Problem Session 4

Problem 4-1. Sequence Rotations

Below is a Sequence AVL Tree `T`. Perform operation `T.delete_at(8)` and draw the tree after each rotation operation performed during the operation.

```
         ______6_____
    _____4____   __12_____
  __11__   __5   7    __10
__3    1   9          8
2
```

Problem 4-2. Fick Nury

Fick Nury directs an elite group of n superheroes called the Revengers. He has heard that supervillian Sanos is making trouble on a distant planet and needs to decide whether to confront her. Fick surveys the Revengers and compiles a list of n polls, where each poll is a tuple matching a different Revenger's name with their integer opinion on the topic. Opinion $+s$ means they are for confronting Sanos with strength s, while opinion $-s$ means they are against confronting Sanos with strength s. Fick wants to generate a list containing the names of the $\log n$ Revengers having the strongest opinions (breaking ties arbitrarily), so he can meet with them to discuss. For this problem, assume that the record containing the polls is **read-only** access controlled (the material in classified), so any computation must be written to alternative memory.

(a) Describe an $O(n)$-time algorithm to generate Fick's list.

(b) Now suppose Fick's computer is only allowed to write to at most $O(\log n)$ space. Describe an $O(n \log \log n)$-time algorithm to generate Fick's list.

Problem 4-3. SCLR

Stormen, Ceiserson, Livest, and Rein are four academics who wrote a very popular textbook in computer science, affectionately known as SCLR. They just found k first editions in their offices, and want to auction them off online for charity. Each bidder in the auction has a unique integer bidder ID and can bid some positive integer amount for a single copy (but may increase or decrease their bid while the auction is live). Describe a database supporting the following operations, assuming n is the number of bidders in the database at the time of the operation. For each operation, state whether your running time is worst-case, expected, and/or amortized.

`new_bid(d, b)`	record a new bidder ID d with bid b in $O(\log n)$ time
`update_bid(d, b)`	update the bid of existing bidder ID d to bid b in $O(\log n)$ time
`get_revenue()`	return revenue from selling to the current k highest bidders in $O(1)$ time

Problem 4-4. Receiver Roster

Coach Bell E. Check is trying to figure out which of her football receivers to put in her starting lineup. In each game, Coach Bell wants to start the receivers who have the highest **performance** (the average number of points made in the games they have played), but has been having trouble because her data is incomplete, though interns do often add or correct data from old and new games. Each receiver is identified with a unique positive integer jersey number, and each game is identified with a unique integer time. Describe a database supporting the following operations, each in **worst-case** $O(\log n)$ time, where n is the number of games in the database at the time of the operation. Assume that n is always larger than the number of receivers on the team.

`record(g, r, p)`	record p points for jersey r in game g
`clear(g, r)`	remove any record that jersey r played in game g
`ranked_receiver(k)`	return the jersey with the k^{th} highest performance

Problem 4-5. Warming Weather

Gal Ore is a scientist who studies climate. As part of her research, she often needs to query the maximum temperature the earth has observed within a particular date range in history, based on a growing set of measurements which she collects and adds to frequently. Assume that temperatures and dates are integers representing values at some consistent resolution. Help Gal evaluate such range queries efficiently by implementing a database supporting the following operations.

`record_temp(t, d)`	record a measurement of temperature t on date d
`max_in_range(d1,d2)`	return max temperature observed between dates d_1 and d_2 inclusive

To solve this problem, we will store temperature mesurements in an AVL tree with binary search tree symantics keyed by date, where each node `A` stores a measurement `A.item` with a date property `A.item.key` and temperature property `A.item.temp`.

(a) To help evaluate the desired range query, we will augment each node with: `A.max_temp`, the maximum temperature stored in `A`'s subtree; and both `A.min_date` and `A.max_date`, the minimum and maximum dates stored in `A`'s subtree respectively. Describe a $O(1)$-time algorithm to compute the value of these augmentations on node `A`, assuming all other nodes in `A`'s subtree have already been correctly augmented.

(b) A subtree **partially overlaps** an inclusive date range if the subtree contains at least one measurement that is within the range **and** at least one measurement that is outside the range. Given an inclusive date range, prove that for any binary search tree containing measurements keyed by dates, there is at most one node in the tree whose left and right subtrees both partially overlap the range.

(c) Let `subtree_max_in_range(A, d1, d2)` be the maximum temperature of any measurement stored in node `A`'s subtree with a date between d_1 and d_2 inclusive (returning `None` if no measurements exist in the range). Assuming the tree has been augmented as in part (a), describe a **recursive** algorithm to compute the value of `subtree_max_in_range(A, d1, d2)`. If h is the height of `A`'s subtree, your algorithm should run in $O(h)$ time when `A` partially overlaps the range, and in $O(1)$ time otherwise.

(d) Describe a database to implement operations `record_temp(t, d)` and `max_in_range(d1, d2)`, each in **worst-case** $O(\log n)$ time, where n is the number of unique dates of measurements stored in the database at the time of the operation.

(e) Implement your database in the Python class `Temperature_DB` extending the `Set_AVL_Tree` class provided; you will only need to implement parts (a) and (c) from above. You can download a code template containing some test cases from the website. Submit your code online at `alg.mit.edu`.