1. **Question:** What is a data structure?

   Answer: A data structure is a way of organizing and storing data in a computer so that it can be accessed and modified efficiently. It defines the relationship between the data, how the data is stored, and the operations that can be performed on the data. Data structures are fundamental building blocks in computer science and are used in various algorithms and applications.

2. **Question:** What is an algorithm?

   Answer: An algorithm is a step-by-step procedure or method for solving a problem. It is a finite sequence of instructions that specify a sequence of operations to be executed in order to solve a problem or perform a task. Algorithms can be expressed in various forms, including natural language, pseudocode, flowcharts, and programming languages. They are essential in solving computational problems efficiently and are used in various domains such as computer science, mathematics, engineering, and more.

3. **Question:** What is the difference between an array and a linked list?

   Answer: An array is a sequential collection of elements stored in contiguous memory locations. Elements in an array are accessed using an index, and the size of the array is fixed at the time of declaration. In contrast, a linked list is a collection of elements (nodes) where each element points to the next one in the sequence through pointers. Linked lists do not require contiguous memory allocation, and their size can dynamically grow or shrink during program execution. Arrays have constant-time access to elements but may require resizing operations for dynamic sizes, while linked lists offer efficient insertion and deletion at any position but have slower access time due to traversal.

4. **Question:** What is the time complexity of searching in a sorted array using binary search?

   Answer: Binary search has a time complexity of O(log n) since it divides the search interval in half at each step, resulting in a logarithmic time complexity. This efficiency is achieved by exploiting the sorted nature of the array. Binary search repeatedly divides the search interval in half and compares the target value with the middle element, reducing the search space by half with each iteration until the target is found or the search interval becomes empty.

5. **Question:** What is the difference between a stack and a queue?

   Answer: A stack is a Last-In-First-Out (LIFO) data structure, meaning the last element added is the first one to be removed. It supports two main operations: push (to add an element to the top of the stack) and pop (to remove the top element from the stack). In contrast, a queue is a First-In-First-Out (FIFO) data structure, where the first element added is the first one to be removed. It supports two main operations: enqueue (to add an element to the rear of the queue) and dequeue (to remove the front element from the queue). Stacks are often used in function calls, expression evaluation, and backtracking algorithms, while queues are commonly used in scheduling, buffering, and breadth-first search algorithms.

6. **Question:** What is a hash table?

   Answer: A hash table is a data structure that stores key-value pairs. It uses a hash function to compute an index into an array of buckets or slots, from which the desired value can be found. Hash tables offer fast access to elements based on their keys, with average-case time complexity O(1) for insertion, deletion, and lookup operations. However, the performance of hash tables can degrade under certain conditions such as hash collisions, where two different keys hash to the same index. To handle collisions, various collision resolution techniques such as chaining or open addressing are employed.

7. **Question:** What is the difference between breadth-first search (BFS) and depth-first search (DFS)?

   Answer: BFS explores all the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level. It starts at the root node and explores all the nodes at the current depth level before moving to the next level. BFS uses a queue data structure to maintain the order of exploration. In contrast, DFS explores as far as possible along each branch before backtracking. It starts at the root node and explores as far as possible along each branch before backtracking to explore other branches. DFS can be implemented using recursion or a stack data structure. BFS is often used to find the shortest path in unweighted graphs, while DFS is useful for topological sorting, cycle detection, and solving puzzles.