CS 225

Data Structures

February 14 – Circular Lists and Trees

G Carl Evans

Queue.h

```cpp
#pragma once

template <typename T>
class Queue {
  public:
    void enqueue(T e);
    T dequeue();
    bool isEmpty();

  private:
    T *items_;
    unsigned capacity_;
    unsigned size_;
};
```

What type of implementation is this Queue?

How is the data stored on this Queue?

Queue.h

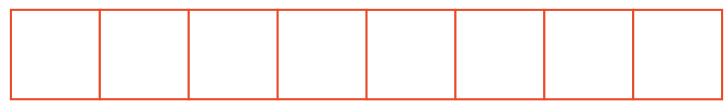
```cpp
#pragma once

template <typename T>
class Queue {
  public:
    void enqueue(T e);
    T dequeue();
    bool isEmpty();

  private:
    T *items_;
    unsigned capacity_;
    unsigned size_;
};
```

What type of implementation is this Queue?

How is the data stored on this Queue?



```cpp
Queue<int> q;
q.enqueue(3);
q.enqueue(8);
q.enqueue(4);
q.dequeue();
q.enqueue(7);
q.dequeue();
q.dequeue();
q.enqueue(2);
q.enqueue(1);
q.enqueue(3);
q.enqueue(5);
q.dequeue();
q.enqueue(9);
```

Queue.h

```cpp
#pragma once

template <typename T>
class Queue {
  public:
    void enqueue(T e);
    T dequeue();
    bool isEmpty();

  private:
    T *items_;
    unsigned capacity_;
    unsigned size_;
};
```

				m	o	n	

```cpp
Queue<char> q;
...
q.enqueue(m);
q.enqueue(o);
q.enqueue(n);
...
q.enqueue(d);
q.enqueue(a);
q.enqueue(y);
q.enqueue(i);
q.enqueue(s);
q.dequeue();
q.enqueue(h);
q.enqueue(a);
```

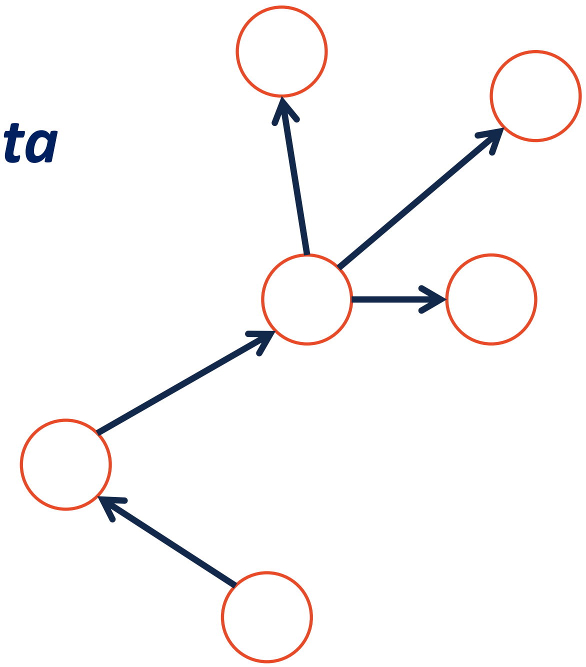
Trees

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

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

A tree is:

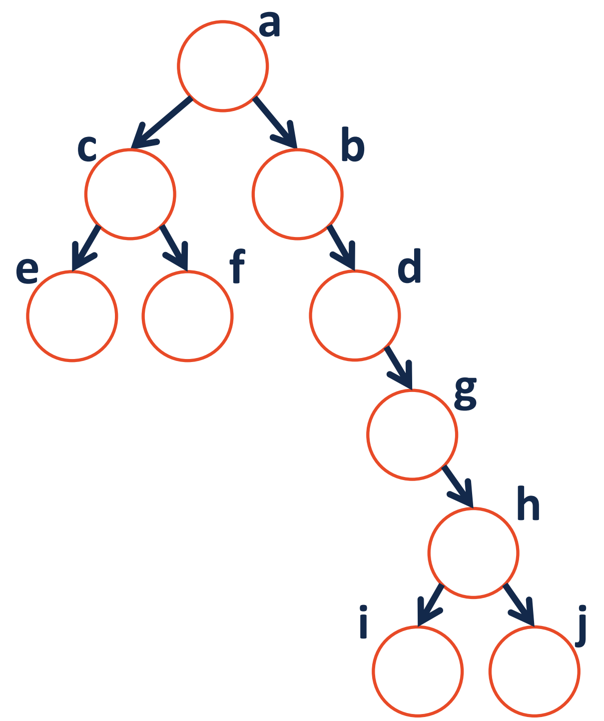
-
-



More Specific Trees

We'll focus on **binary trees**:

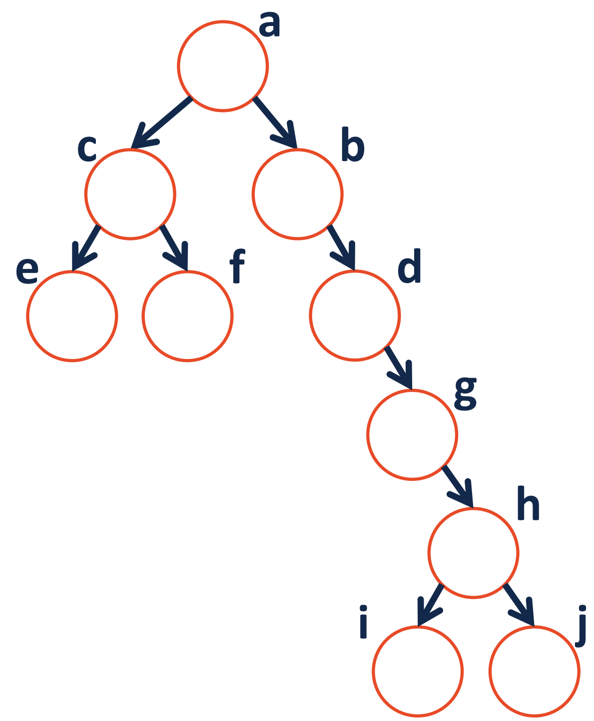
- A binary tree is **rooted** – every node can be reached via a path from the root



More Specific Trees

We'll focus on **binary trees**:

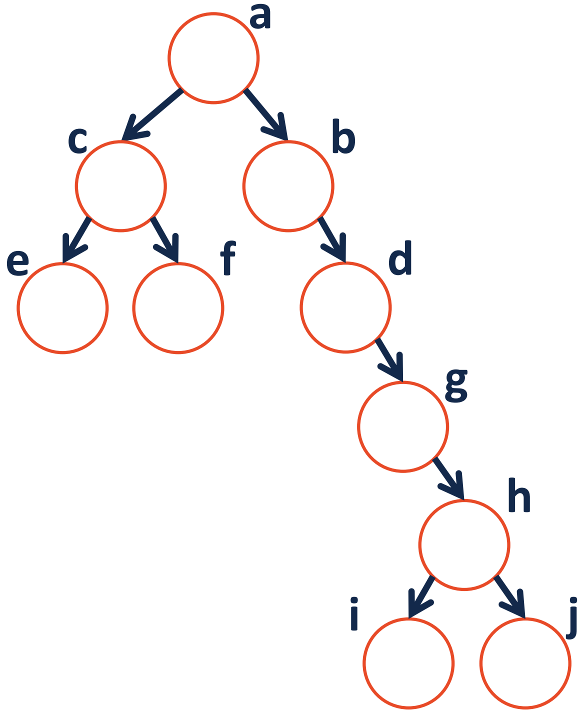
- A binary tree is **acyclic** – there are no cycles within the graph



More Specific Trees

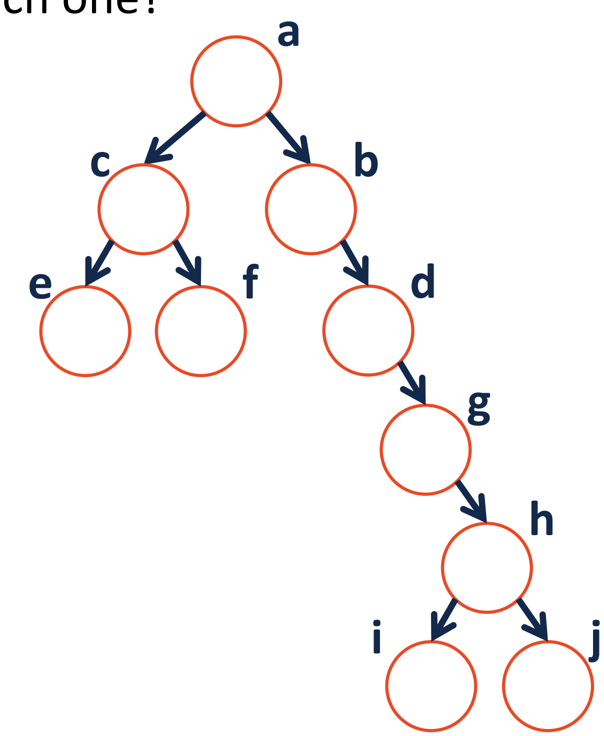
We'll focus on **binary trees**:

- A binary tree contains **two or fewer children** – where one is the "left child" and one is the "right child":



Tree Terminology

- 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.



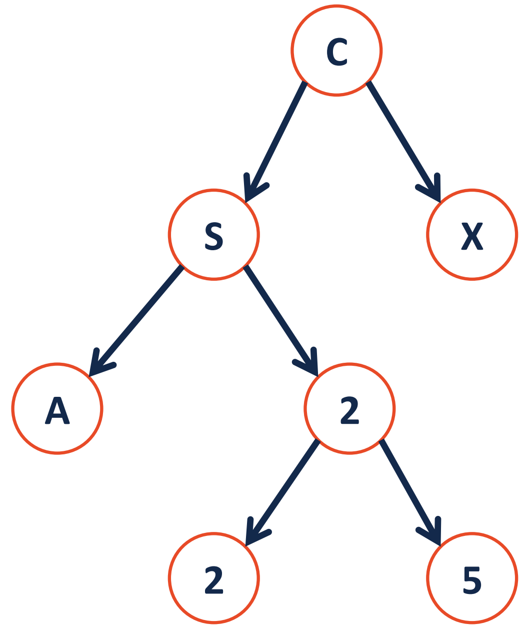
Binary Tree – Defined

A *binary tree* T is either:

-

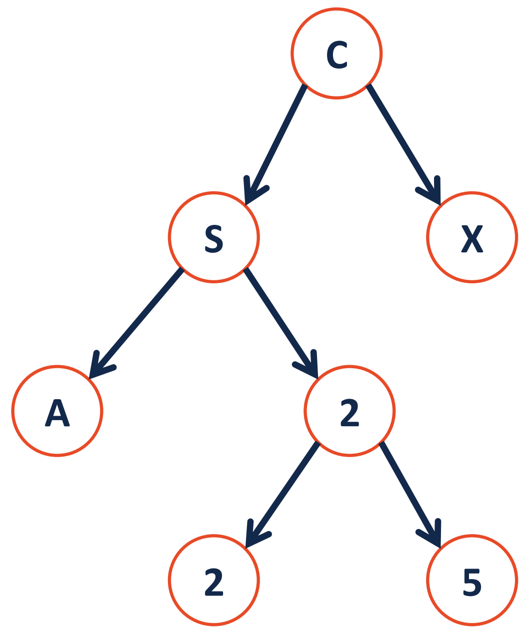
OR

-



Tree Property: height

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



Given a binary tree T:

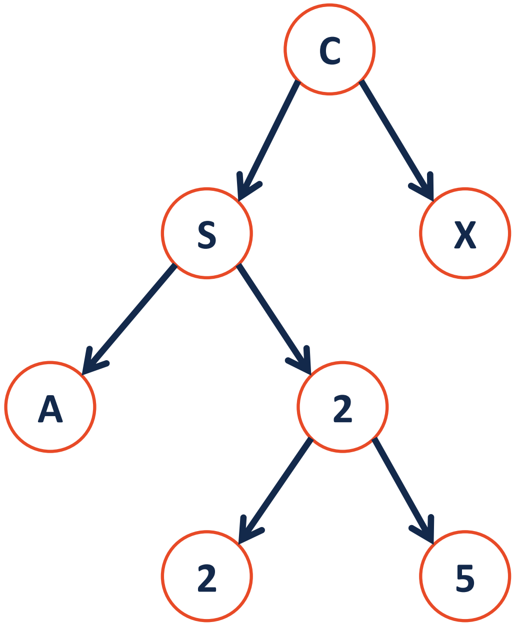
height(T) =

Tree Property: full

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

1.

2.

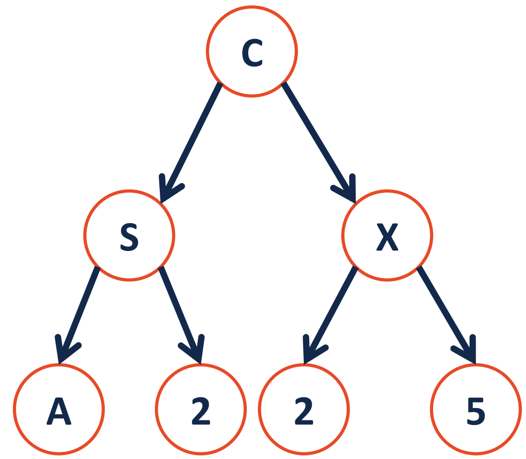


Tree Property: perfect

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

1.

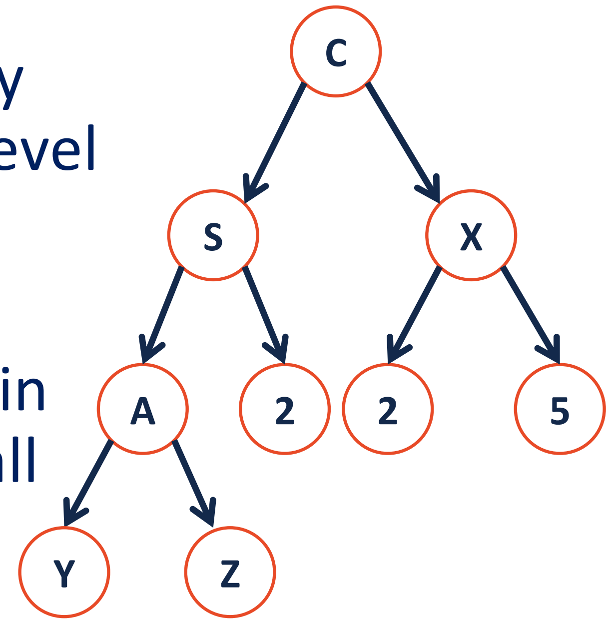
2.



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

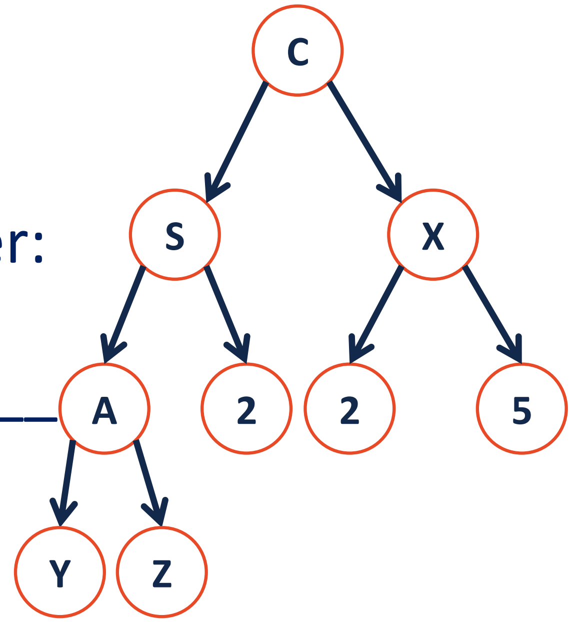
A **complete** tree ***C*** of height **h**, $\mathbf{C_h}$:

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