* **DSA**

-Abdul Bari

-Striver Sheet DSA

-Neetcode & LeetCode DSA questions

* **Data science**

-Certifications and knowledge

* **Cloud**

-halkirat Singh

* **Aptitude**

-Youtube

* **DevOps**

-youtube

* **Machine Learning**

-Deep Learning

-Coursera

-pratice

Here are advanced and challenging **CSE (Computer Science and Engineering)** exam questions, covering a range of topics relevant for high-paying jobs, including **software engineering, algorithms, machine learning, distributed systems, and computer architecture**. The questions are divided into theoretical and practical sections.

**Theoretical Questions**

**1. Data Structures & Algorithms**

1. **Dynamic Programming**:
   * Describe how you would solve the **Travelling Salesman Problem (TSP)** using dynamic programming. What is the time complexity of your solution?
   * Given an array of n integers, design an algorithm to find the **longest increasing subsequence**. Explain its time complexity.
2. **Graph Algorithms**:
   * Explain the difference between **Dijkstra's algorithm** and the **Bellman-Ford algorithm**. When would you prefer one over the other?
   * Design an algorithm to find all **strongly connected components (SCCs)** in a directed graph.
3. **Time Complexity**:
   * Explain the significance of **amortized analysis** in data structures like hash maps and dynamic arrays. Provide an example.

**2. Operating Systems**

1. Explain the differences between **processes** and **threads**. Why are threads more lightweight than processes?
2. What are the primary benefits of using a **microkernel** architecture over a **monolithic kernel**? Provide examples of systems implementing each.
3. Describe how **deadlock detection and prevention** mechanisms work. Can we always avoid deadlock?
4. Explain the concept of **paging** in memory management. How does it differ from **segmentation**?

**3. Databases**

1. **SQL**:
   * Write a query to find the second-highest salary in a table without using the LIMIT or TOP clause.
2. Explain the **ACID properties** of transactions. Why is it critical for high-frequency trading systems?
3. What is **sharding** in distributed databases? Discuss its benefits and challenges.
4. Describe how a **B+ tree** works for indexing in relational databases.

**4. Distributed Systems**

1. Explain the **CAP theorem** and provide examples of systems that prioritize availability over consistency.
2. How do **consensus algorithms** like **Raft** or **Paxos** ensure fault-tolerance in distributed systems?
3. Describe the role of a **load balancer** in a distributed web application.
4. What are **eventual consistency** and **strong consistency**? Provide use cases for both.

**5. Machine Learning & AI**

1. **Math Fundamentals**:
   * Derive the gradient descent update rule for a linear regression problem.
   * Explain the concept of **bias-variance tradeoff** in machine learning.
2. **Deep Learning**:
   * Describe the vanishing gradient problem. How do **ReLU** activations help mitigate this?
   * Explain the architecture of a **Transformer model** and its advantages over recurrent neural networks.
3. **Optimization**:
   * What is **stochastic gradient descent (SGD)**, and how does it differ from batch gradient descent?
   * Discuss the importance of **learning rate scheduling**.

**6. Computer Networks**

1. Explain how **TCP congestion control** works. What are slow start, congestion avoidance, and fast recovery?
2. Compare **IPv4** and **IPv6**. Why was IPv6 introduced, and what are its key benefits?
3. What is the difference between **packet switching** and **circuit switching**? Provide real-world examples.
4. How does a **DNS server** resolve a domain name? Describe the recursive query process.

**7. Computer Architecture**

1. Describe how **pipelines** improve CPU performance. What are **hazards**, and how are they handled?
2. Explain the differences between **RISC** and **CISC** architectures.
3. Discuss how **cache memory** improves system performance. What are **cache replacement policies** like LRU or LFU?
4. What is **virtual memory**, and how is it implemented using **paging**?

**Practical Coding Questions**

**1. Algorithms**

1. Write a function to perform **merge sort** on a list of integers.
2. Implement a program to find the **shortest path** between two nodes in an unweighted graph using **Breadth-First Search (BFS)**.
3. Given an array, implement an algorithm to find the **maximum sum subarray** using **Kadane's algorithm**.

**2. System Design**

1. Design a URL shortening service (e.g., **Bitly**).
   * Explain the database schema, hashing techniques for URL shortening, and methods for handling collisions.
2. Design a **ride-hailing system** (e.g., **Uber**).
   * Discuss how you would handle matching drivers and riders, geolocation updates, and scaling for millions of users.
3. Design a **distributed file storage system** (e.g., **Google Drive**).
   * Explain how files are uploaded, replicated, and retrieved in a fault-tolerant manner.

**3. Database Queries**

1. Write a SQL query to find the employees who earn more than their manager.
2. Write a query to find the **Nth highest salary** in a table using a subquery.

**4. Machine Learning Implementation**

1. Implement **linear regression** from scratch in Python.
2. Write a Python program to train a neural network using a framework like TensorFlow or PyTorch.
3. Build a program to implement **k-means clustering**.

**5. Debugging and Problem-Solving**

1. Debug the following code snippet and identify why it fails on edge cases:
2. public int divide(int a, int b) {
3. return a / b;
4. }
5. A multithreaded program occasionally crashes with a **deadlock**. Explain how you would identify the root cause using tools like a thread dump.

**Tips:**

* These questions are designed to test both theoretical knowledge and practical skills.
* To prepare, focus on understanding **foundational concepts** and their real-world applications.
* Be ready to explain trade-offs, scalability, and optimization in system design problems.

Good luck! 🚀

**Popular Interview Problems:**

•Reverse An Array

• Print Longest Palindromic Subsequence

• Largest Subarray With O Sum

• Valid Parentheses

•Rod Cutting Problem Bottom View Of Binary Tree Remove Loop In Linked List

• Minimum Platforms

•Reverse Level Order Traversal

• Count Inversions