CS/Math 240: Introduction to Discrete Mathematics 6/28/2007

Homework 2

Instructor: Jeff Kinne TA: Mike Kowalczyk

This homework is due at the beginning of class on Thursday July 5, 2007. Mike will hold a review session at 12:45-1:45 on July 5 to discuss the solutions to these problems (that is after you have handed in the homework).

Note: All logarithms are base 2 unless otherwise specified.

Problem 1

Is the following proof correct? If not, what is the error in the proof?

Claim 1. *All people have the same IQ.*

Proof. The proof is by induction. Let $P(n)$ be the assertion that all sets of n people have the same IQ. We want to show that $P(n)$ is true for all $n \geq 1$.

Base Case: For $n = 1$, we have a set of 1 person, so clearly everyone in the set has the same IQ.

Inductive Step: Suppose that for all sets of k people, all people in the set have the same IQ. Consider a set S of size $k+1$. Consider two different subset of S_1 and S_2, each of size k, so that $S = S_1 \cup S_2$. By inductive assumption, all people within set S_1 have the same IQ; and all people within set S_2 have the same IQ. Notice that S_1 and S_2 must have non-empty intersection, call this person in the intersection x. x has some IQ, and this must be the same as all people in both sets S_1 and S_2. We conclude that all people in set S have the same IQ.

By induction, we have shown that all people have the same IQ. $\square$

Problem 2

This problem is concerned with finding the k^{th} smallest number in an unordered list. For the case of $k = 1$, this is the same as finding the smallest number in an unordered list.

Part a)

Give pseudocode for an algorithm to find the k^{th} smallest number. The input and desired output are:

Input: $\{x_1, x_2, ..., x_m\}$, k, with $k \leq m$ and $m \geq 1$.

Correct Output: k^{th} smallest number in the list.

Part b)

Give a correctness proof that the algorithm you designed from Part a) is correct. Your proof should use either a loop invariant or induction.

Part c)

Analyze the number of comparisons that are made by your algorithm in the worst case. Give a Θ estimate, and an exact estimate if possible.

Part d)

The easiest algorithm to give pseudocode for and prove correctness for is not as efficient as possible, in particular for large k. Give the general idea of an algorithm that is faster (running time does not depend on k). If you think your algorithm from part a) already has this property, check with us to make sure.

Problem 3

There is an alternate definition of big-O that makes use of limits: $f(x)$ is $O(g(x))$ if and only if

$$\lim_{x\to\infty} \frac{f(x)}{g(x)} \text{ is finite.}$$

Convince yourself that this definition is equivalent to the one given in class for positive functions. This new characterization of big-O is useful because now we can use tools from real analysis to compare the big-O status of functions. In particular, we can use l'Hôpital's rule to show that a function $f(x)$ is not $O(g(x))$ by showing that $\lim_{x\to\infty} \frac{f(x)}{g(x)}$ is not finite.

We have said that any exponential function is larger than any polynomial function, and that any polynomial function is larger than any logarithmic function. This problem shows that there are functions in between each of these as well.

Part a)

Give a function who's big-O complexity is between logarithmic and polynomial. That is, give $f(x)$ such that: 1) $f(x)$ is not $O(\log(x^d))$ for any constant $d > 0$, and 2) x^d is not $O(f(x))$ for any constant $d > 0$. Use the definition of big-O to prove that your function has these two properties. You may use the definition given in class, or the one given above.

Part b)

Use the result of part a) to give a function $h(x)$ between polynomial and exponential. That is, give $h(x)$ such that: 1) $h(x)$ is not $O(x^d)$ for any constant $d > 0$, and 2) 2^{x^d} is not $O(h(x))$ for any constant $d > 0$. Prove that your function has these two properties.
Hint: First show that for any functions f and g, if $f(x)$ is not $O(g(x))$, then $2^{f(x)}$ is not $O(2^{g(x)})$.

Problem 4

Consider the summation

$$S(n) = \frac{1}{1\cdot 2} + \frac{1}{2\cdot 3} + ... + \frac{1}{n(n+1)}.$$

So, for example $S(1) = \frac{1}{2}$ and $S(2) = \frac{2}{3}$. Find a formula for this summation by looking at the value of $S(n)$ for small values of n. Then use induction to prove the correctness of the formula for all $n \geq 1$.

Problem 5

In class, we used a proof by induction to show that for any finite set S, the number of subsets of S is equal to $2^{|S|}$. For this problem, give an alternate proof using the following facts:

- The number of subsets of S is equal to the size of the power set of S.
- For finite sets A and B, $|A| = |B|$ if and only if there is a one-to-one and onto function mapping from A to B.
- The number of different bit strings of length n is equal to 2^n.