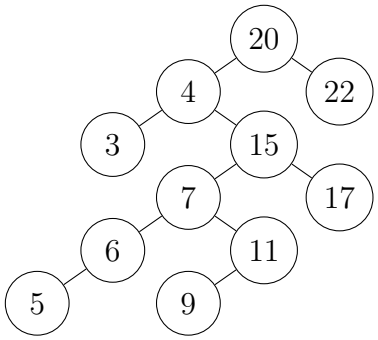
Problem Set 3

All parts are due on March 2, 2018 at 11PM. Please write your solutions in the LaTeX and Python templates provided. Aim for concise solutions; convoluted and obtuse descriptions might receive low marks, even when they are correct. Solutions should be submitted on the course website, and any code should be submitted for automated checking on `py.mit.edu/6.006`.

Problem 3-1. [20 points] **Binary Tree Practice**

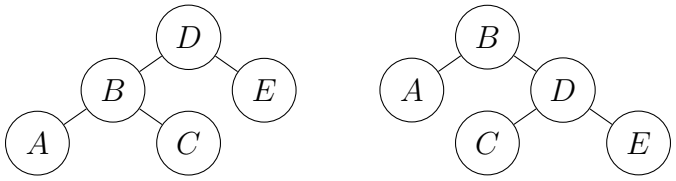
(a) [5 points] Perform the following operations in sequence on the binary search tree T below. Draw the modified tree after each operation.

1. insert key 2
2. delete key 5
3. insert key 13
4. delete key 6
5. delete key 7



(b) [3 points] In the original tree T, list keys from nodes that are **not height balanced**, i.e. the heights of the node's left and right sub-trees differ by more than one.

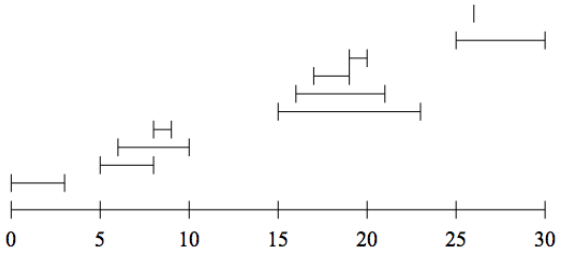
(c) [5 points] A **rotation** rearranges five nodes of a binary search tree between two states, maintaining the binary search tree property as shown below. Perform a sequence of at most four rotations to make the original tree T height balanced. To indicate your rotations, draw the tree after each rotation, circling or listing the keys of the five nodes participating in the rotation.

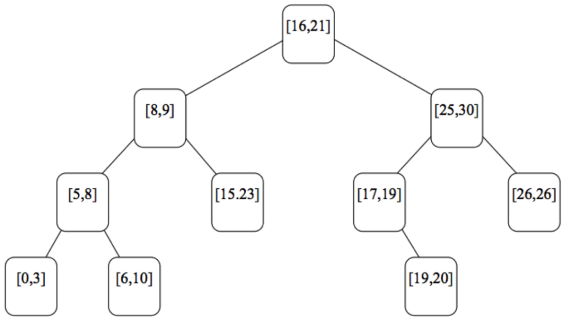


(d) [7 points] Starting at the root, find the minimum key with `min`, then repeatedly find the next key using `successor`. This process will touch every key in the tree via an **in-order traversal**. Draw the path of nodes touched by an in-order traversal on tree T. Because `min` and `successor` each take time proportional to the height h of a tree containing n keys, one can naively calculate a $O(nh)$ upper bound for in-order traversal. Show that in fact, in-order traversal requires at most $O(n)$ time.

Problem 3-2. [20 points] **Interval Search**

We would like to maintain a dynamic set of intervals that supports a query operation: a queried number q should return an interval $[a, b]$ from that set that **contains** q (i.e. $a \leq q \leq b$), or None if none of the intervals contains q. To do this, we will use an augmented BST structure to store these intervals. Each BST node will store an interval, sorting according to the left endpoint of each interval stored. Such a tree is called an **interval tree**. An example of an interval tree and its corresponding intervals are both shown below. You should verify that the interval tree is a BST.





Unfortunately, using the normal BST `find` operation on a query point does not solve our problem. Searching for $q = 22$ would search to the right of the root node $[16, 21]$, while the only interval containing 22 is to the left. To aid you in finding a correct algorithm, we will augment the tree with additional information. For each node x containing interval $[a, b]$ and having children $x.left$ and $x.right$, compute and store the attribute $M(x) = \max(b, M(x.left), M(x.right))$.

(a) [10 points] For the interval tree in the figure above, compute the value of $M(x)$ for each node x in the tree. Argue that you can augment any interval tree with these values in linear time.

(b) [10 points] Using $M(x)$, describe and give pseudocode for an algorithm `query(x,q)`, that finds an interval from the interval tree rooted at x that contains q (or None), and runs in $O(h)$ time, where h is the height of the interval tree. Prove that your algorithm is correct.

Problem 3-3. [20 points] **Consulting**

Briefly describe a database for each of the following clients. Describe any operations that your database should support, and describe how your database supports them. Single operations should run in at most logarithmic time relative the number of items stored in the database (e.g. starting from an empty database, you may use logarithmic time each time you add an item).

(a) [10 points] **PriceTree:** Silliam Whatner runs a discount travel website, which maintains an inventory of available tour packages. Silliam often adds new tour packages to the site. Each package includes an itinerary, a price per person per day, and a duration measured in an integer number of nights. A client tells Silliam a desired tour length and budget per person per day, and Silliam will show them the five most expensive tour packages matching the customer's desired duration that are also within the customer's budget. If a customer decides to purchase a package, Silliam removes the package from the available inventory. Help Silliam design a database to organize the tour package information, and efficiently provide responses to customer requests.

(b) [10 points] **Leywand Corporation:** Ren Lipley is the leader of a group of space explorers colonizing a new planet. Their base is at the center of a large, perfectly-circular island of hospitable terrain, surrounded by an ocean of toxic liquid. Living on the coastline are thousands of alien creatures with strange movement patterns. While they can't seem to walk, they can instantaneously teleport to other locations on the coast, sometimes multiple times per day. Each day, space biologist teams journey to the coast to study and tag coastal aliens. Throughout the day, each team may ask the base to identify the twenty closest tagged aliens to their location. The base is equipped with a scanner that can detect and log the start and end locations of any tagged coastal alien when it teleports. Describe a database for the base to keep track of the tagged coastal aliens, and to support the research teams.

Problem 3-4. [40 points] **Programming**

Cole the cold coder is fed up. Every day he gets up, dress appropriately for the weather from yesterday, and leaves for class. But alas! The weather is drastically different: now either too hot or too cold. Cole decides to build a system to predict today's temperature given only the temperature from the day before, using a crazy new AI technique called *wide learning* (which he hears is all the rage). Like deep learning, wide learning attempts to approximate a function $y = f(x)$ by looking at many examples, i.e., pairs (x_i, y_i) of observed inputs and outputs. Cole's wide learning algorithm will accept two inputs: yesterday's temperature x and a confidence interval w. The algorithm will return a prediction for today's temperature y^* by taking the average temperature from all example outputs y_i from the wide set of sample inputs x_i that are within w degrees of x (i.e. $|x - x_i| < w$). Cole writes a quick Python program to implement his algorithm:

```
examples = []

def add_example(xi, yi):
    examples.append((xi, yi))

def predict(x, w):
    total, count = 0, 0
    for xi, yi in examples:
        if abs(xi - x) < w:
            total += yi
            count += 1
    return (total / count) if (count != 0) else 0
```

(a) [5 points] What is the running time of Cole's prediction function terms of the number of examples? What about the running time of his `add_example` function?

Cole is tired of waiting so long for his predictor to run. He decides to store his examples in an augmented binary search tree, hoping that adding an example and making a prediction will each be fast, requiring at most logarithmic time.

(b) [10 points] To compute averages in sub-linear time, Cole cannot afford to visit every node contained in a range. Prove that at most **one** node of a BST can have left and right sub-trees that each contain keys both inside and outside a given range. If you know that every node in a sub-tree is within range, can you think of a way to pre-compute information to store at each node to shortcut computation?

(c) [10 points] Describe how Cole can augment his binary search tree to efficiently add examples and calculate predictions, each in logarithmic time with respect to the number of examples stored in the tree. Remember to argue correctness and demonstrate the running time of your algorithm.

(d) [15 points] Implement `predict(x, w)` in a Python class `TemperatureLog` that extends the `AVL` class provided. You can download a code template containing some test cases from the website. Submit your code online at `py.mit.edu/6.006`.

```
class TemperatureLog(AVL):
    def __init__(self, key = None, parent = None):
        "Augment AVL with additional attributes"
        super().__init__(key, parent)
        ######################################################
        # TODO: Add any additional sub-tree properties here #
        ######################################################

    def update(self):
        "Augment AVL update() to fix any properties calculated from children"
        super().update()
        ############################################################
        # TODO: Add any maintenence of sub-tree properties here #
        ############################################################

    def add_sample(self, x, y):
        "Add a transaction to the transaction log"
        super().insert((x, y))

    def predict(self, x, w):
        '''
        Return a temperature estimate given:
            x: yesterday's temperature
            w: confidence interval
        If there are no samples within the confidence interval, return 0.
        '''
        if self.key is None:
            return None
        ######################
        # TODO: Implement me #
        ######################
        return 0
```