CMPE 300 ANALYSIS OF ALGORITHMS 19.7.2007

MIDTERM

1. Suppose that you have an algorithm for which you want to measure the performance, but you do not know anything about how the time complexity (execution time) of an algorithm can be measured. So, you decide on running the algorithm on a computer and observing how much clock time (sec., min, etc.) it takes to execute for some sample runs.

   We actually know that this is a bad style of performance analysis and the performance of an algorithm should in fact be stated as $\theta(\ldots)$ (or some other asymptotic notation).

   Explain *clearly* how and why it was decided (by researchers after plenty of research) that performances of algorithms should not be measured by executing on some particular computers, instead should be measured by asymptotic symbols. Make your explanation in steps: Beginning from the method of running on a computer, say why this is not desirable, so how we can improve this approach; then what deficiency still exists in this new approach, so how we can improve it further; …. In other words, via a step by step reasoning, you are asked to arrive at the approach used today (i.e. expressing complexity as $f(n) \in \theta(\ldots)$).

2. Solve the following questions and express the solution in terms of θ class. For each, the θ class is one of those: $\theta(1)$, $\theta(\log n)$, $\theta(n)$, $\theta(n \log n)$, and $\theta(n^2)$. (It is not necessary that each item below will have a different θ class. Some may have the same class, thus some of the above given classes may not be the result for any item.) You may use Master Theorem if needed.

   a) $\sum_{i=1}^{n} \frac{i}{n}$

   b) $\sum_{i=1}^{n} \frac{n}{i}$

   c) How many bits do you need to write 10^n in binary?

   d) $T(n) = 9T(\frac{n}{3}) + n$

   e) $T(n) = T(n-2) + \frac{3}{n}$, n is even and T(0)=0

3. Find the average complexity of linear search algorithm by taking into account the repeated elements. That is, determine A(n,m), where n is the number of elements in the list L and m is the maximum number of distinct elements in the list, $1 \le m \le n$.

   We have the following assumptions:

   - The m distinct elements are drawn from a given fixed set $S=\{s_1,s_2,\ldots,s_m\}$.
   - The searched element $X \in S$.
   - It is equally likely that L[i], $1 \le i \le n$, can be any one of the m distinct elements.

   (Note that we do not require that the list has exactly m distinct elements. The list may have fewer than m distinct elements. Thus, X may or may not exist in the list.)

   An example case for n=10, m=5 is:
   S={1,2,3,4,5}, L=[1 3 3 1 4 5 1 1 5 1], X=3.

4. Suppose that we are given an array A[1:n] with the special property that $A[1] \ge A[2]$ and $A[n-1] \le A[n]$. We say that an element A[i] is a *local minimum* if it is less than or equal to both of its neighbors, or more formally, if $A[i-1] \ge A[i]$ and $A[i] \le A[i+1]$. For example, there are six local minima in the following array (shown in bold):

9	**7**	7	2	**1**	3	7	5	**4**	7	**3**	**3**	4	8	**6**	9

We can obviously find a local minimum in O(n) time by scanning through the array. Write in pseudocode an algorithm that finds a local minimum (it is sufficient to find only one local minimum) in O(log n) time in the worst-case. Explain clearly that your algorithm is correct. Briefly analyze W(n) (you do not need to give a complete analysis; just show that it executes in O(log n) time).

(Note that, with the given boundary conditions, the array must have at least one local minimum. You can assume that n≥3.)

Some equations that you may use:

- $\sum_{i=1}^{n-1} ix^{i-1} = \frac{(n-1)x^n - nx^{n-1} + 1}{(x-1)^2}$
- $\sum_{i=0}^{\infty} x^i = \frac{1}{1-x}, |x| < 1$

Notes:

- Where pseudocode is required, the syntax of the pseudocode must be strictly followed. No points will be given if the syntax is not followed or any other language (e.g. C) is used.
- Questions 1-4 : 25 points
- Time: 1:30 hours
- Close notes and books