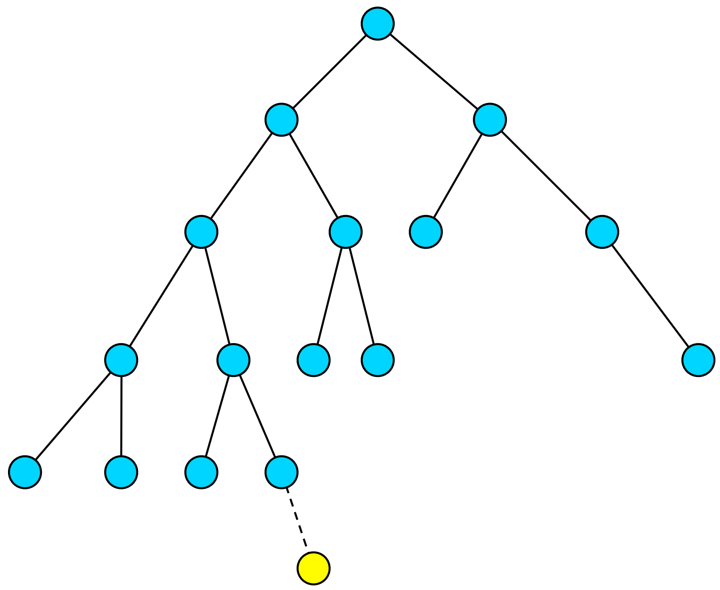
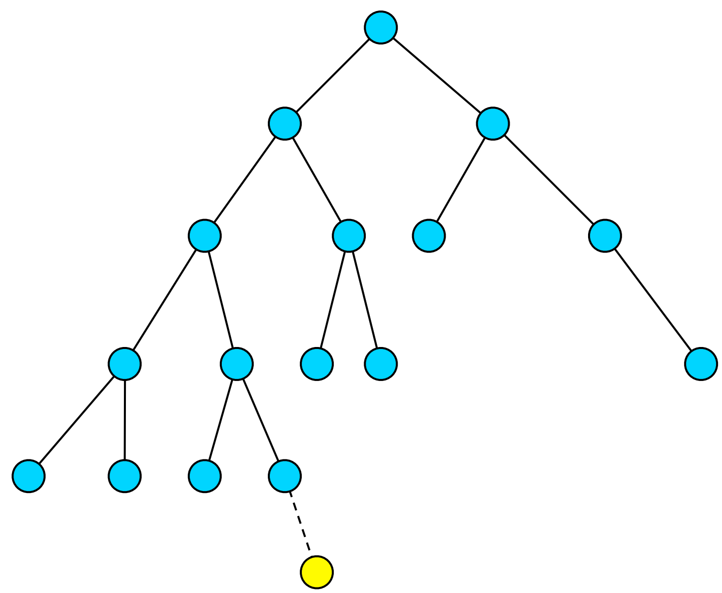
Announcements

MP5 available, due 3/29, 11:59p. EC due 3/15, 11:59p.



AVL trees:

```
struct treeNode {
   T key;
   int height;
   treeNode * left;
   treeNode * right;
};
```



Insert:

insert at proper place

check for imbalance

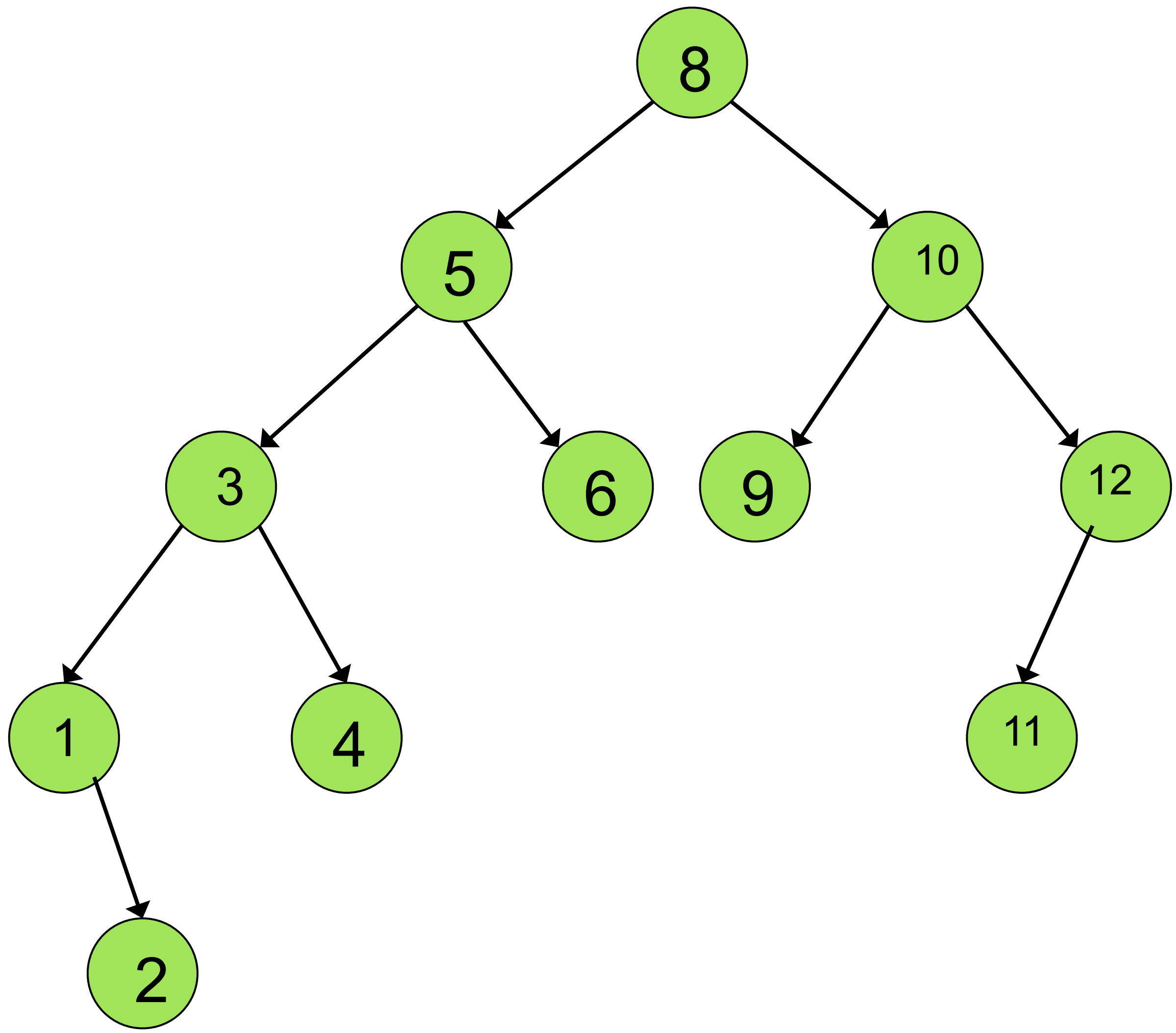
rotate if necessary

update height

AVL tree insertions:

```
template <class T>
void AVLTree<T>::insert(const T & x, treeNode<T> * & t ){
   if( t == NULL ) t = new treeNode<T>( x, 0, NULL, NULL);
   else if( x < t->key ){
      insert( x, t->left );
      int balance = height(t->right)-height(t->left);
      int leftBalance = height(t->left->right)-height(t->left->left);
      if( balance == -2 )
         if( leftBalance == -1 )
            rotate_____________( t );
         else
            rotate_____________( t );
   }
   else if( x > t->key ){
      insert( x, t->right );
      int balance = height(t->right)-height(t->left);
      int rightBalance = height(t->right->right)-height(t->right->left);
      if( balance == 2 )
         if( rightBalance == 1 )
            rotate_____________( t );
         else
            rotate_____________( t );
      }
   t->height=max(height(t->left ), height(t->right))+ 1;
}
```

AVL tree removal:

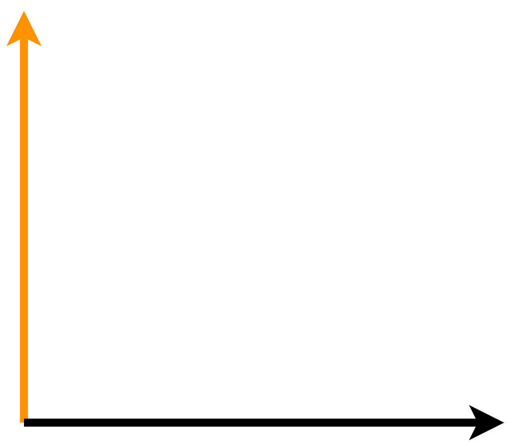


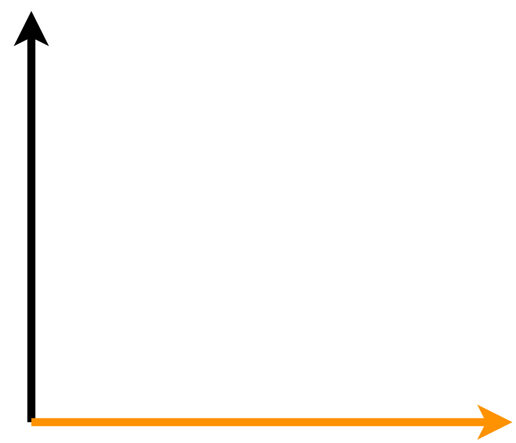
AVL tree analysis:

Since running times for Insert, Remove and Find are O(h), we'll argue that h = O(log n).

- Defn of big-O:

- Draw two pictures to help us in our reasoning:





- Putting an upper bound on the height for a tree of n nodes is the same as putting a lower bound on the number of nodes in a tree of height h.

AVL tree analysis:

Putting an upper bound on the height for a tree of n nodes is the same as putting a lower bound on the number of nodes in a tree of height h.

- Define N(h):
- Find a recurrence for N(h):
- We simplify the recurrence:
- Solve the recurrence: (guess a closed form)

AVL tree analysis: prove your guess is correct.

- Thm: An AVL tree of height h has at least $2^{h/2}$ nodes, ________.

Consider an arbitrary AVL tree, and let h denote its height.

Case 1: ________

Case 2: ________

Case 3: ________ then, by an Inductive Hypothesis that says

________________________________, and since

________________________________, we know that

________________________________.

Punchline:

Classic balanced BST structures:

- Red-Black trees – max ht $2\log_2 n$.

  Constant # of rotations for insert, remove, find.

- AVL trees – max ht $1.44\log_2 n$.

  O(log n) rotations upon remove.

Balanced BSTs, pros and cons:

- Pros:
  - Insert, Remove, and Find are always O(log n)
  - An improvement over:
  - Range finding & nearest neighbor
- Cons:
  - Possible to search for single keys faster
  - If data is so big that it doesn't fit in memory it must be stored on disk and we require a different structure.