CSC 316 – Data Structures

Homework Assignment #1
Due Date: September 1, 2016

Problem 1.1 (12 Points)
Gaussian elimination, the classic algorithm for solving systems of n linear equations in n unknowns, requires about $\frac{1}{3}n^3$ multiplications, which is the algorithm's basic operation.

(a) How much longer should we expect Gaussian elimination to work on a system of 1000 equations versus a system of 500 equation?

(b) You are considering buying a computer that is 1000 times faster than the one you currently have. By what factor will the faster computer increase the sizes of systems solvable in the same amount of time as on the old computer?

Problem 1.2 (10 Points)
Order the following functions according to their order of growth, from the lowest to the highest:
$(n-1)!,\ 5\log_2(n+100)^{10},\ 2^{2n},\ 0.001n^4+3n^3+1,\ \log_2^2 n,\ \sqrt[3]{n},\ 3^n$.

Problem 1.3 (30 Points)
Compute the following sums:

(a) $1+3+5+7+\cdots+999$

(b) $2+4+8+16+\cdots+1024$

(c) $\sum_{i=3}^{n+1} 1$

(d) $\sum_{i=3}^{n+1} i$

(e) $\sum_{i=1}^{n}\sum_{j=1}^{n} ij$

Problem 1.4 (18 Points)
Consider the following algorithm.

```
procedure Secret(A[0..n − 1])
// Input: An array A[0..n − 1] of integers
    minval ← A[0]; maxval ← A[0]
    for i ← 1 to n − 1 do
        if A[i] < minval
            minval ← A[i]
        if A[i] > maxval
            maxval ← A[i]
    return maxval − minval
```

(a) What does this algorithm compute?

(b) What is its basic operation?

(c) How many times is the basic operation executed?

(d) Provide an exact expression for the running time $T(n)$ of the algorithm.

(e) What is the order of $T(n)$?

(f) Suggest an improvement or a better algorithm altogether and indicate the order of its running time. If you cannot find an improvement or a better algorithm, try to prove that in fact no such better algorithm exists.

Problem 1.5 (18 Points)
Consider the following algorithm.

```
procedure Enigma(A[0..n − 1, 0..n − 1])
// Input: An array A[0..n − 1, 0..n − 1] of real numbers
    for i ← 0 to n − 2 do
        for j ← i + 1 to n − 1 do
            if A[i, j] ≠ A[j, i]
                return false
    return true
```

(a) What does this algorithm compute?

(b) What is its basic operation?

(c) How many times is the basic operation executed?

(d) Provide an exact expression for the running time $T(n)$ of the algorithm.

(e) What is the order of $T(n)$?

(f) Suggest an improvement or a better algorithm altogether and indicate the order of its running time. If you cannot find an improvement or a better algorithm, try to prove that in fact no such better algorithm exists.

Problem 1.6 (10 Points)
Find the number $L(n)$ of different ways to climb an n-stage ladder when each step is either one or two stages. For example, $L(3) = 3$, since a 3-stage ladder can be climbed three ways: 1-1-1, 1-2, 2-1. You are only required to *set up* a recurrence relation for $L(n)$, you *do not* have to solve it.

Problem 1.7 (12 Points)
Write recursive function `Swap` that takes as input a string and swaps every two elements of the original string. For instance, when given string "abcdefgh" as input, the function will return "badcfehg." Make sure that the function works even when the length of the input string is an odd integer. For instance, if the input string is "abcdefg" then the function must return "badcfeg." You may provide your answer as pseudocode or a (working) Java program.