Lesson:

Generic Trees

Pre Requisites:

- Recursion
- Trees

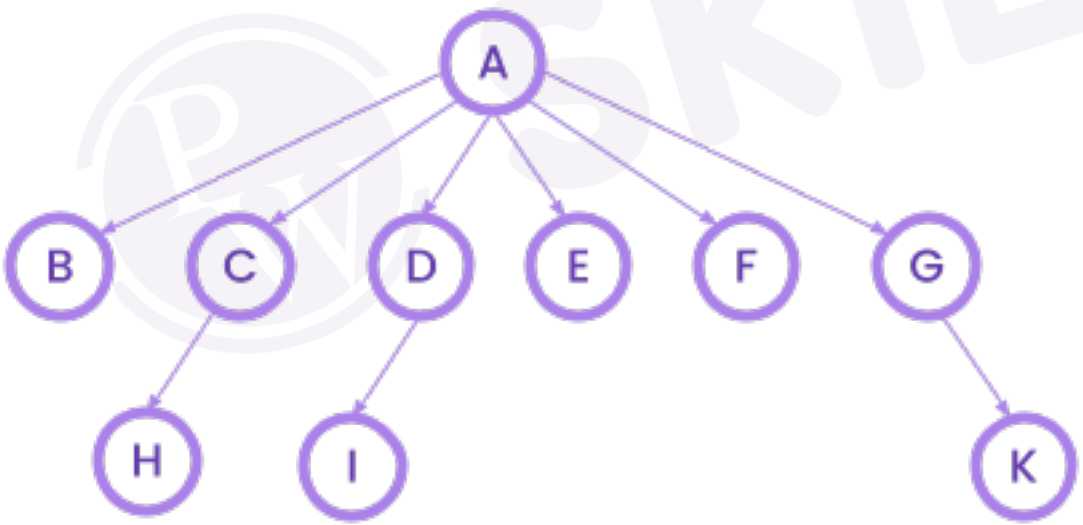
What we will learn:

- What are Generic Trees
- Representation
- Implementation
- Traversals
- Problems

TOPIC: What are Generic Trees

- Generic trees are a collection of nodes where each node is a data structure that consists of data and a list of references to its children(duplicate references are not allowed).
- The Generic trees have many children at every node.
- The number of nodes for each node is not known in advance.
- Also known as an N-ary tree.

TOPIC: Representation



- Here, we can see that A is the root node of the tree and all other nodes are descended from it.
- A tree will always be represented by its root node.
- Each node has different number of nodes. There is no restriction on the number of child nodes.
- Each node contains its value(data) and a list to represent its children.

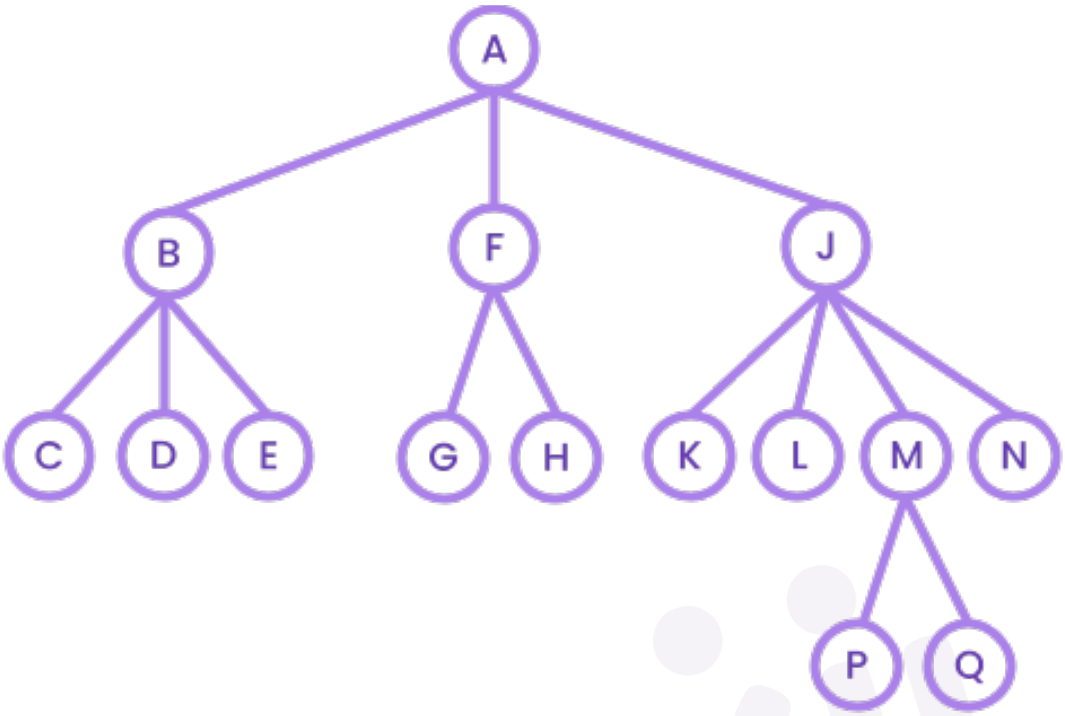
TOPIC: Implementation

Creating a node class

```
class Node{
    int data;
    ArrayList< Node> children = new ArrayList<>();
}
```

TOPIC: Traversals

For this, let us consider this generic tree:



• DFS

DFS means Depth First Search Traversals. We go down a single path until we come across a dead end and return to further nodes from there.
There are three types of traversing a tree in depth.
These are generally done using recursion.
Time complexity of any DFS traversal of a generic tree = O(n) where n is the size of the tree.
Space complexity = O(1), (not including the recursion stack space)

- **Preorder Traversal**

All the child nodes of a parent are traversed from left to right in a sequence. The order of approaching a node in this fashion is -: Node -> All children.
Preorder traversal for given tree is: A B C D E F G H J K L M P Q N

Code:
https://pastebin.com/tMvjtLSm

OUTPUT:

```
/Library/Java/JavaVirtualMachines/jdk-19.jdk/
Preorder traversal for the tree is
A B C D E F G H J K L M P Q N
Process finished with exit code 0
```

- **Inorder Traversal**

The inorder traversal of a generic tree is defined as visiting all the children except the last then the root and finally the last child.

The order of approaching a node in this fashion is - All children except last -> Node -> Last Child.
The inorder traversal for given tree: C D B E G F H A K L P M Q J N

Code:
https://pastebin.com/2YCnpHsd

OUTPUT:

```
/Library/Java/JavaVirtualMachines/jdk-19.jdk/
Preorder traversal for the tree is
C D B E G F H A K L P M Q J N
Process finished with exit code 0
```

- **Postorder Traversal**

In postorder traversal of a generic tree, all children of a node are traversed first in an in-depth fashion and then the node is traversed.
The order of approaching a node in this fashion is -: All children-> Node.
The postorder traversal for the given tree is: C D E B G H F K L P Q M N J A

Code:
https://pastebin.com/D5khHH4Y

OUTPUT:

```
/Library/Java/JavaVirtualMachines/jdk-19.jdk/
Preorder traversal for the tree is
C D E B G H F K L P Q M N J A
Process finished with exit code 0
```

- # BFS

BFS means Breadth First Search Traversal. Here, we traverse the tree level by level. For this tree, the BFS traversal would be:
A
B F J
C D E G H K L M N
P Q
For this, we do this iteratively with the help of a Queue data structure.
Time complexity = O(n) where n is the size of the tree.
Space complexity = O(n) as in the worst case there could be n nodes present in the queue at a time.

Code:
https://pastebin.com/WyPhCNy4

OUTPUT:

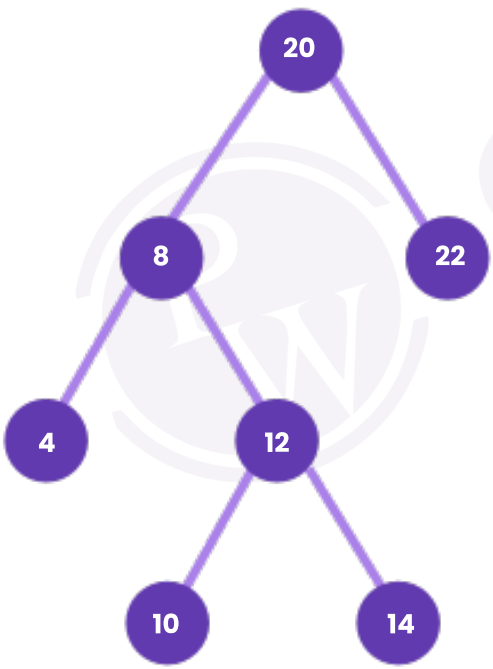
```
/Library/Java/JavaVirtualMachines/jdk-19.jdk/
Level order traversal for the tree is
A
B F J
C D E G H K L M N
P Q

Process finished with exit code 0
```

TOPIC: Problems

PROBLEM 1: NEXT GREATER ELEMENT IN N-ARY TREE

Q1. Given a generic tree and an integer x. Find and return the node with the next larger element in the tree i.e. find a node just greater than x. Return NULL if no node is present with value greater than x.

Input:



x = 10
Output:
12

Approach:

- The idea is to maintain a node pointer res, which will contain the final resultant node.
- Traverse the tree and check if the node data is greater than x. If so, then compare the node data with res data.

- If node data is greater than n and less than res data update res.

CODE:
https://pastebin.com/GA3jBvsw

OUTPUT:

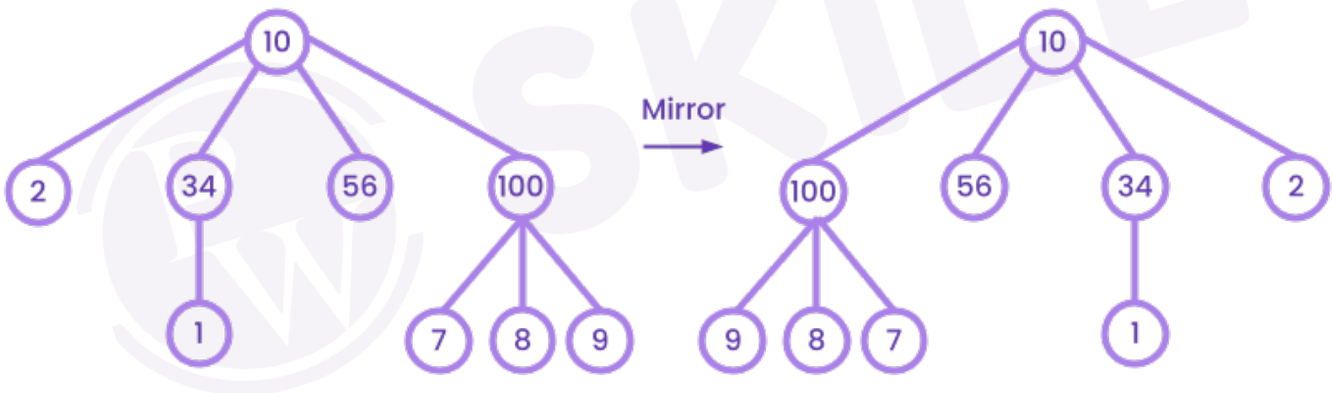
```
"C:\Program Files\Java\jdk-14.0.1\
12
Process finished with exit code 0
```

Time Complexity: O(n), where n is the number of nodes in the n-ary tree. The reason for this is that we traverse each node in the tree once, and for each node, we perform a constant amount of work to find its next larger element.

Space Complexity: O(h) where h is the height of the tree.

PROBLEM-2: MIRROR OF N-ARY TREE

Q2. Given a Tree where every node contains a variable number of children, convert the tree to its mirror.



Explanation:

- For every node, we first recur for all of its children and then reverse the array of children pointers. We can also do these steps in other ways, i.e., reverse array of children pointers first and then recur for children.
- We have done this via BFS traversal.

Code:
https://pastebin.com/tbhcvDA9

Output:

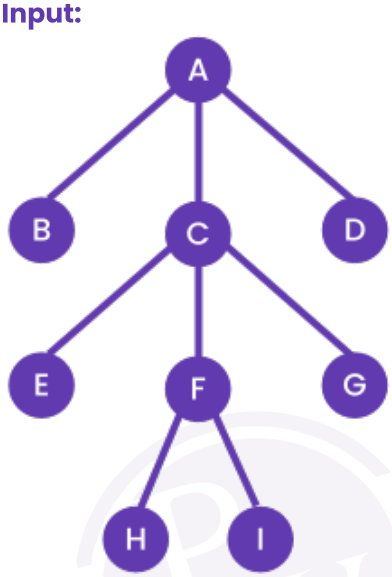
```
"C:\Program Files\Java\jdk-14.0.1\
10
100 56 34 2
9 8 7 1

Process finished with exit code 0
```

Time Complexity: O(n) where n is the number of nodes in the tree because we traverse all the nodes exactly once.
Space Complexity: O(n) as the maximum number of nodes that can be in the queue at a point of time is the maximum number of nodes that can be on the same level which is n.

PROBLEM 3: SERIALIZE DESERIALIZE AN N-ARY TREE

Q3. Serialize and Deserialize an N-ary tree.
Serialization is the process of converting an object into a format that can be stored or transmitted.
Deserialization is the process of converting the serialized string back into an object.
Serialization is to store tree in a file so that it can be later restored. The structure of tree must be maintained.
Deserialization is reading tree back from file.

Input:



Output:
A sample way of serializing: **node_val: num_children: child_1_val,child_2_val,...,child_n_val**

A: 3: B,C,D
B: 0:
C: 3: E,F,G
E: 0:
F: 0:
H: 0:
I: 0:
G: 0:
D: 0:

Approach:
- We can use pre-order traversal to serialize the N-ary tree. In pre-order traversal, we visit the root node first, then the left subtree, and then the right subtree.
- To deserialize the N-ary tree, we can split the serialized string into nodes, and then create a node for each line. For each node, we can split the line into its values, and then create child nodes for each child value.

Code:
https://pastebin.com/vZz5vPhi

Output:

```
"C:\Program Files\Java\jdk-14.0.1\
A:3:B,C,D
B:0:
C:3:E,F,G
E:0:
F:2:H,I
H:0:
I:0:
G:0:
D:0:
```

Time Complexity: O(n) where n is the number of nodes.
Space Complexity: O(h + n) , where h is height of tree and n is number of nodes.

Upcoming Class Teasers:

- Miscellaneous problems on Trees