CS 317, Assignment 2

Due on 8 October in class

1. **algorithm** DoSomething(A, m, n)**:**

```
if n − m = 0 then
    return
else
    for i = m to n do
        A_{i,j} ← A_{j,i}
        for j = m to n do
            A_{i,j} ← A_{i,j} + i + j
    DoSomething(A, m + 1, n)
```

algorithm DoSomethingElse(A, m, n)**:**

```
if n − m = 0 then
    return
else
    for i = m to n do
        A_{i,j} ← A_{j,i} + i
    p ← (n − m)/3
    DoSomethingElse(A, m, m + p)
    DoSomethingElse(A, m + p + 1, m + 2p)
    DoSomethingElse(A, m + 2p + 1, n)
```

   (a) Write down the recurrence relation for the time complexity of each of the above algorithms. Justify your answer fully.

   (b) Solve the two recurrence relations you just developed *using the characteristic equation technique* and thus give the running time of each algorithm in Θ notation. Note that I am only asking for the complexity and so you do not need to worry about constants.

2. Consider a recursive version of insertion sort that goes like this: *To sort a list of size n, sort the last $n-1$ elements recursively, then deal with the first element.*

   (a) Write down the algorithm.

   (b) Write down the recurrence relation for the running time of your algorithm.

   (c) Solve the recurrence relation with full details and thus give the running time of your algorithm in Θ notation.

3. Design an algorithm that received a binary tree and returns true iff that tree is a binary search tree. Establish the correctness and analyze the running time of your algorithm.

4. Design a non-recursive algorithm for the CFindset (collapsing find) disjoint set algorithm. Establish the correctness and analyze the running time of your algorithm.

5. The transpose $G^{\mathbb{T}}$ of a directed graph $G = (V, E)$ reverses all its edges that is, $G^{\mathbb{T}} = (V, \{(v, u) \in V \times V : (u, v) \in E\})$. Design efficient algorithms for computing $G^{\mathbb{T}}$ from G for *both* the adjacency-list and adjacency-matrix representations of G. Establish the correctness and analyze the running times of your algorithms.

Make sure you review the submission guidelines posted on the course's Web site before submitting. Note in particular that the only acceptable ways to describe an algorithm are pseudo-code (preferred) or actual code. Textual descriptions in particular are not acceptable.