Practice Assignment 4

Due: Thursday, February 18 at 2PM

To get credit, each student must submit their own solutions (which need not be typeset but need to be easily legible) to Canvas by the due date above – no late submissions are allowed. Note that while you must each submit solutions individually, you are free to work through the problem sets in groups. Solutions will be posted shortly after class and may be discussed in class. These will be graded on the basis of *completion* alone, not *correctness*.

1. Try to solve the following recurrence relations and give a Θ bound for each of them:

   (a) $T(n) = 2T(n/3) + 1$
   (b) $T(n) = 5T(n/4) + n$
   (c) $T(n) = 7T(n/7) + n$
   (d) $T(n) = 9T(n/3) + n^2$
   (e) $T(n) = 8T(n/2) + n^3$
   (f) $T(n) = T(n-1) + 2$
   (g) $T(n) = T(n-1) + c^n$, where $c > 1$ is some constant
   (h) $T(n) = 2T(n-1) + 1$

   The following ones are a bit more challenging. Try your best to solve them, but they will not be counted toward completeness.

   (i) $T(n) = 49T(n/25) + n^{3/2}\log n$
   (j) $T(n) = T(n-1) + n^c$, where $c \geq 1$ is a constant
   (k) $T(n) = T(\sqrt{n}) + 1$

(a) $T(n) = 2T(n/3) + 1$.
Θ *bound:* $\Theta(n^{\log_3 2})$.

(b) $T(n) = 5T(n/4) + n$.
Θ *bound:* $\Theta(n^{\log_4 5})$.

(c) $T(n) = 7T(n/7) + n$.
Θ *bound:* $\Theta(n \log n)$.

(d) $T(n) = 9T(n/3) + n^2$.
Θ *bound:* $\Theta(n^2 \log n)$.

(e) $T(n) = 8T(n/2) + n^3$.
Θ *bound:* $\Theta(n^3 \log n)$.

(f) $T(n) = T(n-1) + 2$.
Θ *bound:* $\Theta(n)$.

(g) $T(n) = T(n-1) + c^n, c > 1$.
Θ *bound:* $\Theta(c^n)$.
Use Geometric Series Lemma.

(h) $T(n) = 2T(n-1) + 1$.
Θ *bound:* $\Theta(2^n)$.

(i) $T(n) = 49T(n/25) + n^{3/2}\log n$.
Θ *bound:* $\Theta(n^{3/2}logn)$.

(j) $T(n) = T(n-1) + n^c$.
Θ *bound:* $\Theta(n^{c+1})$.
Lower bound: consider the second half of the series
$1^c + 2^c + ... + n^c \geq (n/2)^c + (n/2)^c + ... + (n/2)^c = (n/2)^{c+1}$.
Upper bound: $1^c + 2^c + ... + n^c \leq n^c + n^c + ... + n^c = n^{c+1}$.

(k) $T(n) = T(\sqrt{n}) + 1$.
Θ *bound:* $\Theta(\log \log n)$.
At level i of the recursion tree, the problem size is $n^{1/2^i}$. Assume at the lowest level, the problem size equals a, where a is a number very close to 1. Then the height of the recursion tree would be $\log_2 \log_a n$ ($n^{1/2^i} = a, 2^i = \log_a n$). At each level, the time complexity is O(1), so the total complexity is $\Theta(\log \log n)$.

2. The well-known mathematician George Polya posed the following false "proof" showing through mathematical induction that actually, all horses are of the same color. Identify what is wrong with this proof.
Base case: If there's only one horse, there's only one color, so of course its the same color as itself.
Inductive case: Suppose within any set of n horses, there is only one color. Now look at any set of $n+1$ horses. Number them: $1, 2, 3, \ldots, n, n+1$. Consider the sets $\{1, 2, 3, ..., n\}$ and $\{2, 3, 4, ..., n+1\}$. Each is a set of only n horses, therefore within each there is only one color. But the two sets overlap, so there must be only one color among all $n+1$ horses.

For the case of 2 horses, the two subsets do not overlap. So we could not go from n=1 to n=2.

3. Given two sorted arrays $a[1, ..., n]$ and $b[1, ..., n]$, given an $O(\log n)$ algorithm to find the median of their combined $2n$ elements. (Hint: use divide and conquer).

For the base case, if n is small, say 1 or 2. We can just directly compute the combined median. For larger n, the algorithm works by comparing the medians of the two array (the median of sorted arrays can be computed in constant time). If the two medians are the same, then the combined median is the same value, because exactly 50% of a and 50% b are smaller than this median. If a's median is greater than b's median, the combined median must be either in the first half of a or the second half of b. So it can be found by recursively finding the combined median for a_L and b_R. In contrast, if a's median is smaller than b's, the combined median must be either in the second half of a or the first half of b and can be found by recursively finding the combined median of a_R and b_L. This algorithm will have one recursive call on a half-sized subproblem and constant time non-recursive computation. That is $T(n) = T(n/2) + c = \Theta(\log n)$.

4. (a) Explain why the following algorithm sorts its input.

```
STOOGESORT(A[0 . . . n − 1])
    if n = 2 and A[0] > A[1]
        swap A[0] and A[1]
    else if n > 2
        k = ⌈2n/3⌉
        STOOGESORT(A[0 . . . k − 1])
        STOOGESORT(A[n − k . . . n − 1])
        STOOGESORT(A[0 . . . k − 1])
```

(b) Would STOOGESORT still sort correctly if we replaced $k = \lceil 2n/3 \rceil$ with $m = \lfloor 2n/3 \rfloor$? *(Hint: what happens when $n = 4$?)*

(c) State a recurrence for the number of comparisons executed by STOOGESORT.

(d) Solve the recurrence. Simplify your answer.

(a) See: https://web.engr.oregonstate.edu/~glencora/wiki/uploads/Induction-to-Debug.pdf

(b) No ... try some small examples of arrays of length 4.

(c) $T(n) = 3T(2n/3) + c$ *for a constant* c.

(d) $T(n) = \Theta(n^{\frac{1}{\log_3 1.5}}) \approx \Theta(n^{2.71})$