CS 225

Data Structures

Oct. 2 – Trees

Queue.h

```
 4 template <class QE>
 5 class Queue {
 6   public:
 7     class QueueIterator : public std::iterator<std::bidirectional_iterator_tag, T> {
 8       public:
 9         QueueIterator(unsigned index);
10         QueueIterator& operator++();
11         bool operator==(const QueueIterator &other);
12         bool operator!=(const QueueIterator &other);
13         QE& operator*();
14         QE* operator->();
15       private:
16         int location_;
17
18     };
19
20
21   /* ... */
22
23   private:
24     QE* arr_; unsigned capacity_, count_, entry_, exit_;
25 };
26
```

Big Ideas

Member functions and variables are only inherited in derived classes.

Big Ideas

Member functions and variables are only inherited in derived classes.

Class scope determines access to private members.

Mattox Monday

Exam 2

Exam 2 Re-take

- **Public**: Can be taken anywhere (**not** a CBTF exam)
- **Time Limited**: 50 minute exam
- **Availability Limited:** Must take before Tuesday, Oct 2nd @ 11:59pm

Exam 4

Exam 4 (MP2-like)

- Programming Exam
- Ongoing right now!

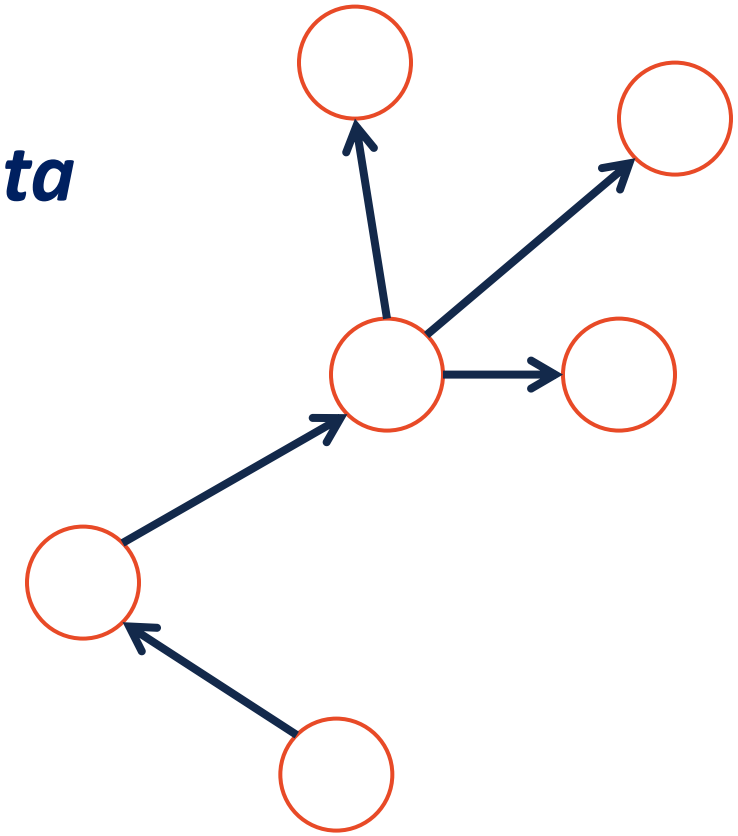
Exam 5

- Theory Exam
- **Topics**: Lists, Stacks, Queues, and related topics
  - All topics covered in lecture/lab/MPs through **right now**!

Trees

"The most important non-linear data structure in computer science."

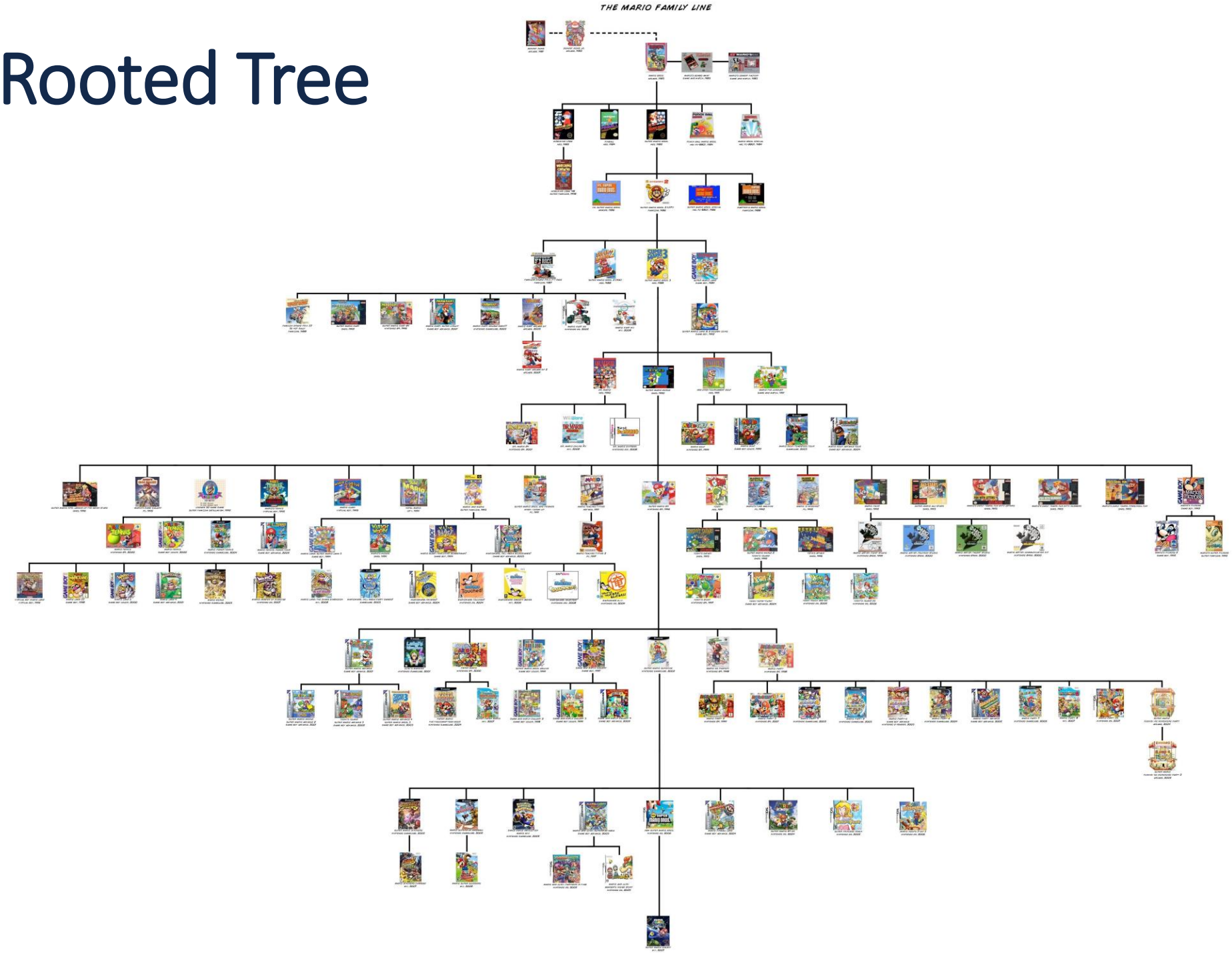
- David Knuth, The Art of Programming, Vol. 1



A tree is:

-
-

A Rooted Tree



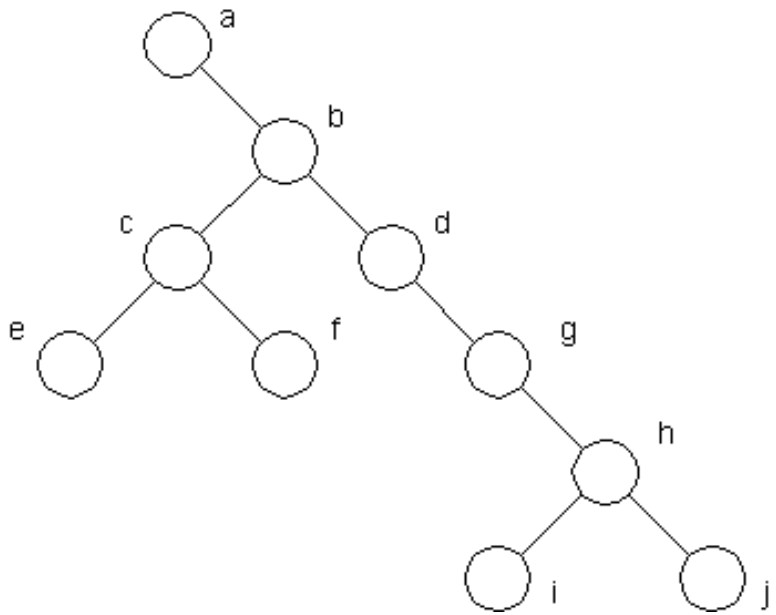
More Specific Trees

We'll focus on a specific type of trees:

-
-

Tree Terminology

- What's the longest "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?
- Make an "word" containing the names of the vertices that have a parent but no sibling.
- 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.



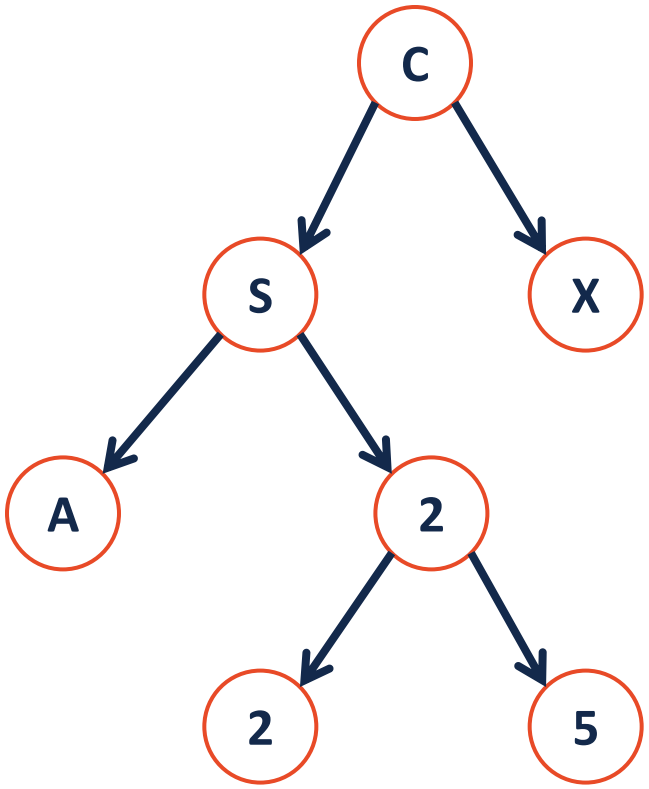
Binary Tree – Defined

A *binary tree* T is either:

-

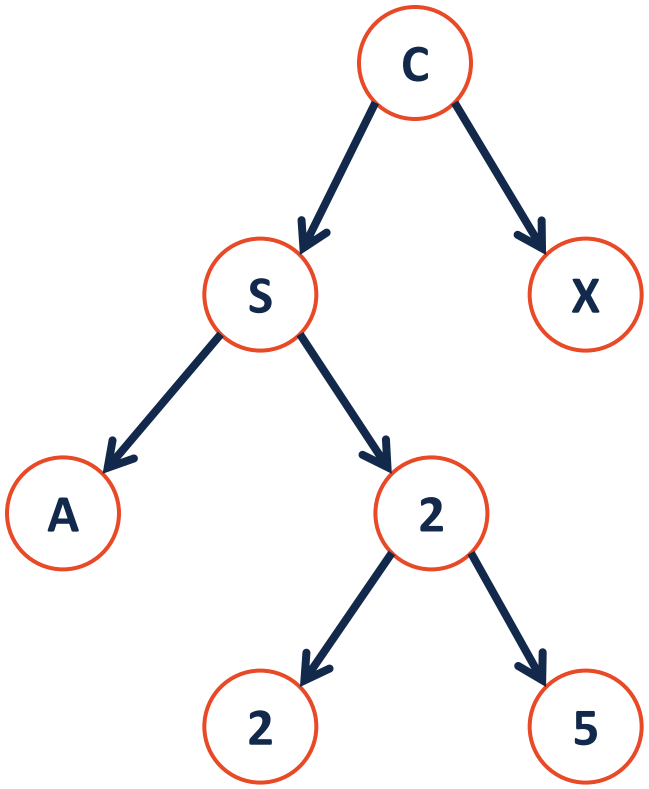
OR

-



Tree Property: height

***height(T)*: length of the longest path from the root to a leaf**



Given a binary tree T:

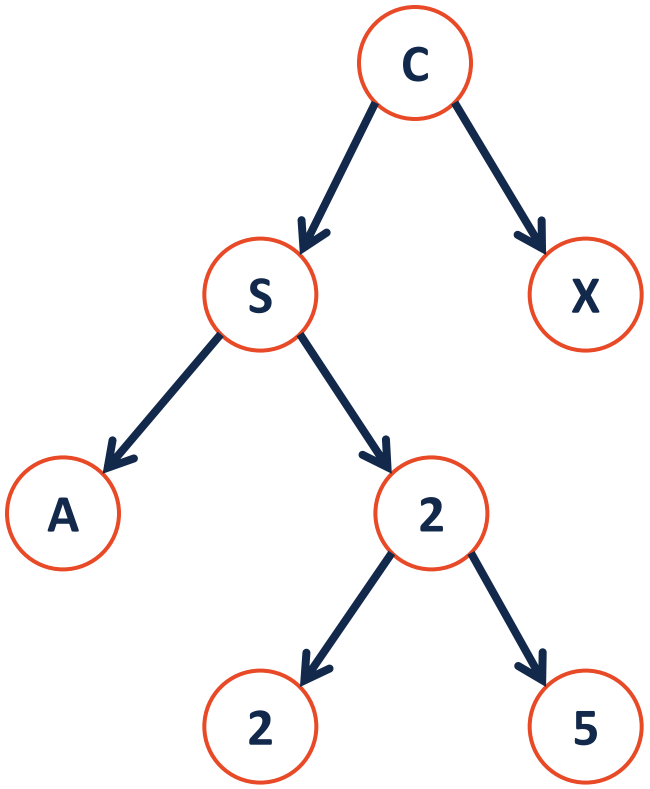
height(T) =

Tree Property: full

A tree ***F*** is **full** if and only if:

1.

2.

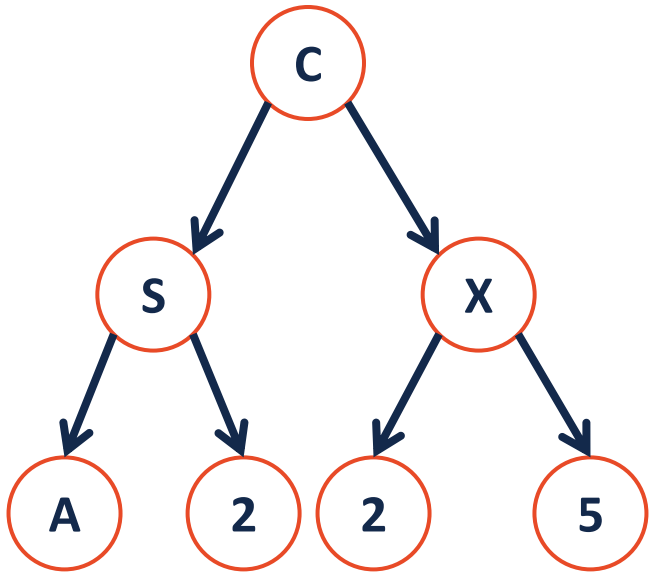


Tree Property: perfect

A **perfect** tree ***P*** is:

1.

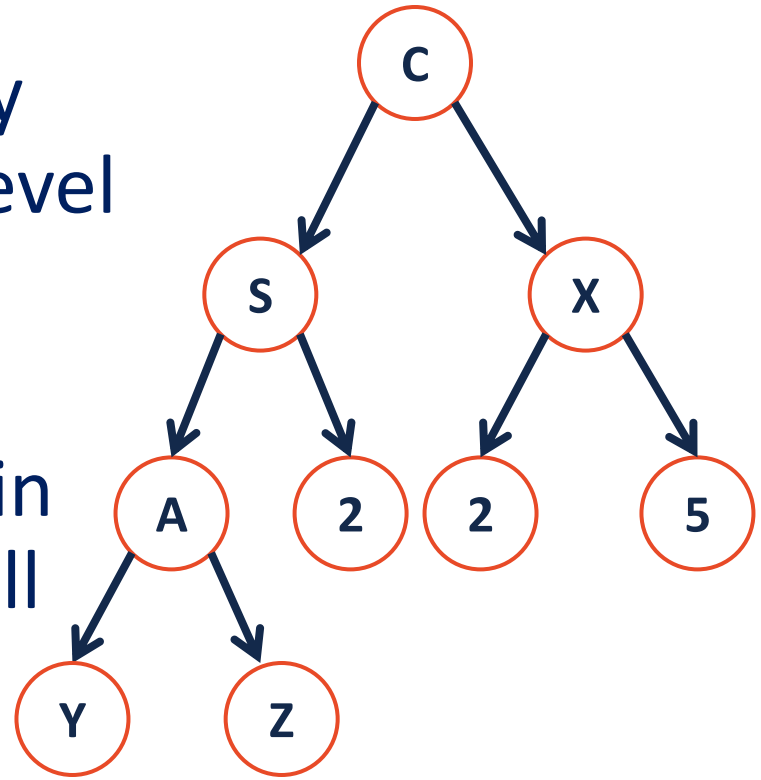
2.



Tree Property: complete

Conceptually: A perfect tree for every level except the last, where the last level if "pushed to the left".

Slightly more formal: For any level k in [0, h-1], k has 2^k nodes. For level h, all nodes are "pushed to the left".



Tree Property: complete

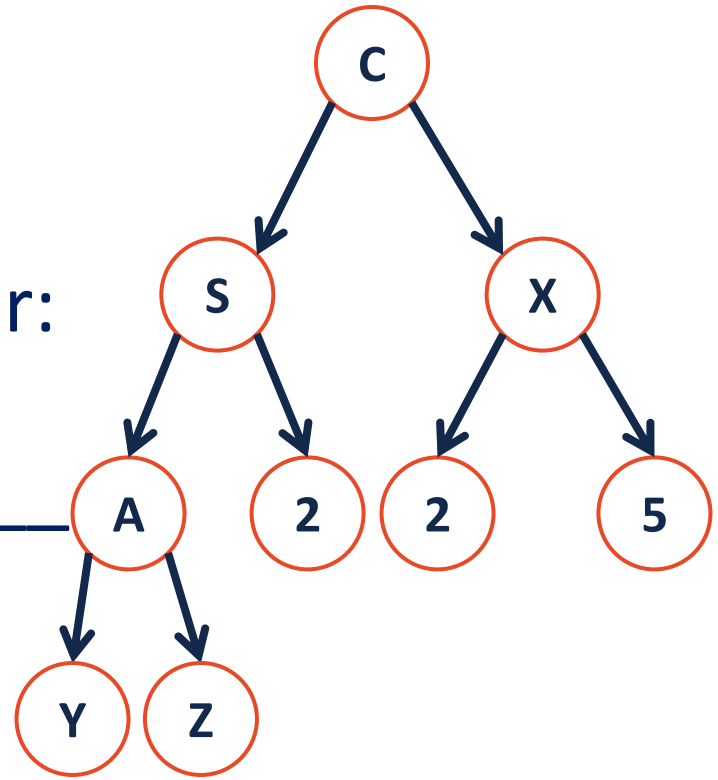
A **complete** tree ***C*** of height **h**, $\mathbf{C_h}$:

1. $\mathbf{C_{-1}} = \{\}$
2. $\mathbf{C_h}$ *(where h>0)* **= {r, $\mathbf{T_L}$, $\mathbf{T_R}$}** and either:

T_L is _____ and T_R is _____

OR

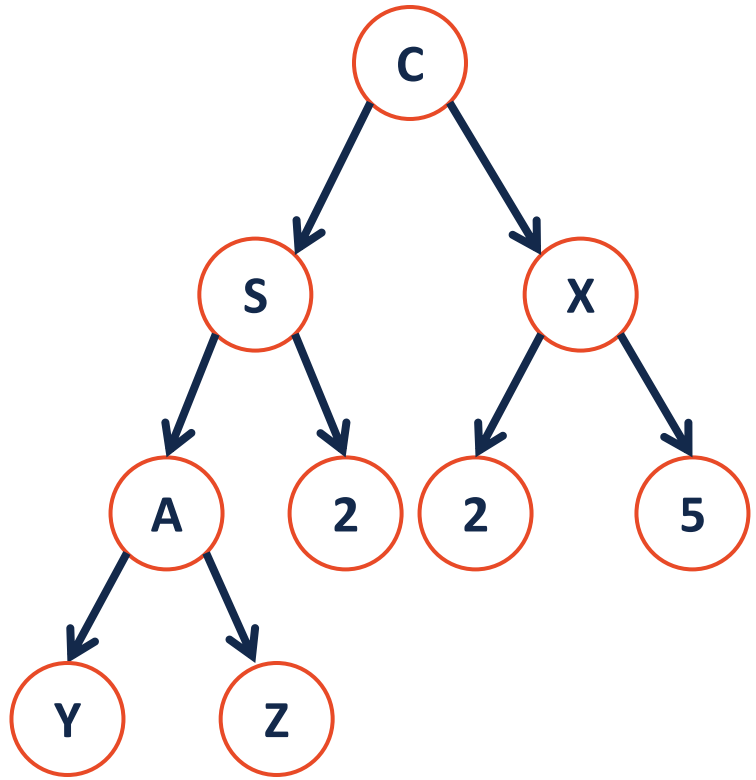
T_L is _____ and T_R is _____



Tree Property: complete

Is every **full** tree **complete**?

If every **complete** tree **full**?



CS 225 – Things To Be Doing

Exam 4 (Programming/MP2) currently ongoing!

More Info: https://courses.engr.illinois.edu/cs225/fa2017/exams/

MP3: Available now!

Up to +7 extra for submission by tonight!

Lab: lab_tree coming this week!

+6 for completion and +4 for attendance

POTD

Every Monday-Friday – *Worth +1 Extra Credit /problem (up to +40 total)*