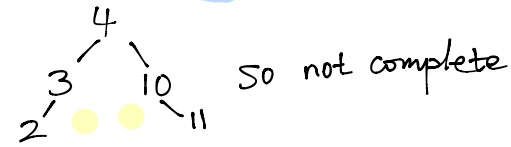
next wednesday midterm 2
cover up and until last week's material.

last week: BSTs, search, insert, delete

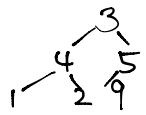
a new ADT: priority queue
example: insert some items with priority
related methods: find max, extract max, insert

implement priority queue with an ADT called Heaps.
binary trees with two invariants:
1. "complete"
2. Heap property
but note that it is different from BST

what does complete mean? "up and left"



So not complete



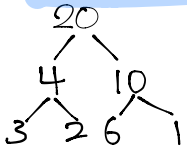
complete ⟹ write in a list [3, 4, 5, 1, 2, 9]
b/c there's no gap

index 1 2 3 4 5 6

left = 2 * index
right = 2 * index + 1

list representation only works for heaps

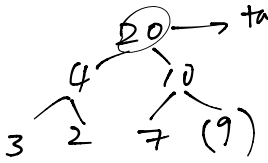
heap property:
node item >= all items in subtrees



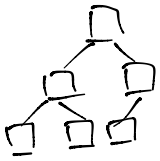

```
def __init__(self):
    self.items = [None]

def is_empty(self):
    return self.items == [None]

#helper
def swap(self, i, j):
    self.items[i], self.items[j] = self.items[j], self.items[i]    # multiple assignment
```

```
def get_max(self):
    if self.is_empty:
        raise IndexError
    else:
        #remember, self.items[1] is the ROOT of the heap
        return self.items[1]
```

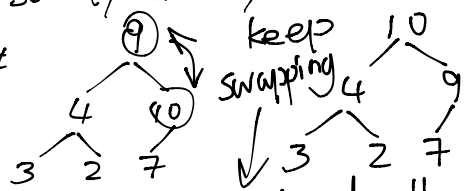
```
def extract_max(self):
    if self.is_empty:
        raise IndexError
    else:
        largest = self.items[1]
        self.items[1] = self.items.pop()
            # last leaf removed
            and make it the new root
        self.bubble_down(1)

    return largest
```



take out 1 node
the final tree
will be like
b/c it's
COMPLETE



So try this way:



until it reaches the right spot

```
def bubble_down(self, index):
    """(Heap, int) -> NoneType
    Make it satisfy Heap property.
    """
```