Name_____

Net-ID_____

Prelim Two Solution

CS211 — Fall 2006

Closed book; closed notes; no calculators. Write your name and Net-ID above. Write your name clearly on ***each*** page of this exam. For partial credit, you must show your work.

Do not begin until we instruct you to do so. You have 90 minutes to complete the problems.

Name_____

1. (*20 points; 2 each*) True or false?

a) **T** F In a hash table using chaining and table-doubling, the worst-case time to remove an item is O(n).

b) T **F** In QuickSort, if you always choose the item in the center of the array as the pivot then QuickSort runs in worst-case O(n log n) time.

c) T **F** If item A is an ancestor of item B in a heap (used as a Priority Queue) then it must be the case that the Insert operation for item A occurred before the Insert operation for item B.

d) **T** F If $f(n) = O(n^2)$ and $g(n) = O(n^3)$ then we can conclude that $f(n) + g(n) = O(n^3)$.

e) **T** F $f(n) + g(n) = O(h(n))$ where $h(\ )$ is defined, for all n, as $h(n) = \max\{f(n), g(n)\}$.

f) **T** F If array A is sorted from smallest to largest then A (excluding A[0]) corresponds to a min-heap.

g) **T** F Operation Insert for a balanced BST takes time O(log n) in the worst case.

h) T **F** Operation Insert for a sorted array takes time O(log n) in the worst case.

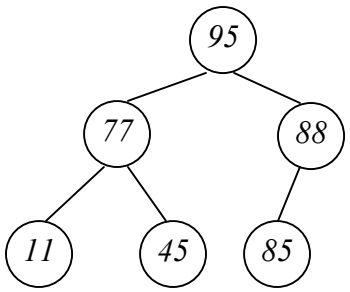
i) **T** F Operation Insert for a heap (using the standard array implementation) takes time O(log n) in the worst case.

j) T **F** Operation Insert for a hash table (using chaining and table-doubling) takes time O(log n) in the worst case.

Name_____

2. (*Heaps; 15 points; 5 each*)

a. The following array represents a heap stored in an array in the manner discussed in lecture and in the text. Show the tree represented by this array.



0	unused
1	95
2	77
3	88
4	11
5	45
6	85

b. Show the contents of the array after 105 is inserted into the original heap using the algorithm for insert discussed in lecture and in the text. It is possible to receive partial credit for this problem, but only if you show your work.

0	*unused*
1	*105*
2	*77*
3	*95*
4	*11*
5	*45*
6	*85*
7	*88*

c. Show the contents of the array after getMax is executed ***twice*** on the ***original heap*** using the algorithm for getMax discussed in lecture and in the text. It is possible to receive partial credit for this problem, but only if you show your work.

0	*unused*
1	*85*
2	*77*
3	*45*
4	*11*
5	
6	

Name_____

3. (*Data Structures; 15 points; 5 each*)
For this question you need to solve the same task using three different algorithms with three different runtimes. The task is as follows: Given an unsorted array of integers, find and print any items that are duplicates. Given the array {3, 2, 4, 3}, the algorithm should print the number 3.

Give your answer in the form of a high-level description or high-level pseudo-code for each time bound below. DO NOT use more than a few lines for your answer; we are not looking for mini-novels. Do not write Java code.

For each problem, the goal is to find an algorithm for which the given runtime is the best possible bound for that algorithm. Algorithms that achieve the desired bound by using a faster algorithm and then doing busy-work to consume the remaining time will not receive credit.

Hint: Use standard data structures and sorting techniques.

a. Print the duplicates in time $O(n^2)$. Is the time bound for your algorithm expected or worst-case?

Use nested for-loops to compare all pairs of numbers. This takes time $O(n^2)$ in the worst-case.

b. Print the duplicates in time O(n log n). Is the time bound for your algorithm expected or worst-case?

Load the items into a balanced BST. Duplicates will be found as an attempt is made to insert them. There are n items and each Insert takes time O(log n) in the worst-case, so total time is O(n log n) in the worst-case.

Alternately, you can sort the items using any O(n log n) sort (e.g., MergeSort, HeapSort, or QuickSort). Once the items are sorted, you can run through the list in linear time to find any duplicates. The total time for this is O(n log n). This is worst-case for MergeSort or HeapSort, expected-case for QuickSort.

You can also load everything into a heap and then pull items out one-at-a-time; the items come out in order, so this is essentially the same idea as the previous method. In effect, you're doing HeapSort.

c. Print the duplicates in time O(n). Is the time bound for your algorithm expected or worst-case?

Load the items into a hash table. Duplicates will be found as an attempt is made to insert them. Each Insert takes expected time O(1), so the total time is O(n), expected.

Name_____

4. (*Data Structures; 20 points; 5 each*)
We want to implement an Abstract Data Type (ADT) with operations Insert, ReportMin, GetMax, and Contains. The operation ReportMin reports the min without deleting it, while GetMax both reports the max and deletes it from the ADT. The operation Contains(x) returns a boolean value indicating whether x is in the ADT or not.

For each of the following requirements, describe a data structure (either a standard data structure or a modification of a standard data structure or a data structure of your own design) that implements the ADT and satisfies the requirements. Your description should be brief, but be sure to give enough description that we can tell what you have in mind. The value *n* represents the number of items in the data structure.

a. Insert, ReportMin, Contains, and GetMax each take time O(log n) in the worst case.

A balanced BST works fine here.

b. Insert and ReportMin each take time O(1) in the worst case.

Use a linked list together with a single variable that holds the current min. During Insert(x), we need to save x in the list [O(1) time] and check the min-variable to see if it needs to be updated [O(1) time]. For ReportMin, we just read out the value of the min-variable. For other operations, we need to take whatever time is necessary to update the min-variable.

c. Insert, Contains, and ReportMin each take expected time O(1).

Use a hash table (with chaining and table-doubling) together with a single variable that holds the current min. During Insert(x), we need to save x in the hash table [O(1) expected time] and, if necessary, update the min-variable [O(1) time]. Contains takes O(1) expected time. ReportMin is trivial [O(1) time]. For other operations, we need to take whatever time is necessary to update the min-variable.

d. GetMax takes time O(1) in the worst-case and Contains takes time O(log n) in the worst case.

For this one, we need a sorted array. For most ADTs, we can find the max quickly, but we can't update the ADT for the next GetMax. A sorted linked-list would work for GetMax, but we need the sorted array in order to implement Contains in O(log n) time (using Binary Search).

Name_____

5. (*Big-O; 10 points; 5 each*)

For each of the following claims either prove the claim true (by providing a witness pair) or prove the claim false (by providing a counterexample). Indicate clearly whether you believe the claim to be true or false.

a. Let $f(n) = kn^3$ where k is a constant, and let $g(n) = n^3$. Claim: $f(n) = O(g(n))$

True.
It's obvious that $kn^3 \leq k(n^3)$ for all values of n.
Thus, by definition, kn^3 is $O(n^3)$ for $c = k$ and $N = 1$.

b. Suppose $f(n) = O(n)$ and $g(n) = O(n \log n)$. Claim: $f(n) = O(g(n))$.

False.
Define f(n) as n; define g(n) as log n. f and g satisfy the requirements (remember that big-O gives an upper bound), but it's obvious that f(n) is not O(g(n)).

Name_____

6. (*Sorting; 20 points; 5 each*)

Consider the following sorting algorithms: InsertionSort, SelectionSort, MergeSort, QuickSort, and HeapSort. For the questions below, assume that each sort is implemented as described in lecture and in the text.

a. Which of the above sorting techniques are stable?

Ins, Mer

b. Which of the above sorting techniques run in worst-case time O(n log n)?

Mer, Hea

c. Which of the above sorting techniques run in time O(n) in the best case?

Ins (You can prepare a Heap in time O(n), but you can't take items out of it fast enough)

d. Which of the above sorting techniques are based on the Divide-and-Conquer strategy?

Mer, Qui