Data Structures Concepts

What is a Data Structure?

A data structure is a way of organizing the data so that the data can be used efficiently. Different kinds of data structures are suited to different kinds of applications, and some are highly specialized to specific tasks.

Recursion:

Recursion is a programming technique where a function calls itself during its execution.

Base Case: The recursive function starts with a base case, which is a condition that determines when the function should stop recursing and return a result. It serves as the termination condition for the recursion.

Recursive Step: If the base case is not met, the function calls itself (recurses) with a modified version of the problem. By doing so, it solves a smaller or simpler subproblem of the original problem.

Example:

```
int factorial(int n) {

    if (n == 0 || n == 1) {

        return 1; // Base case: factorial of 0 or 1 is 1

    } else {
```

```
        return n * factorial(n - 1);

    }

}
```

What are linear and non linear data Structures?

- **Linear:** A data structure is said to be linear if its elements form a sequence or a linear list. Examples: Array. Linked List, Stacks and Queues
- **Non-Linear:** A data structure is said to be non-linear if traversal of nodes is nonlinear in nature. Example: Graph and Trees.

Data Structures:

- Array
- Linked List
- Stack
- Queue
- Tree
- Binary Tree
- Binary Search Tree (BST)

- Heap
- Hash Table
- Graph
- AVL Tree
- B-tree

What are the various operations that can be performed on different Data Structures?

● **Insertion** ? Add a new data item in the given collection of data items. ● **Deletion** ? Delete an existing data item from the given collection of data items. ● **Traversal** ? Access each data item exactly once so that it can be processed. ● **Searching** ? Find out the location of the data item if it exists in the given collection of data items.

● **Sorting** ? Arranging the data items in some order i.e. in ascending or descending order in case of numerical data and in dictionary order in case of alphanumeric data.

What is a Linked List?

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. In simple words, a linked list consists of nodes where each node contains a data field and a reference(link) to the next node in the list.

How is an Array different from Linked List?

● The size of the arrays is fixed, Linked Lists are Dynamic in size.

● Inserting and deleting a new element in an array of elements is expensive, Whereas both insertion and deletion can easily be done in Linked Lists.

● Random access is not allowed in Linked Listed.

- Extra memory space for a pointer is required with each element of the Linked list.
- Arrays have better cache locality that can make a pretty big difference in performance.

What is Stack and where it can be used?

Stack is a linear data structure which orders LIFO(Last In First Out) or FILO(First In Last Out) for accessing elements. Basic operations of stack are : Push, Pop , Peek.

Applications of Stack:

1. Infix to Postfix Conversion using Stack
2. Evaluation of Postfix Expression
3. Reverse a String using Stack
4. Implement two stacks in an array
5. Check for balanced parentheses in an expression

Implement stack using linked list

https://www.geeksforgeeks.org/implement-a-stack-using-singly-linked-list/

What is a Queue, how it is different from stack and how is it implemented?

Queue is a linear structure which follows the order is **F**irst **I**n **F**irst **O**ut (FIFO) to access elements. Mainly the following are basic operations on queue: **Enqueue, Dequeue**, **Front, Rear**

The difference between stacks and queues is in removing. In a stack we remove the item the most recently added; in a queue, we remove the item the least recently added. Both Queues and Stacks can be implemented using Arrays and Linked Lists.

Implement queue using linked list

https://www.geeksforgeeks.org/queue-linked-list-implementation/

What are Infix, prefix, Postfix notations?

- **Infix notation:** X **+** Y – Operators are written in-between their operands. This is the usual way we write expressions. An expression such as
  `A * ( B + C ) / D`
- **Postfix notation (also known as "Reverse Polish notation"):** X Y **+** Operators are written after their operands. The infix expression given above is equivalent to `A B C + * D/`

● **Prefix notation (also known as "Polish notation"):** + X Y Operators are written before their operands. The expressions given above are equivalent to
/ * A + B C D

Converting between these notations: Click here

What is a Linked List and What are its types?

A linked list is a linear data structure (like arrays) where each element is a separate object. Each element (that is node) of a list is comprising of two items – the data and a reference to the next node.Types of Linked List :

1. **Singly Linked List :** In this type of linked list, every node stores address or reference of next node in list and the last node has next address or reference as NULL. For example 1->2->3->4->NULL
2. **Doubly Linked List :** Here, here are two references associated with each node, One of the reference points to the next node and one to the previous node. Eg. NULL<-1<->2<->3->NULL
3. **Circular Linked List :** Circular linked list is a linked list where all nodes are connected to form a circle. There is no NULL at the end. A circular linked list can be a singly circular linked list or doubly circular linked list. Eg. 1->2->3->1 [The next pointer of last node is pointing to the first]

Tree Vs Graph:

The basis of Comparison	Graph	Tree
Definition	Graph is a non-linear data structure.	Tree is a non-linear data structure.
Structure	It is a collection of vertices/nodes and edges.	It is a collection of nodes and edges.
Structure cycle	A graph can be connected or disconnected, can have cycles or loops, and does not necessarily have a root node.	A tree is a type of graph that is connected, acyclic (meaning it has no cycles or loops), and has a single root node.
Edges	Each node can have any number of edges.	If there is n nodes then there would be n-1 number of edges

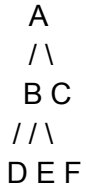
The basis of Comparison	Graph	Tree
Types of Edges	They can be directed or undirected	They are always directed
Root node	There is no unique node called root in graph.	There is a unique node called root(parent) node in trees.
Loop Formation	A cycle can be formed.	There will not be any cycle.
Traversal	For graph traversal, we use Breadth-First Search (BFS), and Depth-First Search (DFS).	We traverse a tree using in-order, pre-order, or post-order traversal methods.
Applications	For finding shortest path in networking graph is used.	For game trees, decision trees, the tree is used.

Difference Between BFS and DFS

Breadth First Search

BFS stands for **Breadth First Search** is a vertex based technique for finding the shortest path in a graph. It uses a Queue data structure which follows first in first out. In BFS, one vertex is selected at a time when it is visited and marked then its adjacent are visited and stored in the queue. It is slower than DFS.

Ex



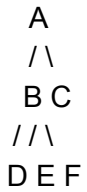
Output is:
A, B, C, D, E, F

Depth First Search

DFS stands for **Depth First Search** is an edge based technique. It uses the Stack data

structure, performs two stages, first visited vertices are pushed into stack and second if there are no vertices then visited vertices are popped.

Ex



Output is:

A, B, D, C, E, F

S.No	BFS	DFS
1	BFS stands for Breadth First Search	DFS stands for Depth First Search
2	BFS uses Queue data structure for finding the shortest path	DFS uses Stack data structure
3	BFS is more suitable for searching vertices which are closer to the given source	DFS is more suitable when there are solutions away from source
4	The Time complexity of BFS is O(V + E), where V stands for vertices and E stands for edges	The Time complexity of DFS is O(V + E), where V stands for vertices and E stands for edges

How to find cycle in graph?

Answer:

To find a cycle in a graph, you can use a depth-first search (DFS) algorithm. The basic idea is to traverse the graph and keep track of the visited vertices. During the traversal, if you encounter a visited vertex that is not the parent of the current vertex, then a cycle is present in the graph.

Binary Tree:

A tree whose elements have at most 2 children is called a binary tree. Since each element in a binary tree can have only 2 children, we typically name them the left and right child.

Binary Tree Representation in C++: A tree is represented by a pointer to the topmost node in a tree. If the tree is empty, then the value of the root is NULL.

A Tree node contains the following parts.

1. Data

2. Pointer to left child

3. Pointer to right child

Class node

```
{
   int data;
    node *left;
    node *right;
};
```

Binary Search Tree

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.
- The left and right subtree each must also be a binary search tree.
- There must be no duplicate nodes.

Sorting Algorithm:

https://www.geeksforgeeks.org/sorting-algorithms/