CIS 435: Homework 1 (Due: February 3, 2004)

Solve problems 1-3 (Group 1) and either Problem 4 (Group 2) or 5 (Group 3).

Problem 1. (8 points)

Use mathematical induction to show that when n is an exact power of two, the solution of the recurrence

$$T(n) = \begin{cases} 3 & \text{if } n = 1 \\ 2T(n/2) + 4n & \text{if } n = 2^k, \text{ for } k \geq 1 \end{cases}$$

is $T(n) = 4n \lg n + 3n$.

Problem 2. (7 points)

(a) Rewrite insertion sort to sort keys in nonincreasing order. (3 points)

(b) Give pseudocode that implements the algorithm for adding two n-bit binary numbers (bits are 0, 1). (4 points)

Problem 3. (5 points)

You are given three algorithms one whose running time is $32n$ (call it A), one whose running time is $4n \lg n$ (call it B), and one whose running time is n^2 (call it C). Which of the three algorithms is faster? Explain. Which of the three algorithms is asymptotically faster? Explain.

Do ONE of problems 4 and 5 below

Problem 4. (30 points)

```
(a)
1. int Search (int key, int A[0..n-1]) // Search for key in A.
2. int i;
3. for(i=0;i<n;i++)
4.   if (A[i] == key ) return(i);
5. return(-1);
```

(i) If `key` is not in A, how many times (express answer in terms of n) does the == of line 4 is tested? (3 points)
(ii) Suppose that `key` is in A and is equally likely to be in position $A[0]$, or $A[1]$, or ..., or $A[n-1]$. What is he average number of comparisons performed to find `key`? Explain. (3 points)
(iii) If A is sorted, a better method for finding `key` is a procedure that is known as binary search. Binary search is described in Exercise 2.3-5 on page 37 of the textbook. Give pseudocode that implements binary search for searching `key` in a now sorted array A. (4 points).
(b) You are given an array $A[1..n]$. The first $n - 1000$ (we assume implicitly that $n > 1000$) keys of the array have already been sorted among themselves, i.e. $A[1..n - 1000]$ is a sorted sequence. The remaining 1000 keys have no particular order i.e. they may be smaller or larger than the other keys of A. Give a stable and in-place efficient algorithm for sorting $A[1..n]$. **Give** means write your algorithm in pseudocode, or cite an already known algorithm, analyze its running time, and show what the running time is. **Efficient** means your algorithm can not be improved upon as far as running time is concerned. Just naming an algorithm will not gain you points; if you name/give the correct algorithm but the running time you give is not the best possible you will not gain points either. (10 points)
(c) An input array B of n integer keys is given. We would like to identify whether there are duplicate keys in the arrays i.e. whether there exist two keys with the same value. Give an efficient algorithm that determines whether duplicate keys exist in B. Note: Your algorithm suffices to answer with a YES (there are duplicate keys) or NO. You don't need to find the set of duplicate keys. Efficient means that your algorithm cannot be improved upon using the techniques we learnt in class. (10 points)

Problem 5. (30 points)

Do the Programming Module that is outlined in the electronic handout available at the course web page
`http://www.cs.njit.edu/~alexg/courses/cis435/handouts/phw1.ps`
or
`http://www.cs.njit.edu/~alexg/courses/cis435/handouts/phw1.pdf`