Alright Aryan, gloves off 🥊 — you want **top SDE at FAANG++/AI labs level** → you need to be **dangerously good** across the *entire stack*. Here’s the **no-fluff roadmap**:

**🚀 Mastery Roadmap for Top SDE**

**1. Computer Science Foundations**

* **Data Structures**: arrays, strings, linked lists, stacks, queues, hash tables, heaps, trees (BST, AVL, Segment, Fenwick), tries, graphs.
* **Algorithms**: sorting, searching, greedy, divide & conquer, DP, graph algos (DFS, BFS, Dijkstra, Floyd-Warshall, Bellman-Ford, Kruskal/Prim), string algorithms (KMP, Rabin-Karp, Z-algo, suffix array/tree), number theory, combinatorics, geometry basics.
* **Complexity**: Big-O, amortized analysis, space vs time tradeoffs.
* **System Design basics**: scalability, caching, load balancing, queues, consistency models.

**2. Programming Mastery**

* **At least 2 languages**:
  + Low-level (C++/Rust/Go) → performance, memory mgmt.
  + High-level (Python/Java/TypeScript) → productivity, ML/DS ecosystems.
* **Concurrency & Parallelism**: threads, async/await, locks, race conditions, deadlocks, distributed systems concurrency.
* **Error handling, debugging, profiling**.
* Writing **clean, testable, maintainable code**.

**3. Systems & OS**

* Processes, threads, context switching.
* Memory management, virtual memory, paging.
* File systems, I/O, interrupts.
* Networking (TCP, UDP, HTTP, gRPC, DNS, TLS).
* Linux mastery (bash, scripting, system calls, perf tools).

**4. Databases**

* **SQL**: joins, indexes, transactions, isolation levels.
* **NoSQL**: MongoDB, DynamoDB, Redis, Cassandra.
* Query optimization, sharding, replication, CAP theorem.

**5. Large-Scale System Design**

* REST vs GraphQL vs gRPC.
* Caching layers (CDNs, Redis, Memcached).
* Message queues (Kafka, RabbitMQ, SQS).
* Horizontal scaling, microservices, service discovery.
* Storage design: object stores (S3), blob storage, data lakes.
* Event-driven & stream processing.
* Monitoring, observability, logging.

**6. DevOps & Cloud**

* Containers: Docker, Kubernetes.
* CI/CD pipelines.
* AWS, GCP, or Azure deep dive (at least one).
* Infra as code (Terraform, Ansible).
* Security best practices.

**7. Software Engineering Practices**

* **Version control**: Git (advanced branching, rebasing, bisect).
* **Testing**: unit, integration, load, fuzzing.
* **Agile/XP**: Scrum, code reviews, pair programming.
* **Design patterns** & SOLID principles.
* **Refactoring** large codebases.

**8. Math & Problem-Solving**

* Discrete math, probability, linear algebra.
* Cryptography basics (hashes, signatures).
* ML basics (since top-tier SDEs touch AI infra now).

**9. Specialized Topics (pick depending on domain)**

* **High-performance computing**: SIMD, GPU programming, CUDA.
* **Networking systems**: packet routing, distributed consensus (Paxos, Raft).
* **Compiler design / interpreters**.
* **Blockchain / cryptosystems** (if infra/fintech).
* **Search & IR systems**: indexing, ranking, recommendation engines.

**10. Mindset & Career Edge**

* **Contribute to OSS** → credibility + GitHub profile.
* **Read large codebases** (Kubernetes, Linux kernel, TensorFlow).
* **Practice real system design interviews**.
* Stay updated with **tech trends** (AI infra, LLM systems, edge compute).
* Learn to **communicate complex systems simply** → that’s how you get promoted to senior/staff.

⚡ **TL;DR**:  
Think of it like building a skyscraper:

* **Foundation** = DSA, OS, networks, DB.
* **Floors** = system design, programming languages, scaling.
* **Top floors** = domain specialization + leadership.

If you can walk into a room, whiteboard **how to build Twitter at scale**, implement **Dijkstra in C++ in 10 mins**, and debug **a deadlock in prod**, you’re *top 1% SDE material*.

Do you want me to give you a **stepwise study plan** (like Year 1 → DSA grind, Year 2 → systems, Year 3 → system design & OSS), or a **daily/weekly grind plan** for max output?