Difficulty Level **Question** **Hint**
Easy (20)
1. Reverse a string Use the `split`, `reverse`, and `join` methods.
2. Check if a number is even or odd Use the modulus operator `%`.
3. Find the largest number in an array Use `Math.max` and `apply` or the spread operator.
4. Count the number of vowels in a string Create a set of vowels and count occurrences using `forEach` or `filter`.
5. Find the factorial of a number Use a loop or recursion to multiply numbers from 1 to `n`.
6. Check if a string is a palindrome Compare the string with its reverse.
7. Sum all numbers in an array Use `reduce` method to accumulate the sum.
8. Find the average of numbers in an array Divide the sum of elements by the length of the array.
9. Remove duplicates from an array Use a `Set` or filter with `indexOf`.
10. Find the length of a string Use the `length` property of the string.
11. Check if a string contains a specific substring Use the `includes` method.
12. Convert a string to lowercase Use the `toLowerCase` method.
13. Convert a string to uppercase Use the `toUpperCase` method.
14. Concatenate two strings Use the `+` operator or `concat` method.
15. Find the maximum value in an object Use `Object.values` combined with `Math.max`.
16. Check if an array is sorted Compare the array to its sorted version.
17. Count the number of words in a string Split the string by spaces and count the resulting array's length.
18. Find the index of a specific element in an array Use the `indexOf` method.
19. Replace all occurrences of a substring in a string Use the `replace` method with a global regular expression.
20. Remove whitespace from the beginning and end of a string Use the `trim` method.
Medium (30) 1. Merge two sorted arrays Use two pointers to compare and merge arrays.

2. Find the median of an array Sort the array and find the middle value or average of the two middle values.
3. Implement a basic calculator Use switch cases or if-else to handle different operations.
4. Check if two strings are anagrams Sort both strings and compare them.
5. Rotate an array to the right by `k` positions Use array slicing and concatenation.
6. Remove all falsy values from an array Use `filter` method with a Boolean callback.
7. Implement a queue using two stacks Use one stack for enqueue and another for dequeue operations.
8. Find the longest substring without repeating characters Use a sliding window technique or a hash set.
9. Check if a number is a prime number Check divisibility up to the square root of the number.
10. Find all pairs in an array that sum up to a specific value Use a hash set to track seen values or nested loops.
11. Find the intersection of two arrays Use a `Set` or `filter` method to find common elements.
12. Check if a string is a valid number Use regular expressions or `isNaN` to validate.
13. Calculate the nth Fibonacci number Use recursion or iteration with memoization.
14. Flatten a nested array Use recursion or `flat` method if supported.
15. Convert a number to its binary representation Use `toString(2)` method for binary conversion.
16. Find the common elements between two arrays Use `filter` with a `Set` or hash map.
17. Implement a basic todo list Use arrays to store tasks and methods to add, remove, or display them.
18. Find the longest word in a string Split the string and use `reduce` to find the longest word.
19. Calculate the power of a number Use `Math.pow` method or exponentiation operator `**`.
20. Check if a string contains only digits Use a regular expression or `isNaN` with `parseInt`.
21. Find the unique elements in an array Use a `Set` or `filter` method.
22. Implement a simple debounce function Use `setTimeout` to delay function execution.
23. Count the number of occurrences of each character in a string Use an object or `Map` to count characters.
24. Find the first non-repeating character in a string Use a hash map to count characters and find the first with count `1`.
25. Convert a binary string to a decimal number Use `parseInt` with base `2`.
26. Implement a simple event emitter Use an object to store event listeners and trigger them.

27. Determine the longest sequence of consecutive numbers in an array Use a loop to find and track the length of sequences.
28. Check if a string is a valid email address Use a regular expression to validate email format.
29. Count the number of elements in an array greater than a specific value Use `filter` method with a comparison.
30. Implement a basic throttle function Use `setTimeout` to limit the rate of function execution.
Advanced (50)
1. Implement a linked list Define nodes and methods for insertion, deletion, and traversal.
2. Find the shortest path in a graph Use Dijkstra's or A* algorithm.
3. Solve the N-Queens problem Use backtracking to place queens on a chessboard.
4. Implement a binary search tree Define nodes and methods for insertion, search, and traversal.
5. Find all permutations of a string Use recursion or iterative algorithms to generate permutations.
6. Perform a depth-first search on a tree Use recursion or stack to explore nodes.
7. Implement a priority queue Use a heap or priority queue data structure.
8. Solve the knapsack problem Use dynamic programming to solve the problem.
9. Find the longest common subsequence of two strings Use dynamic programming to compare subsequences.
10. Implement a hash table Define hash functions and handle collisions.
11. Solve the traveling salesman problem Use dynamic programming or approximation algorithms.
12. Implement a red-black tree Define nodes and balancing methods for the tree.
13. Solve the Sudoku puzzle Use backtracking to fill in the grid.
14. Perform a breadth-first search on a graph Use a queue to explore nodes level by level.
15. Implement a LRU (Least Recently Used) cache Use a doubly linked list and a hash map to maintain order.
16. Solve the maximum subarray problem Use Kadane's algorithm to find the maximum sum subarray.
17. Calculate the minimum spanning tree Use Kruskal's or Prim's algorithm.
18. Find the shortest path between two nodes in a graph Use Dijkstra's or Bellman-Ford algorithm.
19. Implement a Trie (prefix tree) Define nodes and methods for insertion and search.
20. Solve the Longest Increasing Subsequence problem Use dynamic programming to find the increasing subsequence.

21. Find all subsets of a set Use recursion or bit manipulation to generate subsets.
Advanced (50) 22. Implement a segment tree Use a tree structure to store range-based data for efficient queries.
23. Solve the rod cutting problem Use dynamic programming to maximize the profit from cutting.
24. Implement an AVL tree Use rotations to maintain balance after insertion and deletion.
25. Find the strongly connected components in a graph Use Kosaraju's or Tarjan's algorithm.
26. Implement the Rabin-Karp string matching algorithm Use hashing to find patterns in the string.
27. Solve the convex hull problem Use Graham's scan or Jarvis's march algorithm.
28. Implement a machine learning algorithm from scratch Start with simple algorithms like linear regression or k-nearest neighbors.
29. Solve the sequence alignment problem Use dynamic programming techniques like Needleman-Wunsch or Smith-Waterman.
30. Implement a real-time data processing system Use streams or event-driven architectures to handle real-time data.
31. Solve the minimal spanning tree problem using Kruskal's algorithm Sort edges and use a union-find structure to build the tree.
32. Implement a real-time collaborative editing system Use operational transformations or CRDTs (Conflict-free Replicated Data Types).
33. Solve the multidimensional knapsack problem Use dynamic programming with multiple constraints.
34. Implement an efficient search algorithm for large datasets Explore algorithms like binary search, B-trees, or Bloom filters.
35. Solve the cycle detection problem in a graph Use depth-first search or Union-Find to detect cycles.
36. Implement a distributed computing framework Consider using message passing or MapReduce for distributed processing.
37. Solve the optimal task scheduling problem Use dynamic programming or greedy approaches depending on constraints.
38. Implement a real-time recommendation system Use collaborative filtering, content-based filtering, or a hybrid approach.
39. Solve the large-scale optimization problem Use optimization algorithms like simulated annealing, genetic algorithms, or gradient descent.
40. Implement a data compression algorithm Start with simple techniques like Huffman coding or LZW compression.
41. Solve the data clustering problem Use algorithms like K-means, DBSCAN, or hierarchical clustering.
42. Implement a robust error correction algorithm Explore algorithms like Hamming codes, Reed-Solomon codes, or convolutional codes.
43. Solve the multi-agent system problem Use game theory, distributed algorithms, or reinforcement learning.

44. Implement a predictive analytics model Use statistical methods or machine learning models like regression or decision trees.
45. Solve the large-scale data integration problem Use ETL processes, schema matching, or data lakes for integration.
46. Implement a distributed ledger technology Explore blockchain technologies or consensus algorithms like Paxos or Raft.
47. Solve the network flow problem using advanced algorithms Use the Ford-Fulkerson or Edmonds-Karp algorithm.
48. Implement a decentralized system Use peer-to-peer networks, distributed hash tables, or consensus protocols.
49. Solve the probabilistic data structure problem Explore data structures like Bloom filters, Count-Min Sketch, or HyperLogLog.
50. Implement an artificial intelligence algorithm Start with simple AI algorithms like A* search, minimax, or reinforcement learning.
Very Advanced (100) 1. Solve the complex network analysis problem Use graph theory and network analysis techniques like centrality, community detection, or clustering coefficients.
2. Implement an advanced cryptography algorithm Explore elliptic curve cryptography, RSA, or AES encryption algorithms.
3. Solve the distributed database problem Use consensus protocols, sharding, or NoSQL databases for large-scale data management.
4. Implement a large-scale graph processing system Consider frameworks like Apache Giraph, GraphX, or Neo4j for graph processing.
5. Solve the big data analysis problem Use Hadoop, Spark, or data warehousing solutions to manage and analyze big data.
6. Implement a scalable web application Use microservices, load balancing, and distributed databases to handle high traffic.
7. Solve the time-series forecasting problem Use ARIMA, exponential smoothing, or machine learning models like LSTM.
8. Implement a sophisticated search engine Use indexing, crawling, and ranking algorithms like PageRank or BM25.
9. Solve the resource allocation problem Use linear programming, greedy algorithms, or dynamic programming to optimize resource use.
10. Implement a real-time streaming data system Use Apache Kafka, Spark Streaming, or Flink for handling real-time data streams.
11. Solve the robust machine learning problem Address overfitting, handle noisy data, and implement ensemble methods like Random Forests or Gradient Boosting.
12. Implement a complex event processing system Use rules-based systems or event-driven architectures to handle complex events.
13. Solve the advanced data warehousing problem Use ETL processes, OLAP cubes, or data lake architectures for large-scale data storage and querying.
14. Implement a high-performance computing system Use parallel processing, distributed computing, or GPU acceleration for intensive computations.

15. Solve the distributed algorithm problem Explore algorithms like distributed consensus, leader election, or distributed sorting.
16. Implement a scalable cloud infrastructure Use cloud orchestration tools like Kubernetes, Terraform, or AWS to manage resources.
17. Solve the secure communication problem Implement protocols like SSL/TLS, PGP, or secure APIs for encrypted communication.
18. Implement an advanced data mining algorithm Use algorithms like Apriori for association rule learning, or SVMs for classification tasks.
19. Solve the complex system simulation problem Use simulation frameworks or languages like SimPy, AnyLogic, or MATLAB for modeling complex systems.
20. Implement a high-availability system Use redundant systems, failover strategies, or load balancing for uptime.
21. Solve the advanced anomaly detection problem Use statistical methods, machine learning, or deep learning for detecting anomalies.
22. Implement a large-scale machine learning pipeline Use distributed frameworks like TensorFlow, PyTorch, or Apache Spark MLlib.
23. Solve the real-time data visualization problem Use tools like D3.js, Plotly, or real-time dashboards to visualize streaming data.
24. Implement a complex web crawler Handle dynamic content, rate limits, and large-scale data extraction using a robust crawler framework.
25. Solve the advanced recommendation engine problem Use collaborative filtering, matrix factorization, or deep learning to improve recommendations.
26. Implement a sophisticated data integration system Use schema matching, ETL processes, or APIs to integrate data from multiple sources.
27. Solve the high-dimensional data analysis problem Use dimensionality reduction techniques like PCA, t-SNE, or LDA for analysis.
28. Implement a scalable search index Use inverted indexes, term frequency-inverse document frequency (TF-IDF), or Elasticsearch for efficient search.
29. Solve the large-scale event processing problem Use event-driven architectures or complex event processing engines for handling events at scale.
30. Implement a robust data security system Use encryption, secure authentication, and access control mechanisms for protecting data.
31. Solve the distributed transaction management problem Use two-phase commit, distributed locking, or eventual consistency for managing transactions across systems.
32. Implement a high-performance data retrieval system Use caching, indexing, or in-memory databases for fast data access.
33. Solve the advanced pattern recognition problem Use machine learning models, neural networks, or image processing techniques for pattern recognition.
34. Implement a real-time analytics platform Use streaming data platforms, real-time databases, and visualization tools for live analytics.
35. Solve the large-scale text analysis problem Use natural language processing (NLP), text mining, or sentiment analysis techniques for text analysis.
Certainly! Here’s the continuation and completion of the table with hints included:

26. Implement an advanced recommendation algorithm Use collaborative filtering, matrix factorization, or neural networks for improving recommendations.
27. Solve the high-dimensional data clustering problem Use dimensionality reduction techniques like PCA before applying clustering algorithms.
28. Implement a secure multi-party computation protocol Research cryptographic protocols that allow computation on encrypted data.
29. Solve the computational biology problem Use algorithms for sequence alignment, phylogenetic trees, or protein structure prediction.
30. Implement a neural network from scratch Understand backpropagation and gradient descent to train your network.
31. Solve the large-scale distributed machine learning problem Implement distributed training using frameworks like TensorFlow or PyTorch on multiple nodes.
32. Implement a blockchain consensus algorithm Study algorithms like Proof of Work, Proof of Stake, or Practical Byzantine Fault Tolerance (PBFT).
33. Solve the advanced graph traversal problem Explore algorithms for traversing large or complex graphs, like Bidirectional Search or Iterative Deepening.
34. Implement a data lake architecture Use a combination of storage and processing frameworks to handle and query large datasets efficiently.
35. Solve the problem of real-time fraud detection Use anomaly detection models or real-time data analysis tools to identify fraudulent activities.
36. Implement an efficient document retrieval system Use inverted indexing, ranking algorithms, and caching for quick retrieval of relevant documents.
37. Solve the multi-armed bandit problem Research the trade-off between exploration and exploitation in decision-making algorithms.
38. Implement a complex scheduling algorithm Explore scheduling algorithms like Round Robin, Priority Scheduling, or Multi-Level Queue Scheduling.
39. Solve the real-time traffic prediction problem Use machine learning models like time-series forecasting or deep learning for predicting traffic patterns.
40. Implement a distributed hash table (DHT) Research consistent hashing and peer-to-peer systems for distributed data storage and retrieval.
41. Solve the problem of optimizing neural network architecture Experiment with hyperparameter tuning, network pruning, or neural architecture search algorithms.
42. Implement a secure voting protocol Study cryptographic protocols that ensure privacy and integrity in electronic voting systems.
43. Solve the problem of developing a large-scale recommendation system Implement and optimize algorithms like collaborative filtering and content-based filtering at scale.
44. Implement a predictive maintenance system Use IoT data, machine learning, and time-series analysis to predict equipment failures before they happen.
45. Solve the problem of data integration across heterogeneous systems Use data mapping, schema matching, and ETL processes to integrate data from various sources.
46. Implement a high-performance key-value store Research and implement in-memory databases like Redis or LevelDB for fast key-value storage.
47. Solve the multi-objective optimization problem Use algorithms like genetic algorithms or particle swarm optimization to handle multiple objectives.

48. Implement a decentralized identity management system Study blockchain-based identity systems or decentralized authentication protocols like OAuth 2.0.
49. Solve the problem of building a secure multi-tenant cloud environment Use virtualization, containerization, and strong access control mechanisms for security.
50. Implement an algorithm for detecting communities in large-scale social networks Explore algorithms like Girvan-Newman, Louvain, or Infomap for detecting communities in networks.
51. Solve the problem of large-scale natural language processing Use distributed computing and advanced NLP models like BERT, GPT, or transformers for large datasets.
52. Implement an efficient distributed file system Study systems like Hadoop's HDFS or Google File System for storing and processing large files across multiple nodes.
53. Solve the problem of optimizing large-scale parallel computing Use task scheduling algorithms, load balancing, and data partitioning for efficient parallel computation.
54. Implement a real-time stock trading system Combine data streaming, real-time analytics, and low-latency trading algorithms for financial markets.
55. Solve the problem of secure data sharing in distributed systems Implement cryptographic protocols for secure data exchange and storage in distributed environments.
56. Implement a robust fraud detection system Use machine learning, anomaly detection, and real-time data analysis to detect fraudulent activities.
57. Solve the problem of building a scalable microservices architecture Use service discovery, API gateways, and container orchestration for managing microservices at scale.
58. Implement a high-availability distributed system Use techniques like replication, failover, and quorum-based systems to ensure system reliability and availability.
59. Solve the problem of large-scale video streaming Optimize content delivery networks (CDNs), video compression, and buffering strategies for scalable streaming.
60. Implement an advanced AI-driven chatbot Use natural language understanding (NLU), dialogue management, and reinforcement learning for intelligent responses.
61. Solve the problem of real-time multiplayer game synchronization Implement efficient networking protocols, state synchronization, and latency compensation techniques.
62. Implement a secure end-to-end encryption protocol for messaging apps Research and implement encryption protocols like Signal Protocol for secure messaging.
63. Solve the problem of building a decentralized marketplace Use blockchain technology, smart contracts, and decentralized storage for a secure marketplace platform.
64. Implement a distributed AI model training system Use federated learning, data parallelism, and model compression for training AI models across multiple devices.
65. Solve the problem of building a real-time financial analytics platform Use streaming data, machine learning models, and real-time visualization for financial market analysis.
66. Implement a robust IoT device management system Use MQTT, CoAP, and device shadowing techniques for managing large fleets of IoT devices.

67. Solve the problem of optimizing distributed databases for large-scale transactions Use sharding, indexing, and replication strategies to optimize distributed databases for high-volume transactions.
68. Implement a high-performance neural network inference engine Research and implement optimizations like quantization, pruning, and hardware acceleration for fast inference.
69. Solve the problem of scaling blockchain networks Use sharding, layer-2 solutions, or alternative consensus mechanisms to scale blockchain networks.
70. Implement a secure and efficient smart contract platform Use formal verification, gas optimization, and secure coding practices to develop reliable smart contracts.
71. Solve the problem of large-scale real-time fraud detection Combine anomaly detection, machine learning, and streaming analytics for scalable fraud detection systems.
72. Implement an AI-powered recommendation engine for e-commerce Use collaborative filtering, deep learning, and contextual data for personalized product recommendations.
73. Solve the problem of building a decentralized social media platform Use blockchain, IPFS, and decentralized identity for a secure and private social media platform.
74. Implement a scalable real-time chat application Use WebSockets, message queues, and distributed databases for scalable real-time messaging.
75. Solve the problem of real-time video analytics Use computer vision, edge computing, and streaming data for analyzing video feeds in real-time.
76. Implement a large-scale image classification system Use convolutional neural networks (CNNs) and distributed training for large-scale image classification.
77. Solve the problem of building a secure multi-cloud infrastructure Use cloud security best practices, identity and access management, and encryption for securing multi-cloud deployments.
78. Implement a decentralized finance (DeFi) protocol Research and develop smart contracts, tokenomics, and decentralized exchanges (DEXs) for DeFi applications.
79. Solve the problem of building a scalable content delivery network (CDN) Use caching, load balancing, and edge computing for efficient content distribution.
80. Implement a secure and scalable voting system for large-scale elections Use blockchain, cryptographic protocols, and secure identity verification for an election voting system.
81. Solve the problem of building a scalable recommendation system Implement and optimize recommendation algorithms, handling big data and providing personalized suggestions.
82. Implement an AI-driven predictive maintenance system Use machine learning, IoT data, and time-series analysis for predicting equipment failures before they occur.
83. Solve the problem of building a scalable decentralized storage solution Use blockchain, distributed hash tables, and secure encryption for decentralized data storage.
84. Implement a large-scale recommendation engine for real-time personalization Use collaborative filtering, deep learning models, and A/B testing to optimize real-time recommendations.
85. Solve the problem of securing distributed AI models Use techniques like differential privacy, federated learning, and homomorphic encryption for secure AI models.

86. Implement a multi-cloud deployment strategy for high availability Use cloud-agnostic tools, CI/CD pipelines, and disaster recovery planning for multi-cloud deployments.
87. Solve the problem of optimizing deep learning models for edge devices Use model quantization, pruning, and efficient architectures like MobileNet for deployment on edge devices.
88. Implement a decentralized AI system Combine blockchain, federated learning, and privacy-preserving techniques for decentralized AI solutions.
89. Solve the problem of building a large-scale distributed search engine Implement distributed indexing, ranking algorithms, and efficient query processing for scalable search systems.
90. Implement a secure and scalable IoT ecosystem Use secure communication protocols, device authentication, and data encryption for a scalable IoT ecosystem.
91. Solve the problem of building a distributed ledger for secure financial transactions Research and implement consensus algorithms, transaction validation, and cryptographic security for ledgers.
92. Implement an AI-driven fraud detection system for financial institutions Use advanced machine learning models, anomaly detection, and real-time data analysis for fraud detection.
93. Solve the problem of building a large-scale autonomous vehicle control system Combine sensor fusion, real-time decision-making algorithms, and advanced control systems for autonomous vehicles.
94. Implement a secure end-to-end encryption system for cloud storage Use advanced encryption protocols, key management, and secure access controls for cloud data security.
95. Solve the problem of optimizing neural networks for large-scale image processing Use distributed computing, GPU acceleration, and data parallelism for efficient image processing at scale.
96. Implement a scalable and secure distributed consensus protocol Research Byzantine Fault Tolerance, Raft, or Paxos algorithms for achieving consensus in distributed systems.
97. Solve the problem of building a scalable real-time analytics platform Use data streaming, distributed processing, and real-time visualization tools for scalable analytics solutions.
98. Implement an AI-powered content recommendation system Use deep learning, natural language processing, and user behavior analysis for personalized content recommendations.
99. Solve the problem of optimizing large-scale distributed databases for high availability Use techniques like replication, sharding, and consensus algorithms to ensure database availability.
100. Implement a scalable AI-driven customer support chatbot Use natural language understanding, sentiment analysis, and real-time learning for responsive customer support.

This table completes the 200 JavaScript coding problems categorized by difficulty level, along with hints to guide you through solving each problem. These problems should help you significantly improve your problem-solving skills and understanding of advanced programming concepts.8