Analysis of Algorithms: Assignment 3

Due date: September 16 (Thursday)

Problem 1 (4 points)
Determine asymptotic upper and lower bounds for each of the following recurrences. Make your bounds as tight as possible.

(a) $T(n) = 16T(n/4) + n$
(b) $T(n) = 16T(n/4) + n^2$
(c) $T(n) = 2T(n/4) + \sqrt{n}$
(d) $T(n) = T(\sqrt{n}) + 1$

Problem 2 (6 points)
Consider the following sorting algorithm:

```
STOOGE-SORT(A, i, j)
if A[i] > A[j]
    then exchange A[i] ↔ A[j]
if i + 1 ≥ j
    then return
k ← ⌊(j − i + 1)/3⌋
STOOGE-SORT(A, i, j − k)      ▷ first two-thirds
STOOGE-SORT(A, i + k, j)      ▷ last two-thirds
STOOGE-SORT(A, i, j − k)      ▷ first two-thirds again
```

(a) Argue that STOOGE-SORT$(A, 1, n)$ correctly sorts the input array $A[1..n]$.

(b) Give the recurrence for the worst-case running time of STOOGE-SORT and a tight asymptotic (Θ-notation) bound on the worst-case time.

(c) Compare the worst-case time of STOOGE-SORT with that of INSERTION-SORT and MERGE-SORT. Is it a good algorithm?

Problem 3 (bonus)
This problem is optional, and it does not affect your grade for the homework; however, if you solve it, then you will get 2 bonus points toward your **final grade** *for the course. You cannot submit this bonus problem after the deadline.*

Consider the following algorithm, which inputs a natural number n and returns a natural number m. The algorithm uses two data structures for storing intermediate results: (1) a square matrix $A[1..n, 1..n]$ and (2) a set S of natural numbers, which may be represented by an array or linked list. The algorithm calls a MIN-OUTSIDE(S) procedure which finds the minimal natural number that does not belong to the set S; if S is empty, this procedure returns 0.

```
SLOW-COUNTER(n)
for i ← 1 to n
    do for j ← 1 to n
            do S ← ∅          ▷ make the set S empty
               for k ← 1 to i − 1
                  do S ← S ∪ {A[k, j]}      ▷ add the A[k, j] value to S
               for k ← 1 to j − 1
                  do S ← S ∪ {A[i, k]}      ▷ add the A[i, k] value to S
               A[i, j] ← MIN-OUTSIDE(S)      ▷ find the minimal natural number not in S
m ← A[n, n]
return m
```

Give a much faster algorithm that computes the same value m, without using a matrix.