree of Another Tree](#)

[Construct Binary Tree from Preorder and Inorder Traversal](#)

[Validate Binary Search Tree](#)

- ✓ [Kth Smallest Element in a Binary Search Tree](#)

[Lowest Common Ancestor of Binary Search Tree](#)

[Implement Trie \(Prefix Tree\)](#)

[Add and Search Word](#)

[Word Search II](#)

Heap

- ✓ [Top K Frequent Elements](#)

[Find Median from Data Stream](#)

String

- ✓ [Longest Substring Without Repeating Characters](#)
- ✓ [Longest Repeating Character Replacement](#)
- ✓ [Minimum Window Substring](#)
- ✓ [Valid Anagram](#)
- ✓ [Group Anagrams](#)
- ✓ [Valid Parentheses](#)
- ✓ [Valid Palindrome](#)

[Longest Palindromic Substring](#)

[Palindromic Substrings](#)

[Encode and Decode Strings](#) ★

Linked List

- ✓ [Reverse a Linked List](#)
- ✓ [Detect Cycle in a Linked List](#)
- ✓ [Merge Two Sorted Lists](#)
- ✓ [Merge K Sorted Lists](#)
- ✓ [Remove Nth Node From End Of List](#)
- ✓ [Reorder List](#)

Graph

- ✓ [Clone Graph](#)
- ✓ [Course Schedule](#)
- ✓ [Pacific Atlantic Water Flow](#)
- ✓ [Number of Islands](#)
- ✓ [Longest Consecutive Sequence](#)

[Alien Dictionary](#) ★

- ✓ [Graph Valid Tree](#) ★

✓ [Number of Connected Components In an Undirected Graph](#) ★

Interval

- ✓ [Insert Interval](#)
- ✓ [Merge Intervals](#)
- ✓ [Non-overlapping Intervals](#)
- ✓ [Meeting Rooms](#) ★
- [Meeting Rooms II](#) ★

Other problems

Tree

- ✓ [Maximum Level Sum of a Binary Tree](#)
- ✓ [Minimum Number of Increments on Subarrays to Form a Target Array](#)
- ✓ [Leaf-Similar Trees](#)
- ✓ [Count Good Nodes in Binary Tree](#)

Min Cost Climbing Stairs

Longest Palindromic Subsequence

Space Complexity

Space Complexity

- Count only extra space needed (**exclude output**)
- The space complexity of a recursive tree traversal is **$O(h)$** , where h is the height of the tree. This is because each recursive call adds a frame to the call stack, and in the worst case, the maximum stack depth is proportional to the tree's height