CSCI 136:
Data Structures
and
Advanced Programming

Lecture 21

Trees

Instructor: Dan Barowy

Williams

Announcements

No Barowy office hours this Friday

PRE-LAB 1: due in lab on today

More one-on-one meetings

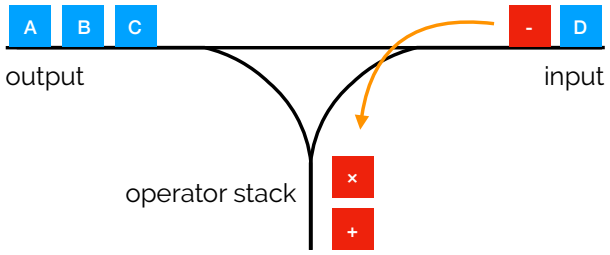
`OrderedVector`

What is the biggest limitation of an `OrderedVector`?

It's elements are `Comparable<T>`,
so there is only one order possible.

How might you overcome that limitation?

Shunting yard algorithm

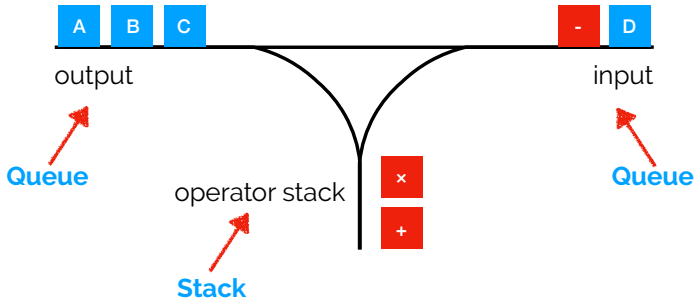


Shunting yard algorithm

Converts **infix** expressions to **postfix** expressions.

Utilizes a **stack** and a **queue**.

Shunting yard algorithm



Shunting yard algorithm

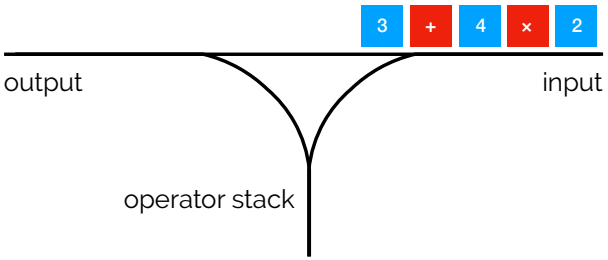
Pseudocode (slightly simplified):

```
while there are tokens to be read:
    read a token.
    if the token is a number, then:
        push it to the output queue.
    if the token is an op, then:
        while ((there is an operator at the top of the op stack with greater
precedence) or
               (the op at the top of the op stack has equal precedence and
is left associative)):
            pop ops from the op stack onto the output queue.
        push it onto the op stack.
if there are no more tokens to read:
    while there are still op tokens on the stack:
        pop the op from the op stack onto the output queue.
exit.
```

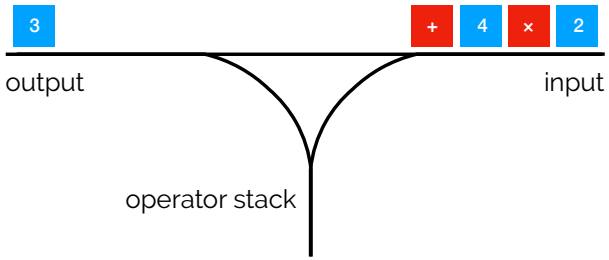
Shunting yard algorithm

Example: 3 + 4 × 2

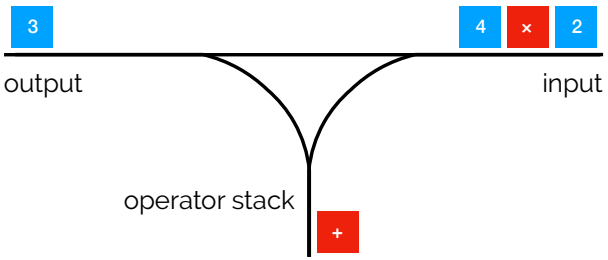
Example: 3 + 4 × 2



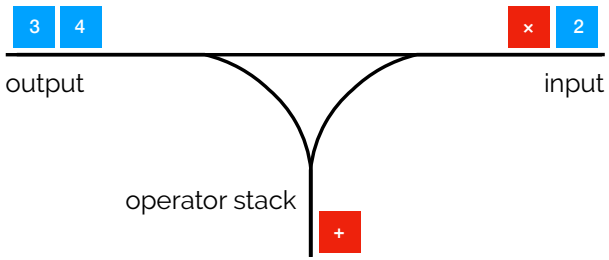
Example: 3 + 4 × 2



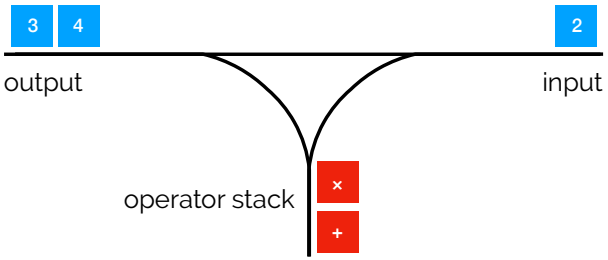
Example: 3 + 4 × 2



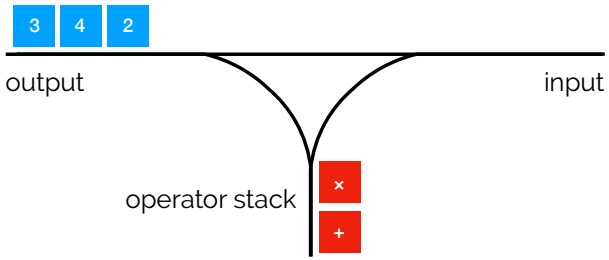
Example: 3 + 4 × 2



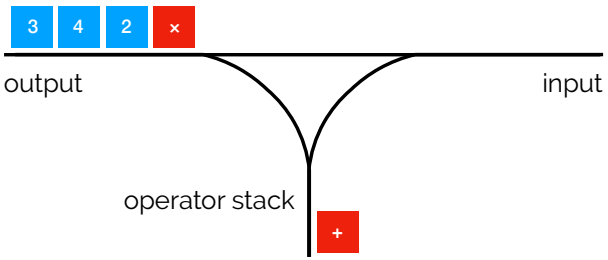
Example: 3 + 4 × 2



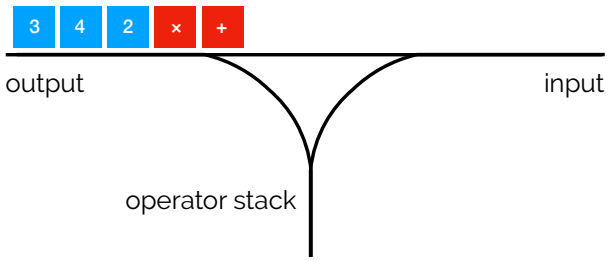
Example: 3 + 4 × 2



Example: 3 + 4 × 2



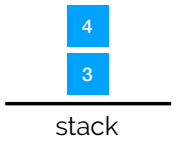
Example: 3 + 4 × 2



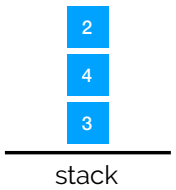
Is 3 4 2 × + correct?

Is 4 2 × + correct?

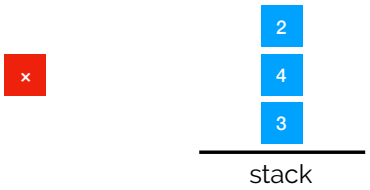
Is 2 × + correct?



Is × + correct?

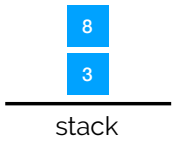


Is + correct?

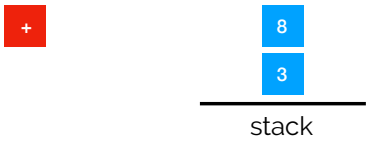


Is + correct?

Is + correct?



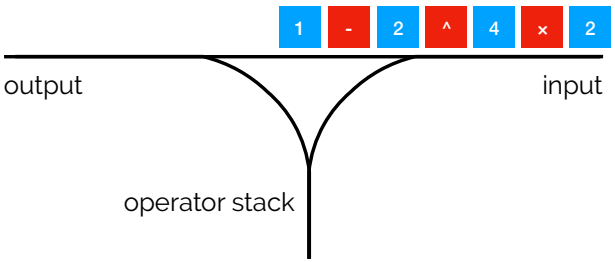
Is correct?



Is correct?

Is correct?

Activity: $1 - 2^4 \times 2$

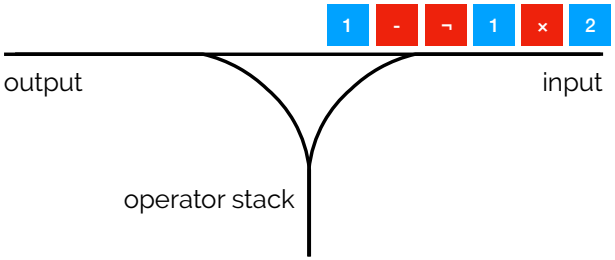


Shunting yard algorithm

Pseudocode (slightly simplified):

```
while there are tokens to be read:
    read a token.
    if the token is a number, then:
        push it to the output queue.
    if the token is an op, then:
        while ((there is an operator at the top of the op stack with greater
precedence) or
                (the op at the top of the op stack has equal precedence and
is left associative)):
            pop ops from the op stack onto the output queue.
        push it onto the op stack.
if there are no more tokens to read:
    while there are still op tokens on the stack:
        pop the op from the op stack onto the output queue.
exit.
```

Activity: 1 – – 1 × 2



Trees

List

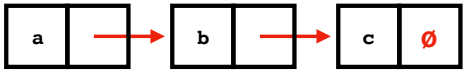
A **list** is a recursive data structure that stores information sequentially. A list is either:

- **empty** (i.e., Ø) or
- a **node** containing a **value** and a reference to a **list**.

The empty list: Ø

List of length 1:

List of length 3:



Tree

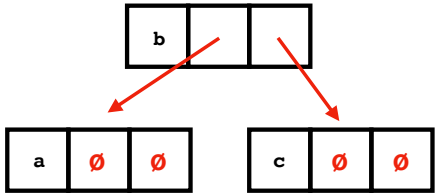
A **tree** is a recursive data structure that stores information hierarchically. A tree is either:

- **empty** (i.e., Ø), or
- a **node** containing a **value** and references to one or more **trees**.

The empty tree:

Ø

A non-empty tree:

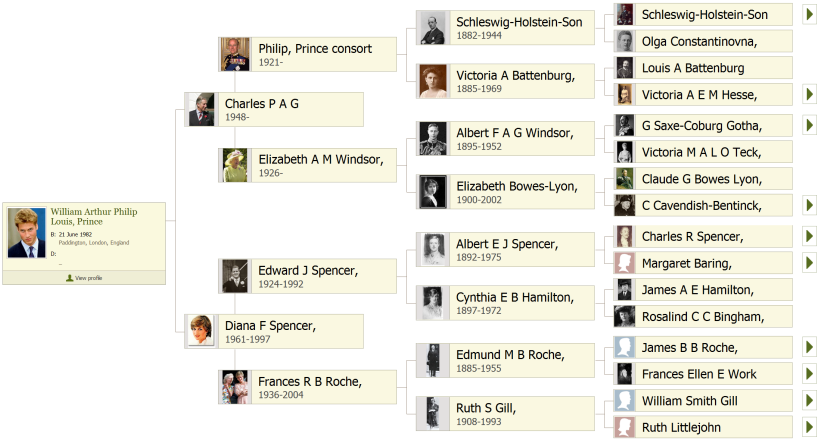


Uses

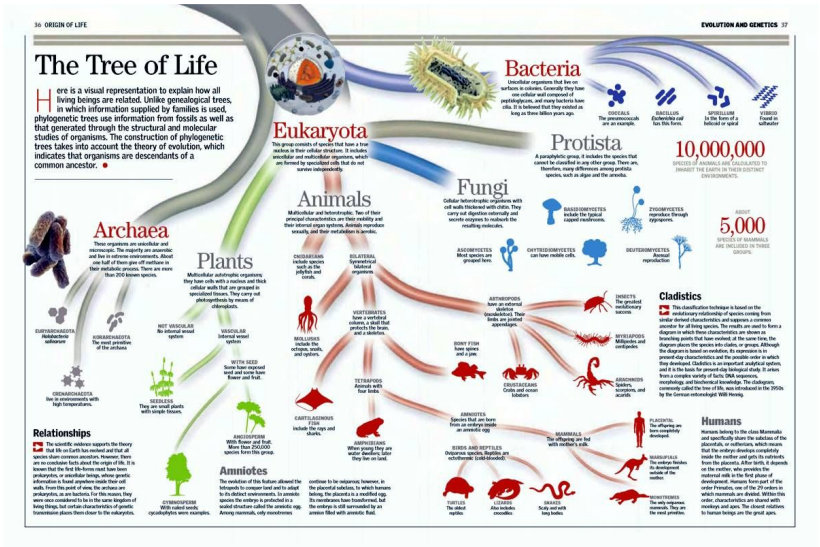
Many kinds of information is **hierarchical**.

Trees facilitate **encoding** such information on a computer.

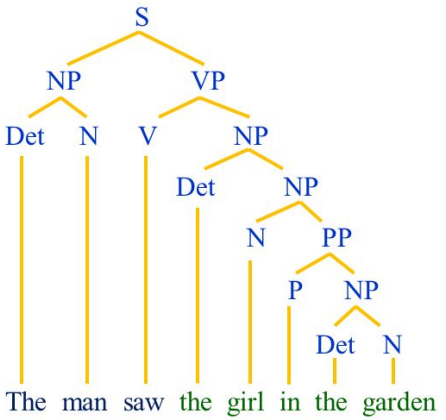
Uses



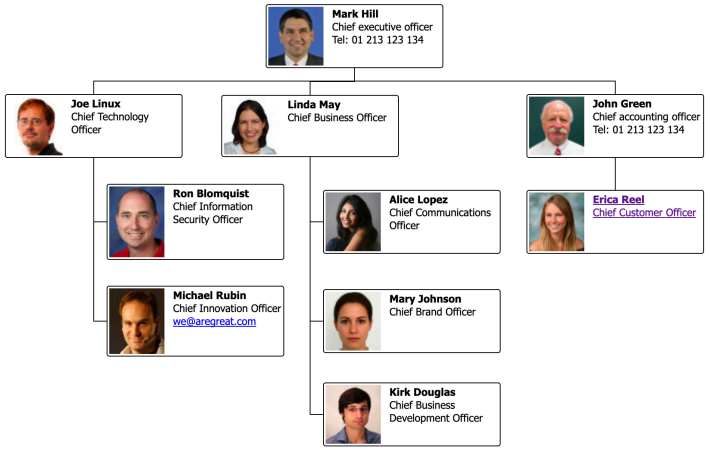
Uses



Uses

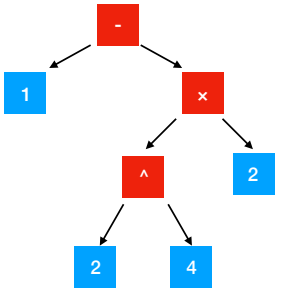


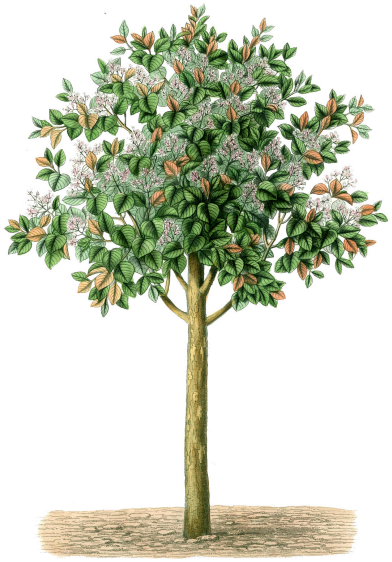
Uses

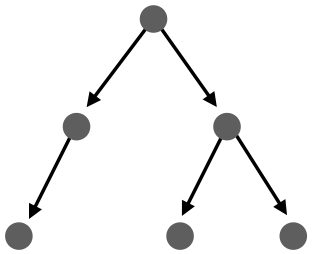


Uses

$$1 - 2^4 \times 2$$

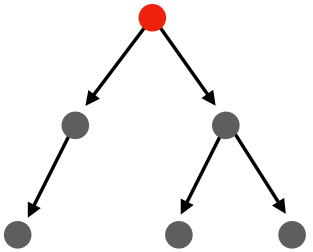






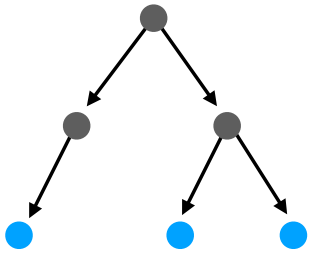
Terminology

The topmost node is called the **root**.



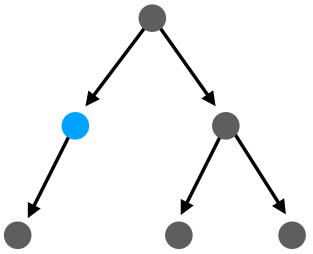
Terminology

The nodes at the bottom of a tree are called **leaves**.



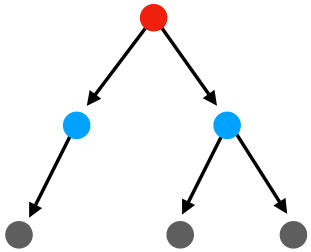
Terminology

Any node that is not a leaf is an **interior node**.



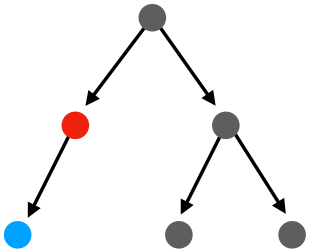
Terminology

A **node** may have **children**.



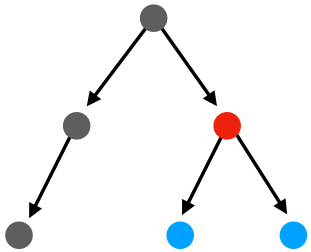
Terminology

A **node** may have **children**.



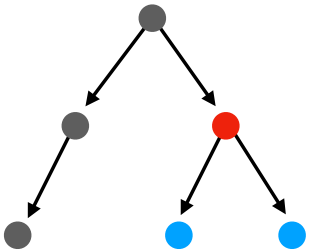
Terminology

A **node** may have **children**.



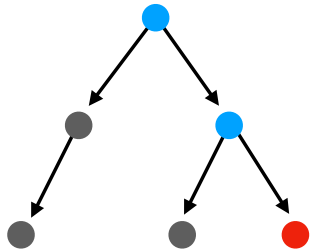
Terminology

A **node** that has **children** is called the **parent** of those children.



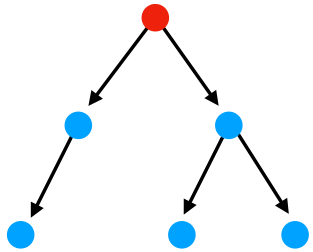
Terminology

For a **given node**, all of the nodes above it are called **ancestors**.



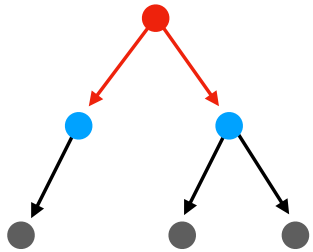
Terminology

For a **given node**, all of the nodes below it are called **descendants**.



Terminology

The **degree** of a tree is the maximum number of **children** had by any node.

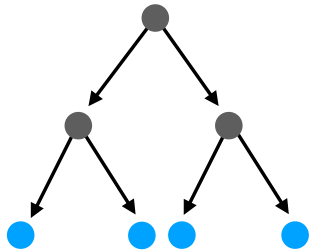


Degree of this tree: 2

Degree 2 trees are common: we call them **binary trees**.

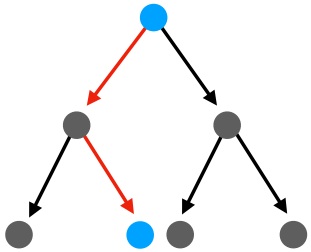
Terminology

A tree that is missing no leaves is **full**.



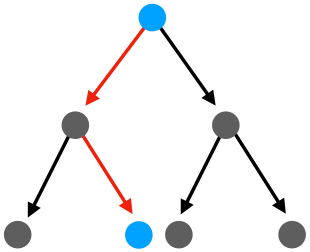
Terminology

A **path** is a sequence of edges between **two nodes**.



Terminology

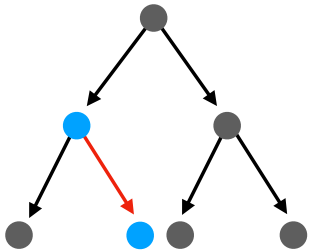
The **length** of a path is the **number of edges** in the path.



Length = **2**

Terminology

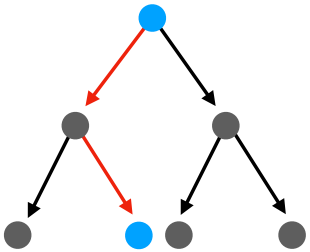
The **height** of **node n** is the length of the longest path between **n** and **any leaf**.



Height of **n** = **1**

Terminology

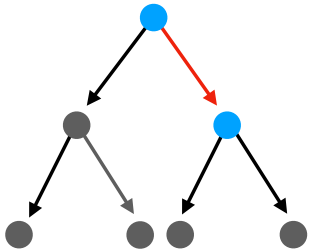
The **height** of **a tree** is the length of the longest path between **the root** and **any leaf**.



Height of **tree** = **2**

Terminology

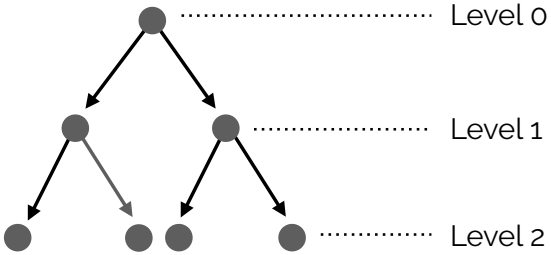
The **depth** of **node n** is the length of the longest path between **the root** and **n**.



Depth of **n** = **1**

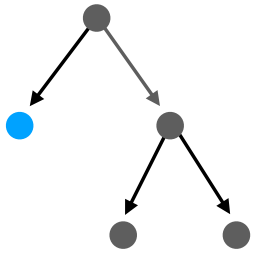
Terminology

The **level** of **any node** is its depth.



Terminology

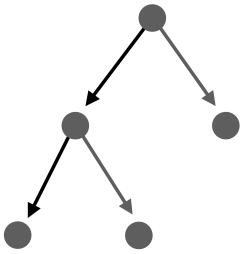
The depth of **n** + the height of **n** ≤ the height of the tree.



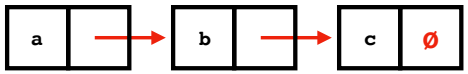
(depth of **n**: **1**) + (height of **n**: **0**) ≤ (height of **tree**: **2**)

Terminology

A **complete** tree of **height h** is a full tree with zero or more rightmost leaves of **level h** removed.



Is a **list** a **tree**?



Yes, a list is a tree whose nodes have **degree** 1.

We call such trees **degenerate**.

Binary Tree

Let's implement this together.

Recap & Next Class

Today we learned:

Trees

Next class:

Binary Search Trees