SDE ROADMAP

120 concepts to become a software engineer

8 areas, 15 concepts each. Every concept explained in one line, the order to study each area, and two practice questions to test yourself.

How to use it: one area at a time, tick each concept only when you can explain it out loud in under a minute, then answer the two practice questions without notes.

@chalkesahil · chalkesahil.ai

AREA 01 / 08

Programming language

Study order: Pick one of C++, Java or Python and code in it every day.

01	**Variables and types**	How data is stored.
02	**Conditions and loops**	Control the flow.
03	**Functions**	Reusable logic.
04	**Recursion**	A function calling itself.
05	**Arrays and lists**	The first data structure.
06	**Strings**	Text and its common operations.
07	**Hash maps**	Fast lookups by key.
08	**Sets**	Unique values.
09	**Sorting built-ins**	Sort with custom rules.
10	**Input and output**	Read fast for coding tests.
11	**Classes**	Group data and behaviour.
12	**Error handling**	Fail with useful messages.
13	**Standard library**	STL, Collections or Python built-ins.
14	**Memory basics**	Stack vs heap.
15	**Writing clean code**	Names that explain themselves.

Practice

1. Reverse each word in a sentence.
2. Count how often each character appears in a string.

AREA 02 / 08

DSA basics

Study order: Learn to measure code before you learn more patterns.

01	**Time complexity**	How runtime grows with input.
02	**Space complexity**	How memory grows.
03	**Big O**	The notation interviewers use.
04	**Brute force first**	Then improve it.
05	**Constraints**	Input size tells you the target complexity.
06	**Linear search**	Check every element.
07	**Binary search**	Halve the search space each step.
08	**Sorting algorithms**	Merge, quick and their costs.
09	**Two pointers**	Two indexes moving through data.
10	**Sliding window**	A moving range over an array.
11	**Prefix sums**	Answer range totals fast.
12	**Hashing patterns**	Trade memory for speed.
13	**Stack**	Last in, first out.
14	**Queue**	First in, first out.
15	**Edge cases**	Empty, one element, duplicates.

Practice

1. Find two numbers in an array that add up to a target.
2. Find the longest substring without repeating characters.

AREA 03 / 08

Arrays and strings

Study order: These are the most asked topics in coding rounds.

01	**Kadane's algorithm**	Maximum subarray sum.
02	**Moving zeros**	In-place array changes.
03	**Rotate an array**	Common trick question.
04	**Merge intervals**	Combine overlapping ranges.
05	**Dutch national flag**	Sort 0s, 1s and 2s.
06	**Majority element**	Find the element seen most.
07	**Next permutation**	Classic medium problem.
08	**Matrix traversal**	Spiral and diagonal walks.
09	**Anagrams**	Compare letter counts.
10	**Palindromes**	Read the same both ways.
11	**Pattern matching**	Find a word inside text.
12	**String building**	Avoid slow concatenation.
13	**Frequency counting**	Hash maps on characters.
14	**Subarray with given sum**	Prefix sums plus hashing.
15	**Practice set**	50 easy and medium problems.

Practice

1. Merge overlapping meeting times.
2. Group words that are anagrams of each other.

AREA 04 / 08

Linked lists and trees

Study order: Draw every problem on paper before coding it.

01	**Linked list basics**	Nodes pointing to the next.
02	**Reverse a linked list**	The classic first question.
03	**Fast and slow pointers**	Find the middle or a cycle.
04	**Merge two sorted lists**	Common follow-up.
05	**Binary trees**	Nodes with two children.
06	**Tree traversals**	Inorder, preorder, postorder.
07	**Level order traversal**	Breadth-first with a queue.
08	**Height and diameter**	Recursion on trees.
09	**Binary search trees**	Left smaller, right bigger.
10	**Lowest common ancestor**	Classic tree question.
11	**Heaps**	Always get the smallest or largest fast.
12	**Top K problems**	Heaps in action.
13	**Tries**	Prefix search for words.
14	**Recursion on trees**	Think in subproblems.
15	**Practice set**	40 list and tree problems.

Practice

1. Detect a cycle in a linked list.
2. Find the k most frequent words.

AREA 05 / 08

Graphs and DP

Study order: Product companies separate candidates here.

01	**Graph representation**	Adjacency list vs matrix.
02	**BFS**	Shortest path in unweighted graphs.
03	**DFS**	Explore as deep as possible.
04	**Connected components**	Count separate groups.
05	**Cycle detection**	In directed and undirected graphs.
06	**Topological sort**	Order tasks with dependencies.
07	**Dijkstra**	Shortest path with weights.
08	**Union-Find**	Merge groups fast.
09	**Grid problems**	Islands and flood fill.
10	**DP basics**	Reuse answers to subproblems.
11	**Memoization**	Top-down DP.
12	**Tabulation**	Bottom-up DP.
13	**Knapsack**	Pick items under a limit.
14	**Longest common subsequence**	Classic string DP.
15	**Practice set**	30 graph and 30 DP problems.

Practice

1. Count the islands in a grid.
2. Find the minimum coins needed to make an amount.

AREA 06 / 08

OOP and low-level design

Study order: Many rounds ask you to design classes for a small system.

01	**Classes and objects**	Blueprint and instance.
02	**Encapsulation**	Hide internal details.
03	**Inheritance**	Reuse behaviour.
04	**Polymorphism**	Same call, different behaviour.
05	**Abstraction**	Show what, hide how.
06	**Interfaces**	Contracts between classes.
07	**Composition**	Has-a over is-a.
08	**SOLID principles**	Five rules for clean design.
09	**Singleton**	One instance only.
10	**Factory**	Create objects in one place.
11	**Strategy**	Swap algorithms at runtime.
12	**Observer**	Notify many on one event.
13	**UML basics**	Sketch classes and relations.
14	**Parking lot design**	The most common LLD question.
15	**BookMyShow design**	Seats, shows and bookings.

Practice

1. Design classes for a parking lot.
2. Design a library management system.

AREA 07 / 08

DBMS, OS and networks

Study order: Service and product companies both ask CS fundamentals.

01	**SQL queries**	Joins, GROUP BY and subqueries.
02	**Normalisation**	Remove duplicate data.
03	**Indexes**	Faster lookups.
04	**Transactions and ACID**	Safe writes.
05	**Processes vs threads**	Units of execution.
06	**Scheduling**	How the OS picks what runs.
07	**Deadlocks**	Waiting forever on each other.
08	**Memory management**	Paging and virtual memory.
09	**Synchronisation**	Mutexes and semaphores.
10	**OSI and TCP/IP**	Layers of networking.
11	**TCP vs UDP**	Reliable vs fast.
12	**HTTP**	How browsers talk to servers.
13	**DNS**	Names to IPs.
14	**What happens when you type a URL**	Classic question.
15	**System design basics**	Load balancers, caches, databases.

Practice

1. Find the second highest salary in SQL.
2. Explain a deadlock with a real example.

AREA 08 / 08

Projects and interviews

Study order: Two solid projects and steady practice beat random grinding.

01	**Project 1: full CRUD app**	Login, database, deployed.
02	**Project 2: something unique**	Solves a real problem.
03	**GitHub profile**	Clean repos with READMEs.
04	**Resume**	One page, impact with numbers.
05	**Online assessments**	Timed coding tests.
06	**Mock interviews**	Practise out loud.
07	**Explaining your approach**	Talk while you code.
08	**Behavioural questions**	STAR format answers.
09	**Referrals**	Most interviews come through people.
10	**LinkedIn**	Show your work.
11	**Company-wise questions**	Practise past patterns.
12	**Contest practice**	LeetCode or Codeforces weekly.
13	**Revision sheet**	Re-solve the problems you missed.
14	**Salary negotiation**	Know your market rate.
15	**Using AI tools**	Learn faster, still write it yourself.

Practice

1. Explain your best project in two minutes.
2. Tell me about a time you fixed a hard bug.