JAVA ROADMAP

Stage 1. Introduction to JAVA

- Basic Syntax
- Data types, Variables
- OOPs, Interfaces, Classes
- Functions
- Conditionals
- Package and their working

Stage 2. Diving Deep In Java

- How Memory Mangement Works?
- How JVM Works?
- Threads and threadpool
- How Garbage Collection works and why it is needed?
- How Serialazion works and why it is needed?

Stage 3. IDEs, Build Tools and Frameworks

- IDEs
  - IntelliJ
  - Eclipse
  - Spring Tool Suite
  - VS Code
- Build Tools
  - Maven
  - Gradle
  - Ant
- Frameworks
  - Spring
  - Play
  - Quarkus
  - Spark
- Servers
  - Tomcat
  - JBoss
  - Websphere
  - Web logic

Stage 4. Testing and Logging tools

- Testing
  - Unit Testing → JUnit, TestNG
  - Integration Testing → Rest As, JMeter
  - Behaviour Testing → Cucumb, Cukes, J Behave
- Logging library - → Log4j, Log4j2, LogBack

Stge 5. Databases

- SQL → ACID, Joins, Indexes, Constraints.
- NoSQL → MongoDB
- ORM → JPA data, Hibernate, Spring D

Must Knows

- * Array list vs Vectors
- * Hashmaps Vs Hashtables
- * Optional Class and its use.
- * Lists vs Sets.
- * Stream APIs, Tree Set.

BACKEND ROADMAP

Step 1. Basics Backend Concept

week

- Choose any Language like (Python, Java, Rust, C#, PHP)
- Git / Github
- Learn Databases (MySQL, MariaDB, MSSQL, SQLite)
- Server Side / Client Side Caching
- CDN
- Redis / Memcached

Must Know-

1. How does the internet work?
2. Protocols
3. Hosting and Servers
4. DNS
5. Browsers and how they work?

Step 2. Apis and Their Working

3 Week

- Learn about Apis (CRUD)
- What are Restful Apis
- End point Hostings / Node Hosting
- Hashing Algos (MD5, SHA, crypt)
- Security Practices (HTTPS / CORS | CSP | SSL-TLS)
- Database Terminology (ORMs / ACID / N+1 / Normalisation)
- CI / CD Pipelines

DataBases-

Document DBs	(Mongo DB, Couch DB)
Key-Value	(Redis)
Realtime	(Firebase, Rethink)
Time Series	(Influx DB)
Column DBs	(Casandra, Base)
Graph DBs	(Neo4j)

Step 3. DataBase Terminologies-

2 months

- Database Scaling
- Indexations
- Sharding Strategy
- CAP theorem
- Schema Design.
- Design patterns
- Service Mesh.

* Mitigations

a) Gracefull
b) Throttling
c) Back Pressure

* Web Servers

a) Nginx
b) Apache
c) Caddy

Containerisatio

a) Kubernetes
b) Docker LXC

Visualisation

a) Message Brokers
b) RTDs (Web Sockets,
c) Graph QL

FRONTEND ROADMAP

Stage 1. Learn HTML / CSS

1-2 Weeks

- Try to Create Webpages
- Markups Understanding
- Semantic Codes
- Layouts Configurations -

Must Haves -

1. Web architecture
2. How Browser Works
3. DNS
4. Web Protocols
5. Working of Internet
6. Hosting

* This is the roadmap from basics
* Topics need to be cleared and practised thoroughly
* Time allocated can be changed as person to person

Stage 2. Learn Javascript

3-4 Weeks

- What is DOM?
- Web Apis (setTimeout, fetch()), DOM apis)
- Javascript Engines (V8 / Spider Monkey)
- Events Manipulations
- Throttling / Bubbling
- DOM Event Lifecycle
- Use of Callbacks, Promises, Callback hell, Hoistings etc.
- Thorow understanding of DOM, Console Elements
- Memory Allocation

Stage 3. Pick a Frame Work (React / Vue.js / Angular / Solid Js)

1 month

- Components / Lifecyle (Mounting, Updation, DeMounting, Deletion)
- Props
- Effects and Hooks
- State Mangement
- Why React ?
- What React solves.?
- Reconciliation
- Performance Optimations
- How and What are different mechanism of all frameworks.

DSA ROADMAP

Stage 1. Basics for Problem Solving

- → Mathematics
- → Time Complexities
- → Space Complexities
- → Language I/O operations (C++/JAVA/PY)

Stage 2. Introduction to Data Structures

- → Primitives (Int / Char / Long / Float)
- → Arrays
- → Strings
- → Stacks
- → Queues
- → LinkedList
- → Tree → Binary, Binary Search, AVL, B-
- → Graph → Directed / Undirected.

Arrays, Strings, Stacks, Queues } Linear Ones (30 Questions per topic)

* Advanced / Complex Data Structures (Not required)

- → Trie
- → Segment Tree
- → Fenwick Tree
- → DSU
- → Suffix Tree
- → B/ B+ Trees
- → Skip List

Stage 3. Optimisations Techniques

- → Brute Force (Layman Logic)
- → Back Tracking
- → Recursion
- → Dynamic Programming
- → Greedy
- → Divide and Conquer
- → Two - Pointer / Sliding Window

Stage 4 Algorithms

- → Searching (Linear, Binary, Ternary)
- → Sorting (Merge, Bubble, Insertion etc,
- → Tree Algos
  - → BFS / DFS
  - → Traversals (In, Pre, Post
- → Graph Algos
  - → Dijkstra
  - → Bellman Ford
  - → Prism's
  - → Kruskal's

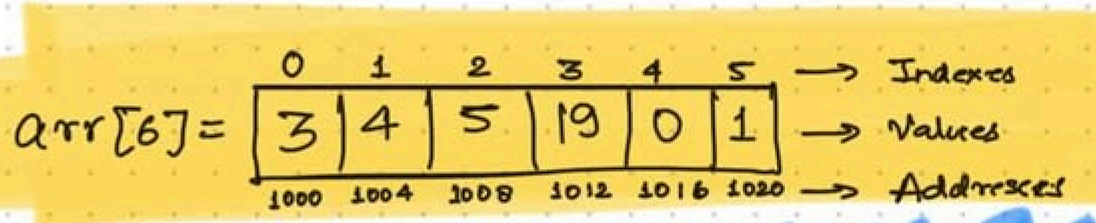
Practice Platforms -

1. Leetcode, Hackerrank, IB.
2. Codechef, Codeforces.

Key Pointers -

- * Practice Daily
- * Dont Go to Solution Directy
- * Clist.by (Use this website to stay updated about upcoming contests)
- * Easy → 15 - 30min
  Medium → 30 - 60 min
  Hard → 2 hr
  } This is the maximum time you can put On one question, before moving to editorials.
- * Participate in Contests
- * Dont try to remember the solution
- * Stay Consistent.

1. Arrays

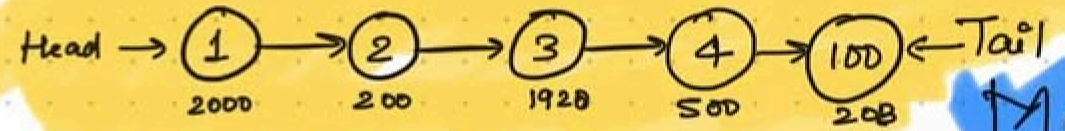


- Consecutive Memory allocation
- Insert Complexity → O(N)
- Accessing an Element Complexity → O(1)
- Topics to be covered related to Arrays.
  - Kadane's Algorithm
  - Two Pointer Algorithm
  - Dutch National Flag's Algorithm
  - Prefix Sum / Suffix Sum.
  - Linear Search / Binary Search.
  - Divide and Conquer.
  - Sliding Window
    - Fixed Size
    - Variable Size (Twopointer algo) →
  - Sub Array - Sub sequence
  - Precomputation

Most Asked Questions

- Min / Max of an Array.
- Duplicates / Occurrence of elements
- Sort 0s, 1s and 2s in an array.
- Majority Element
- Factorial of a Large number
- Peak Element
- Coinchange Problem
- Subarray with given sum or 0s
- Rain water Problem
- Stocks Buy / Sell
- Minimum Number of Jumps / Frog Jum
- Itna Kafi Hai !!!

2. Linked Lists

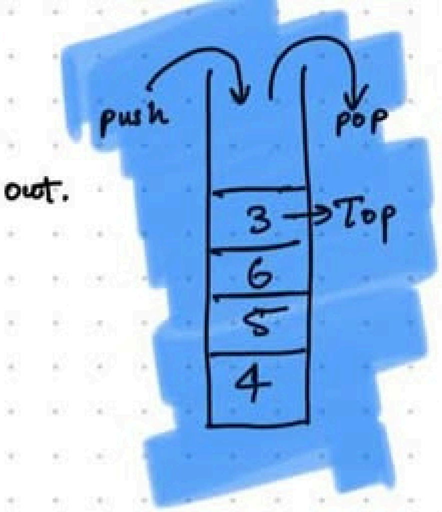


- Kuch ni hai is topic me.
- Node {data, next}
  - If doubly link list {data, next, prev}
  - ek raaz ki bat batau (Linked list hi tree hai)
  - If tree {data, childrens} (badme aate hai !!)
- Time Complexities
  - Insert / Delete O(1) (Only for Head) otherwise O(N) →
  - Access O(N)
- Topics you should Know about linked list
  - Two pointer
  - Slow fast pointer (Tortoise-Rabbit algo)
  - Bas yahi hai isme 🙂 !!

Most Common Asked Questio

- Find a loop
- Detect a middle element
- Pairwise swap of elements
- Reverse linked list in group of given si
- Multiply two number represented a
- In-place arrangment of elements
- Flattening a linked list
- Break the LL among an element.
- Rotate Doubly linked list by N nodes
- Delete without Head Pointer.

3. Stacks

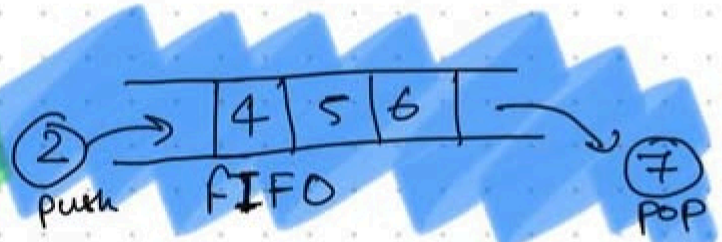


- Based on last in first out.
- Use case specific ds.
- very imp.
- Complexity
  - → push O(1)
  - → pop O(1)
  - → find O(N)
- Most Important topics related to Stacks
  - → Recursion
  - → Expression evaluation and Parsing
  - → Depth for Search (DFS)
  - → Undo / Redo Operations.
  - → Browser History
  - → Function Calls.
  - → Cache In memory (LRU/LFU)
  - → Browser Engines

Most asked Questions Related to Stacks

- → Paranthesis checker (({}))
- → Reverse a string using stack.
- → Postfix and Prefix amotations.
- → Delete amiddle from a stack.
- → Find next smaller of next greater element.
- → Implement two stacks in an array.
- → Maximum product of Indices of next greater in left and right.
- → Next greater frequency element.
- → Check if two expressions with brackets are same or not.
- → The Celebrity Problem ✓.
- → Range Queries based questions.
- → Largest area rectangle in a histogram.
- → Find min/max in every window of given size.
- → Print ancestors of a given binary tree.
- → Jitna Karlo Kam Hai !!!

4. Queue



- Based on First in First Out.
- Generally wherever you find scheduling something.
- Job-Schedulers.
- Push → O(1) Pop → O(1)
- Most Important topics related to Queues.
  - → Deques / Priority Queues
  - → Other functions in Ques provided by STL (peak, full)
  - → Wherever you can see the use of stacks, queues can be used.
  - → Sliding Window (Variable / Static)
  - → If a usecase is specific to finding something in a range →
  - → Advanced Queue (Priority Queues / Heaps)
  - → Whenever we need something max and min we use heaps.
  - → Breadth First Search / Depth First Search

Most asked Interview Question

- → Reverse first K elements of a Queue.
- → Level with maximum number of Nodes
- → Minimum Depth of a Binary Tree.
- → Implement Circular Queue / Deque.
- → Implement Stack using one/two Q
- → Design a queue to get maximum and
- → Reverse a queue using recursion
- → Flatten a multilevel linked list.
- → Flood Fill Algorithm. ✓
- → Shortest distance / Geek in Maz
- → Connect nodes at same level.
- → Implement LRU using Queue.

System Design

Stage 1. Get a Basic Understanding

- How web architecture Works?
- How Domains are working (DNS)?
- DB Scaling / Sharding
- Normalisation
- How Encryption works?
- Start learning about CDN (Push / POll)
- Latency Vs Throughput
- Availabilty Patterns

Stage 2. Diving deep in Concepts

- Caching types
- Communication terms
- Load Balancers
- Redis / Kafka Usage
- Design Notify me.
- Schedulers and Krons

Stage 3. Start Implementing.

- Topics as unlimited to learn
- URL Shortner
- Web Crawler
- Key-Value Hashing Mecha
- Rate - Limiter
- Job Scheduler
- Consistent Hashing
- Parking Lot **
- Notify Me **
- News feed System.
- Google Docs **
- Google Drive
- G map
- Search Autocomplete
- Keep Going.....