数据结构与算法分析 （2017年秋季学期），Assignment #1

截止日期：2017年9月21日星期四

姓名：__________________ 学号：______________________ 班级：_________________

在答题之前，请仔细阅读以下注意事项：

(1) 请**独立**完成作业。

(2) 本作业包含五个大题，请用**英语**简单明确作答。

(3) 请于截止日期前将作业纸质版本交给本班学习委员，学习委员在**9月21日星期四上课开始前**将作业集中交给我。

(4) 请务必**按时提交作业**，从9月21日10点40分开始计时，迟交24小时内作业总分扣30%，迟交24小时之后，该次作业计0分。

1. Define an ADT for a set of integers (remember that a set may not contain duplicates). Your ADT should contain the following seven operations:

a) Insert: insert an integer into a set;

b) Delete: remove an integer from a set;

c) Size: return the size of a set currently;

d) Empty: return if a set is empty;

e) Union: return the union of two sets;

f) Intersection: return the intersection of two sets;

g) SetDifference: return the difference of two sets.

Each operation should be clearly defined in terms of its input and output.

2. State whether each of the following relations is a partial ordering, and explain why or why not.

a) "isFatherOf" on the set of people.

b) "isOlderThan" on the set of people.

c) "noLessThan" on the set of integers.

d) $\{(a,b),(a,a),(b,a)\}$ on the set of $\{a,b\}$.

e) $\{(2,1),(1,3),(2,3)\}$ on the set of $\{1,2,3\}$.

3. Answer the following two questions.

a) Prove that $\boldsymbol{x}^{\log_a y} = \boldsymbol{y}^{\log_a x}$ for any $a > 0$, $x > 0$, and $y > 0$.

b) Derive the closed form of the recurrence relation: $f(n) = 2\, f(n/2) + 2n$, with $f(1) = 1$.
To simplify the problem, you may assume that n is a power of 2. That is, the relation holds for $n = 2^t$ for some non-negative integer t.

4. For each pair of the following functions, determine whether $f(n)$ is in $O(g(n))$, $f(n)$ is in $\Omega(g(n))$, or $f(n) = \Theta(g(n))$.

a) $f(n) = \log(n^2)$; $g(n) = \log n + 7$

b) $f(n) = \log(n^2)$; $g(n) = \sqrt{n}$

c) $f(n) = \log n$; $g(n) = n \log n + n$

d) $f(n) = n$; $g(n) = (\log n)^2$

5. Let ***P*** be an array storing integers.

a) Write in pseudocode an algorithm to find a sub-array of *P* with the largest sum. That is, your algorithm takes as input an array ***P,*** its size ***n***, and returns two array indexes *i* and *j* with $i \leqslant j$, such that the sum: ***P[i]+P[i+1]+...+P[j−1]+P[j]*** is as large as possible.
For example, if ***P***={−1,5,−3,7,−2}, your algorithm should return 1 and 3.

b) Analyze the time complexity of your algorithm in the worst case.