6.006  
Recitation 3

Agenda
- Overhead
- Warmup
  - defn
  - searching
  - insertion
  - next-largest
- objects in python
- representations of BSTs

Warmup Answers

```
<table>
<thead>
<tr>
<th>5 9 4 7 8 2 1 0 1 6 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4 5 6 7 8 9 10</td>
</tr>
<tr>
<td>10 1 9 2 8 3 7 4 6 5</td>
</tr>
</tbody>
</table>
```

Concept of Binary Search Tree
- **binary**: Splits in 2 each time
- **search**: Optimizes search time

Specs of data structure
- **recursive structure**
- **node holds a value**
- **left child tree must nodes must be ≤ the value**
- **right child tree nodes must be > the value**
example BST

```
  8
 /   \
-4  20
 / \
3.2 10.1
 / \   \n0  4  21   oo
```

find 0, -3, 7, oo

- compare x to root
  - x ≤ root done
  - x > root goto left child
  - x > root goto right child
repeat
(unless child doesn't exist—
in which case x & tree)

why bother?
- can handle ∞ size spaces - not just ints!
- nice insertion properties
- sometimes nice searching properties
- some natural operations easy: min/max

SUXX!
- sorted input → AWFUL tree (degenerate case)
- deletion is a bitch
- some natural operations difficult: next biggest
- no more random indexing

SEARCHING
- best case time: balanced tree: lg(n)
- worst case time: linear: N

INSERTION
- same running times as ↑
- same

NEXT LARGER

given x, find y ∈ BST s.t. x < y and ∀z ∈ BST if z > x z ≥ y ≥ x

case 1: x has a right child subtree y ⇒ min(y)

case 2: no right child ⇔ p(x) > x ⇒ p(x)

case 3: no right child ⇔ p_i(x) ≤ p_{i-1}(x) ≤ ... ≤ p_1(x) ≤ x ⇒ p_i+1(x)

and p_i(x) ≥ p_d(x)


classes in python

```python
class Name:
    class_var = <value>
    def class_fcn(self):
        do stuff
    def fcn2(i, j):
        do more
    def __init__(self):
        auto defined does nothing
        can put more code
    def __init__(self, x, y):
        ...

x = Name()
x = Name(x, y)

example: class counter:
    __count = 0
    def inc(self):
        self.__count += 1
    def inc(self, a):
        self.__count += a
    def __init__(self):
        self.__count = 0
    def __init__(self, x):
        self.__count = 10

x = counter()
x.inc()
x.inc(10)
print x.count
print x.inc(10)
x.name = "foo"
print x.name
def x.name

BST representation

- in class we used code that doesn't make a new class.
node = [val, ] array of 3 elts
    list
  list
- now use objects to make code more understandable.
    [see handout]  I implemented next larger + delete for you

exercise: implement printing all elements in order
          (or rest. an array of all in order)
"
array

\[
\text{low child (i)} \rightarrow 2i \\
\text{high child (i)} \rightarrow 2i + 1
\]

delete frag
For each problem:

1. Construct the Binary Search Tree formed by inserting the given elements in order

2. Determine how many steps the following operations take:
   - `min()`
   - `max()`
   - `find(3)`
   - `find(9)`

\[
\begin{align*}
\text{Initial Order:} & \quad 5 \quad 9 \quad 4 \quad 7 \quad 8 \quad 2 \quad 10 \quad 1 \quad 6 \quad 3 \\
\text{Resultant Order:} & \quad 5 \quad 4 \quad 3 \quad 2 \quad 1 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10 \\
\text{Initial Order:} & \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10 \\
\text{Resultant Order:} & \quad 10 \quad 1 \quad 9 \quad 2 \quad 8 \quad 3 \quad 7 \quad 4 \quad 6 \quad 5
\end{align*}
\]