CSCI 5454 **Prof. Aaron Clauset**
Problem Set 1, due Feb. 7 **Spring 2012, CU-Boulder**

This is a long problem set. I encourage you to discuss the problems with each other, but remember that you *must* write up your solutions independently *and* list the names of your collaborators (a footnote is okay).
Note: the programming problems (9 and 10) will take considerable time. Plan accordingly.

1. (10 pts total) Acme Corp. has asked Professor Flitwick to develop a faster algorithm for their core business. The current algorithm runs in $f(n)$ time. (For concreteness, assume it take $f(n)$ microseconds to solve a problem of size exactly n.) Flitwick believes he can develop a more clever algorithm that takes only $g(n)$ time, but developing it will take t days. Acme only needs to solve a problem of size n once. Should Acme pay Flitwick to develop the faster algorithm or should they stick with their current algorithm? Explain.

   (a) (5 pts) Let $n = 41$, $f(n) = 2^n$, $g(n) = n^3$ and $t = 17$ days.

   (b) (5 pts) Let $n = 10^6$, $f(n) = n^2$, $g(n) = n \lg n$ and $t = 12$ days.

   (Hint: start with the *largest* n Acme can solve with Flitwick after t days of waiting.)

2. (5 pts) Show that for any real constants a and b, where $b > 0$, that $(n + a)^b = \Theta(n^b)$.

3. (5 pts) Is $2^{n+k} = O(2^n)$, for $k = O(1)$? Is $2^{nk} = O(2^n)$ for $k > 1$? Explain.

4. (15 pts total) Use the (classic, deterministic) QuickSort algorithm to sort the following functions by order of growth such that the final arrangement of functions $g_1, g_2, \ldots, g_{12}$ satisfies the ordering constraint $g_1 = \Omega(g_2)$, $g_2 = \Omega(g_3)$, …, $g_9 = \Omega(g_{12})$.

   QuickSort defines a sequence of comparison operations that take an input array A and returns its elements in sorted order; in this case, the comparison operation is non-trivial, but the principle and goal is the same.

   (a) (10 pts) For each level of the recursion tree, identify the chosen pivot element and give the array ordering after the pivot element has been moved into place.

   (b) (5 pts) Give the final sorted list and identify out which pair(s) functions $f(n), g(n)$, if any, are in the same equivalence class, i.e., $f(n) = \Theta(g(n))$.

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

5. (15 pts total) Consider the following function:

```
int f (int n) {
   if (n==0) return 3;
   else if (n==1) return 5;
   else {
      int 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; identify the base cases.

   (b) (3 pts) Solve the value recurrence relation using the method of characteristic polynomials.

   (c) (2 pts) Show by induction that your solution is correct.

   (d) (2 pts) Write down the recurrence relation for the *running time* of f.

   (e) (3 pts) In three sentences or less, explain why the running time is $O(2^n)$.

   (f) (3 pts) Using the method of characteristic polynomials, solve the time recurrence relation for a *tight* upperbound. (Hint: $O(2^n)$ is not tight.)

6. (10 pts total) Solve the following recurrences using the method specified:

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

   (b) (3 pts) $T(n) = T(n-1) + n$ by "unrolling" (tail recursion)

   (c) (4 pts) $T(n) = 2T(n/2) + n^3$ by the recurrence tree method

7. (15 pts total) Professor Snape has n computer chips that are supposedly both identical and capable of testing each other's correctness. Snape's test apparatus can hold two chips at a time. When it is loaded, each chip tests the other and reports whether it is good or bad. A good chip always reports accurately whether the other chip is good or bad, but the answer of a bad chip cannot be trusted. Thus, the four possible outcomes of a test are as follows:

Chip A says	Chip 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) (5 pts) Show that if more than n/2 chips are bad, Snape cannot necessarily determine which chips are good using any strategy based on this kind of pairwise test. Assume that the bad chips can conspire to fool Snape.

(b) (5 pts) Consider the problem of finding a single good chip from among the n chips, assuming that more than $n/2$ of the chips are good. Show that $\lfloor n/2 \rfloor$ pairwise tests are sufficient to reduce the problem to one of nearly half the size.

(c) (5 pts) Show that the good chips can be identified with $\Theta(n)$ pairwise tests, assuming that more than $n/2$ of the chips are good. Give and solve the recurrence that describes the number of tests.

8. (5 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 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 $a \leq b \leq c$. Give a *smallest possible* counterexample to the professor's claim.

9. (15 pts) Implement the skip list data structure.
No credit if you don't include your source code.

10. (10pts) Define the running time as the number of atomic operations (lookups), not clock time, used to complete a function call. Now, use your implementation from 9. to conduct a numerical experiment that verifies the asymptotic running times of the (i) insert, (ii) delete and (iii) find operations.

The deliverable here is a single figure (like the one below) showing how the number of operations T grows as a function of n. Show two trend lines of the form $O(\log n)$ that bound from above and below your results. Label your axes and the trend lines. Include a clear and concise description (1-2 paragraphs) of exactly how you ran your experiment.

No credit will be given if you don't label your axes and trend lines.

Hint 1: To count the number of operations, you'll need to implement some measurement code inside your skip list data structure to count and report the number of elements you access in the data structure as you complete a function call.

Hint 2: To clearly show the asymptotic behavior, choose at least 10 values of n spaced out logarithmically over several factors of 10, e.g., $\log_2 n = \{4, 5, \ldots 13\}$.

Hint 3: Because skip lists are a randomized data structure, each particular measurement is itself a random variable. Thus, for each value of $T(n)$, you should compute the *average* over several *independent* trials. The greater the number of independent trials you average over, the smoother the trend line. A smooth trend line is your goal.

Hint 4: If your figure doesn't show smooth $O(\log n)$ behavior over a wide range of n, there's almost surely a bug in your implementation.

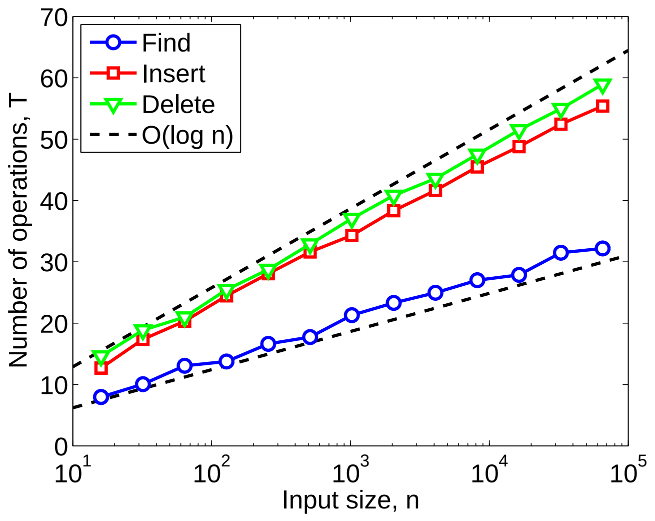

Figure 1: An example of what your skiplist results should look like. (Note: these are not real data.)