Instructions: In-class exercises are meant to introduce you to a new topic and provide some practice with the new topic. **Work in a team of up to 4 people to complete this exercise.** You can work simultaneously on the problems, or work separate and then check your answers with each other. **Turn in one copy of the exercise per group**.

Names:

7.6 Graph Theory: Binary Trees

7.6.1 Intro to Trees

Terminology

Tree: A collection of **Nodes (or vertices)** and **Edges**.

Edge: A path that connects two Nodes together. If we have N nodes, then there are $N - 1$ edges.

Nodes: A vertex in the tree, usually associated with some data.

Root Node: The source Node of the tree; it has no parents. Each Tree has one Root Node, usually drawn at the top. All other Nodes descend from the Root Node.

Leaf Node: A Node with no children.

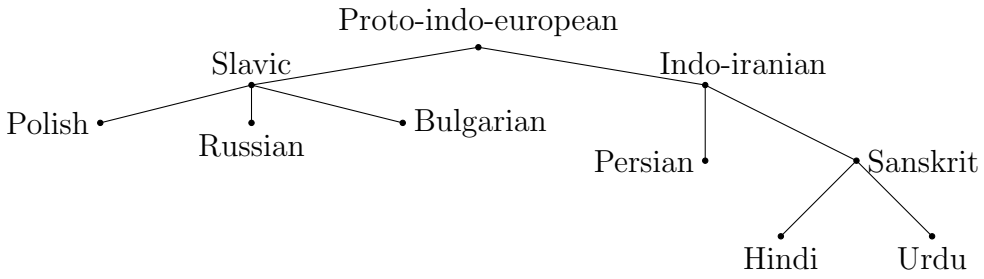
Terminology

Node Family: We use family terminology to talk about how Nodes are related to each other.

- **Parent node:** Given some Node n, n's parent is the Node immediately above n, in the path between n and the root node. Each Node can have only 0 or 1 parent.
- **Ancestor node:** Given some Node n, an Ancestor of n is any Node along the path from n to the root node.
- **Child node:** Given some Node n, n's child is a Node that comes immediately below it in the tree. Node n lies in the path from its child to the root node. Each Node can have 0 or more children. With a Binary Search Tree, a Node can have 0, 1, or 2 children.
- **Descendant node:** Given some Node n, a Descendant is a Node that comes below it in the tree, where the Node n lies in the path from that descendant to the root node.
- **Sibling node:** Given some Node n, a Sibling of n is another Node where n and that Sibling share the same Parent node.

Question 1

For the given tree:



a. What are all the (listed) ancestors of *Russian*?

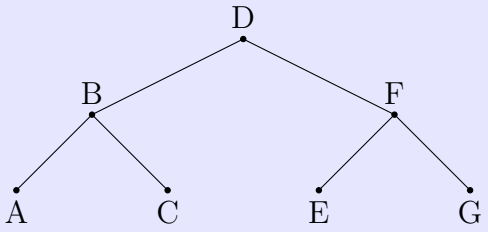
b. What are all the (listed) descendants of $Indo - iranian$?

c. What are all the (listed) siblings of *Polish*?

d. What are all the (listed) leaves of the tree?

7.6.2 Traversals

Terminology



Since a Tree is not a linear structure, what order do you display its contents? There are three main methods you will see for traversing through a tree. Each of these are recursive, beginning at the root node. Once the end of a path is reached (by hitting a leaf), the recursion causes it to step back upwards through the tree.

Pre-order traversal Begin at the Root r node of some Tree/Subtree...

1. Process r
2. Traverse left, if available
3. Traverse right, if available

With the above tree, we process nodes as such:
`D - B - A - C - F - E - G`

In-order traversal Begin at the Root r node of some Tree/Subtree...

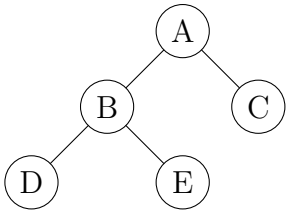
1. Traverse left, if available
2. Process r
3. Traverse right, if available

With the above tree, we process nodes as such:
`A - B - C - D - E - F - G`

Post-order traversal Begin at the Root r node of some Tree/Subtree...

1. Traverse left, if available
2. Traverse right, if available
3. Process r

With the above tree, we process nodes as such:
`A - C - B - E - G - F - D`

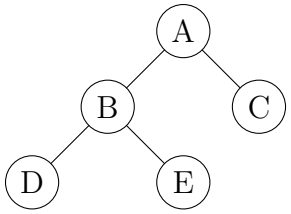


Preorder example

1. Node A: Display "A", traverse left.
2. Node B: Display "B", traverse left.
3. Node D: Display "D", no left child, no right child, return.
4. Node B: Traverse right.
5. Node E: Display "E", no left child, no right child, return.
6. Node B: Done, return
7. Node A: Traverse right.
8. Node C: Display "C", no left child, no right child, return.
9. Done. Result: "ABDEC"

Inorder example

1. Node A: Have left child, traverse left.
2. Node B: Have left child, traverse left.
3. Node D: No left child, display "D", no right child, return.
4. Node B: Left child done, display "B", traverse right.
5. Node E: No left child, display "E", no right child, return.
6. Node B: Right child done, return.
7. Node A: Left child done, display "A", traverse right.
8. Node C: No left child, display "C", no right child, return.
9. Done. Result: "DBEAC"

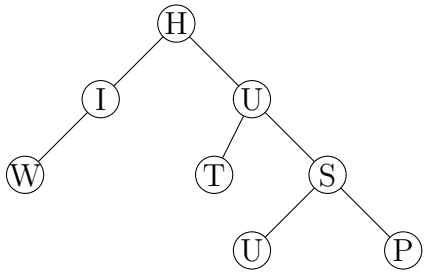


Postorder example

1. Node A: Have left child, traverse left.
2. Node B: Have left child, traverse left.
3. Node D: No left child, no right child, display "D", return.
4. Node B: Have right child, traverse right.
5. Node E: No left child, no right child, display "E", return.
6. Node B: Display "B", return.
7. Node A: Have right child, traverse right.
8. Node C: No left child, no right child, display "C", return.
9. Node A: Display "A".

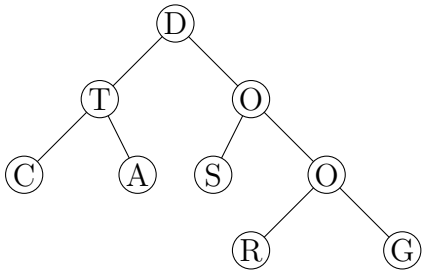
Question 2

Traverse the following tree using **pre-order** traversal. Write out each Node as you "process" it.



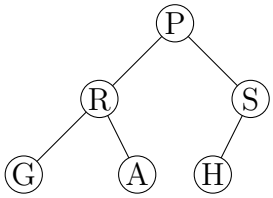
Question 3

Traverse the following tree using **post-order** traversal. Write out each Node as you "process" it.



Question 4

Traverse the following tree using **in-order** traversal. Write out each Node as you "process" it.


P
R
S
G
A
H

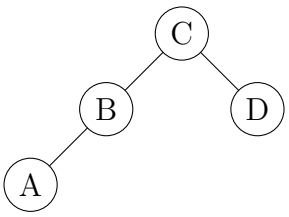
Question 5

A **binary search tree** is a type of tree where each node can have 0, 1, or 2 children, but no more than 2. For any Node n, any nodes to the **left** of n are less than n. Similarly, any nodes to the **right** of n are greater than n.

When the first node is added to a binary tree, it becomes the **root**. When subsequent nodes are added, we traverse the tree, moving to the left or right until we find an available space.

For the following, there is a list of nodes in the order added to a tree. Draw out the binary tree once all nodes are added.

Example: Add: `C, B, D, A`



a. Add: `B, A, D, C, F, E`

b. Add: `A, B, C, D, E`

c. Add: `E, D, C, B, A`

By Rachel Singh, last updated September 30, 2018