Getting Started with LeetCode & Competitive Programming

Resource Handout and Guide Linux Users Group Bits Dubai

Language Selection for Competitive Programming

Choosing Your Programming Language

When starting competitive programming, your choice of programming language significantly impacts your learning curve and contest performance. The key is to select a language that balances **ease of implementation** with **execution speed**, while matching your current skill level. Most successful competitive programmers recommend starting with **Python** for its simplicity, then transitioning to **C++** as you advance and need better performance. The goal isn't to master every language, but to become proficient enough in one primary language that you can implement any algorithm quickly and correctly. Focus on learning the language's standard libraries, built-in data structures, and common programming patterns rather than getting caught up in syntax details.

Programming Languages for LeetCode & Competitive Programming

Language	LeetCode Usefulness	Speed	Learning Curve	Best For
Python	Excellent	Medium	Easy	Beginners, interviews, rapid prototyping
C++	Excellent	Very Fast	Hard	Advanced users, contests, performance-critical problems
Java	Good	Fast	Medium	Enterprise background, structured programming
JavaScript	Moderate	Medium	Easy	Web developers transitioning to algorithms
C#	Moderate	Fast	Medium	Microsoft stack developers
Go	Moderate	Fast	Medium	Systems programming background
Rust	Fair	Very Fast	Very Hard	Advanced systems programmers

What is Competitive Programming?

Competitive programming is solving algorithmic problems under time constraints. It's essentially puzzle-solving with code - you're given a problem, and you need to write efficient code to solve it. This skill has become essential for tech interviews at companies like Google, Microsoft, and Amazon.

Why Start with LeetCode?

LeetCode bridges the gap between learning programming and getting hired. It offers 2,500+ problems organized by difficulty and topic, with a focus on interview-style questions. The platform includes company-specific problem collections and weekly contests to keep you sharp.

Unlike pure competitive programming sites, LeetCode emphasizes practical skills you'll actually use in technical interviews.

The Practical Roadmap

Master the Basics

- Solve 2-3 **Easy** problems daily
- Focus on arrays, strings, and basic math
- Don't worry about optimization yet - just get solutions working
- **Goal**: Complete 30-40 easy problems

Learn Core Patterns

Start recognizing common problem types. Focus on these 5 essential patterns first:

Pattern	What It Does	Example Problems
Two Pointers	Use two indices to traverse arrays	Two Sum, Valid Palindrome
Sliding Window	Move a window across data	Maximum Subarray, Longest Substring
Binary Search	Find elements in sorted data	Search Insert Position, First Bad Version
DFS/BFS	Explore trees and graphs	Maximum Depth of Binary Tree, Number of Islands
Dynamic Programming	Break problems into smaller parts	Climbing Stairs, House Robber

Goal: Solve 5-10 problems per pattern

Expand Pattern Knowledge

Add these 10 intermediate patterns:

Pattern	Usage	Key Problems
Fast & Slow Pointers	Detect cycles	Linked List Cycle
Merge Intervals	Handle overlapping ranges	Merge Intervals

Backtracking	Generate combinations	Subsets, Permutations
Tree Traversals	Navigate tree structures	Inorder/Preorder Traversal
Heap/Priority Queue	Find top K elements	Kth Largest Element
Union Find	Group connected components	Number of Connected Components
Trie	Efficient string searching	Word Search II
Monotonic Stack	Next greater/smaller problems	Daily Temperatures
Topological Sort	Order by dependencies	Course Schedule
Bit Manipulation	Work with binary operations	Single Number

Contest Practice & Medium Problems

- Join weekly LeetCode contests
- Attempt 1-2 **Medium** problems daily
- Review solutions in discussion sections
- **Goal**: Solve under time pressure

Essential Resources

Practice Platforms

Platform	Best For	Difficulty
LeetCode	Interviews, structured learning	Beginner-friendly
CSES Problem Set	Algorithm fundamentals	Intermediate
Codeforces	Regular contests	All levels

Books

Resource	Author/Source	Link
Competitive Programmer's Handbook	Antti Laaksonen	cses.fi/book/book.pdf
Introduction to Algorithms (CLRS)	Cormen, Leiserson, Rivest, Stein	
Programming Challenges	Steven Skiena, Miguel Revilla	cs.stonybrook.edu/~skiena/programming/
Looking for a Challenge	Various Contest Writers	mimuw.edu.pl
CS Guide	Samuel Hsiang, Alexander Wei, Yang Liu	github.com/kartikkukreja/blog-codes

Video Learning Channels

Channel	Focus Area	Skill Level	Link
takeUforward (Striver)	Complete DSA Course	Beginner to Advanced	youtube.com/@takeUforward
NeetCode	LeetCode Solutions	Beginner to Intermediate	youtube.com/@NeetCode
NeetCodeIO	Extended Tutorials	Intermediate	youtube.com/@NeetCodeIO
Abdul Bari	Algorithm Fundamentals	Beginner	youtube.com/@abdul_bari
Errichto	Advanced Techniques	Advanced	youtube.com/@Errichto
Algorithms Live!	Contest Problems	Intermediate to Advanced	youtube.com/@AlgorithmsLive

Structured Learning Platforms

Platform	Specialization	Link
USACO Guide	Contest Preparation	usaco.guide
GeeksforGeeks	Theory + Practice	geeksforgeeks.org
W3Schools DSA	Beginner Concepts	w3schools.com/dsa
Programiz	Visual Learning	programiz.com/dsa
CP-Algorithms	Algorithm Reference	cp-algorithms.com

Curated Problem Sets

Resource	Problems Count	Difficulty Range	Link
NeetCode 150	150 problems	Easy to Hard	neetcode.io/practice
Sean Prashad's Patterns	169 problems	Easy to Hard	seanprashad.com/leetcode-patterns
CSES Problem Set	200+ problems	Easy to Expert	cses.fi/problemset
LeetCode Top Interview Questions	145 problems	Easy to Hard	leetcode.com/explore/interview
Striver's SDE Sheet	191 problems	Medium to Hard	takeuforward.org/interviews/strivers-sde-sheet

Practice Platforms

Platform	Problems	Community	Contests	Best For	Link
LeetCode	2,500+	12M+ users	Weekly/Biweekly	Interview prep, pattern learning	leetcode.
Codeforces	8,000+	600K+ users	2-3 times/week	Contest practice, rating system	codeforce
CodeChef	7,000+	Active forums	Monthly	Student-friendly contests	codechef.
AtCoder	3,000+	Growing	Weekly	Clean problems, beginner contests	atcoder.j
HackerRank	1,000+	Large	Regular	Structured learning paths	hackerran
TopCoder	2,000+	Professional	Weekly SRM	Advanced competitions	topcoder.

Community and Discussion Forums

Platform	Type	Activity Level	Best For	Link
Reddit r/leetcode	Discussion Forum	Very Active	Problem discussions, study groups	reddit.com/r/leetcode
Reddit r/csMajors	Career Forum	Very Active	Interview prep, career guidance	reddit.com/r/csMajors
Codeforces Blogs	Technical Blogs	Active	Algorithm tutorials, contest analysis	codeforces.com/blog
LeetCode Discuss	Problem Discussion	Very Active	Solution explanations, hints	leetcode.com/discuss
Discord Communities	Real-time Chat	Active	Live help, study partnerships	Various servers
Stack Overflow	Q&A Platform	Very Active	Technical implementation	stackoverflow.com

			questions	

Additional Resources

Type	Resource	Description	Link
Template Libraries	Competitive Programming Templates	Ready-to-use code snippets	github.com/kth-competitive-programming
Visualizers	Algorithm Visualizer	Interactive algorithm demonstrations	algorithm-visualizer.org
Reference	Big-O Cheat Sheet	Time/space complexity reference	bigocheatsheet.com
Practice Sheets	A2OJ Ladders	Graduated difficulty problem sets	a2oj.com/ladders

Contest Strategy & Preparation Guide

Understanding Contest Types

Competitive programming contests come in two main formats: **individual contests** where you compete alone, and **team contests** where 2-3 people work together on one computer. Most online weekly contests are individual, while major championships like ICPC are team-based. Contest durations range from 90 minutes for quick weekly rounds to 10 days for long challenges.

International Championships

These are the "Olympics" of competitive programming - the most prestigious contests that can change your career.

Contest	Level	Participants	Prize/Recognition
ICPC World Finals	University teams	3-person teams from 3,400+ universities	Ultimate prestige, internships
International Olympiad in Informatics (IOI)	High school	Individual, national representatives	Gold/Silver/Bronze medals
TopCoder Open (TCO)	All levels	Individual	$15,000+ prizes, onsite finals
Google Code Jam	All levels	Individual	$15,000 grand prize, T-shirts
Meta Hacker Cup	All levels	Individual	Cash prizes, recognition
AtCoder World Finals	All levels	Individual	Cash prizes, Tokyo finals

Company-Sponsored Contests

Tech companies host these contests to discover talent and promote their brand. Performing well can lead to direct interview opportunities.

Contest	Company	Frequency	Focus
Google Code Jam	Google	Annual	Algorithmic problem solving
Google Kick Start	Google	Quarterly	Interview preparation
Meta Hacker Cup	Meta	Annual	Advanced algorithms
Microsoft Imagine Cup	Microsoft	Annual	Innovation projects
Apple WWDC Swift Student Challenge	Apple	Annual	Swift/iOS development

Regular Platform Contests

These are your weekly practice grounds - perfect for building consistency and improving ratings gradually.

Platform	Contest Type	Schedule	Duration
Codeforces	Div 1/2/3 Rounds	2-3 times/week	2-2.5 hours
AtCoder	Beginner/Regular Contest	Weekly (Sat/Sun)	100-120 minutes
CodeChef	Long/Cook-off/Lunchtime	Monthly	3-10 days/3.5 hours
LeetCode	Weekly/Biweekly	Every week	90 minutes
TopCoder	SRM (Single Round Match)	Weekly	90 minutes
HackerRank	Weekly/Monthly Challenges	Regular	Varies

Time Management Framework

Think of contests like exams - you need to maximize points within the time limit, not solve every problem perfectly.

Contest Phase	Time Allocation	Strategy
Problem Reading	10-15% of total time	Read all problems, identify easiest
Easy Problems	40% of time	Solve 2-3 easiest problems first
Medium Problems	35% of time	Focus on problems you can solve
Hard Problems	15% of time	Attempt only if time permits

Problem Selection Priority

1. **Implementation problems** - straightforward logic, minimal algorithms
2. **Pattern recognition** - problems matching known templates
3. **Partial credit problems** - where brute force gives some points

4. **Novel problems** - attempt only if others are solved

Team Coordination

In team contests, clear roles prevent chaos and maximize efficiency since only one person can code at a time.

- **Coder**: Implements solutions, handles keyboard
- **Strategist**: Reads problems, plans approaches, debugs
- **Implementer**: Works on secondary problems, provides backup

Remember: **Contest participation is the fastest way to improve**. Start with easier contests, focus on learning over ranking, and gradually work up to more challenging competitions.