CSCI 3104 **Prof. Aaron Clauset**
Problem Set 2 **Spring 2017, CU-Boulder**

Advice 1: For every problem in this class, you must justify your answer: show how you arrived at it and why it is correct. If there are assumptions you need to make along the way, state those clearly.

Advice 2: Purely verbal or purely mathematical reasoning is typically insufficient for full credit. Instead, use mathematics to make your argument more precise and logical, and use language to explain your mathematical reasoning more clearly.

---

1. (15 pts total) Prove or disprove each of the following claims, where $f(n)$ and $g(n)$ are functions on positive values.
   (a) $f(n) = O(g(n))$ implies $g(n) = \Omega(f(n))$
   (b) $f(n) = O(g(n))$ implies $2^{f(n)} = O(2^{g(n)})$
   (c) $f(n) = O((f(n))^2)$

2. (10 pts total) Consider the following Python function:

```
def find_max (L):
    max = 0
    for x in L:
        if x > max:
            max = x
    return max
```

   Suppose list `L` has n elements.

   (a) In asymptotic notation, determine the best case running time as function of n.
   (b) In asymptotic notation, determine the worst case running time as function of n.
   (c) Now, assume `L` is sorted. Give an algorithm that takes asymptotically less time than the above algorithm, but performs the same function. Prove that your algorithm is correct, and explain why this algorithm is useless.

3. (30 pts total) Solve the following recurrence relations using the method specified. Show your work.
   (a) $T(n) = T(n-2) + n$ by "unrolling" (tail recursion)
   (b) $T(n) = 2T(n/2) + n^3$ by the Master method
   (c) $T(n) = 2T(n/2) + n^3$ by the recurrence tree method; include the tree diagram.

(d) $T(n) = 2T(n/4) + \sqrt{n}$ by the Master method

(e) $T(n) = 3T(n/2) + n$ by the recurrence tree method; include the tree diagram

4. (15 pts) Exercise 4.5-4 in CLRS

5. (30 pts total) Professor Snape has n magical widgets that are supposedly both identical and capable of testing each other's correctness. Snape's test apparatus can hold two widgets at a time. When it is loaded, each widget tests the other and reports whether it is good or bad. A good widget always reports accurately whether the other widget is good or bad, but the answer of a bad widget cannot be trusted. Thus, the four possible outcomes of a test are as follows:

Widget A says	Widget B says	Conclusion
B is good	A is good	both are good, or both are bad
B is good	A is bad	at least one is bad
B is bad	A is good	at least one is bad
B is bad	A is bad	at least one is bad

(a) Prove that if $n/2$ or more widgets are bad, Snape cannot necessarily determine which widgets are good using any strategy based on this kind of pairwise test. Assume a worst-case scenario in which the bad widgets are intelligent and conspire to fool Snape.

(b) (Divide) Consider the problem of finding a single good widget from among the n widgets, assuming that more than $n/2$ of the widgets are good. Prove that $\lfloor n/2 \rfloor$ pairwise tests are sufficient to reduce the problem to one of nearly half the size.

(c) (And Conquer!) Prove that the good widgets can be identified with $\Theta(n)$ pairwise tests, assuming that more than $n/2$ of the widgets are good. Give and solve the recurrence that describes the number of tests.

6. (15 pts extra credit)

Asymptotic relations like O, Ω, and Θ represent relationships between *functions*, and these relationships are transitive. That is, if some $f(n) = \Omega(g(n))$, and $g(n) = \Omega(h(n))$, then it is also true that $f(n) = \Omega(h(n))$. This means that we can sort *functions* by their asymptotic growth.[1]

[1]The notion of sorting is entirely general: so long as you can define a pairwise comparison operator for a set of objects $\mathcal{S}$ that is transitive, then you can sort the things in $\mathcal{S}$. For instance, for strings, we use a comparison based on lexical ordering to sort them. Furthermore, we can use any sorting algorithm to sort $\mathcal{S}$, by simply changing the comparison operators $>$, $<$, etc. to have a meaning appropriate for $\mathcal{S}$. For instance, using Ω, O, and Θ, you could apply QuickSort or MergeSort to the functions here to obtain the sorted list.

Sort the following *functions* by order of asymptotic growth such that the final arrangement of functions $g_1, g_2, \ldots, g_{12}$ satisfies the ordering constraint $g_1 = \Omega(g_2)$, $g_2 = \Omega(g_3)$, $\ldots$, $g_{11} = \Omega(g_{12})$.

n	n^2	$(\sqrt{2})^{\lg n}$	$2^{\lg^* n}$	$n!$	$(\lg n)!$	$\left(\frac{3}{2}\right)^n$	$n^{1/\lg n}$	$n \lg n$	$\lg(n!)$	e^n	1

Give the final sorted list and identify which pair(s) functions $f(n), g(n)$, if any, are in the same equivalence class, i.e., $f(n) = \Theta(g(n))$.