CS 6212 August 25, 2023
Youssef

Homework 1
Due Date: September 15, 2023

Problem 1: (20 points)

a. Show by induction on n that $1^2 + 2^2 + 3^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$.

b. Let $T(n)$ be defined recursively as follows: $T(1) = c,$ and $T(n) = 5T\left(\frac{n}{5}\right) + c\ \forall n \geq 5$ where c is an arbitrary constant, and $\boldsymbol{n = 5^k}$ for some non-negative integer $k.$ Prove by induction on k that $T(n) = \frac{5c}{4}n - \frac{c}{4}$.

c. Let $T(n)$ be defined recursively as follows: $T(1) = c,$ and $T(n) = 2T\left(\left\lfloor\frac{n}{2}\right\rfloor\right) + cn$, for all integers $n \geq 2$, where c is an arbitrary positive constant and $\lfloor x \rfloor$ for any number x is the *floor* value of x. Prove by induction on n that $T(n) \leq cn\log n + cn$ for all integers $n \geq 1$.

d. Let $T(n)$ be defined recursively as follows: $T(1) = c,$ and $T(n) = 4T\left(\frac{n}{2}\right) + n^5$. Use the Master Theorem to find the Θ value of $T(n)$.

Problem 2: (20 points)

A binary tree is called an *AVL tree* if the subtrees of every node differ in height by at most 1. (Assume the height of an empty tree is “-1”.)

a. Show all AVL trees of height 0 and height 1, and show 8 AVL trees of height 2.

b. A *left-heavy* AVL is an AVL where for every internal node, the height of its left subtree is one more than the height of its right subtree. Show all right-heavy AVL trees of height 0, height 1, height 2, and height 3.

c. Let $N(h)$ be the number of nodes of a left-heavy AVL tree of height h. Express $N(h)$ in terms of $N(h-1)$ and $N(h-2)$.

d. It can be shown that $N(h) = a(\frac{1+\sqrt{5}}{2})^h + b(\frac{1-\sqrt{5}}{2})^h - 1$ for some constants *a* and *b* (you do not have to prove this formula). Find the values of *a* and *b*.

Problem 3: (20 points)

a. Let T be a binary tree and assume that every node has one numerical data field (called “data”) of type double. Write a recursive algorithm that takes as input the tree T and computes the following values: (1) number of nodes in T that have positive values in the data field, (2) number of nodes in T that have negative values in the data field, and (3) number of nodes in T that have value 0 in the data field. Note: You should give one single algorithm (not three separate algorithms) that computes all three numbers together.

b. Write a recursive algorithm that takes as input a binary tree T and returns whether or not the tree is full. Give the time complexity of your algorithm.

(NOTE: Your grade in all the homework assignments and exams depends on the correctness **and** speed of your algorithms.)

Problem 4: (20 points)

a. Build step by step a min-heap for 27, 13, 56, 35, 48, 8, 18, 67, 5, 62, 7 by inserting those elements (in order) into an initially empty heap. Show how the heap looks like (in a tree form and in an array form) after each insertion of the first 10 elements, then show the insertion of the last element (7) step by step in tree form and array form.
b. Perform delete-min() on the min-heap you built in part a. Show how the heap looks like (in a tree form and array form) after each step.
c. Insert 27, 13, 56, 35, 48, 8, 18, 67, 5, 62, 3, 12, 14, 7, 10, 16, 17, 15 (as you read them) into an originally empty binary search tree. Show how the tree looks like after each insertion. Perform delete (13) on this tree step by step, showing how the tree looks like after each step.

Problem 5: (20 points)

Consider the elements 1, 2, ... , 12. Perform the following sequence of Unions (U) and Finds (F) using the *path compression* algorithm (i.e., 3rd implementation), show how the forest looks like after each operation, and display the PARENT array alongside each snapshot of the forest:

U(1,5) U(3,4) U(2,3) U(1,3) U(6,7) U(8,10) U(8,9) U(6,8) U(11,12) U(3,11) U(8,3) F(7)

(Tie-breaking note: in U(i,j), if the two trees rooted at i and j are of equal size, make i the root of the new tree.)

Bonus Problem: (5 points)
Let c be a positive constant, and let $T(n)$ be as follows: $0 \leq T(n) \leq c$ for all $n = 1, 2, 3, \ldots, 15$, and $T(n) = T(\lfloor\sqrt{n}\rfloor) + c\lfloor\sqrt{n}\rfloor$ for all integers $n \geq 16$. Prove that $T(n) = O(\sqrt{n})$ for all positive integers n. Hint: Prove that $\forall n \geq 1, T(n) \leq 2c\sqrt{n}$, by induction on n.