CS2040S: Data Structures and Algorithms

Discussion Group Problems for Week 5

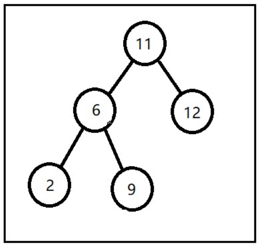
For: Feb. 8–Feb. 12

1 Review

Discuss questions, if you have any, with the tutor and the rest of the class, about the material and content so far.

2 Problems

Problem 1. Review AVL tree insertion and deletion algorithms in class. What would the tree be like after inserting 7? What if we delete 6 after that?



Problem 2. (A few puzzles involving duplicates or array processing)

Problem 2.a. Given an array A, decide if there are any duplicated elements in the array.

Problem 2.b. Given an array A, output another array B with all the duplicates removed. Note the order of the elements in B does not need to follow the same order in A. That means if array A is $\{3, 2, 1, 3, 2, 1\}$, then your algorithm can output $\{1, 2, 3\}$.

Problem 2.c. Given arrays A and B, output a new array C containing all the items in both A and B. You are given that array A and array B already have their duplicates removed.

Problem 2.d. Given array A + target value, output two elements x and y in A where (x+y) equals the target value.

Problem 3. (**Chicken Rice**)
Imagine you are the judge of a chicken rice competition. You have in front of you n plates of chicken rice. Your goal is to identify which plate of chicken rice is best.

Problem 3.a. A simple algorithm:

- Put the first plate on your table.
- Go through all the remaining plates. For each plate, taste the chicken rice on the plate, taste the chicken rice on the table, decide which is better. If the new plate is better than the one on your table, replace the plate on your table with the new plate.
- When you are done, the plate on your table is the winner!

Assume each plate begins containing n bites of chicken rice. When you are done, in the worst-case, how much chicken rice is left on the winning plate?

Problem 3.b. Oh no! We want to make sure that there is as much chicken rice left on the winning plate as possible (so you can take it home and give it to all your friends). Design an algorithm to maximize the amount of remaining chicken rice on the winning plate, once you have completed the testing/tasting process. How much chicken rice is left on the winning plate? How much chicken rice have you had to consume in total? (Give a tight asymptotic bound.)

Problem 3.c. Now I do not want to find the best chicken rice, but (for some perverse reason) I want to find the median chicken rice. Again, design an algorithm to maximize the amount of remaining chicken rice on the median plate, once you have completed the testing/tasting process. How much chicken rice is left on the median plate? How much chicken rice have you had to consume in total? (Give a tight asymptotic bound. If your algorithm is randomized, give your answers in expectation.)

Problem 4. (**Order Maintenance**)

The goal of the order maintenance problem is to maintain a total order over some (unspecified) objects. The data structure supports two operations:

- `InsertAfter(A, B)`: insert B immediately after A;
- `InsertBefore(A, B)`: insert B immediately before A;
- `isAfter(A, B)`: is B after A in the total order?

Notice that the insert operation adds B directly after A, while the query operation isAfter asks whether B is anywhere after A in the total order.

Problem 5. What solutions did you find for Contest 1 (Catch the Spies)?

Problem 6. (**Ancestor Queries**) (**Just for fun**)

Our job is now to *simulate* a binary tree. Each node has zero, one, or two children, and the tree is of height h. Unfortunately, it is not a balanced tree. By pre-processing the binary tree, design and implement an auxiliary data structure to support the following operations efficiently:

- `InsertLeft(x, y)`: insert y as a left child of x in the binary tree.
- `InsertRight(x, y)`: insert y as a right child of x in the binary tree.
- `isAncestor(x, y)`: is x an ancestor of y in the binary tree that contains them?

Hint: Think about how you answer the `isAncestor(x, y)` query without the extra data structure. What would be the cost of that operation? How can you improve this?