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: Exercises 9.2-2, 9.2-3 (optionally use $A = \langle 3, 2, 9, 0, 7, 5, 4, 8, 6, 1 \rangle$ instead to match the 3rd ed. 9.2-4) [3rd ed. Pages 219–220: Exercise 9.2-3, 9.2-4]

Page 241: Exercise 9.3-3 [rewording of 3rd ed. Page 223: Exercise 9.3-3, asking you to use SELECT as a subroutine]

Pages 243–244: Problems 9-1 (express the run time of each solution in terms of both n and i), 9-3(b,c) (you may assume $x_1 \ldots, x_n$ are distinct). [slight rewordings of 3rd ed. Page 224: Problems 9-1, 9-2b, 9-2c]

Pages 281–282: Exercises 11.2-2, 11.2-5 (you can assume $|U| > nm$), 11.2-6 [minor rewordings of 3rd ed. Page 261: Exercises 11.2-2, 11.2-5, 11.2-6]

Page 292: Exercise 11.3-1 [3rd ed. Pages 268–269: Exercise 11.3-1, slightly reworded]

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: Exercise 12.1-1]

Pages 319–320: Exercises 12.2-1, 12.2-4 [3rd ed. Page 293: Exercises 12.2-1, 12.2-4, slightly reworded]

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]

NIT3: Figure 13.10 shows a treap variant called a min-treap, which differs from the version in the notes only because the priorities from a min-heap rather than a max-heap. Show the result of inserting the key 'J', with priority 8, into the min-treap shown in Figure 13.10f. Then show the result of inserting the key 'J', with priority 1, into the min-treap shown in Figure 13.10f. In each case, list the rotations performed by the insertion.