Ordered Maps and BSTs

Recall: a map

* Stores key-value pairs (k,v).
* Key values must be unique.
* Must allow for insertion and deletion of (k,v) pairs.
* Should allow for fast search by key value.

Textbook defines the Element class.

All our maps have an iterator p

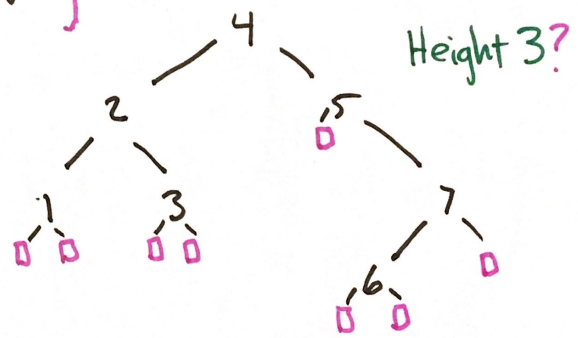
For an ordered map, the iterator produces (k,v) pairs in order by key values.

(So there must be an ordering of keys)

Employees What company?

Number	Name
1	Larry Page
2	Sergey Brin
3	Craig Silverstein
4	Heather Cairns
5	Ray Sidney
6	Harry Cheung
7	Amit Patel

A possible BST



Height 3?

We already know that search is O(h).

We also know there is a bad case when h = n+1 (i.e a list)

The bad case will be addresed later!

What about the iterator?

Our iterator needs to implement inorder traversal. We know how that works!

That only leaves two questions:

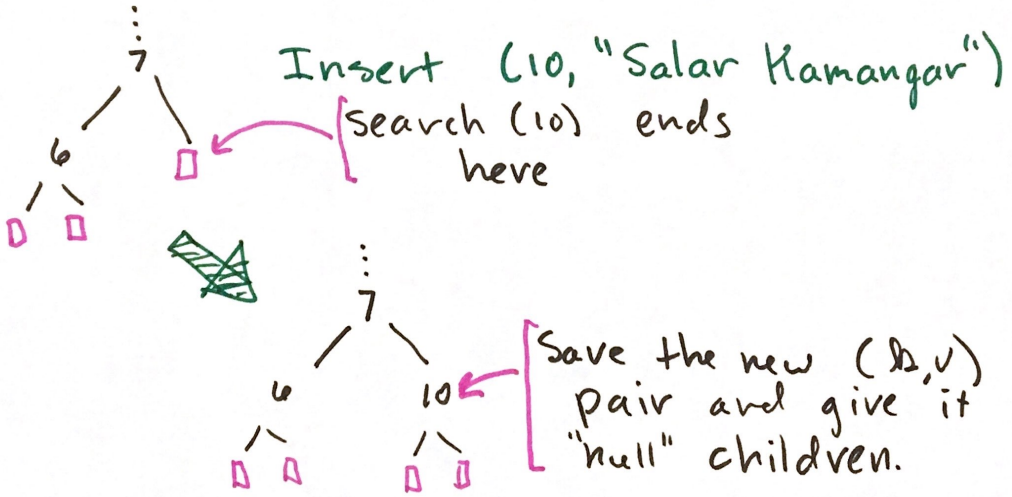
(1) How do we insert a (k, v) element?

(2) How do we erase an element?

Insertion

Given a (k,v) pair:

If k is not already used, search for k will end at the leaf node where (k,v) should be inserted:



Insertion (continued)

If k is already used, search finds it in the tree; update its stored value to v.

For example

```
Insert(1, "Boss")
```

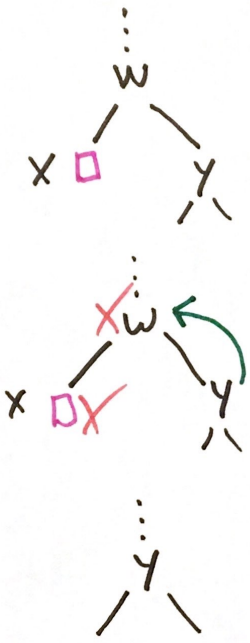
just changes the name for employee #1 from "Larry Page" to "Boss"

Removal (Deletion)

To remove the node with key k, first search the tree to find k.

* If k is not found, it is an error (could throw an exception).
* If k is found, let w denote the node with key k. There are two cases:
  (1) At least one of w's children is external (easy case).
  (2) Both of w's children are internal (less easy case).

Removal - Case (i)



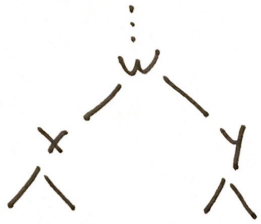
We want to remove w.

Delete w and x.

Move y to w's position

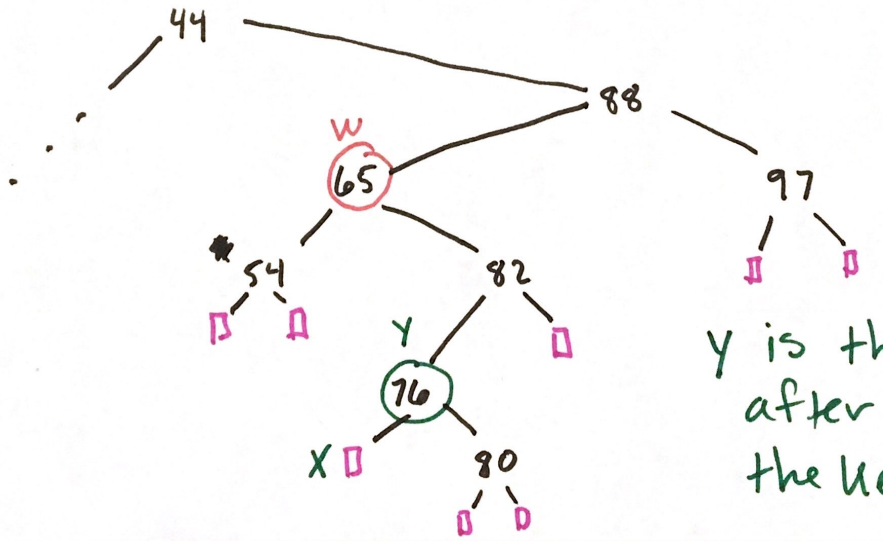
Looks like this.

Removal – Case (2)



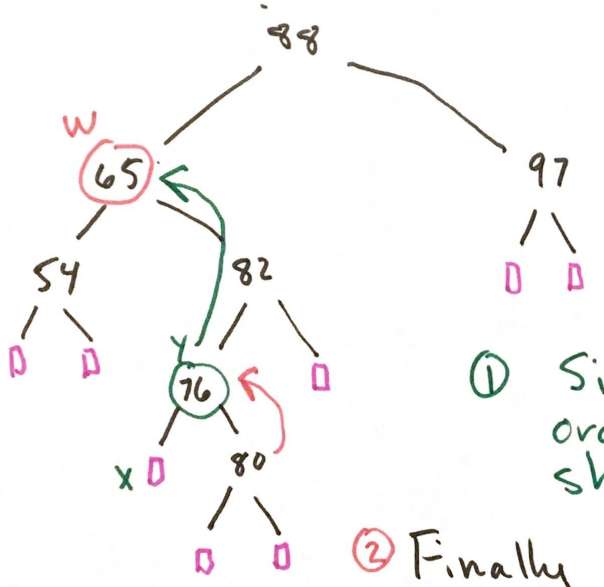
We want to remove w.

Hmm... we need a bigger picture!



y is the next key after w (using the key order)

Removal - Case (2) (Continued)

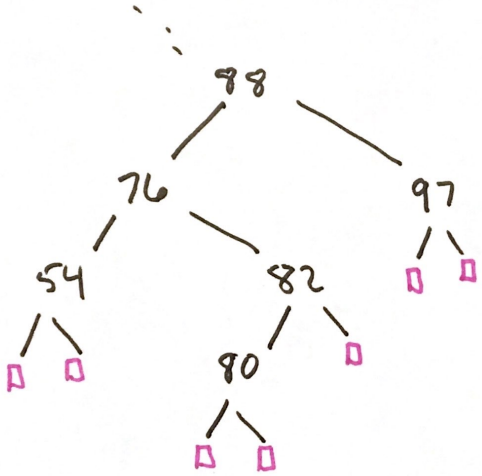


Another way to describe y:
"The left-most internal node of the right subtree of w"

① Since y is next in order after w, it should replace w.

② Finally, remove x and y, replacing y with x's sibling.

Et Voila!



We have maintained the order property of the BST!

Running Time?

Find w.

Descend subtree to find y.

The rest is O(1).

For Case (1): Find w: then O(1) stuff.