CMPT 225 - Recursion on trees

Recursion

Recursion: A definition of a function is recursive if the body contains an application of itself.

```
Consider: S(n) = sigma(i)
or S(n) = {0 if n = 0, and n + S(n-1) if n >= 0}
```

These two descriptions of S(n) suggest two implementations:

```
S(n) {
  s = 0
  for i = 1 ... n
    s += i
  return s
}
```

or recursively

```
S(n) {
  if n = 0
    return 0
  return n + S(n-1)
}
```

The call stack: https://youtu.be/X8lmsAekDEY?t=485

Recursion on Trees

We will often use recursion and induction on trees
eg) The tree rooted at V has some property if its subtrees have some related property

eg) The height of a node V in a binary tree may be defined by:
`height(V) = 0 if V is a leaf, and 1 + max{height(left(v)), height(right(v))} otherwise`
We can define h(left(V)) to be -1 if left(V) does not exist and same for right(V).

Pseudo-code height finder

```
height(V) {
  if V is a leaf
    return 0
  if V has one child u
    return 1 + height(u)
  else
    return 1 + max(height(left(V)), height(right(V)))
}
```

Traversals of Binary Trees

A traversal of a graph is a process that visits each node in the graph once
We consider 4 standard traversals:

1. level order
2. pre-order
3. in-order
4. post-order

2, 3, 4 begin at the root and recursively visit the nodes in each subtree and the root. They vary in the relative order.
Level order later.

```
preOrder(V) {
  visit V
  preOrder(left(V))
  preOrder(right(V))
}
```

V is visitited before any of its descendants.
Every node in the left subtree is visited before any node int he right subtree.

```
inOrder(V) {
  inOrder(left(V))
  visit V
  inOrder(right(V))
}
```

```
postOrder(V) {
  postOrder(left(V))
  postOrder(right(V))
  visit V
}
```

Notice the postorder is not just a reverse of preorder.