nHere’s a list of some complex data structures, including both classic ones and a few specialized structures used in advanced computing applications:

**1. Trees**

* **Binary Tree**: Each node has up to two children.
* **Binary Search Tree (BST)**: A binary tree with ordered nodes for efficient searching.
* **AVL Tree**: A self-balancing binary search tree.
* **Red-Black Tree**: Another self-balancing binary search tree with specific balancing rules.
* **Segment Tree**: Useful for range query operations, such as finding sums over intervals.
* **Fenwick Tree (Binary Indexed Tree)**: Useful for prefix sums and range queries.
* **B-tree**: Common in databases and filesystems, where nodes have multiple children.
* **Trie (Prefix Tree)**: Efficient for prefix-based searching, often used in dictionaries.
* **Suffix Tree**: Useful for string matching problems.
* **Merkle Tree**: Used in cryptographic applications for data verification.

**2. Graphs**

* **Adjacency Matrix/Adjacency List**: Ways to represent graphs in memory.
* **Directed Acyclic Graph (DAG)**: Used in dependency resolution, scheduling, and versioning.
* **Weighted Graphs**: Where edges carry weights, used in shortest-path algorithms like Dijkstra's.
* **Flow Network**: A graph with capacities on edges, useful in network flow problems.
* **Sparse/Dense Graphs**: Based on the number of edges relative to the number of vertices.
* **Planar Graph**: A graph that can be drawn on a plane without crossing edges.

**3. Heaps**

* **Binary Heap**: Often used to implement priority queues.
* **Min-Heap/Max-Heap**: Binary heaps where each parent is either less than or greater than its children.
* **Fibonacci Heap**: Optimized for decrease-key operations, used in algorithms like Dijkstra’s.
* **Binomial Heap**: Efficient for merging heaps.

**4. Hash-Based Structures**

* **Hash Table**: Key-value storage for fast lookup.
* **Hash Map**: A variant of the hash table often found in programming languages.
* **Hash Set**: Used for storing unique values with fast insertion and lookup.
* **Count-Min Sketch**: Used for approximate frequency counting.
* **Bloom Filter**: Probabilistic data structure for set membership testing.

**5. Arrays and Lists**

* **Dynamic Array**: Arrays that resize themselves, like Python’s lists or Java’s ArrayList.
* **Linked List**: A series of nodes, where each node points to the next (or previous).
* **Doubly Linked List**: Nodes have links to both the next and previous nodes.
* **Skip List**: A layered, probabilistic list for faster search times.

**6. Queues and Deques**

* **Queue**: A First-In-First-Out (FIFO) data structure.
* **Deque (Double-Ended Queue)**: Supports insertion and deletion at both ends.
* **Priority Queue**: A queue where elements have priorities; higher priority items are dequeued first.
* **Circular Queue**: A queue in which the end connects back to the beginning.

**7. Advanced Lists and Stacks**

* **Stack**: A Last-In-First-Out (LIFO) structure.
* **Rope**: An efficient data structure for storing and manipulating long strings.
* **Zipper List**: Used in functional programming for efficient editing of lists.

**8. Specialized Data Structures**

* **Union-Find (Disjoint Set)**: Useful for handling dynamic connectivity problems.
* **Bloom Filter**: Probabilistic data structure for checking membership.
* **Sparse Table**: Used for range query problems where array elements do not change.
* **Suffix Array**: Useful for string searching problems.
* **Voronoi Diagram**: A way to partition space into regions based on distance to points.
* **Convex Hull**: Useful in computational geometry for determining shape outlines.

**9. Spatial and Geometric Data Structures**

* **Quad Tree**: A tree used to partition 2D space.
* **KD-Tree**: A tree used for partitioning N-dimensional space.
* **R-Tree**: Often used for spatial access methods, like indexing multi-dimensional information.
* **Bounding Volume Hierarchy (BVH)**: Used in graphics and collision detection.

**10. Probabilistic and Approximate Data Structures**

* **Skip List**: A probabilistic alternative to balanced trees.
* **Bloom Filter**: For set membership testing with false positives.
* **HyperLogLog**: Used to approximate the number of distinct elements.
* **Count-Min Sketch**: Used for frequency estimation in streaming data.

**11. Specialized Application Data Structures**

* **Blockchain**: Used to maintain a decentralized ledger.
* **Dancing Links**: Used in exact cover problems like Sudoku solvers.
* **Expression Tree**: Used to represent mathematical expressions, common in compilers.

To make this list easier to follow, I'll organize it from foundational to advanced, helping you build a solid understanding before tackling more complex topics. Here’s an ordered approach:

**1. Basic Programming and Concepts**

* **Python Programming (4 videos)**: Learn the basics of Python if you're using it as your primary language.
* **Object-Oriented Programming (OOPs) in C++ (1 video)**: Covers OOP principles which are essential for understanding advanced data structures.
* **Linked List (15 videos)**: An introduction to a basic yet fundamental data structure, key for understanding more complex structures.
* **Stack & Queue (20 videos)**: Learn these linear structures next; they’re the basis for many algorithms.
* **Hashing Full Course (6 videos)**: Covers efficient data retrieval techniques and basic hashing principles.

**2. Intermediate Data Structures**

* **Binary Trees and Trees (24 videos)**: Trees are foundational for many advanced structures, so get comfortable here.
* **Graph (37 videos)**: Graph theory knowledge is crucial for advanced algorithms in many areas.
* **Heap (15 videos)**: Heaps are essential for priority queue operations and efficient sorting.
* **Segment Tree (3 videos)**: Useful in scenarios with range queries, common in competitive programming.
* **Trie (8 videos)**: Important for solving string-related problems efficiently.
* **Advanced Data Structures (8 videos)**: Explore more specialized structures, including balanced trees or self-balancing trees.

**3. Algorithms**

* **Sorting Algorithms (7 videos)**: Covers a range of sorting algorithms, foundational for understanding algorithm efficiency.
* **Searching Algorithms (15 videos)**: Covers essential search algorithms.
* **Mathematical (4 videos)** and **Geometric Algorithms (4 videos)**: Helpful for specialized problem-solving in mathematics or geometry-related tasks.
* **Greedy Algorithms (1 video)**: Covers the greedy approach, a common problem-solving strategy.
* **Dynamic Programming Newbie to Expert (47 videos)**: Key for tackling complex problems with overlapping subproblems.
* **Graph Algorithms (37 videos)**: Focuses on graph-specific algorithms for pathfinding, spanning trees, etc.

**4. System and Memory Management**

* **Operating Systems (13 videos)**: Covers essential OS concepts, crucial for understanding memory, processes, and thread management.
* **Memory Management in Operating System (1 video)**: In-depth focus on how memory is managed.
* **Linux Tutorials (11 videos)**: Familiarize yourself with Linux, commonly used in programming environments.

**5. Specialized Topics**

* **Data Visualization and Analysis using Python (1 video)**: Useful if you are interested in data analytics or visualization.
* **Machine Learning Concepts (6 videos)**: A good foundation for ML if you want to dive into data science.
* **Image Processing using Python (8 videos)** and **Image Operations in Python (6 videos)**: Learn image processing techniques if working in computer vision.
* **Face Recognition (8 videos)**: Specifically for face recognition applications if you’re interested in computer vision.

**6. Interview Preparation and Challenges**

* **Programming Interview Questions (246 videos)**: Covers a wide range of interview-focused questions.
* **Top Amazon Interview Problems (20 videos)**: Prepares you for Amazon-specific interview questions.
* **30-days Leetcode Challenges (MAY, JUNE, APRIL)** and **August Challenge (9 videos)**: These month-long challenges help you get familiar with coding problems, focusing on common patterns and problem-solving.
* **Interview Tips (2 videos)**, **Interview Puzzles (5 videos)**, and **Placement Guide (14 videos)**: General interview prep and puzzle-solving tips.
* **Our Success Stories (12 videos)**: Motivational, can be viewed at any time.

Following this structure will ensure a logical flow from fundamental concepts to advanced problem-solving and interview preparation. Let me know if you’d like more details on any specific section!

Docker, GitHub, postman, rest api, Javascript, node.js, react, Java, spring boot, ajax,

 **Express.js**

 **MongoDB**

 **MySQL**

 **AWS (Amazon Web Services)**

 **Nginx**

 **Linux/Unix Commands**

 **CI/CD (Continuous Integration/Continuous Deployment)**

 **TypeScript**

 **Redux (React state management)**

 **HTML/CSS**

 **Jest (JavaScript testing framework)**

 **GraphQL**

 **Microservices Architecture**

 **Jenkins**

 **Kubernetes**

 **Terraform**

 **GitLab**

 **WebSockets**

 **JSON Web Tokens (JWT)**

 **OAuth2 (Authentication)**

 **Web Security (CSRF, XSS, etc.)**

 **Socket.IO**

 **RESTful API Design**

 **Swagger (API Documentation)**

Docker, kubernetes, GitHub, postman, rest api, microservices AWS, severless development, Javascript, node.js, react.js, redux, express.js,, Java, spring boot, ajax, jwt, web sockets Networking - Cisco certified POSTGRES, ORACLE Is - Linux, windows