CSE 321 - Introduction to Algorithm Design
Homework 01

Deadline: 13:00 October 21st, 2016

Some Reminders

Informally	Formally
$O \approx \leq$	**O-notation** $O(g(n)) = \{f(n) :$ there exist positive constants c and n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0\}$.
$\Omega \approx \geq$	**Ω-notation** $\Omega(g(n)) = \{f(n) :$ there exist positive constants c and n_0 such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0\}$.
$\Theta \approx =$	**Θ-notation** $\Theta(g(n)) = \{f(n) :$ there exist positive constants c_1, c_2, and n_0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0\}$.
$o \approx <$	**o-notation** $o(g(n)) = \{f(n) :$ for all constants $c > 0$, there exists a constant $n_0 > 0$ such that $0 \leq f(n) < cg(n)$ for all $n \geq n_0\}$.
$\omega \approx >$	**ω-notation** $\omega(g(n)) = \{f(n) :$ for all constants $c > 0$, there exists a constant $n_0 > 0$ such that $0 \leq cg(n) < f(n)$ for all $n \geq n_0\}$.

1. Prove the statements below whether it is true or not.

   a. It is true for all $f(n)$ and $g(n)$ functions $f(n) \in O(g(n))$ or $g(n) \in O(f(n))$
   b. Let $f(n)$ and $g(n)$ be functions. If $f(n) \in O(g(n))$ then the following is true: $\frac{f(n)}{g(n)} \in O(\frac{f(n)}{g(n)})$
   c. Let $f(n)$ and $g(n)$ be functions and k be an integer. If $f(n) \in O(g(n))$ then the following is true: $f(n)^k \in O(g(n)^k)$

2. Prove or disprove the following statements

   a. $n^3 \in O(2^n)$
   b. $2^n \in o(3^n)$
   c. $n! \in O(100^n)$

3. Write the pseudocode of linear search with repeated elements. Analyze its best case, worst case, and average case complexities.
4. Enigma algorithm is given below.

```
Algorithm Enigma(A[0…n-1,0…n-1])
//input the matrix A 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 is problem size (input size)?
b) What is the main operation of the algorithm?
c) What is calculated by this algorithm?
d) Show this algorithm's complexity using the O, Ω, and Θ asymptotic notations for following states below.
    i. Worst case scenario
    ii. Best case scenario
    iii. Average case scenario

5. You are given the following function.

```
Function myFun2(A[1...n])
// Input       : an array of integers
// Output  : an integer
if (n<=1)
     return 4
else
     return myFun2(A[1...n/3]) + myFun2(A[2n/3...n])
```

a) What is problem size (input size)?
b) Write as a recursive function how many times the main operation of the algorithm, which is addition, is called.
c) By solving the recursive function, calculate how many times the main operation is called in the function.
d) Show this function's complexity using the most proper asymptotic notation.

6. Let A be a set, n be an even number, and A consist of n positive integers. Design an efficient algorithm to separate set A into set A_1 and set A_2, each of which includes n/2 items, in order to maximize the difference between the addition of elements in set A_1 and the addition of elements in set A_2. Analyze and show your algorithm's complexity using proper asymptotic notations.

7. Design a recursive algorithm to find out how many 0s are included in an array which consists of only 0s and 1s. Analyze and show your algorithm's complexity using proper asymptotic notations.
   PS: Algorithm is expected to run by dividing an array into two equal parts.

8. Solve the recursive equations below. Calculate exact values and explain using theta notation.
   a) $T(n) = -4*T(n-1) - 4*T(n-2)$, $T(0) = 0$, $T(1) = 1$
   b) $T(n) = T(n-1) + 6*T(n-2)$, $T(0) = 3$, $T(1) = 6$
   c) $T(n) = -5*T(n-1) - 6*T(n-2) + 42*4^n$, $T(1) = 56$, $T(2) = 278$

9. Give exact solutions for T(n) in each of the following recurrences.
   a) $T(n) = T(n-1) + (n^2 + 1)$, $T(0) = 3$
   b) $T(n) = \sum_{i=1}^{n-1} T(i) + n^2$, $T(1) = 1$
   c) $T(n) = 2T(n-1) - T(n-2) + n$, $T(0) = 0$, $T(1) = 0$

10. Solve the recursive equations below asymptotically.

    a) $f(n) = 3f(n/2) + n^2$, $f(1)=4$
    b) $f(n) = 3f(n/2) + n^2\log n$, $f(1)=1$