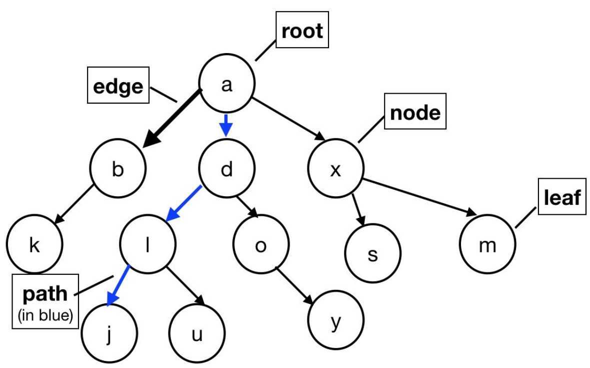
CS 225 Spring 2019 :: TA Lecture Notes
2/15 Tree

By Wenjie

- **Tree Terminology review**
    - Vertex: “nodes”
    - Edge: a connection between two vertices
    - Path: sequence of edges
    - Parents: Node **b, d, x** have Node **a** as their parent
    - Children: **b, d, x,** are the children of **a**
    - Siblings: **b, d, x,** are siblings of each other
    - Ancestors: **u** has ancestors **l, d, a**
    - Descendants: **x** has **s, m** as its descendants
    - Leaves: Vertices with no children



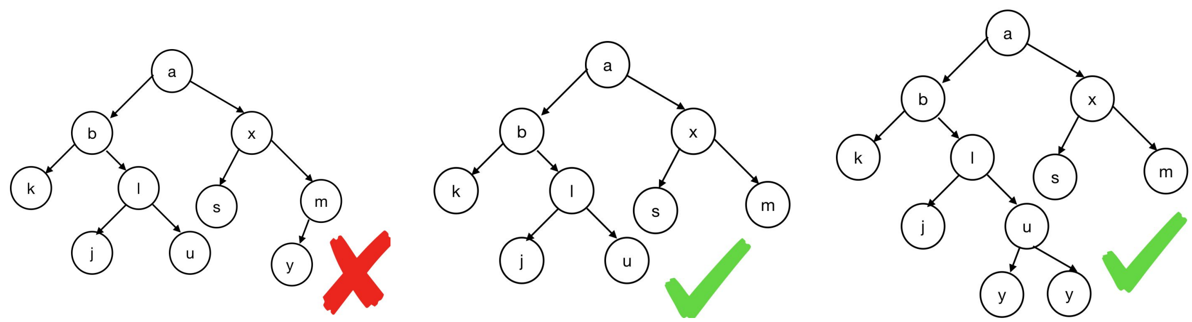
- **Binary Tree**
    - A binary tree is either
        - $T = \{T_L, T_R, r\}$, where T_L, T_R are binary trees
        - $T = \{\} = \varnothing$

- **Computation of the tree height**
    - The length of the longest path from the root to the leaf (count edges).
    - If we want to compute recursively:
        - height(T) = 1 + max(height(T_L), height(T_R)), where if height(null) = -1, which might be counter-intuitive but it follows the mathematical definition of tree height

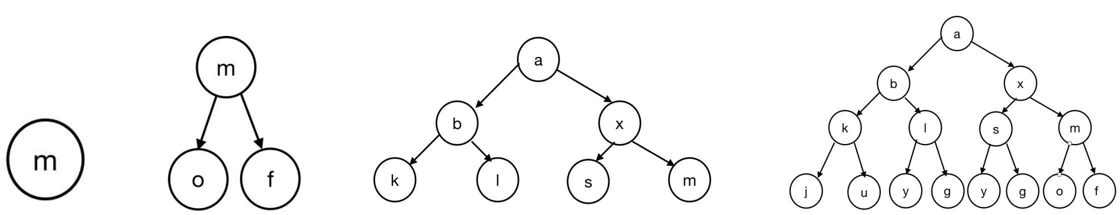
CS 225 Spring 2019 :: TA Lecture Notes
2/15 Tree

By Wenjie

- **Full Tree**:
    - A binary tree is **full** *if and only if*
        - Either: F = {}
        - Or: $F = \{T_L, T_R, r\}$ where T_L, T_R both have either 0 or 2 children



- **Perfect Tree**
    - A perfect tree P_h is defined by its height
        - P_h is a tree of height **h**, with
            - $P_{-1} = \{\}$
            - $P_h = \{r, P_{h-1}, P_{h-1}\}$ when $h \geq 0$

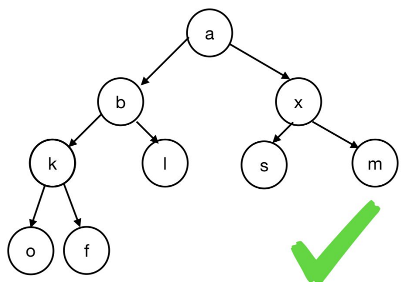


P_0 P_1 P_2 P_3

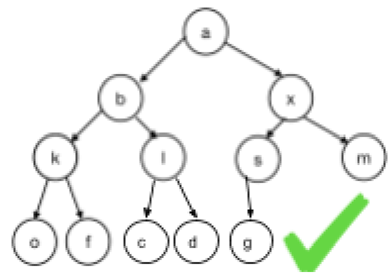
- **Complete Tree**
    - A complete tree is
        - A perfect tree except for the last level
        - All leaves must be pushed to the **left**
    - Or, recursively, a complete tree C_h of height **h** is
        - $C_{-1} = \{\}$
        - $C_h = \{r, T_L, T_R\}$ where
            - Either: $T_L = C_{h-1}$ and $T_R = P_{h-2}$
            - Or: $T_L = P_{h-1}$ and $T_R = C_{h-1}$

CS 225 Spring 2019 :: TA Lecture Notes 2/15 Tree

By Wenjie

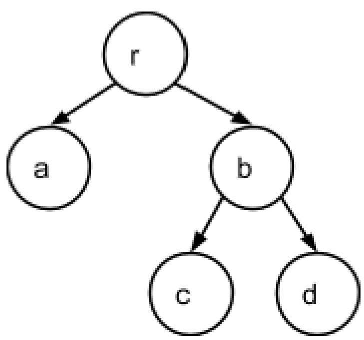


$T_L = C_{h-1}$ and $T_R = P_{h-2}$

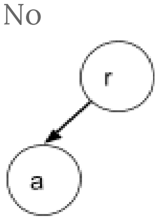


$T_L = P_{h-1}$ and $T_R = C_{h-1}$

- **Tree property**
  - Is every full tree has to be complete?
    - No



  - How about the other way - is every complete tree has to be full?
    - No

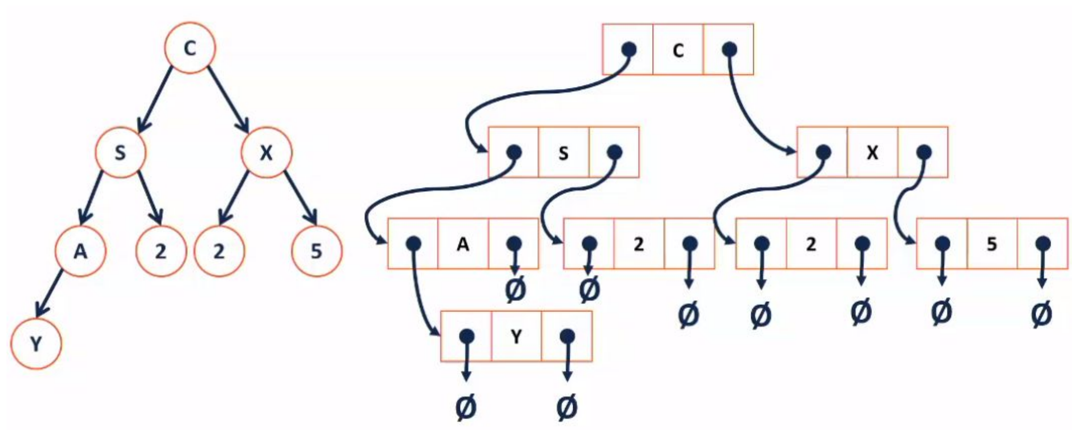


  - Also,
    - Full does not imply perfect, so as complete does not imply perfect
    - Not full implies not perfect, thus perfect implies full; perfect also implies complete too.

- **Tree ADT**
  - Operations of Tree ADT
    - Insert

- Remove
- Traverse

○ A binary tree is just like a fancy linked list since they both traverse between nodes/TreeNode



BinaryTree.h

```
#pragma once

template <typename T>
class BinaryTree {
      public:

            /* … */

      private:
            class TreeNode {
                  TreeNode * left;  // pointer to the left child
                  TreeNode * right; // pointer to the right
child
                  T & data;
                  TreeNode(T & t) :
                      data(t), left(NULL), right(NULL) {};
                      // constructor (initialization list)
            };
```

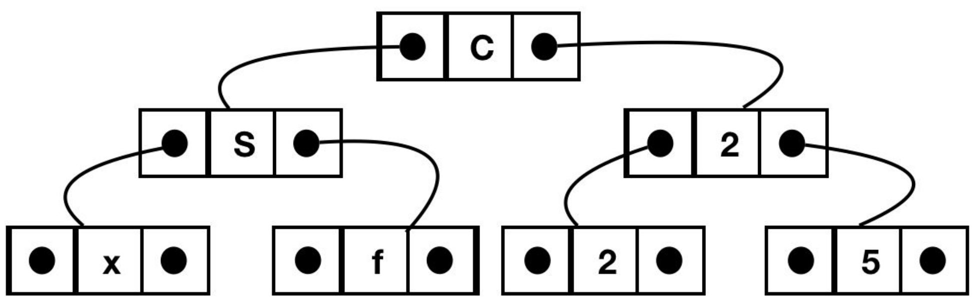
CS 225 Spring 2019 :: TA Lecture Notes
2/15 Tree

By Wenjie

```
            TreeNode * root_;
            // root of the tree: similar to head in linked list
}
```

- Drawing

  The actual tree



  - Every pointer not pointing to another node is NULL

- Number of null pointers in a binary tree
  - **Theorem**: A binary tree with n data items has n+1 null pointers.