Frequently Ask Questions for DSAL Lab

1. Total how many assignments are there in DSAL Lab?

Ans:- 12

2. List out all assignments in DSAL Lab?

3. Explain regarding your Mini-project of DSAL Lab.

4. What are Linear Data Structure and Non-Linear Data Structure?

Ans:- In **linear data structure**, **data** elements are sequentially connected and each element is traversable through a single run. In **non-linear data structure**, **data** elements are hierarchically connected and are present at various levels. ... **Linear data structures** can be traversed completely in a single run.

5. List out various Linear Data Structure and Non-Linear Data Structure.

Ans:- Array, Queue, Stack, Linked **List** are **linear data structures**. Trees, graphs are **non-linear data structures**. ... A Tree is a collection **of** nodes where these nodes are arranged hierarchically and form a parent-child relationship. A Graph is a collection **of** a finite number **of** vertices and edges.

6. What is difference between tree and graph?

Ans:-

1	Graph is a non-linear data structure.	Tree is a non-linear data structure.
2	It is a collection of vertices/nodes and edges.	It is a collection of nodes and edges.
3	Each node can have any number of edges.	General trees consist of the nodes having any number of child nodes.

7. Terminology of Tree and Graph

Ans:- **Tree** is a non-linear data structure which organizes data in a hierarchical structure and this is a recursive **definition**. OR. A **tree** is a connected **graph** without any circuits. OR. If in a **graph**, there is one and only one path between every pair of vertices, then **graph** is called as a **tree**.

8. What is Binary Tree? List out properties of Binary Tree.

Ans:- A **binary tree** is a finite set of nodes that is either empty or consist a root node and two disjoint **binary trees** called the left subtree and the right subtree. In other words, a **binary tree** is a non-linear data structure in which each node has maximum of two child nodes. The **tree** connections can be called as branches.

9. List out applications of Binary tree?

Ans:- **Applications of Binary tree:**

- Implementing routing table in router.
- Data compression code.
- Implementation of Expression parsers and expression solvers.
- To solve database problem such as indexing.
- Expression evaluation.

10. What is Time and space complexity of Binary Tree?

Ans:- Therefore, searching in **binary search tree** has worst case **complexity** of O(n). In general, **time complexity** is O(h) where h is height of BST. ... Therefore, we need to traverse all elements (in order 3, 2, 1) to insert 0 which has worst case **complexity** of O(n). In general, **time complexity** is O(h).

11. What is Binary Search Tree? List out properties of Binary Search Tree.

Ans:- A **binary tree** is a finite set of nodes that is either empty or consist a root node and two disjoint **binary trees** called the left subtree and the right subtree. In other words, a **binary tree** is a non-linear data structure in which each node has maximum of two child nodes.

Binary Search Tree is a node-based binary tree data structure which has the following properties:

- The left subtree of a node contains only nodes with keys lesser than the node's key.
- The right subtree of a node contains only nodes with keys greater than the node's key.

12. What is Time and space complexity of Binary Search Tree?

Ans:- The **space complexity** of a **binary search tree** is O (n) O(n) O(n) in both the average and the worst cases.

13. List out operations which you can perform on BST.

Ans:- **Basic operations on a BST**

- Create: creates an empty tree.
- Insert: insert a node in **the** tree.
- Search: Searches for a node in **the** tree.
- Delete: deletes a node from **the** tree.
- Inorder: in-order traversal **of the** tree.
- Preorder: pre-order traversal **of the** tree.
- Postorder: post-order traversal **of the** tree

14. How we can perform insert operation in BST.

Ans:- **Insertion**

1. Go the root node of Tree, compare the value **to** be **inserted** with the current node value.

2. If the value **to** be **inserted** is smaller then or equal **to** the root node value, go **to** the left subtree, else go **to** the right subtree.
3. Compare the value again with the root node value of the subtree, and follow the step2 again.

15. How we can perform delete operation in BST.

Ans:- **Deleting a binary tree using the delete keyword in C++ program**

1. Write **a** class called Node.
2. Write **a** constructor function that accepts data for the node.
3. Write **a** destructor function. **Delete** the left node. **Delete** the right node. Print the current node data.
4. Initialize the **binary tree** with dummy data.
5. **Delete** the **binary** tress using the **delete** root statement.

16. List out applications of Binary Search tree?

Ans:- **Applications of BST**

1. BSTs are used for indexing and multi-level indexing.
2. They are also helpful to implement various **searching** algorithms.
3. It is helpful in maintaining a sorted stream of data.
4. TreeMap and TreeSet data structures are internally implemented using self-balancing BSTs.

17. What is OBST?

1. Ans:- An **optimal binary search tree** is a **binary search tree** for which the nodes are arranged on levels such that the **tree** cost is minimum. For the purpose of a better presentation of **optimal binary search trees**, we will consider "extended **binary search trees**", which have the keys stored at their internal nodes.

18. How we can find OBST in general?

Ans:- As we know that in binary search tree, the nodes in the left subtree have lesser value than the root node and the nodes in the right subtree have greater value than the root node. The frequency and key-value determine the overall cost of searching a node. ...

19. How we can find OBST using Dynamic Approach?

Ans:- In computer science, an **optimal binary search tree** (**Optimal** BST), sometimes called a weight-balanced **binary tree**, is a **binary search tree** which provides the smallest possible **search** time (or expected **search** time) for a given sequence of accesses (or access probabilities).

20. What is AVL tree?

Ans:- An **AVL tree** is another balanced binary search **tree**. Named after their inventors, Adelson-Velskii and Landis, they were the first dynamically balanced **trees** to be proposed. Like red-black **trees**, they are not perfectly balanced, but pairs of sub-**trees** differ in height by at most 1, maintaining an O(logn) search time.

21. What is Height Balance Tree?

Ans:- A **tree** whose subtrees differ in **height** by no more than one and the subtrees are **height-balanced**, too. An empty **tree** is **height-balanced**.

22. List out advantages of AVL Tree.

Ans:- **Advantages of AVL Trees**

- The height of the **AVL tree** is always balanced. The height never grows beyond log N, where N is the total number of nodes in the **tree**.
- It gives better search time complexity when compared to simple **Binary** Search **trees**.
- **AVL trees** have self-balancing capabilities.

23. How we can find out balance factor in AVL Tree?

Ans:- The **balance factor** of a node is the height of its right subtree minus the height of its left subtree and a node with a **balance factor** 1, 0, or -1 is considered **balanced.**

24. What is hashing?

Ans:- **Hashing** is simply passing some data through a formula that produces a result, called a **hash**. That **hash** is usually a string of characters and the **hashes** generated by a formula are always the same length, regardless of how much data you feed into it.

25. List out different hashing techniques.

Ans:- **Hashing Algorithms – A Closer Look at the Methods and Applications for Encryption**

- **Hash** Values. ...
- **Hash** Functions. ...
- Collision. ...
- The Division-remainder **Method**. ...
- The Folding **Method**. ...
- The Radix Transformation **Method**. ...
- The Digit Rearrangement **Method**. ...
- Applications in Encryption.

26. List out collision handling techniques. Explain it.

Ans:- **Collision Resolution Techniques** in data structure are the **techniques** used for handling **collision** in hashing. Separate Chaining is a **collision resolution technique** that handles **collision** by creating a linked list to the bucket of hash table for which **collision** occurs.

27. List out applications of hashing.

Ans:- **Some of these applications are listed below:**

- Message Digest.
- Password Verification.
- Data Structures(Programming Languages)
- Compiler Operation.

- Rabin-Karp Algortithm.
- Linking File name and path together.

28. What is ADT?

Ans:- abstract data type

In computer science, an abstract data type (**ADT**) is a mathematical model for data types. ... Formally, an **ADT** may be defined as a "class of objects whose logical behavior is defined by a set of values and a set of operations"; this is analogous to an algebraic structure in mathematics.

29. What is Skip List? How we can find element in skip list.

Ans:- **Searching an element in Skip list**

The **skip list** is **used to** store a sorted **list** of elements or data with a linked **list**. It allows the process of the elements or data to view efficiently. In one single step, it skips several elements of the entire **list**, which is why it is known as a **skip list**.

1. Key of next node is less than **search** key then **we** keep on moving forward on the same level.
2. Key of next node is greater than the key **to** be inserted then **we** store the pointer **to** current node i at update[i] and move **one** level down and continue our **search**.

30. What is TBT?

Ans:- In computing, a **threaded binary tree** is a **binary tree** variant that facilitates traversal in a particular order (often the same order already defined for the **tree**).

31. List out advantages of TBT.

Ans:- **Advantage**

1. By doing threading we neglect the recursive method of traversing a Tree , which makes use of stack and consumes many memory and time .

2. The node can keep record of its root .

32. List out applications of TBT.

Ans:- The idea of **threaded binary trees** is to make inorder traversal faster and do it without stack and without recursion. A **binary tree** is made **threaded** by making all right child pointers that would normally be NULL point to the inorder successor of the node (if it exists). There are two types of **threaded binary trees**.

33. List out different tree traversal.

Ans:- **Generally, we traverse a tree to search or locate a given item or key in the tree or to print all the values it contains.**

- In-order **Traversal**. In this **traversal** method, the left subtree is visited first, then the root and later the right sub-**tree**. ...
- Pre-order **Traversal**. ...
- Post-order **Traversal**.

34. Explain Pre-order, In-order, Post-order Traversal.

Ans:- In-order => In case of binary search trees (BST), **Inorder traversal** gives nodes in non-decreasing order. To get nodes of BST in non-increasing order, a variation of **Inorder traversal** where **Inorder traversal** s reversed can be used.

Preorder Traversal (current-left-right)— Visit the current node before visiting any nodes inside left or right subtrees. ... **Postorder Traversal** (left-right-current) — Visit the current node after visiting all the nodes of left and right subtrees.

35. List out different Graph Traversal and explain it.

Ans:- **Graph traversal** is a **technique used** for searching a vertex in a **graph**. The **graph traversal** is also **used** to decide the order of vertices is visited in the search process. A **graph traversal** finds the edges to be **used** in the search process without creating loops.

These are called the Breadth First Search and Depth First Search.

- Breadth First Search (BFS) The Breadth First Search (BFS) **traversal** is an algorithm, which is used to visit all **of** the nodes **of** a given **graph**. ...
- Algorithm. ...
- Depth First Search (DFS) ...
- Algorithm.

36. What is time and space complexity of DFS and BFS Traversal?

Ans:- The **space complexity** for **BFS** is O(w) where w is the maximum width of the tree. For **DFS**, which goes along a single 'branch' all the way down and uses a stack implementation, the height of the tree matters. The **space complexity** for **DFS** is O(h) where h is the maximum height of the tree.

37. Differentiate DFS and BFS Traversal.

Ans:- **BFS**(**Breadth First** Search) uses Queue data structure for finding the shortest path. **DFS**(**Depth First** Search) uses Stack data structure. ... **BFS** can be used to find single source shortest path in an unweighted graph, because in **BFS**, we reach a vertex with minimum number of edges from a source vertex.

38. Which technique we can use for representation of Graph?

Ans:- Two common ways **to represent graphs** on a computer are as an adjacency list or as an adjacency matrix. . Corresponding **to** each vertex is a list (either an array or linked list) of its neighbours.

39. What is Heap Data Structure ?

Ans:- A **heap** is a tree-based **data structure** in which all the nodes of the tree are in a specific order. For **example**, if is the parent node of , then the value of follows a specific order with respect to the value of and the same order will be followed across the tree.

40. Explain Min Heap and Max Heap.

Ans:- **Min-Heap** − Where the value of the root node is less than or equal to either of its children. **Max-Heap** − Where the value of the root node is greater than or equal to either of its children. Both trees are constructed using the same input and order of arrival.

41. What is file organization?

Ans:- **File organization** refers to the way data is stored in a **file**. **File organization** is very important because it determines the methods of access, efficiency, flexibility and storage devices to use. There are four methods of **organizing** files on a storage media.

42. What are the applications of file organization?

Ans:- **File organization** refers to the way data is stored in a **file**. **File organization** is very important because it determines the methods of access, efficiency, flexibility and storage devices to use. There are four methods of **organizing** files on a storage media.

43. List out different file organization technique.

Ans:- **Types of file organization are as follows:**

1. Sequential **file organization**.
2. Heap **file organization**.
3. Hash **file organization**.
4. B+ **file organization**.
5. Indexed sequential access **method** (ISAM)
6. Cluster **file organization**.

44. What is Sequential File?

Ans:- A **sequential file** contains records organized by the order in which they were entered. The order of the records is fixed. Records in **sequential files** can be read or written only **sequentially**. After you place a record into a **sequential file**, you cannot shorten, lengthen, or delete the record.

45. List out advantages and disadvantages of sequential file.

Ans:- **Advantages and Disadvantages**

- The **sequential file** organization is efficient and process faster for the large volume of data.
- It is a simple **file** organization compared to other available **file** organization methods.
- This method can be implemented using cheaper storage devices such as magnetic tapes.

46. What is Index Sequential File?

Ans:- Records in **indexed sequential files** are stored in the order that they are written to the disk. Records may be retrieved in **sequential** order or in random order using a numeric **index** to represent the record number in the **file**.

47. List out advantages and disadvantages of index sequential file.

Ans:- **Pros of sequential file** organization

It contains a fast and efficient method for the huge amount of data. In this method, **files** can be easily stored in cheaper storage mechanism like magnetic tapes. It is simple in design. It requires no much effort to store the data.

Cons of ISAM

This **method** requires extra space in the disk to store the **index** value. When the new records are inserted, then **these** files have to be reconstructed to maintain the sequence. When the record is deleted, then the space used by it needs to be released.

48. What is difference between Index Sequential File and Sequential

Ans:- Records can be read in **sequential** order just like in **sequential file** organization. Records can be accessed randomly if the primary key is known. **Index file** is used to get the address **of** a record and then the record is fetched from the data **file**.