CMSC351 - Fall 2014, Homework #3

Due on October 20th at the start of class

Name:
Section:

- Grades depend on neatness and clarity.
- Write your answers with enough detail about your approach and concepts used, so that the grader will be able to understand it easily. You should ALWAYS prove the correctness of your algorithms either directly or by referring to a proof in the book.
- Write your answers in the spaces provided. If needed, attach other pages.
- The grades would be out of 100. Four problems would be selected and everyones grade would be based only on those problems. You will also get 25 bonus points for trying to solve all problems.

Problem 1 Consider a hash table of size 7, hash function $h(x) = (3x + 5) \mod 7$, and the following input.

92 98 4 85 86 17 23 79 10 11 99 80.

Show the hash table when the following methods are applied for resolving collision:

(a) Chaining rule.

(b) Linear probing (In this case you can stop once the table is full.)

Problem 2 Construct a heap by inserting the following number one by one (from left to right). Draw the heap after each insertion.

$4, 1, 2, 3, 9, 8, 7, 6, 5, 10.$

Problem 3 Suppose you are given an unsorted array of numbers $x_1, x_2, \ldots, x_n$ and a constant k. The goal is to find whether there are two elements of the array x_i and x_j such that $x_i + x_j = k$.

a) Give an $O(n)$ time algorithm that uses $O(n)$ space.

b) Give an $O(n \log(n))$ algorithm that uses $O(1)$ space.

Problem 4 You are given an array A of size n and a constant number $k \leq n$. Give an $O(n \log(k))$ algorithm to output an array B of size n such that for all $1 \leq i \leq n$, $B[i]$ is the smallest element among $A[i], A[i+1], \ldots, A[i+k-1]$. For the last k elements of the array, just consider the existing elements after them.

Problem 5 Design a data structure such that we have the following operations in $O(\log n)$ where n is the maximum number of insertions:

- INSERT(X): Insert x into the data structure.
- DELETE(X): Delete x from the data structure.
- GET(K): Return the k-th smallest element.

*H*int: Use a binary search tree. You can assume that your binary search tree is balanced.

Problem 6 Consider the following modified version of quicksort algorithm.Pick two elements of the list. Partition based on both of the elements. So the elements smaller than both are to the left, the elements in between are in the middle, and the elements larger than both are to the right.

(a) Give a pseudocode for Partition in this case.

(b) Using the partition procedure, give an algorithm for sorting in this modified version.

(c) What is the average number of comparisons that your partition function uses?

(d) Assuming that partition always splits the array into three equal sized arrays, give a recurrence and analyze the order of the algorithm.

Problem 7 The goal of this exercise is to analyze a selection algorithm. This algorithm runs in $O(n)$ even in the worst case (and not just in expectation).

(a) Assume we split the elements of the array into $\frac{n}{11}$ groups of size 11 each. Find median of each group. Suppose M is median of these medians. Prove that at least $5 \times \frac{n}{22}$ elements are smaller than M and at least $5 \times \frac{n}{22}$ are larger than M.

(b) Write a recurrence for the following recursive algorithm:

---

function SELECTION($A[1, \ldots, n], k$)
  Split the array into $\frac{n}{11}$ subarrays of size 11 each.
  $B \leftarrow$ medians of groups.
  $M \leftarrow$ SELECTION($B, \frac{n}{22}$) (or $M \leftarrow$ Median of medians.)
  Partition A based on M.
  Continue Selection on the appropriate part.
end function

(c) Prove that the algorithm runs in $O(n)$.