CS 242: Data Structures and Algorithms

Second Examination (November 5th, Wednesday)

Due (in class) on 10th November (Monday)

Name: ____________________ *Total Points: 150*

- *This is a* take home, open book, closed person, open computer, closed Internet *exam. However you may discuss the exam with the instructor. Also you can use a computer to check your solutions or to use a tool like Maple but you may not use the Internet to find solutions.*
- *Show your work, as partial credit will be given. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it. If your solution is complex, use examples to help explain your ideas. For each problem, 10% of the grade will be reserved for clarity.*
- *Illegible, stream of consciousness answers will not be graded.*

I. (25 Points) Assume that you have to sort a sequence of keys where the length of the sequence is n and each sequence has only $\lg n$ *distinct* keys. It is easy to sort these sequences in $O(n \log n)$ time using a standard sorting algorithm. Suggest a faster algorithm for these sequences that accomplishes the sorting in worst-case $O(n \lg\lg n)$ time? Would your answer change if we only want expected-case $O(n \lg\lg n)$? (*Your answer for the expected case should not use hashing.*)

II. (25 Points) A **concatenate** operation takes two sets, such that all the keys in one set are smaller than all the keys in the other set, and merges them together. Design an algorithm to concatenate two binary search trees into one binary search tree. The worst-case running time should be $O(h)$, where h is the larger of the heights of the two trees.

III. (25 Points) Design an algorithm that adds a special key to a binary search tree, such that the special key is at the root. (Notice that the problem is to **split** the binary search tree into two trees, one containing keys less than the special key, and one containing keys greater than the special key. These two trees can be used as the left and right children of the root.) The worst-case running time should be $O(h)$, where h is the height of the tree.

IV. (25 Points) Consider the following variation for the 0/1 knapsack problem. We are given n items with sizes s_i, $1 \le i \le n$, and associated values v_i, $1 \le i \le n$, where the values are all positive. We want to maximize the total value of the items in the knapsack (with capacity K), subject to the constraint that there is enough room in the knapsack for the chosen items. *We are no longer restricted to filling the knapsack exactly.* Write out the complete algorithm to solve this variation of the knapsack problem. Your solution should have as much detail as the one given in class.

V. (25 Points) Recall that double hashing uses probes of the form

$$h(k, i) = (h_1(k) + ih_2(k)) \bmod m,$$

where h_1 and h_2 are auxiliary hash functions.

1. Why not let $h_2 = h_1$? Describe any advantages or disadvantages.
2. In many cases, there are many more searches performed than insertions. (For example, in your dictionary programs, there were many more words than unique words, and if a search for a word was successful, no insertion was performed.) Assuming double hashing is used, and that all keys are equally likely, describe how we might improve the average time for searching by increasing the time for insertion.
   (*Hint.* Consider an insertion that used many probes to find an open address. Could any elements already in the hash table be moved to result in a smaller average search time? Which elements, and how?)

VI. (25 Points) Show that any sequence of m MAKE-SET, FIND-SET, and UNION operations, where all the UNION operations appear before any of the FIND-SET operations, takes only $O(m)$ time if both path compression and union by rank (that is, size balancing) are used. What happens in the same situation if only the path-compression heuristic is used?