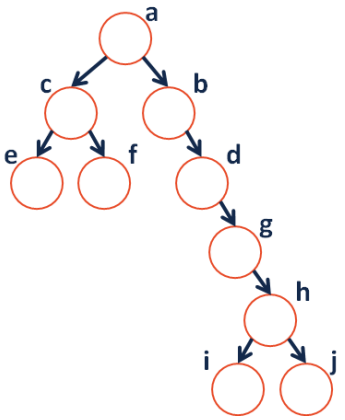
CS 225

#9: Trees

Brad Solomon

Review common tree terminology with the following exercises:

- What's the longest **English word** you can make using the **vertex** labels in the tree (repeats allowed)?
- Find an **edge** that is not on the longest **path** in the tree. Give that edge a reasonable name.
- One of the vertices is called the **root** of the tree. Which one?
- How many parents does each vertex have?
- Which vertex has the fewest **children**?
- Which vertex has the most **ancestors**?
- Which vertex has the most **descendants**?
- List all the vertices is b's left **subtree**.
- List all the **leaves** in the tree.

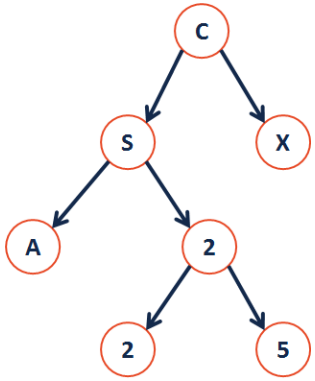


Definition: Binary Tree

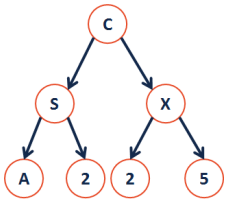
A *binary tree* **T** is either:

Tree Property: Tree Height

Tree Property: Full



Tree Property: Perfect



Towards a Tree Implementation – Tree ADT:

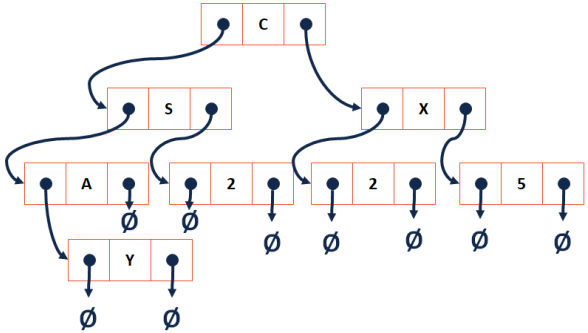
ADT Functionality (English Description)	Function Call

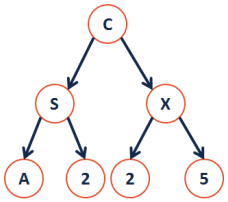
Tree Class

BinaryTree.h

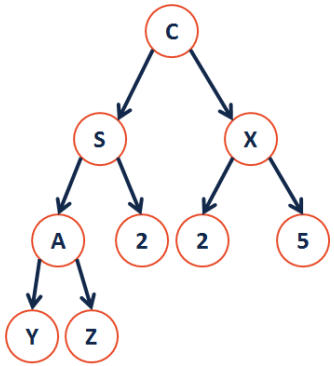
```
1  #pragma once
2
3  template <typename T>
4  class BinaryTree {
5    public:
6      /* ... */
7    private:
8
9
10
11
12 };
```

Trees are nothing new – they're fancy linked lists:

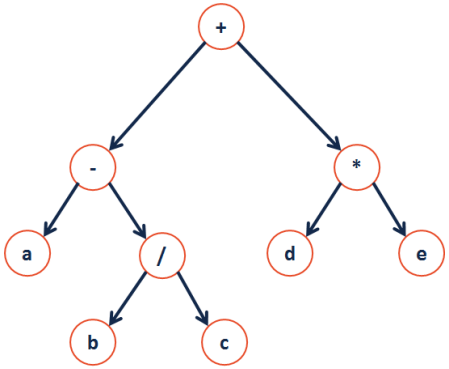




Theorem: If there are n data items in our representation of a binary tree, then there are _____ NULL pointers.



Traversals:



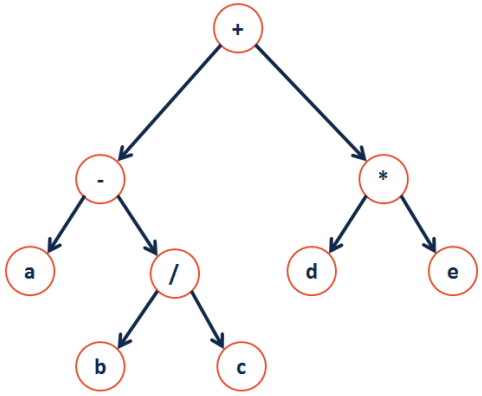
One Algorithm, Three Traversals:

BinaryTree.cpp

```
50  void BinaryTree<T>::___Order(TreeNode * cur) {
51    if (cur != nullptr) {
52
53
54
55
56
57    }
58  }
```

A Different Type of Traversal

Strategy:



BinaryTree.cpp

```
void BinaryTree<T>::levelOrder(TreeNode * croot) {




}
```