COL106: Data Structures and Algorithms

Ragesh Jaiswal, IIT Delhi

- Linked List: A collection of nodes with linear ordering defined on them.
  - Each node holds an element and points to the next node in the order.
  - The first node in the ordering is called the head and the last is called the tail.
  - The tail points to a null reference.
  - The data structure is accessed using a reference to the head node.
- Give the mechanism for performing the following operations along with the running time:
  - Add an element at the beginning of the list: $O(1)$
  - Add an element at the end of the list: $O(n)$
  - Delete a particular node (given its reference): $O(n)$
  - Delete the first node containing element e: $O(n)$
  - Search element e in the linked list: $O(n)$
  - Remove the first element of the list: $O(1)$
  - Reverse the list:

- Linked List: A collection of nodes with linear ordering defined on them.
  - Each node holds an element and points to the next node in the order.
  - The first node in the ordering is called the head and the last is called the tail.
  - The tail points to a null reference.
  - The data structure is accessed using a reference to the head node.
- Question: Can you implement a stack using a Linked List? What is the running time of Push(e) and Pop() as per your implementation?

- Linked List: A collection of nodes with linear ordering defined on them.
    - Each node holds an element and points to the next node in the order.
    - The first node in the ordering is called the head and the last is called the tail.
    - The tail points to a null reference.
    - The data structure is accessed using a reference to the head node.
- What we just considered is more specifically called a singly linked list since we save the links in only one direction. Some natural extensions are:
    - Doubly Linked List
    - Circular Linked List

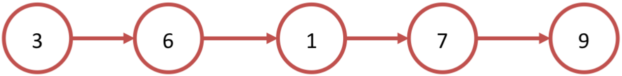
Data Structures: Tree

- Tree: An abstract data type that stores elements hierarchically.
    - It is a collection of nodes storing elements such that there is a parent-child relationship between nodes.
- Basic Definition:
    - Every non-empty tree has a special node called the root that has no parent.
    - Every node v except the root has a *unique* parent node and every node with parent v is the child of v.

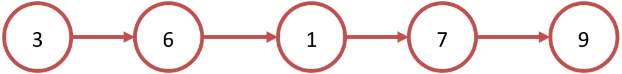
- Tree: An abstract data type that stores elements hierarchically.
  - It is a collection of nodes storing elements such that there is a parent-child relationship between nodes.
- Basic Definition:
  - Every non-empty tree has a special node called the root that has no parent.
  - Every node v except the root has a *unique* parent node and every node with parent v is the child of v.
- Is this a tree?

- Tree: An abstract data type that stores elements hierarchically.
  - It is a collection of nodes storing elements such that there is a parent-child relationship between nodes.
- Basic Definition:
  - Every non-empty tree has a special node called the root that has no parent.
  - Every node v except the root has a *unique* parent node and every node with parent v is the child of v.
- Is this a tree? Yes this is an empty-tree

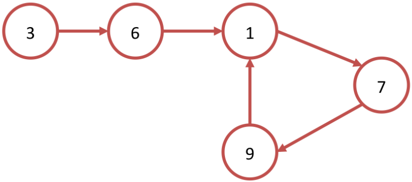
- Tree: An abstract data type that stores elements hierarchically.
  - It is a collection of nodes storing elements such that there is a parent-child relationship between nodes.
- Basic Definition:
  - Every non-empty tree has a special node called the root that has no parent.
  - Every node v except the root has a *unique* parent node and every node with parent v is the child of v.
- Is this a tree?



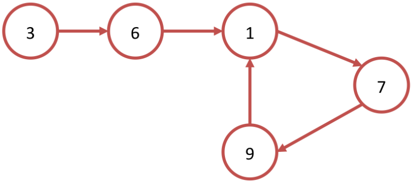
- Tree: An abstract data type that stores elements hierarchically.
  - It is a collection of nodes storing elements such that there is a parent-child relationship between nodes.
- Basic Definition:
  - Every non-empty tree has a special node called the root that has no parent.
  - Every node v except the root has a *unique* parent node and every node with parent v is the child of v.
- Is this a tree? Yes
- Which is the root node?



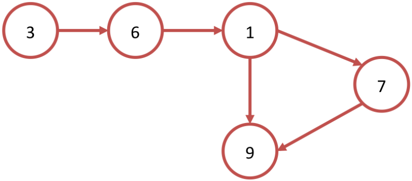
- Tree: An abstract data type that stores elements hierarchically.
  - It is a collection of nodes storing elements such that there is a parent-child relationship between nodes.
- Basic Definition:
  - Every non-empty tree has a special node called the root that has no parent.
  - Every node v except the root has a *unique* parent node and every node with parent v is the child of v.
- Is this a tree?



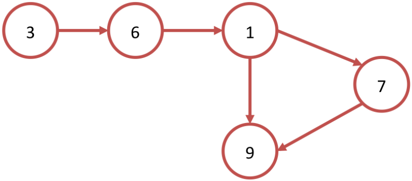
- Tree: An abstract data type that stores elements hierarchically.
  - It is a collection of nodes storing elements such that there is a parent-child relationship between nodes.
- Basic Definition:
  - Every non-empty tree has a special node called the root that has no parent.
  - Every node v except the root has a *unique* parent node and every node with parent v is the child of v.
- Is this a tree? No



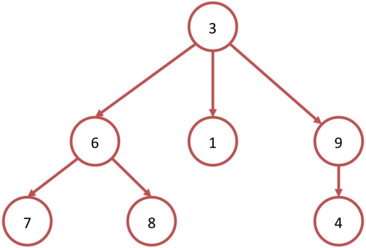
- Tree: An abstract data type that stores elements hierarchically.
  - It is a collection of nodes storing elements such that there is a parent-child relationship between nodes.
- Basic Definition:
  - Every non-empty tree has a special node called the root that has no parent.
  - Every node v except the root has a *unique* parent node and every node with parent v is the child of v.
- Is this a tree?



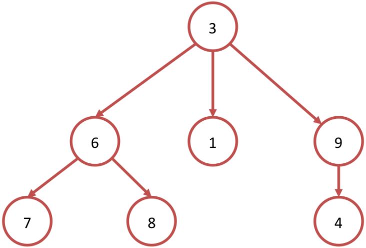
- Tree: An abstract data type that stores elements hierarchically.
    - It is a collection of nodes storing elements such that there is a parent-child relationship between nodes.
- Basic Definition:
    - Every non-empty tree has a special node called the root that has no parent.
    - Every node v except the root has a *unique* parent node and every node with parent v is the child of v.
- Is this a tree? No



- Tree: An abstract data type that stores elements hierarchically.
  - It is a collection of nodes storing elements such that there is a parent-child relationship between nodes.
- Basic Definition:
  - Every non-empty tree has a special node called the root that has no parent.
  - Every node v except the root has a *unique* parent node and every node with parent v is the child of v.
- Is this a tree?



- Tree: An abstract data type that stores elements hierarchically.
  - It is a collection of nodes storing elements such that there is a parent-child relationship between nodes.
- Basic Definition:
  - Every non-empty tree has a special node called the root that has no parent.
  - Every node v except the root has a *unique* parent node and every node with parent v is the child of v.
- Is this a tree? Yes



- Tree: An abstract data type that stores elements hierarchically.
  - It is a collection of nodes storing elements such that there is a parent-child relationship between nodes.
- Basic Definition:
  - Every non-empty tree has a special node called the root that has no parent.
  - Every node v except the root has a *unique* parent node and every node with parent v is the child of v.
- Terminology:
  - Two nodes that are children of the same parent are called siblings.
  - A node is called external node or a leaf node if it has no children.
  - A node is called internal if it has one or more children.
  - A node u is called an ancestor of another node v iff:
    - $u = v$, or
    - u is an ancestor of the parent of v.
  - A node v is a descendent of u iff u is an ancestor of v.
  - A subtree rooted at a node u is the tree consisting of all the descendents of u.

- Basic Definition:
  - Every non-empty tree has a special node called the root that has no parent.
  - Every node v except the root has a *unique* parent node and every node with parent v is the child of v.
- Terminology:
  - Two nodes that are children of the same parent are called siblings.
  - A node is called external node or a leaf node if it has no children.
  - A node is called internal if it has one or more children.
  - A node u is called an ancestor of another node v iff:
    - $u = v$, or
    - u is an ancestor of the parent of v.
  - A node v is a descendent of u iff u is an ancestor of v.
  - A subtree rooted at a node u is the tree consisting of all the descendents of u.

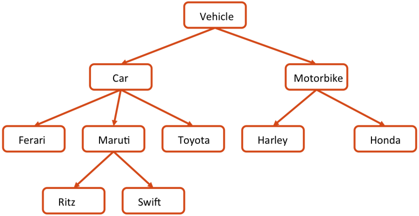


Figure : An example tree.

- Basic Definition:
  - Every non-empty tree has a special node called the root that has no parent.
  - Every node v except the root has a *unique* parent node and every node with parent v is the child of v.
- Terminology:
  - Two nodes that are children of the same parent are called siblings.
  - A node is called external node or a leaf node if it has no children.
  - A node is called internal if it has one or more children.
  - A node u is called an ancestor of another node v iff:
    - $u = v$, or
    - u is an ancestor of the parent of v.
  - A node v is a descendent of u iff u is an ancestor of v.
  - A subtree rooted at a node u is the tree consisting of all the descendents of u.

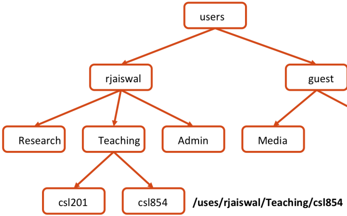
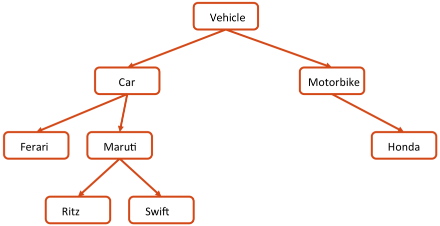


Figure : Another example tree.

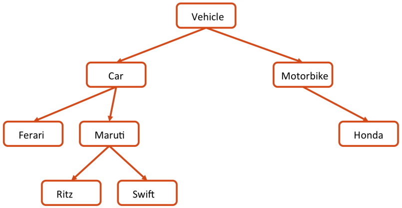
- Terminology:
  - Two nodes that are children of the same parent are called siblings.
  - A node is called external node or a leaf node if it has no children.
  - A node is called internal if it has one or more children.
  - A node u is called an ancestor of another node v iff:
    - $u = v$, or
    - u is an ancestor of the parent of v.
  - A node v is a descendent of u iff u is an ancestor of v.
  - A subtree rooted at a node u is the tree consisting of all the descendents of u.
- More Terms:
  - A tree is called ordered if there is a linear ordering defined on the children of all the nodes.
  - A binary tree is an ordered tree where each node has at most two children.
    - The children are called left child and right child. The left child precedes the right child in the ordering.
  - A binary tree is called proper or full if all the nodes have either 0 or 2 children.
  - A binary tree which is not proper is called improper.
  - An edge in a tree is a pair of nodes (u, v) such that u is the parent of v.
  - A path is a tree is a sequence of nodes such that any two consecutive nodes in the sequence form an edge.

- <u>More Terms</u>:
  - A tree is called ordered if there is a linear ordering defined on the children of all the nodes.
  - A binary tree is an ordered tree where each node has at most two children.
    - The children are called left child and right child. The left child precedes the right child in the ordering.
  - A binary tree is called proper or full if all the nodes have either 0 or 2 children.
  - A binary tree which is not proper is called improper.
  - An edge in a tree is a pair of nodes (u, v) such that u is the parent of v.
  - A path is a tree is a sequence of nodes such that any two consecutive nodes in the sequence form an edge.
- Is this a binary tree?
- Is this a proper binary tree?



- More Terms:
  - A tree is called ordered if there is a linear ordering defined on the children of all the nodes.
  - A binary tree is an ordered tree where each node has at most two children.
    - The children are called left child and right child. The left child precedes the right child in the ordering.
  - A binary tree is called proper or full if all the nodes have either 0 or 2 children.
  - A binary tree which is not proper is called improper.
  - An edge in a tree is a pair of nodes (u, v) such that u is the parent of v.
  - A path is a tree is a sequence of nodes such that any two consecutive nodes in the sequence form an edge.
  - The depth of a node v in a tree is the number of ancestors of v excluding v.
  - The height of a node v in a tree is the length of the longest path from v to a leaf node.

- More Terms:
  - A tree is called ordered if there is a linear ordering defined on the children of all the nodes.
  - A binary tree is an ordered tree where each node has at most two children.
    - The children are called left child and right child. The left child precedes the right child in the ordering.
  - A binary tree is called proper or full if all the nodes have either 0 or 2 children.
  - A binary tree which is not proper is called improper.
  - An edge in a tree is a pair of nodes (u, v) such that u is the parent of v.
  - A path is a tree is a sequence of nodes such that any two consecutive nodes in the sequence form an edge.
  - The depth of a node v in a tree is the number of ancestors of v excluding v.
  - The height of a node v in a tree is the length of the longest path from v to a leaf node.
- What is the depth and height of the node labeled "Maruti"?

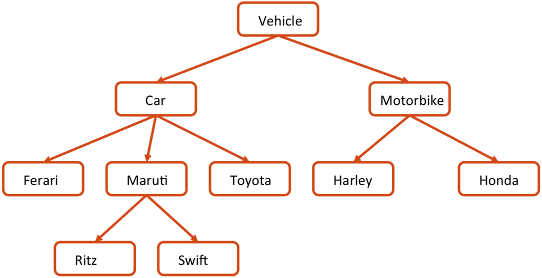


- One of the most basic operations on Trees is Tree Traversal.
- Here are two ways in which the nodes of a rooted tree may be traversed.
  - Pre-order Traversal: Visit the root *before* visiting the children.
  - Post-order Traversal: Visit the root *after* visiting the sub-trees.

Algorithm

`PreorderVisit(v)`
- Visit the root node v
- For every child u of v:
  - `PreorderVisit(u)`
	`PostorderVisit(v)`
	- For every child u of v:
	  - `PostorderVisit(u)`
	- Visit the root node v

- Question: Output the nodes visited while doing a pre-order traversal in the tree below.

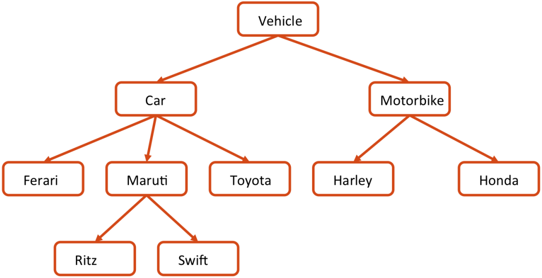


Algorithm

PreorderVisit(v)	PostorderVisit(v)

- Visit the root node v
- For every child u of v:
  - PreorderVisit(u)
	- For every child u of v:
	  - PostorderVisit(u)
	- Visit the root node v

- Question: Output the nodes visited while doing a post-order traversal in the tree below.



- A binary tree is a an ordered tree where all the nodes have at most two children.
- Each node is either is labeled as either being a left child or a right child.
- A binary tree is proper if each internal node has exactly two children or improper otherwise.

- A binary tree is a an ordered tree where all the nodes have at most two children.
- Each node is either is labeled as either being a left child or a right child.
- A binary tree is proper if each internal node has exactly two children or improper otherwise.
- For any given binary tree T, let:
  - N denote the number of nodes in the T.
  - L denote the number of external nodes (or leaves) in T.
  - I denote the height of internal nodes in T.
  - H denote the height of T. Height of a tree is equal to the height of the root.
- Show that:
  1. $H + 1 \leq N \leq 2^{H+1} - 1$
  2. $1 \leq L \leq 2^{H}$
  3. $H \leq I \leq 2^{H} - 1$
  4. $\log (N + 1) - 1 \leq H \leq N - 1$

- A binary tree is a an ordered tree where all the nodes have at most two children.
- Each node is either is labeled as either being a left child or a right child.
- A binary tree is proper if each internal node has exactly two children or improper otherwise.
- For any given binary tree T, let:
    - N denote the number of nodes in the T.
    - L denote the number of external nodes (or leaves) in T.
    - I denote the number of internal nodes in T.
    - H denote the height of T. Height of a tree is equal to the height of the root.
- Show that:
    1. $H + 1 \leq N \leq 2^{H+1} - 1$
    2. $1 \leq L \leq 2^{H}$
    3. $H \leq I \leq 2^{H} - 1$
    4. $\log{(N + 1)} - 1 \leq H \leq N - 1$

- A binary tree is a an ordered tree where all the nodes have at most two children.
- Each node is either is labeled as either being a left child or a right child.
- A binary tree is proper if each internal node has exactly two children or improper otherwise.
- For any given binary tree T, let:
  - N denote the number of nodes in the T.
  - L denote the number of external nodes (or leaves) in T.
  - I denote the number of internal nodes in T.
  - H denote the height of T. Height of a tree is equal to the height of the root.
- Show that:
  1. $H + 1 \leq N \leq 2^{H+1} - 1$
  2. $1 \leq L \leq 2^{H}$
  3. $H \leq I \leq 2^{H} - 1$
  4. $\log(N + 1) - 1 \leq H \leq N - 1$
  5. The number of edges is equal to $(N - 1)$.

End