CMPT 225 Binary Search Trees

```
// Operations
// These O() times are considering the unbalanced linked list tree
insert(x);  // O(h), O(n)
member(x);  // O(h), O(n)
remove(x);  // O(h), O(n)
empty(x);
size(x);
clear(x);
```

Some operations are false and some are slow. With binary search trees, we can get O(log(n)) for many operations.

Bag ADT: like a set but allows duplicates.
Map ADT: unordered collection of <key, value> pairs. At most one value with every key
Dictionary ADT: like map but can have duplicate values for each key

Binary Seach Tree

All the keys to the left are smaller than the current and vice versa with right subtree.

Without balancing the tree can just become a linked list.

`remove(t)`
Three cases:

1. is a leaf: just delete
2. has one child: move child subtree into its place
3. has two children:
   i) Find the node v with key(v) = t
   ii) find the successor of v - call it u
   iii) key(v) <- key(u) `// replace t with its successor`
   iv) delete u:
   a) if u is a leaf, delete it. Like case 1. leaf delete.
   b) if u is not a leaf, it must have one child w. Then do case 2. one child delete.

Perfect binary tree

A perfect binary tree of height h is a binary tree of height h with the max number of nodes.

Every perfect binary tree of height h has `2^(h+1) - 1` nodes