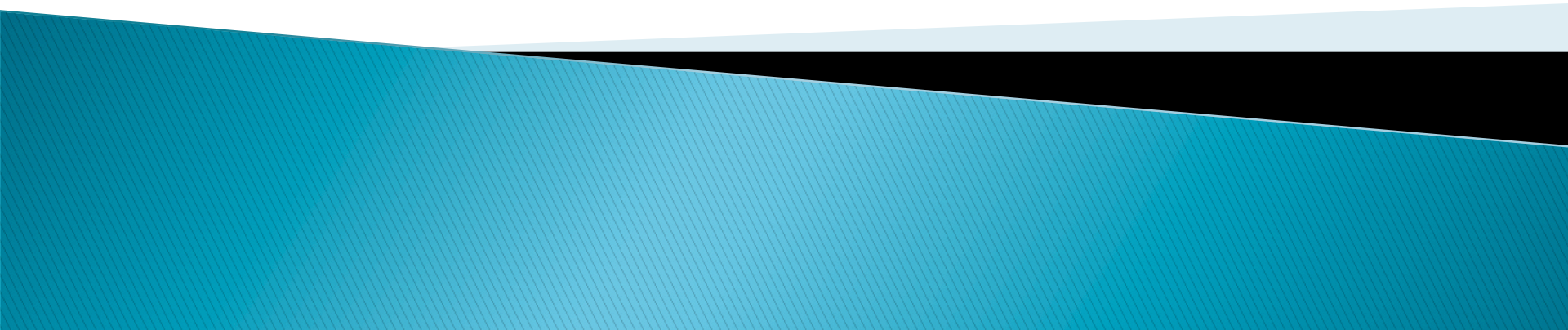
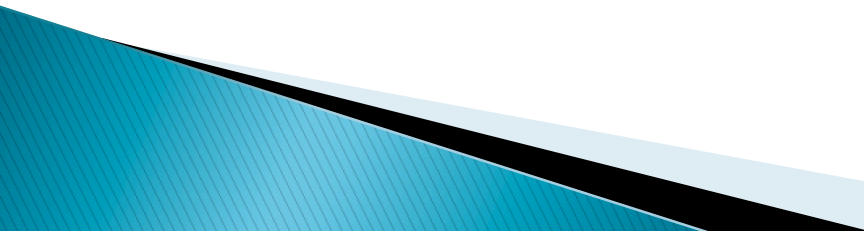


Tree Search Techniques

By Sarah Bailey

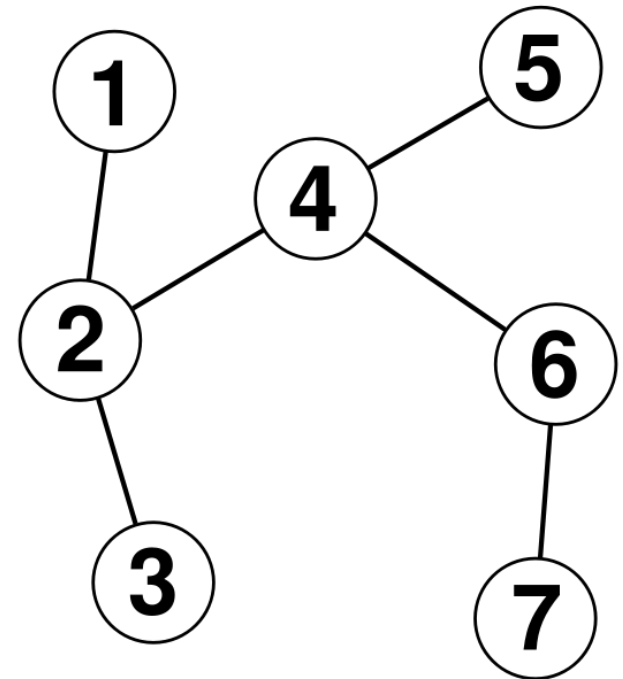


What to Expect?

- ▶ What is a tree?
 - ▶ Types of trees:
 - Spanning tree
 - Rooted tree
 - Binary tree
 - ▶ Searching techniques
 - Depth First Search
 - Breath First Search
 - Min. and Max. Spanning trees
- 

What is a Tree?

- ▶ Connected Graph
- ▶ No cycles
- ▶ Contains: N vertices and $n-1$ edges
- ▶ Not a tree if:
 - Remove an edge from a graph
 - Add an edge to a graph

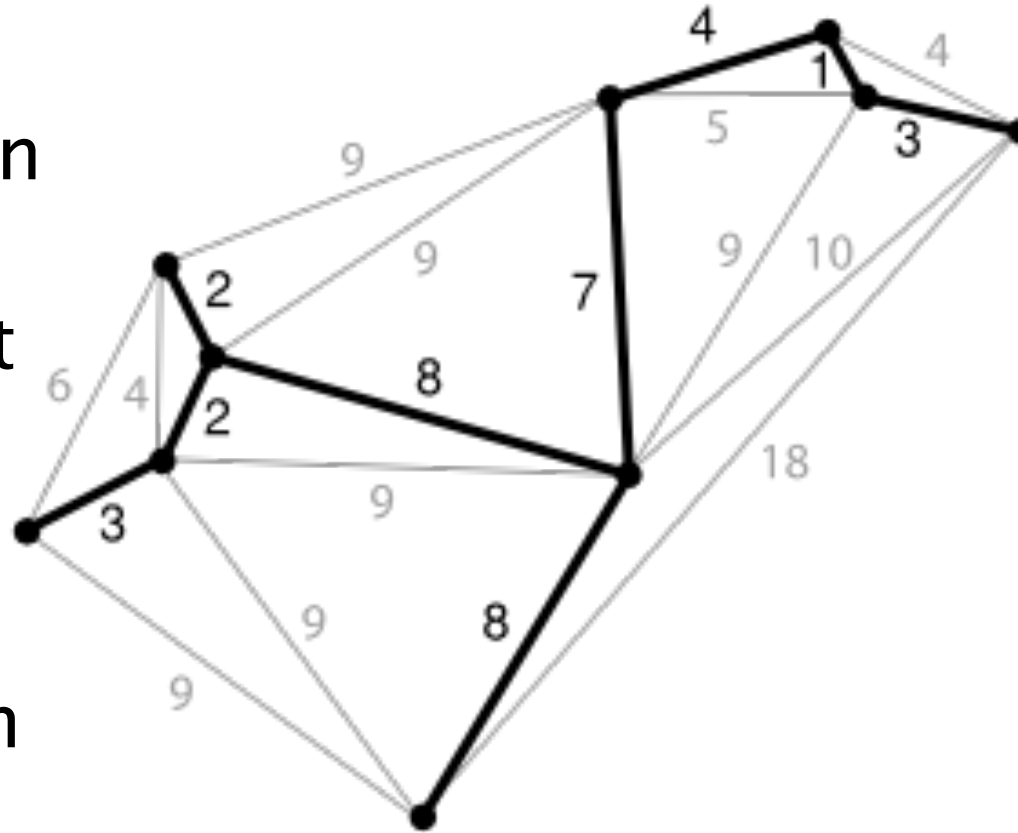


Types of Trees

- ▶ **Spanning Trees**
 - ▶ **Rooted Trees**
 - ▶ **Binary Trees**
- 

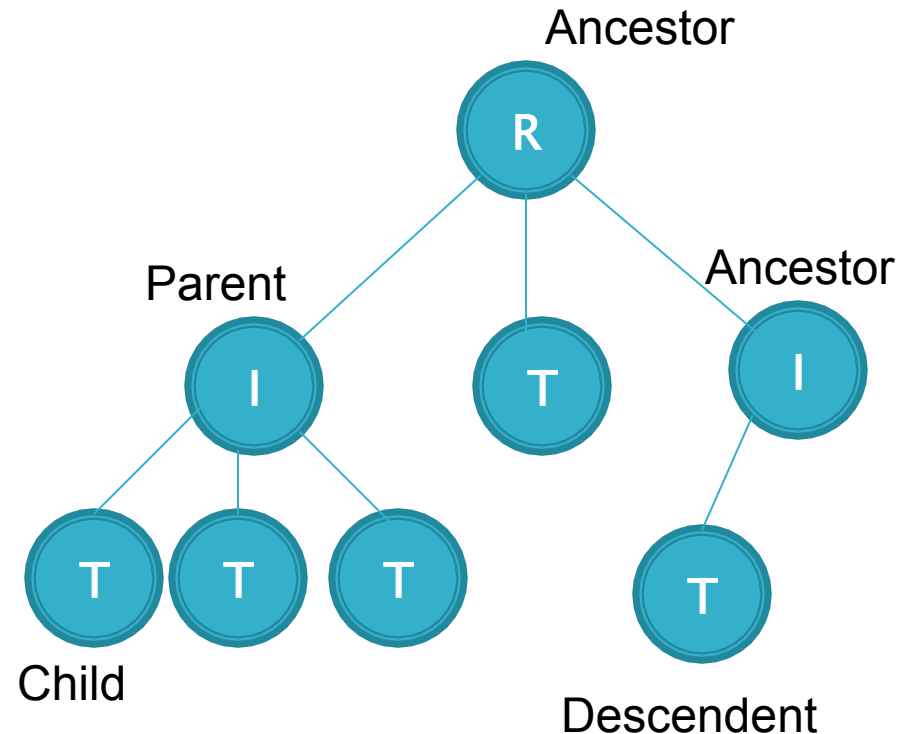
Spanning Tree

- ▶ Includes all vertices in graph G
- ▶ Include all edges that do not create cycles
- ▶ Edges = $n-1$
- ▶ Searches: minimum and maximum search



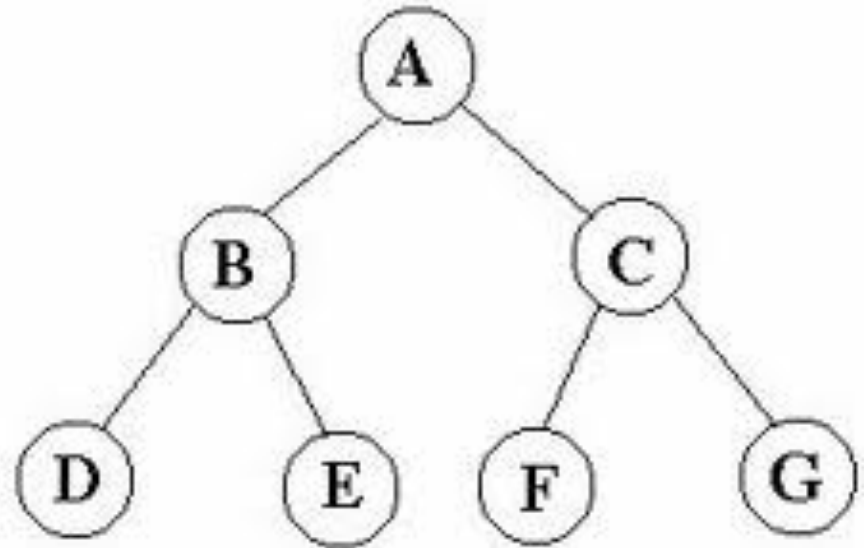
Rooted Tree

- ▶ Directed graph
- ▶ Two conditions:
 - Ignoring directions of graph results in a tree
 - Unique vertex R with in degree(0)
- ▶ Components
 - Root, Internal Vertices, Terminal Vertices, Parents, Children, Ancestors, Descendants



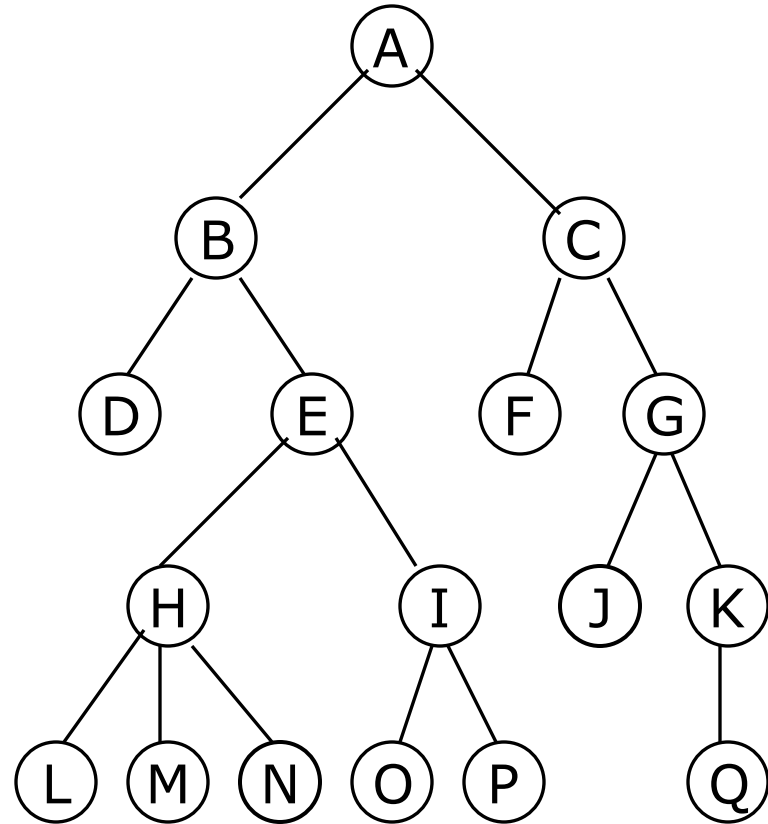
Binary Tree

- ▶ Rooted Tree
 - Root
 - Parent nodes
 - Maximum of two children
 - Child nodes
 - Right or left child

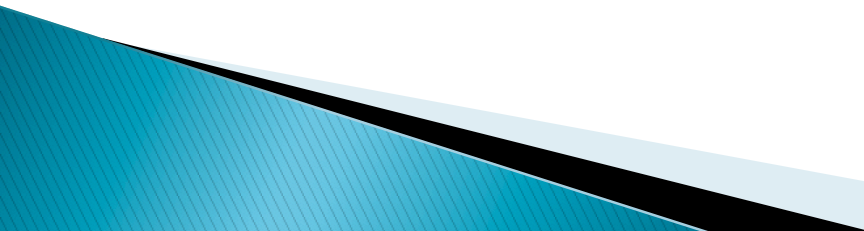


Searching a Tree

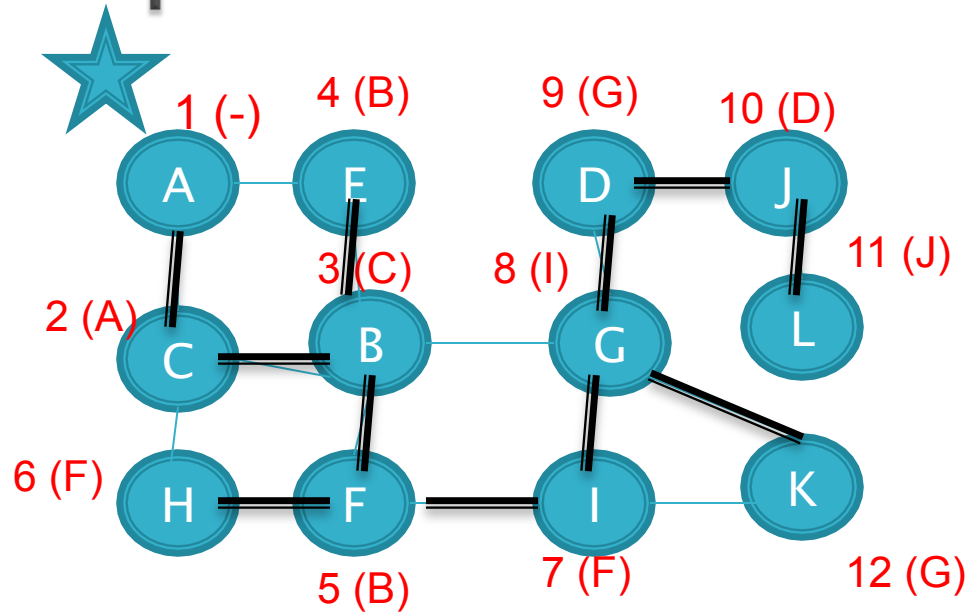
- ▶ Begin at the Root
- ▶ Explore following nodes
- ▶ End at desired node



Depth-First Search Algorithm

- ▶ Goal: Find a spanning tree on a graph G
 - ▶ Designate starting point
 - ▶ Visit all acceptable neighboring nodes (No cycles!)
 - ▶ Backtrack if necessary
 - ▶ End at last node.
 - ▶ *Keep track of the order in which nodes are visited*
- 

Depth-First Search



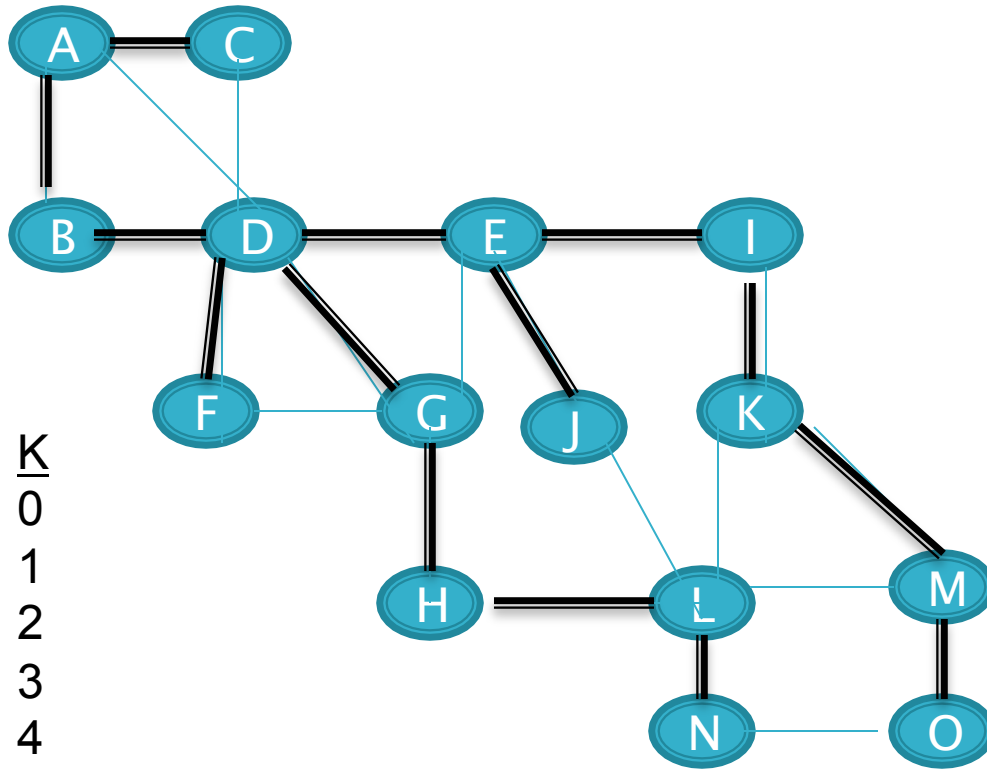
$L = \{A, C, B, E, F, H, I, G, D, J, L, K\}$

GRAPH G

Breadth-First Search Algorithm

- ▶ Goal: Find spanning tree
- ▶ Similar to DFS, but shorter.
- ▶ Designate starting node
- ▶ Visit each appropriate adjacent node (without creating cycles!)
- ▶ End at the n th node with created spanning tree
- ▶ *Keep track of visited nodes (L) and length (k)*

Breadth-First Search



L
A
B,C
D
E,F,G
H,I,J
K,L
M,N
O

K
0
1
2
3
4
5
6
7

Other Spanning Tree Searches

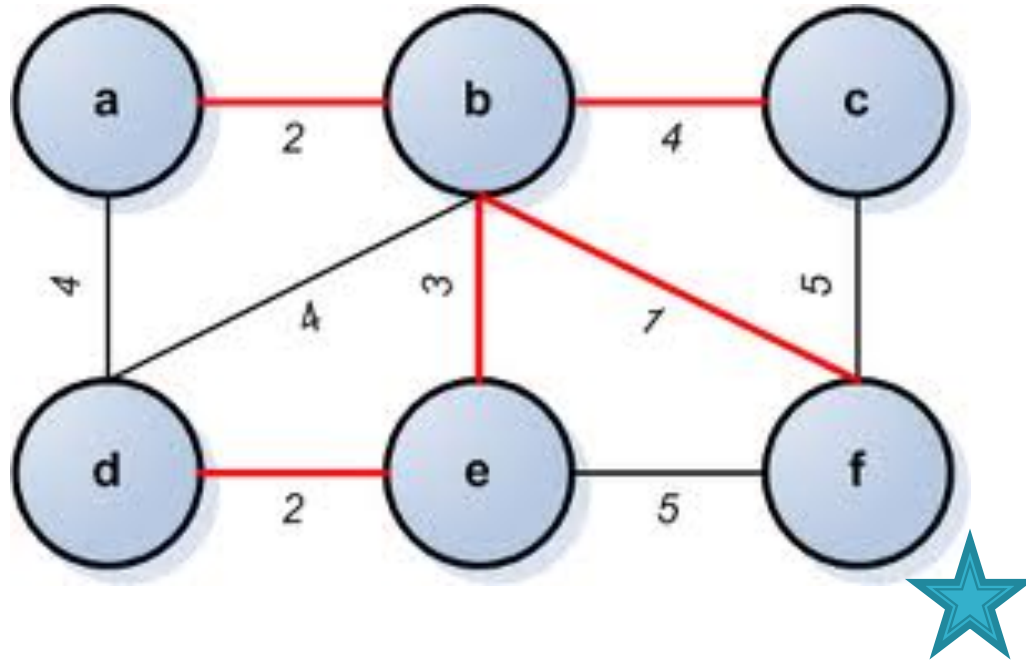
▶ Minimum Spanning Tree:

- Goal: Create a spanning tree with minimum path value
- Start at a designated root
- Select the adjacent node with smallest edge value
- Continue until all nodes are visited
- *Keep running total of the selected edge values*

▶ Maximum Spanning Tree:

- Goal: Create a spanning tree with maximum path value

Example of Minimum Spanning Tree



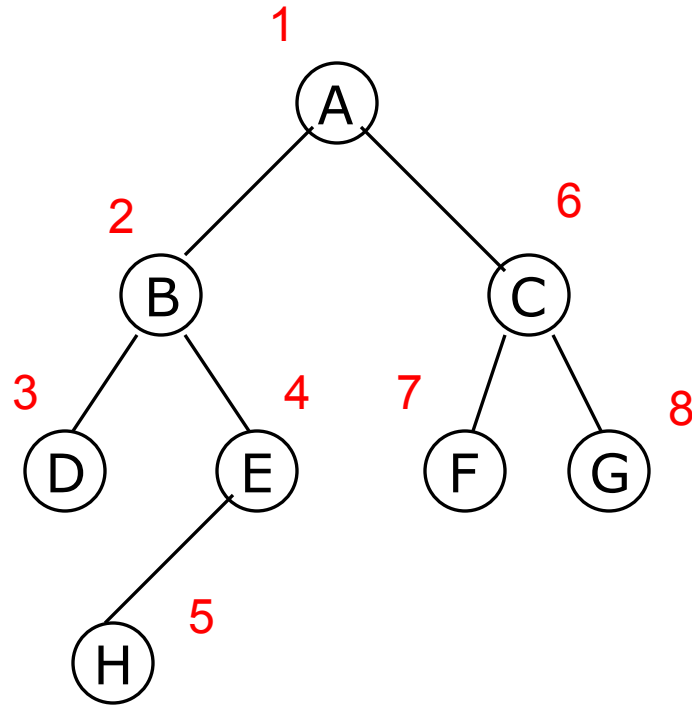
$L = \{f, b, a, e, d, c\}$

$T = 1 + 2 + 3 + 2 + 4 = 12$

Binary Tree Search Techniques

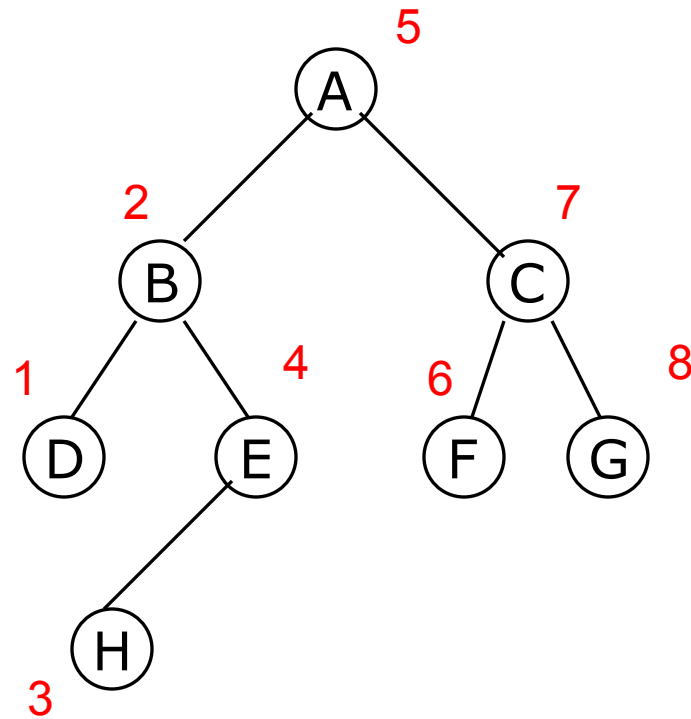
- ▶ Preorder Traversal (“Polish Notation”)
 - Start at Root
 - Root....Left....Right
- ▶ Inorder Traversal
 - Start at Root
 - Left....Root....Right
- ▶ Postorder Traversal (“Reversed Polish Notation”)
 - Start at Root
 - Left....Right....Root

Preorder Traversal



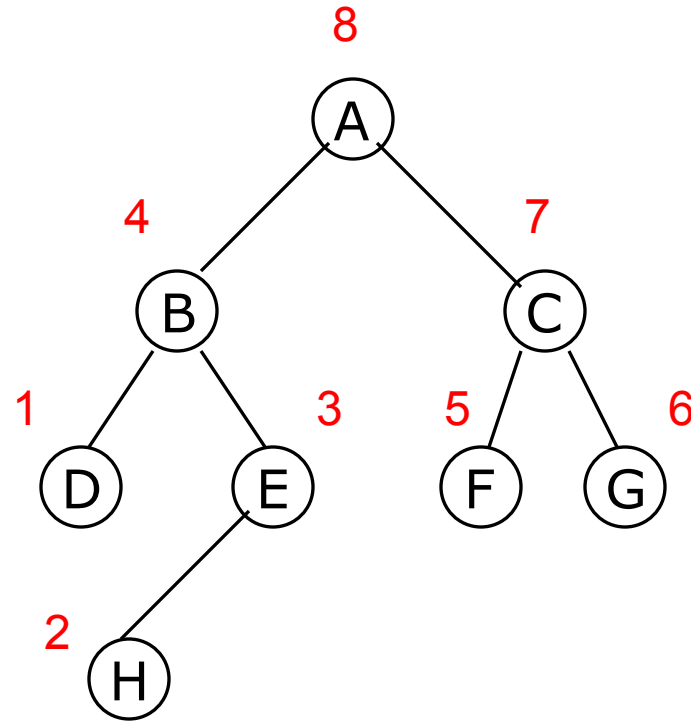
Root - Left - Right

Inorder Traversal



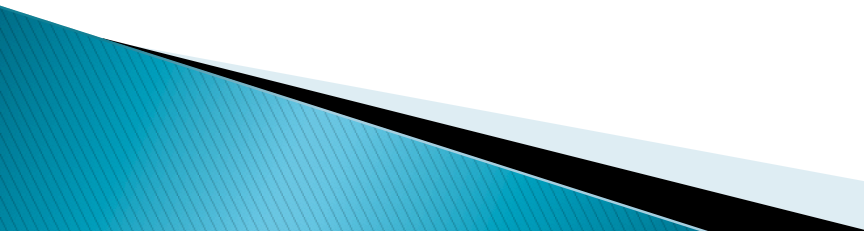
Left – Root - Right

Postorder Traversal



Left – Right - Root

Summary

- ▶ Definition of a tree
 - ▶ Types of trees
 - Spanning
 - Rooted
 - Binary
 - ▶ Types of searches
 - Depth-first search
 - Breath-first search
 - Min. and Max. spanning trees
 - Preorder, Inorder, Postorder traversal
- 

References

- ▶ <http://cs482.elliottback.com/lecture-4-minimum-spanning-trees/>
- ▶ http://en.wikipedia.org/wiki/Binary_tree
- ▶ http://en.wikipedia.org/wiki/Tree_graph
- ▶ http://www.kirupa.com/developer/actionscript/depth_breadth_search2.htm
- ▶ <http://www.i-cherubini.it/mauro/blog/2006/04/06/minimum-spanning-tree-of-urban-tapestries-messages/>