CSCE 750, Homework 4

This assignment covers material from the lectures on Chapters 8, 9, 11, 12, and 13, in preparation for Quiz 4.

NIT1: In the **search problem**, the input is an array A of size n along with a search key k. The output is an integer i such that $A[i] = k$, or -1 if k is not in A. **Prove**, using the decision tree method, that any correct algorithm for this problem based on comparisons ($<, >, \leq, \geq$, and $=$) between elements takes $\Omega(\lg n)$ time.

Page 236, Exercise 9.2-2 [3rd ed. Page 219, Ex 9.2-3]: Write an iterative version of RANDOMIZED-SELECT.

Page 236, Exercise 9.2-3 [3rd ed. Page 220, Ex 9.2-4]: Suppose that RANDOMIZED-SELECT is used to select the minimum element of the array $A = \langle 2, 3, 0, 5, 7, 9, 1, 8, 6, 4 \rangle$. Describe a sequence of partitions that results in a worst-case performance of RANSOMIZED-SELECT. [Optionally use $A = \langle 3, 2, 9, 0, 7, 5, 4, 8, 6, 1 \rangle$ instead to match the 3rd ed.]

Page 241, Exercise 9.3-3: Show how to use SELECT as a subroutine to make quicksort run in $O(n \lg n)$ time in the worst case, assuming that all elements are distinct.

Page 243, Problem 9-1: *Largest i numbers in sorted order*

You are given a set of n numbers, and you wish to find the i largest in sorted order using a comparison-based algorithm. Describe the algroithm that implements each of the following methods with the best asymptotic worst-case running time, and analyze the running times of the algorithms in terms of n and i.

a. Sort the numbers, and list the i largest.

b. Build a max-priority queue from the numbers, and call EXTRACT-MAX i times.

c. Use an order-statistic algorithm to find the ith largest number, partition around that number, and sort the i largest numbers.

Pages 244–245, Problem 9-3(b,c,d): *Weighted median*

Consider n elements $x_1, x_2, \ldots, x_n$ with positive weights $w_1, w_2, \ldots, w_n$ such that $\sum_{i=1}^{n} w_i = 1$. The *weighted (lower) median* is an element x_k satisfying

$$\sum_{x_i < x_k} w_i < \frac{1}{2} \qquad \text{and} \qquad \sum_{x_i > x_k} w_i \leq \frac{1}{2} .$$

For example, consider the following elements x_i and weights w_i:

i	1	2	3	4	5	6	7
x_i	3	8	2	5	4	1	6
w_i	0.12	0.35	0.025	0.08	0.15	0.075	0.2

For these elements, the median is $x_5 = 4$, but the weighted median is $x_7 = 6$. To see why the weighted median is x_7, observe that the elements less than x_7 are x_1, x_3, x_4, x_5, and x_6, and the sum $w_1 + w_3 + w_4 + w_5 + w_6 = 0.45$, which is less than $1/2$. Furthermore, only element x_2 is greater than x_7, and $w_2 = 0.35$, which is no greater than $1/2$.

b. Show how to compute the weighted median of n elements in $O(n \lg n)$ worst-case time using sorting.

c. Show how to compute the weighted median in $\Theta(n)$ worst-case time using a linear-time median algorithm such as SELECT from Section 9.3.

The *post-office location problem* is defined as follows. The input is n points $p_1, p_2, \ldots, p_n$ with associated weights $w_1, w_2, \ldots, w_n$. A solution is a point p (not necessarily one of the input points) that minimizes the sum $\sum_{i=1}^{n} w_i \, d(p, p_i)$, where $d(a, b)$ is the distance between points a and b.

d. Argue that the weighted median is a best solution for the one-dimensional post-office location problem, in which points are simply real numbers and the distance between points a and b is $d(a, b) = |a - b|$.

Page 281, Exercise 11.2-2 [minor rewording of 3rd ed. Page 261, Ex 11.2-2]: Consider a hash table with 9 slots and the hash function $h(k) = k \bmod 9$. Demonstrate what happens upon inserting the keys 5, 28, 19, 15, 20, 33, 12, 17, 10 wiht collisions resolved by chaining.

Page 282, Exercise 11.2-5 [minor rewording of 3rd ed. Page 261, Ex 11.2-5]: You need to store a set of n keys in a hash tabel of size m. Show that if the keys are drawn from a universe U with $|U| > (n-1)m$, then U has a subset of size n consisting of keys that all hash to the same slot, so that the worst-case searching time for hashing with chaining is $\Theta(n)$.

Page 282, Exercise 11.2-6 [minor rewording of 3rd ed. Page 261, Ex 11.2-6]: You have stored n keys in a hash table of size m, with collisions resolved by chaining, and you know the length of each chain, including the length L of the longest chain. Describe a procedure that selects a key uniformly at random from among the keys in the hash table and returns it in expected time $O(L \cdot (1 + 1/\alpha))$.

Page 292, Exercise 11.3-1 [3rd ed. Pages 268–269, Ex 11.3-1, reworded]: You wish to search a linked list of length n, where each element contains a key k along with a hash value $h(k)$. Each key is a long character string. How might you take advantage of the hash values when searching the list for an element with a given key?

NIT2: Write an algorithm that uses a hash table to solve the **element uniqueness** problem:

- Input: An array A of n elements.
- Output: "True" if the elements of A are all distinct, or "False" if A contains at least one pair of duplicate elements.

How efficient is your algorithm in the worst case? How efficient is it under the simple uniform hashing assumption? Can you design a different algorithm, not based on hashing, that performs better?

Page 315, Exercise 12.1-1 [3rd ed. Page 289, Ex 12.1-1]: For the set $\{1, 4, 5, 10, 16, 17, 21\}$ of keys, draw binary search trees of heights 2, 3, 4, 5, and 6.

Pages 319, Exercise 12.2-1 [3rd ed. Page 293, Ex 12.2-1, reworded]: You are searching for the number 363 in a binary search tree containing numbers between 1 and 1000. Which of the following sequences *cannot* be the sequence of nodes examined?

a. $2, 252, 401, 398, 330, 344, 397, 363$.

b. $924, 220, 911, 244, 898, 258, 362, 363$.

c. $925, 202, 911, 240, 912, 245, 363$.

d. $2, 399, 387, 219, 266, 382, 381, 278, 363$.

e. $935, 278, 347, 621, 299, 392, 358, 363$.

Page 320, Exercise 12.2-4 [3rd ed. Page 293, Ex 12.2-4, reworded]: Professor Kilmer claims to have discovered a remarkable property of binary search trees. Suppose that the search for key k in a binary search tree ends up at a leaf. Consider three sets: A, the keys to the left of the search path; B, the keys on the search path; and C, the keys to the right of the search path. Professor Kilmer claims that any three keys $a \in A$, $b \in B$, and $c \in C$ must satisfy $a \leq b \leq c$. Give a smallest possible counterexample to the professor's claim.

Page 337: Exercises 13.2-3, 13.2-4 [Page 314: Exercises 13.2-3 (correction: change "left subtree" to "right subtree"), 13.2-4]

Page 337, Exercise 13.2-3 [3rd ed. Page 314, Ex 13.2-3, corrected]: Let a, b, and c be arbitrary nodes in subtrees α, β, and γ, respectively, in the right tree of Figure 13.2. How do the depths of a, b, and c change when a left rotation is performed on node x in the figure?

Page 337, Exercise 13.2-4 (corrected) [3rd ed. Page 314, Ex 13.2-4]: Show that any arbitrary n-node binary search tree can be transformed into any other arbitrary n-node binary search tree [with the same keys] using $O(n)$ rotations. (*Hint:* First show that at most $n - 1$ right rotations suffice to transform the tree into a right-going chain.)

NIT3: Suppose min-treap T[1] contains the following (key:priority) pairs:

$$(A{:}10), (B{:}7), (C{:}25), (D{:}9), (E{:}23), (F{:}2), (G{:}4), (H{:}5), (I{:}73), (K{:}65) \ .$$

a. Draw T as a binary tree.

b. Show the result of inserting $(J{:}8)$ into T.

c. Show the result of inserting $(J{:}1)$ into T.

For each insertion, list the rotations performed by the insertion.

[1] A *min-treap* differs from the treap described in the notes only in that the priorities from a min-heap rather than a max-heap.