CS 325 - Homework Assignment 1

The following problems are from the 3rd edition of Introduction to Algorithms, CLRS. Attempt to solve these problems independently and then discuss the solutions in your Homework discussion groups. Submit a "professional" looking individual solution in Canvas. A subset of the problems will be graded for correctness.

1) (CLRS) 1.2-2. Suppose we are comparing implementations of insertion sort and merge sort on the same machine. For inputs of size n, insertion sort runs in $8n^2$ steps, while merge sort runs in 64nlgn steps. For which values of n does insertion sort beat merge sort?

Note: lg n is log "base 2" of n or $\log_2 n$. There is a review of logarithm definitions on page 56. For most calculators you would use the change of base theorem to numerically calculate lgn.

That is: $\mathrm{lgn} = \log_2 n = \frac{\log n}{\log 2}$. Where $\log \quad n = \log_{10} n$ and is calculated using the log button on your calculator.

2) (CLRS) Problem 1-1 on pages 14-15. Fill in the given table. Hint: It may be helpful to use a spreadsheet or Wolfram Alpha to find the values.

3) (CLRS) 2.3-3 on page 39. Use mathematical induction to show that when n is an exact power of 2, the solution of the recurrence

$$T(n) = \begin{cases} 2\,, & if\ n = 2 \\ 2T\left(\frac{n}{2}\right) + n, & if\ n = 2^k, \text{for } k > 1 \end{cases}$$

is $T(n) = n \lg n$.

4) For each of the following pairs of functions, either f(n) is O(g(n)), f(n) is Ω(g(n)), or f(n) = Θ(g(n)). Determine which relationship is correct and explain.
   a. f(n) = n; g(n) = n^2-100n
   b. f(n) = $n^{0.25}$; g(n) = $n^{0.5}$
   c. f(n) = n; g(n) = $\log^2$ n
   d. f(n) = lgn; g(n) = $\log^2$ 3n
   e. f(n) = log n; g(n) = lg n
   f. f(n) = 2^n; g(n) = $10n^2$
   g. f(n) = e^n; g(n) = 2^n
   h. f(n) = 2^n; g(n) = 2^{n+1}
   i. f(n) = 2^n; g(n) = 2^{2^n}
   j. f(n) = $n2^n$; g(n) = 2^n
   k. f(n) = 2^n; g(n) = n!
   l. f(n) = (n+1)!; g(n) = n!

5) Let f_1 and f_2 be asymptotically positive functions. Prove or disprove each of the following conjectures.

   a. $f_1(n) = O(f_2(n))$ implies $f_2(n) = O(f_1(n))$.

   b. If $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$, then $f_1(n) \cdot f_2(n) = O(g_1(n) \cdot g_2(n))$.

   c. $\max(f_1(n), f_2(n)) = \Theta(f_1(n) + f_2(n))$.

6) **Fibonacci Numbers**:

The Fibonacci sequence is given by : 0, 1, 1, 2, 3, 5, 8, 13, 21, ….. By definition the Fibonacci sequence starts at 0 and 1 and each subsequent number is the sum of the previous two. In mathematical terms, the sequence F_n of Fibonacci number is defined by the recurrence relation

$$F_n = F_{n-1} + F_{n-2} \text{ with } F_0=0 \text{ and } F_1=1$$

An algorithm for calculating the n^{th} Fibonacci number can be implemented either recursively or iteratively.

Example Recursive:

```
fib (n) {
    if (n = 0) {
        return 0;
    } else if (n = 1) {
        return 1;
    } else {
        return fib(n-1) + fib(n-2);
    }
}
```

Example Iterative:

```
fib (n) {
    fib = 0;
    a = 1;
    t = 0;
    for(k = 1 to n) {
       t = fib + a;
       a = fib;
       fib = t;
    }
    return fib;
}
```

CS 325 - Homework Assignment 1

a) Implement both recursive and iterative algorithms to calculate Fibonacci Numbers in the programming language of your choice. Provide a copy of your code with your written homework assignment.

b) Use the system clock to record the running times of each algorithm for n = 5, 10, 15, 20, 30, 50, 100, 1000, 2000, 5000, 10,000. You may need to modify the values of n if an algorithm runs too fast or too slow.

c) Plot the running time data you collected on graphs with n on the x-axis and time on the y-axis. What type of function (curve) best fits each data set? Discuss the differences in the running times of each algorithm.

7) Describe a $\Theta(n \lg n)$- time algorithm that, given a set S of *n* integers and another integer x, determines whether or not there exist two elements in S whose sum is exactly x. Give an example.