**Problem 1**

Show that using induction.

Base case:

Let N

=4

Left side:

Right side:

The left and right side are equal, so the formula is true when N=4

Inductive Hypothesis:

It is true that

Proof:

Left side:

Right side:

Both sides together:

, theorem on textbook page 5

Since both sides are equal, they are indeed equivalent.

**Problem 2**

Recursive method for number of 1s in binary representation of N

Assuming N is a non-negative integer,

private static int binaryOnes(int n)

{

if (n == 1 || n == 0)

return n;

else if (n % 2 == 0)

return binaryOnes(n / 2); // even

else

return binaryOnes(n / 2) + 1; // odd

}

(Scratch work)

10 => 1010

5 = > 101

6=>110

7 => 111

14 =>1110

28 => 11100

30 => 11110

31 => 11111

31/2=15 => 1111

f(15)+1

15 => 1111

**Problem 3**

Show that using induction.

Base:

Let N=2

Left side:

Right side:

The left and right sides are equal, so the base case is true.

Inductive Hypothesis:

It is true that

Proof

Right side:

Left side:

Both sides together:

The function has been simplified, and results in the original function. Therefore, the function must be true.

**Problem 4**

List the functions from lowest to highest order

Hint, make use of rule #3, (log n) ^k grows slowly

First, some of the functions can be ordered through intuition:

, , , ,

By comparing to these known functions, the order of the rest of the functions can be found:

Since approaches zero as N approaches infinity,

Since is greater than 48 if , and is less than ,

Since if a is greater than one, and ,

Since , and ,

Since , and ,

Since if , and ,

Since from Rule 3 in chapter 2 of the textbook, and ,

Using the same reasoning as in the case of ,

Since, as an exponential, it increases faster than the cubic function, but ,

As a result, this order is found:

, which is equivalent to

**Problem 5**

Determine whether , , or both.

Initially, is greater than , but is later exceeded by , and is again exceeded by . To find the values where these occurs:

Using the quadratic formula, the solutions are: , or about 0.159, and , or about 18.841. Therefore:

If or ,

If ,

is initially greater than , and is later exceeded. To find the values where these occurs:

Graphing software shows that . For this problem, it will be assumed that . Therefore:

If ,

If ,

g(N) is always above f(N). Therefore:

f(N) and g(N) alternate, with f(N) originally above, then g(N), and finally with f(N). Finding the points where they alternate:

The two functions alternate at , or about 0.316, and 10. Therefore,

If or ,

If ,

Considering the higher order values, will be higher than . As a result:

**Problem 7**

Give an analysis of the program fragments.

Fragment 1

This case is very simple, and should result in a Big-Oh of N.

N=10,000, approx. 100 microseconds

N=50,000, approx. 550 microseconds

N=100,000, approx. 1,000 microseconds

The runtime follows the N value in a predictable way, so I believe the Big-Oh of fragment 1 is N

Fragment 2

This is also straightforward, and the Big-Oh should be

N=100, approx. 90 microseconds

N=200, approx. 440 microseconds

N=300, approx. 1000 microseconds

N=400, approx. 1700 microseconds

The number of microseconds is roughly equal to N/100, squared, and times 100. For example, N=400, divided by 100 is 4, squared is 16, and times 100 is 1600, which is close to the 1700 microseconds recorded.

Fragment 3

The inner loop, from the N \* N, appears to produce a Big-Oh of . It is enclosed by a loop that has N, so the result should be .

N=10, approx. 12 microseconds

N=20, approx. 80 microseconds

N=30, approx. 400 microseconds

N=40, approx. 800 microseconds

N=50, approx. 1500 microseconds

The time roughly equals N/10, cubed, times ten. For example, N=50, divided by ten and cubed is 125, and times 10 is 1250. It is somewhat close to 1500 microseconds recorded.

Fragment 4

The inner loop appears similar to logN since it goes to i, which is passed in. It is enclosed in a loop that has a Big-Oh of N. Therefore, the whole algorithm should have a Big-Oh of N(logN).

N=100, approx. 80 microseconds

N=200, approx. 260 microseconds

N=300, approx. 550 microseconds

N=400, approx. 850 microseconds

N=500, approx. 1400 microseconds

The resulting time is somewhat close to N logN, if N is first divided by ten, and the result of the equation is multiplied by ten. For example, 500 is divided by ten, then applied to the function. 50\*log50 is about 85, times ten is 850. Is off by a bit, but the other N values are closer.

Fragment 5

This one was complicated, but we assumed it was N log(( logN )^2). The i\*i, like Fragment 3, was assumed to mean a square. And since it’s passed in to the inner for loop similar to Fragment 4, it’s assumed to mean a logarithm. The inner most for loop goes to j, and since its format is also similar to Fragment 4, it’s assumed to be a logarithm too.

N=100, approx. 5 milliseconds

N=100, approx. 9 milliseconds

N=100, approx. 19 milliseconds

N=100, approx. 35 milliseconds