Ph.D. Qualification Examination

The Design of Algorithms

1. (10%) Argue that the solution to the recurrence $\mathrm{T}(n) = \mathrm{T}(n/3) + \mathrm{T}(2n/3) + n$ is $\Omega(n \lg n)$ by appealing to a recursion tree.

2. (10%) Argue that the solution to the recurrence $\mathrm{T}(n) = 2\mathrm{T}(n/4) + \sqrt{n}$ where T(n) is constant for $n \leq 2$.

3. (10%) Show that the second smallest of n elements can be found with $n + \lceil \lg n \rceil - 2$ comparisons in the worst case. (Hint: Also find the smallest element.)

4. (15%) Which of the following sorting algorithms are stable: insertion sort, merge sort, heapsort, and quicksort? Give a simple scheme that makes any sorting algorithm stable. How much additional time and space does your scheme entail?

5. (15%) Prove that a binary tree is not full cannot correspond to an optimal prefix code.

6. (15%) Given an O(n^2)-time algorithm to find the longest monotonically increasing subsequence of a sequence of n numbers.

7. (15%) Find a feasible solution or determine that no feasible solution exists for the following system of difference constraints:

   x_1-$x_2 \leq 1$, x_1-$x_4 \leq$-4, x_2-$x_3 \leq 2$, x_2-$x_5 \leq 7$, x_2-$x_6 \leq 5$, x_3-$x_6 \leq 10$, x_4-$x_2 \leq 2$, x_5-$x_1 \leq$-1, x_5-$x_4 \leq 3$, x_6-$x_3 \leq$-8

8. (10%) NP problems.

   (a)、 How can we prove that a problem is NP-hard?

   (b)、 How can we prove that a problem is NP-complete?