Please answer the below Data Structure Questions:

1. Why do we need data structure?

We need it because we need efficient ways of organizing different data so that we can retrieve data efficiently.

1. List out the areas in which data structures are applied extensively?

It is applied to every aspect of programming because all programming application uses data. Thus, this applies to all industries.

1. What is an Array?

An array is a container object that holds a fixed number of values of a single type.

1. What is the difference between the Array and ArrayList?

* Array is static in size that is fixed length data structure. ArrayList is dynamic in size.
* ArrayList cannot contain primitive data types. It can only contains Object while Array can contain both primitive data types as well as objects.
* We can use iterator to iterate through ArrayList . We can use for loop or for each loop to iterate through array .
* Length of the ArrayList is provided by the size() method while Each array object has the length variable which returns the length of the array.
* We can insert elements into the arraylist object using the add() method while in array we insert elements using the assignment operator.

1. What is LinkedList?

A linked list is a linear data structure where each element is a separate object. Each element (we will call it a node) of a list is comprising of two items - the data and a reference to the next node.

1. What is queue?

Queue is an abstract data structure, somewhat similar to Stacks. Unlike stacks, a queue is open at both its ends. One end is always used to insert data (enqueue) and the other is used to remove data (dequeue). Queue follows First-In-First-Out.

1. What is stack?

A stack is a basic data structure that can be logically thought as linear structure represented by a real physical stack or pile, a structure where insertion and deletion of items takes place at one end called top of the stack.

1. What is FIFO and LIFO?

First-In-First-Out (the first element added to the queue will be the first one to be removed). Last-In-First-Out (last element added will be first to be removed).

1. What is the order of complexity?

It refers to a certain mathematical expression of the size of the input, and the algorithm finishes between two factors of it. Generally, the smaller the order of complexity of the program's underlying algorithm, the faster it will run and the better it will scale as the input gets larger.

1. What is the best case to search an element from an array?
2. What is the worst case to search an element from an array?
3. What is tree in data structure?

It is a widely used abstract data type that simulates a hierarchical tree structure, with a root value and subtrees of children with a parent node, represented as a set of linked nodes.

1. What is graph in data structure?

A graph is a pictorial representation of a set of objects where some pairs of objects are connected by links. The interconnected objects are represented by points termed as vertices, and the links that connect the vertices are called edges.

1. What is the difference between the HashTable and HashMap?

Hashtable is synchronized, whereas HashMap is not. This makes HashMap better for non-threaded applications, as unsynchronized Objects typically perform better than synchronized ones.

Hashtable does not allow null keys or values. HashMap allows one null key and any number of null values.

1. What are the major data structures used in the following areas: RDBMS, Network data model and Hierarchical data model.

RDBMS: array

Network data model: graph

Hierarchical data model: trees

1. How HashMap works in java?

HashMap maintains an array of buckets. Each bucket is a linkedlist of key value pairs encapsulated as Entry objects. This array of buckets is called table. Each node of the linked list is an instance of a private class called Entry

1. What is ArrayIndexOutOfBoundsException in java? When it occurs?

It indicates that an array has been accessed with an illegal index. The index is either negative or greater than or equal to the size of the array.

1. What are the different ways of copying an array into another array?

* Object.clone(): Object class provides clone() method and since array in java is also an Object, you can use this method to achieve full array copy. This method will not suit you if you want partial copy of the array.
* System.arraycopy(): System class arraycopy() is the best way to do partial copy of an array.
* Arrays.copyOf(): If you want to copy first few elements of an array or full copy of array, you can use this method.
* Arrays.copyOfRange(): If you want few elements of an array to be copied, where starting index is not 0, you can use this method to copy partial array.

1. What is difference between an array and a linked list?

In array, each element is independent, no connection with previous element or with its location. In Linked list, location or address of elements is stored in the link part of previous element/node. In array, no pointers are used like linked list so no need of extra space in memory for pointer.

1. What are DFS and BFS?

Depth-first search (DFS) and Breadth-first search (BFS) are both algorithms for traversing or searching tree or graph data structures. Depth-first search starts at the root (selecting some arbitrary node as the root in the case of a graph) and explores as far as possible along each branch before backtracking. Breadth-first search starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a 'search key') and explores the neighbor nodes first, before moving to the next level neighbors.

1. What is Recursion?

Recursion is a process in which a method calls itself continuously.

1. What is Big-(O)-notation?

Big O notation is used in Computer Science to describe the performance or complexity of an algorithm. Big O specifically describes the worst-case scenario, and can be used to describe the execution time required or the space used (e.g. in memory or on disk) by an algorithm.

1. Outlined different kind of sorting algorithm with time complexity?

Algorithm Time Complexity Space Complexity

Best Average Worst Worst

Quicksort Ω(n log(n)) Θ(n log(n)) O(n^2) O(log(n))

Mergesort Ω(n log(n)) Θ(n log(n)) O(n log(n)) O(n)

Timsort Ω(n) Θ(n log(n)) O(n log(n)) O(n)

Heapsort Ω(n log(n)) Θ(n log(n)) O(n log(n)) O(1)

Bubble Sort Ω(n) Θ(n^2) O(n^2) O(1)

Insertion Sort Ω(n) Θ(n^2) O(n^2) O(1)

Selection Sort Ω(n^2) Θ(n^2) O(n^2) O(1)

Tree Sort Ω(n log(n)) Θ(n log(n)) O(n^2) O(n)

Shell Sort Ω(n log(n)) Θ(n(log(n))^2) O(n(log(n))^2) O(1)

Bucket Sort Ω(n+k) Θ(n+k) O(n^2) O(n)

1. Draw a table with time complexity in which it will summarize all the data structure feature's advantages and disadvantages.
   1. In terms of Accessing element.
   2. In terms of Inserting element.
   3. In terms of Searching element.
   4. In terms of Removing element.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Data Structure | Time Complexity | |  |  |  |  |  |  | Space Complexity |
| Average | Worst |  |  | Worst |  |  |  |  |  |
| Access | Search | Insertion | Deletion | Access | Search | Insertion | Deletion |  |  |
| Array | T(1) | T(n) | T(n) | T(n) | O(1) | O(n) | O(n) | O(n) | O(n) |
| Stack | T(n) | T(n) | T(1) | T(1) | O(n) | O(n) | O(1) | O(1) | O(n) |
| Queue | T(n) | T(n) | T(1) | T(1) | O(n) | O(n) | O(1) | O(1) | O(n) |
| Singly-Linked List | T(n) | T(n) | T(1) | T(1) | O(n) | O(n) | O(1) | O(1) | O(n) |
| Doubly-Linked List | T(n) | T(n) | T(1) | T(1) | O(n) | O(n) | O(1) | O(1) | O(n) |
| Skip List | T(log(n)) | T(log(n)) | T(log(n)) | T(log(n)) | O(n) | O(n) | O(n) | O(n) | O(n log(n)) |
| Hash Table | N/A | T(1) | T(1) | T(1) | N/A | O(n) | O(n) | O(n) | O(n) |
| Binary Search Tree | T(log(n)) | T(log(n)) | T(log(n)) | T(log(n)) | O(n) | O(n) | O(n) | O(n) | O(n) |
| Cartesian Tree | N/A | T(log(n)) | T(log(n)) | T(log(n)) | N/A | O(n) | O(n) | O(n) | O(n) |
| B-Tree | T(log(n)) | T(log(n)) | T(log(n)) | T(log(n)) | O(log(n)) | O(log(n)) | O(log(n)) | O(log(n)) | O(n) |
| Red-Black Tree | T(log(n)) | T(log(n)) | T(log(n)) | T(log(n)) | O(log(n)) | O(log(n)) | O(log(n)) | O(log(n)) | O(n) |
| Splay Tree | N/A | T(log(n)) | T(log(n)) | T(log(n)) | N/A | O(log(n)) | O(log(n)) | O(log(n)) | O(n) |
| AVL Tree | T(log(n)) | T(log(n)) | T(log(n)) | T(log(n)) | O(log(n)) | O(log(n)) | O(log(n)) | O(log(n)) | O(n) |
| KD Tree | T(log(n)) | T(log(n)) | T(log(n)) | T(log(n)) | O(n) | O(n) | O(n) | O(n) | O(n) |