Basic Algorithms (V22.0310); Fall 2005; Yap

HOMEWORK 2

Date Due: Oct 5

Question 1

(10 Points)

Do Problem 2, Chapter 2 (p.67). "Finding the biggest input size you can solve within an hour, for various running times."

Question 2

(10 Points)

Do Problem 4, Chapter 2 (p.67). "Ordering a list of functions by their big-Oh order"

Question 3

(20 Points)

Do Problem 8, Chapter 2 (p.69). "Tradeoffs for stress-testing glass jars". If you get stuck with part (a), send an email to the TA and he will send you a hint.

Question 4

(10 points)

(a) Suppose a binary tree on n nodes has height h. Give an upper bound on n as a function of h. We want you to give a proof by induction on h.

(b) Give an upper bound on h as a function of n.

Question 5

(20 points) I used two concepts in class lectures: eventuality and domination. See below for a recap. Remember that they are just useful alternatives to the big-Oh type notations – you are not really learning a different concept.

(a) Show that $n^k \prec n^{k'}$ for all $k < k'$.

HINT: There are really two statements to show, (i) $n^k \preceq n^{k'}$ and (ii) $n^k \not\succeq n^{k'}$. To show (ii), use proof by contradiction.

(b) Let $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$ for all $n \in \mathbb{N}$. So $H_0 = 0, H_1 = 1, H_2 = 3/2, H_3 = 11/6$. They are called the **harmonic numbers** and arise frequently in algorithm analysis. Show that $H_n \asymp \lg n$.

HINT: Again, there are 2 statements to show, (i) $H_n \preceq \lg n$ and (ii) $H_n \succeq \lg n$. In both cases, first show these results when n is a power of 2. Try to regroup the summation of H_n into $\lg n$ groups, in such a way that each group adds up to $\Theta(1)$.

Question 6

(50 points)

(a) Assume a heap priority queue of size 200. How many comparisons among the input numbers does it take to do an extractMin()?

(b) Suppose we are given an array $A = A[1..n]$ containing some numbers, in no particular order. We want an algorithm to build up a heap priority queue on these numbers. At the end of your algorithm, the array A would be a priority queue.

Here are some rules for your solution: You may assume the subroutines for StartHeap(N), Insert(H,v), etc., in page 64 of text. But you must not use additional arrays (so all the input numbers remain in array A at all times).

Here is the writeup we want: (i) First, briefly describe informally your strategy. (ii) Then make your ideas more concrete by giving a procedure in pseudo-code, in the style of the text (e.g., see the Heapify-down code in p.63). But be sure to give enough details that we can easily turn it into a running Java code.

(c) Run your procedure of part (a) on the input initial array, $A[1..13] = [17, 3, 1, 11, 7, 5, 19, 13, 16, 4, 2, 10, 8]$. Since you are not using additional arrays, all your elements are always in this array A. Thus, at any moment, you can draw the state of your computation by displaying the heap that corresponds to A.

Show the intermediate results by displaying the resulting heap after each call to subroutines such as Insert(H,v), etc.

(d) How may comparisons among the input numbers did your procedure make in part (b)? Show your working, not just a number.

(e) If the input array has size n, give the big-Oh analysis of your procedure.

Practice problems, no credit

Show that $\lg n$ is unbounded (i.e., the function grows arbitrarily large as $n \to \infty$).

Exercise 1, p. 67 (how much slower if you double the input size or increase size by one)

Exercise 3, p.67 (ordering functions by big-Oh order)

Exercise 5, p.68 (true or false)

Can you show that $H_n \prec \sqrt{n}$ by direct argument? HINT: break the sum of H_n into two groups: the first group has the first $\sqrt{n}$ terms.

Eventuality and Domination Notations

If f, g are two real functions, we write

$$f \le g \text{ (ev.)}$$

(read "f is less than or equal to g eventually" if there is an x_0 for all $x > x_0$, $f(x) \le g(x)$. We write

$$f \preceq g$$

(read "f **is dominated by** g" or "g dominates f") if there is a $C > 0$ such that $f \le C \cdot g$ (ev.). Clearly, domination is an alternative way to view the big-Oh notation: $f \preceq g$ iff $f = O(g)$. If $f \preceq g$ but $g \not\preceq f$ then we write

$$f \prec g.$$

Also, if $f \preceq g$ and $g \preceq f$ then we write

$$f \asymp g.$$

Thus $f \asymp g$ iff $f = \Theta(g)$. As you might expect, the triplet of notations $\preceq, \prec, \asymp$ correspond nicely to the well-known relations $\le, <, =$ on real numbers Also, the notations $\succeq, \succ$ are just the reverse of $\preceq, \prec$. For instance, $f \preceq g$ iff $g \succeq f$.