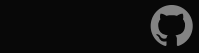
Coding Interview Preparation in the AI Era

Let's get started →



About Me

Suresh Kumar (DSK)

- Passionate Software Engineer
- Senior Software Architect @ Appknox
- Organiser @ Build 2 Learn tech community

  https://build2learn.in
- Tech blogger @ https://sureshdsk.dev
- Open source contributor

Agenda

- Technical Interviews in AI Era
- Fundamentals?
- What Interviewers look for
- Leveraging AI for Preparation
- Preparation strategy
- Learning path
- Data structures overview
- Essential algorithms
- Personal branding

The AI Era: What's Changed?

Old Approach

- Reading documentations
- Google, stack overflow
- Memorize Data structures
- Memorize algorithms
- Solve LeetCode, Hacker rank problems

New Approach

- Chatgpt
- Collaborate with AI tools
- AI-assisted development
- Cursor, windsurf, claud code
- Building side projects
- Participating in hackathons

Fundamentals Still Matter

- Data Structures and Algorithms (DSA) remain crucial.
- AI can help understand concepts and generate examples, but deep understanding is required.
- Practice implementing DSA from scratch to build intuition and problem-solving skills.
- Understand time and space complexity thoroughly.
- Understand Object oriented programming concepts.
- Understand fundamental protocols DNS, TCP/IP, UDP, HTTP, TLS, etc.

What Interviewers Look For

- **Problem Decomposition:** Can you break down a complex problem into smaller, manageable parts?
- **Critical Thinking:** Do you understand the constraints and edge cases?
- **Communication:** Can you clearly explain your thought process, design choices, and trade-offs?
- **Adaptability:** How do you approach a problem when the initial idea doesn't work?

🧠 Problem-Solving Mindset

- Break down complex problems
- Think algorithmically
- Communicate your approach clearly
- Handle ambiguity well

💻 Technical Foundation

- Data structures & algorithms
- System design principles
- Code quality & testing
- Performance optimization

🗣️ Communication Skills

- Explain complex concepts simply
- Ask clarifying questions
- Present trade-offs clearly

Leveraging AI for Preparation

- **Understanding Concepts:** Use AI to explain complex algorithms, data structures, and design patterns.
- **Generating Practice Problems:** Ask AI for variations of common interview questions or generate entirely new ones.
- **Debugging and Code Review:** Use AI to identify errors in your practice code and suggest improvements.
- **Mock Interviews:** Practice explaining your thought process and solutions to an AI.
- **Language and Framework Specifics:** Get quick explanations and code snippets for syntax or specific library usage.

Essential Preparation Strategies

📚 Study Plan

- **Fundamentals First**: Master data structures & algorithms
- **AI Tools Practice**: Get comfortable with ChatGPT, Claude, Copilot
- **Mock Interviews**: Practice AI-assisted problem solving
- **System Design**: Study scalable architecture patterns

🛠️ Websites to checkout

- **GitHub Copilot/Curson/Windsurf**: Code completion and IDE
- **Gemini/Claude/ChatGPT**: Problem explanation and debugging
- **Replit/CodeSandbox**: Collaborative coding environments
- **LeetCode/Hackerrank**: Problem solving

💡 Pro Tip

Red Flags to Avoid

❌ Don’t Do This

- Blindly copy AI solutions
- Skip understanding the logic
- Ignore edge cases
- Assume AI is always right
- Forget to test your code

✅ Do This Instead

- Explain AI-generated code
- Verify and validate solutions - Test Driven Development
- Consider multiple approaches
- Think critically about AI output

⚠️ Warning

Companies can detect over-reliance on AI. Show that you understand the fundamentals and can think independently.

Key Takeaways

- 🎯 **Master the fundamentals** - AI enhances, doesn't replace core skills
- 🤖 **Embrace AI collaboration** - Learn to work effectively with AI tools
- 🗣️ **Communicate clearly** - Explain your reasoning
- ⚡ **Stay adaptable** - The interview process is dynamic
- 🎓 **Keep learning** - Continuous improvement is your superpower

Learning paths

- https://www.hackerrank.com/interview/preparation-kits
- https://www.hackerrank.com/domains/data-structures
- https://leetcode.com/problemset/
- https://leetcode.com/studyplan/
- https://neetcode.io/roadmap
- https://roadmap.sh/

Algorithm Visualisation

- https://algorithm-visualizer.org/
- https://dsa-visualizer-delta.vercel.app/

Design Patterns - https://refactoring.guru/design-patterns/catalog

Essential Data Structures

The Building Blocks of Programming

- Arrays & Strings
- Linked lists
- Stacks & Queues
- Heap
- Trees
- Hash tables
- Graphs

📚 Arrays & Strings

Description

Contiguous memory structures for storing elements of the same type

Examples

- In mathematics and scientific computing, arrays are used to represent matrices, enabling efficient matrix multiplication, inversion, and other linear algebra operations.
- textual data, User input and output.

🔗 Linked Lists

Description

Dynamic structures with nodes containing data and pointers to the next node

Examples

- Netflix "Continue Watching" episode queue
- Spotify song queue in party playlists
- Instagram Stories viewing sequence

📦 Stacks & Queues

Description

Stack: LIFO (Last In, First Out) data structure
Queue: FIFO (First In, First Out) data structure

Examples

- **Stack**: Browser back/forward navigation history
- **Stack**: Undo/Redo in Google Docs while writing essays
- **Queue**: Printer queue in computer lab during finals

🌳 Trees & Binary Search Trees

Description

Hierarchical structures with nodes connected by edges, BST maintains sorted order

Examples

- Comment thread hierarchy (parent → child comments)
- Discord server structure (Server → Categories → Channels)
- Database Indexing

Hash Tables

Description

Key-value pairs with hash function mapping keys to array indices

Examples

- Instagram username to user profile and follower count
- Spotify song ID to track metadata (artist, album, duration)

Graphs

Description

Networks of vertices (nodes) connected by edges, can be directed or undirected

Examples

- Instagram follower network and mutual connections
- LinkedIn professional network recommendations
- DAG workflow in data pipelines - Direct acyclic graph

🔍 Tries

Description

Tree structures optimized for efficient string operations and prefix matching

Examples

- Instagram search autocomplete for usernames and hashtags
- Spotify song/artist autocomplete in search bar
- YouTube video title suggestions as you type
- Amazon product search predictions

Essential Algorithms

- Sorting
- Searching
- Dynamic Programming
- Greedy algorithms
- Breadth-First Search
- Depth-First Search
- Dijkstra's Algorithm
- Minimum Spanning Tree

Algorithm Complexity Cheat Sheet

⚡ Big O Notation

- **O(1)**: O(yeah)
- **O(log n)**: O(nice)
- **O(n)**: O(ok)
- **O(n log n)**: O(no)
- **O(n^2)**: O(sh*t)
- **O(2^n)**: O(mg)

💡 Interview Tip

Always discuss time and space complexity trade-offs. Interviewers love candidates who can optimize for different constraints!

References and Resources

🌐 Online Platforms

- **LeetCode**: leetcode.com
- **HackerRank**: hackerrank.com
- **Neetcode**: neetcode.io
- **Systemdesignschool**: systemdesignschool.io

📺 YouTube Channels

- **ByteByteGo**: System design and interview tips - https://www.youtube.com/@ByteByteGo
- **Back To Back SWE**: Algorithm explanations - https://www.youtube.com/@BackToBackSWE
- **Gaurav Sen**: System design concepts - https://www.youtube.com/@gkcs
- **CS Dojo**: Programming interview prep - https://www.youtube.com/@CSDojo
- **Jenny's Lectures CS IT** - https://www.youtube.com/channel/UCM-yUTYGmrNvKOCcAl21g3w

Personal branding

1. Blogging - Technical writing, communication - https://hashnode.com/ https://medium.com/
2. Side projects - build micro products
3. Open source contributions(optional)
4. 100 days of code

   https://www.100daysofcode.com/

   https://www.100daysofcode.io/learn/python

Be passionately curious.

Thank You!

Connect With Me

💼 **LinkedIn**: linkedin.com/in/sureshdsk
🐱 **GitHub**: github.com/sureshdsk
🐦 **Twitter**: @sureshdsk

Best of luck with your interviews! Remember: You've got this! 🚀