Spring 2017

**CS3230 Tutorial (Analysis of Algorithms, week of 30 January)**

1. For each of the following pairs of functions, determine which has a higher order of growth.
   1) $n(n+5000)$ and $2000n^2$
   2) $10^{1000}n^2$ and $10^{-1000}n^3$
   3) $\log_2 n^{1000}$ and $\log_{1000} n$
   4) $(\log_2 n)^2$ and $\log_2 n^2$
   5) $2^{n-1}$ and $2^n$
   6) $(n-1)!$ and $n!$

2. For each of the following functions $f(n)$, find the simplest $g(n)$ such that $f(n) \in \Theta(g(n))$.
   1) $(3n^2 - n + 13)^9$
   2) $\sqrt{25n^4 + 37n^3 + 20n^2 - 7899n + 9854120}$
   3) $7n\log(n^2 + 8n + 16) + (n + 211)^2 \log\frac{n}{2}$
   4) $2^{n+3} + 3^{n-2000}$
   5) $\lceil \log_2(n+1) \rceil$
   6) $\sum_{i=1}^{n} i^{-1}$
   7) $\log n!$

3. Consider the following algorithm:

```
// Input matrix A[1..n,1..n]
for i from 1 to n-1 do for j from i+1 to n do
  if A[i,j] <> A[j,i] then return false fi
od od
return true
```

   What does the algorithm compute? What is the basic operation? How many times is the basic operation executed? What is the efficiency class of this algorithm?

4. You are facing a wall that stretches infinitely in both directions. The wall has a door but you know neither how far away nor in which direction. The door can only be seen when you are right in front of it. Let *n* (unknown to you) be the number of steps between you and the door. Design an algorithm so that you can get to the door in O(*n*) steps.

5. Consider the Tower of Hanoi algorithm discussed in the lecture. How many times is the *i*-th ($1 \le i \le n$) largest disk moved?

6. Find the number of ways to climb an *n*-rung ladder when each step is either one or two rungs.