**CSCI 3104** **Prof. Aaron Clauset**
**Problem Set 2** **Spring 2014, CU-Boulder**

1. (20 pts) Consider the following Python function:

```
def find(a, target):
    x = 0
    y = len(a)
    while x < y:
        m = (x+y)/2
        if a[m] < target:
            x = m+1
        elif a[m] > target:
            y = m
        else:
            return m
    return -1
```

   Suppose list `a` has $n$ elements and is sorted.

   (a) (10 pts) Using $\Theta$ notation, what is the best case running time as function of $n$?

   (b) (10 pts) Using $\Theta$ notation, what is the worst case running time as function of $n$?

2. (30 pts) In the classic version of Quicksort, the pivot is always chosen as the last element in the input array. When the pivot is compared to another element, we can use any valid comparison operator. Thus, "sorting" generalizes to any set of inputs over which we can mathematically define comparisons.

   Use this version of the algorithm to 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 $g_1 = \Omega(g_2)$, $g_2 = \Omega(g_3)$, $\ldots$, $g_{11} = \Omega(g_{12})$.

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

   (a) (**20 pts extra credit**) For each branch of the recursion tree, identify the chosen pivot element; for each level of the recursion tree and after all pivot elements at that level have been moved into their final location, give the global array ordering.

   (Hint: it may be easier to work out the final ordering by hand and then backtrack through the Quicksort operations to produce the recursion tree.)

   (b) (30 pts) 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., $f(n) = \Theta(g(n))$.

3. (25 pts total) Consider the following recursive function `f`, which takes an integer argument `n` and returns some other integer `f(n)`:

```
f (n) {
    if (n==0) return 3;
    else if (n==1) return 5;
    else {
        val = 3*f(n-1);
        val = val - 2*f(n-2);
        return val;
    }
}
```

   (a) (2 pts) Write down the recurrence relation for the *value* returned by `f(n)`; identify the base cases.

   (b) (6 pts) Using the method of *the characteristic polynomial*, solve the value recurrence relation exactly. (See `http://bit.ly/1dcsNUJ`)

   (c) (4 pts) Show (prove) by induction that your solution is correct.

   (d) (3 pts) Under the `RAM` model of computation, write down the recurrence relation for the *running time* of `f(n)`.

   (e) (4 pts) Give a succinct explanation in terms of the shape of the recursion tree of why $O(2^n)$ is a trivial upper bound on the running time.

   (f) (6 pts) Using the method of characteristic polynomials, solve the time recurrence relation for a *tight* upper bound. (Hint: $O(2^n)$ is not tight.)

4. (15 pts total) Solve the following recurrence relations using the method specified. Show your work.

   (a) (7 pts) $T(n) = T(n-1) + n$ by "unrolling" (tail recursion).

   (b) (8 pts) $T(n) = 2T(n/2) + n^3$ by the Master method.

5. (10 pts) Professor Septima Vector thinks she has discovered a remarkable property of binary search trees. Suppose that the search for key $k$ in a binary search tree ends up in a leaf. Consider three (possibly empty) sets: $A$, the keys to the left of the search path; $B$, the keys on the search path; and $C$, the keys to the right of the search path. Professor Vector claims that any three keys $a \in A$, $b \in B$ and $c \in C$ must satisfy $a \leq b \leq c$. Prove that Vector's claim is false by giving a counterexample that is the *smallest possible.*