Coding Interview Tips Cheatsheet

A concise guide offering effective strategies and techniques for acing coding interviews, covering preparation, problem-solving, communication, and follow-up.

Preparation Strategies

Fundamentals Review

Data Structures: Master arrays, linked lists, trees, graphs, hash tables, stacks, and queues. Understand their properties, time complexities, and use cases.

Algorithms: Grasp sorting (e.g., quicksort, mergesort), searching (e.g., binary search), and graph algorithms (e.g., Dijkstra's, BFS, DFS). Understand their trade-offs.

Time Complexity (Big O): Learn to analyze the efficiency of algorithms. Focus on O(1), O(log n), O(n), O(n log n), O(n$^{2), \text{ and } O(2}$n). Practice determining complexity for common operations.

Space Complexity: Understand how much memory your algorithms use. Be mindful of auxiliary space used in addition to input data.

Practice Platforms

LeetCode	Extensive problem set, active community, and interview simulations.
HackerRank	Diverse challenges, tracks progress, and provides company-specific preparation kits.
GeeksforGeeks	Comprehensive articles, explanations, and coding problems.
Interview Cake	Focuses on understanding underlying principles, not just memorizing solutions.

Mock Interviews

Schedule mock interviews with peers or online services (e.g., Pramp, interviewing.io). Simulate real interview pressure to identify areas for improvement.

Ask for detailed feedback on your problem-solving approach, coding style, and communication skills.

Record yourself to analyze your body language and verbal communication.

During the Interview

Understanding the Problem

Clarify Requirements: Ask clarifying questions to fully understand the problem scope, constraints, and edge cases. Don't assume anything!

Example Inputs/Outputs: Work through a few examples to solidify your understanding and identify potential complexities.

Test Cases: Think about different types of test cases: basic, edge, large-scale, and negative. This demonstrates thoroughness.

Problem Solving Approach

Think Out Loud: Explain your thought process as you explore potential solutions. The interviewer wants to see *how* you think.

Break it Down: Decompose the problem into smaller, manageable subproblems. This makes the overall task less daunting.

Consider Trade-offs: Analyze the time and space complexity of different approaches and discuss the trade-offs with the interviewer.

Optimal Solution: Aim for the most efficient solution, but don't get stuck optimizing prematurely. A working solution is better than no solution.

Coding

Write Clean Code: Use meaningful variable names, proper indentation, and comments to improve readability.

Modularize: Break your code into functions to improve organization and reusability.

Handle Edge Cases: Explicitly address potential edge cases in your code to prevent errors.

Don't Panic: If you get stuck, take a deep breath and revisit your approach. Ask the interviewer for a hint if necessary.

Communication Skills

Verbal Communication

Be Clear and Concise: Articulate your thoughts clearly and avoid rambling. Use precise language to explain your ideas.

Active Listening: Pay attention to the interviewer's questions and instructions. Ask follow-up questions to ensure understanding.

Explain Trade-offs: Clearly articulate the reasoning behind your design choices, highlighting the benefits and drawbacks of each option.

Non-Verbal Communication

Maintain Eye Contact: Show engagement and confidence by making eye contact with the interviewer.

Body Language: Sit upright, avoid fidgeting, and use hand gestures to emphasize your points.

Enthusiasm: Express genuine interest in the problem and the company. Show that you are excited about the opportunity.

Asking Questions

Prepare Questions: Have a few thoughtful questions prepared about the company, the team, or the role. This demonstrates your interest and initiative.

Focus on Culture and Growth: Ask questions that reveal insights into the company culture, opportunities for professional development, and the team's goals.

Avoid Generic Questions: Steer clear of questions easily answered by a quick search on the company website. Show you've done your research.

Post-Interview

Follow-Up

Thank-You Note: Send a personalized thank-you note within 24 hours, reiterating your interest and highlighting key takeaways from the interview.

Be Specific: Reference specific topics discussed during the interview to demonstrate your engagement and attentiveness.

Review and Reflection

Analyze Performance: Review your performance in the interview. What went well? What could you have done better? Identify areas for improvement.

Seek Feedback: Reach out to mock interviewers or mentors for additional feedback on your performance.

Document Learning: Keep a log of the questions you encountered, the solutions you developed, and the lessons you learned. This will help you prepare for future interviews.

Handling Rejection

Don't Take it Personally: Rejection is a common part of the job search process. Don't let it discourage you.

Request Feedback: If possible, ask for specific feedback on why you weren't selected. This can provide valuable insights for future improvement.

Stay Positive: Maintain a positive attitude and continue to refine your skills and strategies. Persistence is key.