**Analysis of Algorithms**

Advanced Algorithms – Assign#1 **Due date: Sept 7th**

1. Order the following functions by growth rate : N, N1/2, N1.5, N2, NlogN, N(logN)2, NlogN2, 2/N, 2N, 2N/2, 37, N3, and N2logN. Also, Indicate which functions grow at the same rate.
2. Describe the order of magnitude of each of the following code sections, using Big-O notation :

**static** **int** Square\_Root(**int** num) {

**int** i = num;

**while**(i \* i >= num) {

i = i - 1;

}

**return** (i + 1);

}

count = 0;

**for**(i = 1; i <= N; i++)

count++;

**for**(j = 1; j <= N; j++)

count++;

value = N;

count = 0;

**while** (value > 1){

value = value/2;

count++;

}

1. Consider four programs—A, B, C, and D—that have the following performances:

A - O(log *n*)

B - O(*n*)

C - O(*n*2)

D - O(2*n*)

If each program requires 10 seconds to solve a problem of size 1000, estimate the time required by each program for a problem of size 2000.

1. Calculating prefix average of a set of values is an important problem, especially in financial calculations. Use the following link to understand background on prefix averages problem - <http://cs-fundamentals.com/tech-interview/dsa/prefix-averages-algorithm-java-program.php>  
   Consider the algorithms (methods) – prefixAverage1 and prefixAverage2 for calculating prefix averages. The source code is given [here](file:///D:\Teaching\AdvAlgorithms\Assignments\PrefixAverage.java). Implement both the algorithms in your choice of programming language and perform an experiment analysis of their running times under different input sizes. Visualize the running times on a chart, where x-axis represents different input sizes and y-axis represents running times of the algorithms.

**Submission**:

Please upload your solution files to d2l.