SDE Interview and Prep Roadmap

linkedin.com/in/aastha-shukla/

aasthas2022@gmaill.com

Overview

Welcome to the SDE Interview Preparation Roadmap! This repository is not just about my personal journey; it's a collaborative space for collective learning. As I prepare for Software Development Engineer (SDE) interviews, I've created a comprehensive checklist to guide my preparation. By sharing this roadmap, I aim to foster a community of learners where we can all grow together. It covers various domains including **Data Structures, Algorithms, System Design, Operating Systems, Networking, Databases, Programming Languages and Concepts, System Architecture, Problem-solving and Coding, as well as Behavioral and Soft Skills**.

Domains and Topics

1. **Data Structures**
    - [ ] **Arrays**
        - [ ] Dynamic arrays
            - [ ] ArrayList
            - [ ] Vector
            - [ ] ArrayList
        - [ ] Sparse arrays
        - [ ] Associative arrays
    - [ ] **Linked Lists**
        - [ ] Singly linked lists
            - [ ] Circularly linked lists
            - [ ] Lock-free linked lists
        - [ ] Doubly linked lists
            - [ ] Circular doubly linked lists
        - [ ] Circular linked lists
        - [ ] Skip lists
        - [ ] Unrolled linked lists
    - [ ] **Stacks**
        - [ ] Implementations using arrays and linked lists
            - [ ] Array-based stack
            - [ ] Linked list-based stack
        - [ ] Applications (e.g., expression evaluation, backtracking)
        - [ ] Priority stacks
    - [ ] **Queues**
        - [ ] Implementations (e.g., array-based, linked list-based, priority queues)
            - [ ] Circular queue
            - [ ] Double-ended queue (Deque)
        - [ ] Applications (e.g., BFS, job scheduling)
    - [ ] **Trees**
        - [ ] Binary Trees
            - [ ] Full binary tree
            - [ ] Complete binary tree
            - [ ] Perfect binary tree
        - [ ] Binary Search Trees (BST)
            - [ ] Self-balancing BST
            - [ ] Scapegoat tree
            - [ ] Tango tree

    - [ ] AVL Trees
    - [ ] Red-Black Trees
    - [ ] Splay Trees
    - [ ] B-Trees
    - [ ] Heap Trees (min-heap, max-heap)
    - [ ] Trie
    - [ ] Radix Trees

- [ ] **Graphs**
    - [ ] Representations (adjacency matrix, adjacency list)
        - [ ] Edge list
        - [ ] Incidence matrix
    - [ ] Traversal algorithms (DFS, BFS)
    - [ ] Weighted graphs
    - [ ] Directed graphs
    - [ ] Acyclic graphs
    - [ ] Bipartite graphs
    - [ ] Spanning trees (Minimum Spanning Tree, Maximum Spanning Tree)
    - [ ] Graphs with special properties (e.g., planar graphs, Eulerian graphs)

- [ ] **Hash Tables**
    - [ ] Collision resolution techniques (chaining, open addressing)
    - [ ] Hash functions
    - [ ] Perfect Hashing
    - [ ] Cuckoo Hashing
    - [ ] Robin Hood Hashing
    - [ ] Count-Min Sketch
    - [ ] Bloom Filters

2. Algorithms

- [ ] **Sorting Algorithms**:
    - [ ] Bubble Sort
    - [ ] Selection Sort
    - [ ] Insertion Sort
    - [ ] Merge Sort
    - [ ] Quick Sort
    - [ ] Heap Sort
    - [ ] Shell Sort
    - [ ] Counting Sort
    - [ ] Radix Sort
    - [ ] Bucket Sort
    - [ ] Comb Sort
    - [ ] Cocktail Shaker Sort
    - [ ] Tim Sort
    - [ ] Cycle Sort
    - [ ] Pancake Sort
    - [ ] Bitonic Sort
    - [ ] Gnome Sort
    - [ ] Strand Sort

- [ ] **Searching Algorithms**:
    - [ ] Linear Search
    - [ ] Binary Search

- [ ] Depth-First Search (DFS)
- [ ] Breadth-First Search (BFS)
- [ ] Jump Search
- [ ] Interpolation Search
- [ ] Exponential Search
- [ ] Fibonacci Search
- [ ] Ternary Search
- [ ] Hashing (Hash Table)

- [ ] **Dynamic Programming**
  - [ ] Memoization
  - [ ] Tabulation
  - [ ] Longest Common Subsequence (LCS)
  - [ ] Longest Increasing Subsequence (LIS)
  - [ ] 0/1 Knapsack Problem
  - [ ] Coin Change Problem
  - [ ] Matrix Chain Multiplication
  - [ ] Edit Distance
  - [ ] Subset Sum Problem
  - [ ] Rod Cutting Problem
  - [ ] Fibonacci Series
  - [ ] Shortest Path Problems (e.g., Dijkstra's Algorithm using DP)
- [ ] **Greedy Algorithms**
  - [ ] Fractional Knapsack Problem
  - [ ] Activity Selection Problem
  - [ ] Huffman Coding
  - [ ] Job Sequencing with Deadlines
  - [ ] Prim's Algorithm (for Minimum Spanning Tree)
  - [ ] Kruskal's Algorithm (for Minimum Spanning Tree)
  - [ ] Dijkstra's Algorithm (for Single Source Shortest Path)
  - [ ] Greedy Coloring of Graphs
  - [ ] Greedy Set Cover
- [ ] **Divide and Conquer**
  - [ ] Binary Search
  - [ ] Merge Sort
  - [ ] Quick Sort
  - [ ] Strassen's Matrix Multiplication
  - [ ] Closest Pair of Points Problem
  - [ ] Karatsuba Algorithm (for Fast Multiplication)
  - [ ] Cooley–Tukey Fast Fourier Transform (FFT)
  - [ ] Finding Maximum Subarray Sum (Kadane's Algorithm)
  - [ ] Finding Peak Element in 1D/2D Array
- [ ] **String Algorithms**:
  - [ ] Rabin-Karp Algorithm
  - [ ] Knuth-Morris-Pratt (KMP) Algorithm
  - [ ] Z-Algorithm
  - [ ] Boyer-Moore Algorithm
  - [ ] Manacher's Algorithm
  - [ ] Suffix Array Construction
  - [ ] Longest Common Prefix (LCP) Array
  - [ ] Aho-Corasick Algorithm
  - [ ] Suffix Tree Construction
  - [ ] Suffix Automaton
  - [ ] Trie (Prefix Tree) Insertion and Search

    - [ ] Breadth-First Search (BFS) in a Trie
    - [ ] Depth-First Search (DFS) in a Trie
    - [ ] Edit Distance (Levenshtein Distance)
    - [ ] Hamming Distance
    - [ ] Longest Palindromic Substring
    - [ ] Longest Repeated Substring
    - [ ] Longest Common Substring
    - [ ] Longest Common Subsequence
    - [ ] Shortest Common Supersequence
    - [ ] Palindrome Check
    - [ ] Count and Say Sequence
    - [ ] String Hashing
    - [ ] Baker-Bird Algorithm
    - [ ] Burrows-Wheeler Transform (BWT)

3. System Design

- [ ] **Design patterns**
    - [ ] **Creational patterns**
        - [ ] Singleton
        - [ ] Factory Method
        - [ ] Abstract Factory
        - [ ] Builder
        - [ ] Prototype
    - [ ] **Structural patterns**
        - [ ] Adapter
        - [ ] Bridge
        - [ ] Composite
        - [ ] Decorator
        - [ ] Facade
        - [ ] Flyweight
        - [ ] Proxy
    - [ ] **Behavioral patterns**
        - [ ] Chain of Responsibility
        - [ ] Command
        - [ ] Iterator
        - [ ] Mediator
        - [ ] Memento
        - [ ] Observer
        - [ ] State
        - [ ] Strategy
        - [ ] Template Method
        - [ ] Visitor
- [ ] **Object-oriented design principles**
    - [ ] DRY (Don't Repeat Yourself)

    - [ ] KISS (Keep It Simple, Stupid)
    - [ ] YAGNI (You Aren't Gonna Need It)
    - [ ] Law of Demeter (Principle of Least Knowledge)
    - [ ] Open/Closed Principle
    - [ ] SOLID principles
        - [ ] Single Responsibility Principle (SRP)
        - [ ] Open/Closed Principle (OCP)
        - [ ] Liskov Substitution Principle (LSP)
        - [ ] Interface Segregation Principle (ISP)
        - [ ] Dependency Inversion Principle (DIP)

- [ ] **Scalability**
    - [ ] Replication vs. Partitioning
    - [ ] Consistent Hashing
    - [ ] Auto-scaling

- [ ] **Distributed systems**
    - [ ] ACID vs. BASE
    - [ ] Eventual consistency
    - [ ] Leader election algorithms (Paxos, Raft)
    - [ ] Distributed tracing
    - [ ] Fault tolerance and resilience

- [ ] **Microservices architecture**
    - [ ] Choreography vs. Orchestration
    - [ ] API gateway
    - [ ] Circuit Breaker pattern
    - [ ] Saga pattern
    - [ ] Caching strategies
        - [ ] Cache aside
        - [ ] Write-through caching
        - [ ] Write-behind caching
        - [ ] Cache stampede prevention
    - [ ] Load balancing
        - [ ] DNS load balancing
        - [ ] Content Delivery Networks (CDNs)
        - [ ] Anycast routing
        - [ ] Adaptive load balancing algorithms

- [ ] **Database design and optimization**
    - [ ] Partitioning
    - [ ] Materialized views
    - [ ] NoSQL databases (document, key-value, column-family, graph)
    - [ ] Database normalization forms (1NF, 2NF, 3NF, BCNF)

4. Operating Systems

- [ ] **Processes**
- [ ] **Threads**
- [ ] **Thread synchronization mechanisms**:
    - [ ] Mutexes

    - [ ] Semaphores
    - [ ] Monitors
- [ ] **Deadlock detection and prevention**
- [ ] **Scheduling algorithms**
    - [ ] Fair share scheduling
    - [ ] Earliest Deadline First (EDF)
    - [ ] Weighted Fair Queuing (WFQ)
- [ ] **Memory management**
    - [ ] Page replacement algorithms (LRU, FIFO, Clock)
    - [ ] Memory-mapped files
    - [ ] Buddy memory allocation
- [ ] **File systems**
    - [ ] Journaling file systems (ext3, ext4)
    - [ ] Network file systems (NFS, SMB)
    - [ ] File system encryption
    - [ ] Distributed file systems (HDFS, Ceph)

5. Networking

- [ ] **TCP/IP stack**
    - [ ] OSI model layers
    - [ ] TCP vs. UDP
- [ ] **HTTP protocol**
    - [ ] Request methods (GET, POST, etc.)
    - [ ] Status codes
- [ ] **DNS (Domain Name System)**
    - [ ] Resolution process
- [ ] **Routing algorithms**
    - [ ] Shortest Path algorithms (Dijkstra's, Bellman-Ford)

6. Databases

- [ ] **Relational databases (SQL)**
    - **[ ] Normalization forms**
        - [ ] First Normal Form (1NF)
        - [ ] Second Normal Form (2NF)
        - [ ] Third Normal Form (3NF)
        - [ ] Boyce-Codd Normal Form (BCNF)
    - **[ ] Joins**
        - [ ] Inner joins
        - [ ] Outer joins (left, right, full)
        - [ ] Cross joins
    - **[ ] SQL Data Manipulation Language (DML)**
        - [ ] SELECT statement
        - [ ] INSERT statement
        - [ ] UPDATE statement
        - [ ] DELETE statement
        - [ ] MERGE statement
        - [ ] TRUNCATE statement
        - [ ] UPSERT operations
        - [ ] Explicit vs. Implicit transactions
        - [ ] Control-of-flow language (e.g., CASE, IF-ELSE)
    - **[ ] SQL Data Definition Language (DDL)**
        - [ ] CREATE statement

        - [ ] ALTER statement
        - [ ] DROP statement
    - **[ ] SQL Data Control Language (DCL)**
        - [ ] GRANT statement
        - [ ] REVOKE statement
    - **[ ] SQL Data Query Language (DQL)**
        - [ ] Subqueries
        - [ ] Aggregate functions (e.g., SUM, AVG, COUNT)
        - [ ] GROUP BY and HAVING clauses
        - [ ] Window functions
    - **[ ] Transaction Control**
        - [ ] COMMIT statement
        - [ ] ROLLBACK statement
    - **[ ] Database Constraints**
        - [ ] Primary key
        - [ ] Foreign key
        - [ ] Unique constraint
        - [ ] Check constraint
        - [ ] Default constraint
    - **[ ] Stored Procedures and Functions**
        - [ ] Creation
        - [ ] Execution
        - [ ] Parameters
    - **[ ] Triggers**
        - [ ] Types of triggers (BEFORE, AFTER)
        - [ ] Trigger execution order
    - **[ ] Views**
        - [ ] Materialized views vs. regular views
        - [ ] Advantages and use cases
- [ ] **NoSQL databases**
    - [ ] Types
        - [ ] Document-based
        - [ ] Key-value
        - [ ] Column-family
        - [ ] Graph
    - [ ] Examples
        - [ ] MongoDB Interview Questions
        - [ ] Redis Interview Questions
- [ ] **ACID properties**
    - [ ] Atomicity
    - [ ] Consistency
    - [ ] Isolation
    - [ ] Durability
- [ ] **Indexing**
    - [ ] B-tree
    - [ ] B+ tree
    - [ ] Bitmap Indexing
- [ ] **Transactions**
    - [ ] ACID properties in transactions
    - [ ] Isolation levels
        - [ ] Read Uncommitted
        - [ ] Read Committed
        - [ ] Repeatable Read

        - [ ] Serializable
- [ ] **Database design and optimization**
    - [ ] Partitioning
    - [ ] Materialized views

7. Programming Concepts

- [ ] **Programming paradigms**
    - [ ] Imperative programming
    - [ ] Declarative programming
    - [ ] Functional programming
    - [ ] Object-oriented programming
    - [ ] Procedural programming
    - [ ] Event-driven programming
    - [ ] Aspect-oriented programming
- [ ] **Memory management**
    - [ ] Stack vs. Heap
    - [ ] Manual vs. Automatic memory management
    - [ ] Garbage collection algorithms (e.g., Mark and Sweep, Generational GC)
- [ ] **Concurrency**
    - [ ] Threads vs. Processes
    - [ ] Synchronization primitives (locks, mutexes, semaphores)
    - [ ] Thread safety
    - [ ] Deadlocks
    - [ ] Race conditions
    - [ ] Parallelism vs. Concurrency
    - [ ] Asynchronous programming
- [ ] **Error handling**
    - [ ] Exceptions vs. Error codes
    - [ ] Exception handling mechanisms
    - [ ] Exception safety
    - [ ] Error propagation
    - [ ] Error recovery
- [ ] **Functional programming**
    - [ ] Higher-order functions
    - [ ] Closures
    - [ ] Lambda expressions
    - [ ] Pure functions
    - [ ] Referential transparency
- [ ] **Object-oriented programming (OOP)**
    - [ ] Encapsulation
    - [ ] Inheritance
    - [ ] Polymorphism
    - [ ] Abstraction
    - [ ] Composition vs. Inheritance
    - [ ] Method overriding vs. Method overloading
- [ ] **Design patterns**
    - [ ] Creational patterns
    - [ ] Structural patterns
    - [ ] Behavioral patterns
- [ ] **Asynchronous programming**
    - [ ] Callbacks
    - [ ] Promises

    - [ ] Futures
    - [ ] Coroutines
- [ ] **Functional vs. Imperative vs. Declarative programming**
- [ ] **Type systems**
    - [ ] Static vs. Dynamic typing
    - [ ] Strong vs. Weak typing
    - [ ] Nominal vs. Structural typing
- [ ] **Lambda calculus**
- [ ] **Garbage collection**
    - [ ] Tracing vs. Reference counting
    - [ ] Generational GC
- [ ] **Regular expressions**
- [ ] **Memory layout**
    - [ ] Stack vs. Heap memory allocation
    - [ ] Data segment vs. Code segment
- [ ] **Recursion**
    - [ ] Tail recursion
    - [ ] Mutual recursion
    - [ ] Anonymous recursion
- [ ] **Virtual Memory**
    - [ ] Demand Paging
    - [ ] Page replacement algorithms (FIFO, LRU, Optimal)
- [ ] **Networking**
    - [ ] Sockets
    - [ ] Client-server architecture
    - [ ] Protocols (TCP, UDP)
    - [ ] Remote Procedure Call (RPC)

8. System Architecture

- [ ] **Client-server architecture**
    - [ ] Basics
    - [ ] Communication protocols
- [ ] **RESTful architecture**
    - [ ] Principles
    - [ ] Advantages
    - [ ] Constraints
- [ ] **Service-Oriented Architecture (SOA)**
    - [ ] Principles
    - [ ] Advantages
    - [ ] Challenges
- [ ] **Message Queuing**
    - [ ] Basics
    - [ ] Use cases
    - [ ] Implementations (e.g., RabbitMQ, Kafka)
- [ ] **Microservices**
    - [ ] Principles
    - [ ] Advantages
    - [ ] Challenges
- [ ] **Event-Driven Architecture (EDA)**
    - [ ] Basics
    - [ ] Components
    - [ ] Advantages
    - [ ] Implementations (e.g., Apache Kafka)

- [ ] **Layered Architecture**
  - [ ] Presentation layer
  - [ ] Business logic layer
  - [ ] Data access layer
  - [ ] Cross-cutting concerns layer
- [ ] **Caching strategies**
  - [ ] Cache aside
  - [ ] Write-through caching
  - [ ] Write-behind caching
  - [ ] Cache stampede prevention

9. Problem-solving and Coding

- [ ] **Problem-solving strategies**
  - [ ] Understand the problem
  - [ ] Break it down
  - [ ] Solve a simpler problem
  - [ ] Look for patterns
  - [ ] Make a plan
  - [ ] Implement the plan
  - [ ] Test your solution
- [ ] **Coding techniques**
  - [ ] Modular programming
  - [ ] Divide and conquer
  - [ ] Recursion
  - [ ] Dynamic programming
  - [ ] Greedy algorithms
  - [ ] Backtracking
  - [ ] Bit manipulation
  - [ ] Sliding window
  - [ ] Two pointers
  - [ ] Binary search
  - [ ] Fast and slow pointers
  - [ ] Hashing
- [ ] **Coding best practices**
  - [ ] Naming conventions
  - [ ] Code readability
  - [ ] Code reusability
  - [ ] Error handling
  - [ ] Testing
  - [ ] Version control (e.g., Git)
  - [ ] Code reviews
- [ ] **Time and space complexity analysis**
  - [ ] Big O notation
  - [ ] Big Omega notation
  - [ ] Big Theta notation
  - [ ] Space complexity analysis
- [ ] **Debugging**
  - [ ] Print debugging
  - [ ] Debugger tools
  - [ ] Rubber duck debugging
- [ ] **Optimization**
  - [ ] Algorithmic optimization
  - [ ] Space-time trade-offs
  - [ ] Profiling tools

Contributions

Contributions are welcome! If you have suggestions for improving the roadmap, additional resources to recommend, or want to fix any errors, feel free to open an issue or submit a pull request on https://github.com/aasthas2022/SDE-Interview-and-Prep-Roadmap

Disclaimer

This roadmap is intended as a general guideline and may not cover every aspect of SDE interviews. It's essential to supplement my preparation with additional resources and adapt based on individual needs and experiences.

Credits

This project is inspired by various interview preparation resources and the collective wisdom of the developer community.

aastha-shukla/

Thank You

linkedin.com/in/aastha-shukla/

aasthas2022@gmaill.com