Problem Set 7 – Due Wednesday, 4:15 pm, November 20, 2013

1. Suppose you have a choice of four algorithms to solve a problem, with the following approximate running time as a function of input size n:

Algorithm 1 takes	2^n seconds
Algorithm 2 takes	$20n^2$ seconds
Algorithm 3 takes	$1000n \lg n$ seconds
Algorithm 4 takes	$10000n$ seconds

   (a) Which algorithm is asymptotically best? (b) Give the range of n values for which each algorithm is optimal.

2. (a) Rank the following functions by order of growth. That is, partition the 25 functions into a list of sets where: when f and g are in the same set, $f \in \Theta(g)$; in going from one set to the next, if f is in the first and g is in second, then $f \in O(g)$. It is not necessary to show your work.

$$\begin{array}{ccccc} \lg n & \ln n & \lg\lg n & \ln^2 n & \ln\ln n \\ \lg^2 n & \lg(n!) & 2^{\sqrt{\lg n}} & n! & n^{0.001} \\ 1 & n & e^n & 2n^2 & 2^n \\ 2^{\sqrt{n}} & n\lg n & 4^{\lg n} & n^{\lg 7} & n^{\log n} \\ n^{\lg\lg n} & 2^{2^n} & n^2 & n^3 & (2/3)^n \end{array}$$

   (b) Give an example of a function h such that for all of the 25 functions f above, $h \notin O(f)$, nor is $f \in O(h)$.

3. For $n \geq 1$, let $B(n)$ be the number of ways to express n as the sum of 1s and 2s, taking order into account. Thus $B(4) = 5$ because $4 = 1+1+1+1 = 1+1+2 = 1+2+1 = 2+1+1 = 2+2$.

   (a) Compute $B(i)$ for $1 \leq i \leq 5$ by showing all the different ways to write these numbers as above.

   (b) Find a recursive definition for $B(n)$ and identify this sequence.

   (c) Compute $B(10)$.

4. Solve the following recurrence relations to within a $\Theta(\cdot)$ result. Assume that $T(n) \in \Theta(1)$ for sufficiently small n. Use repeated substitution to get your answers.

   (a) $T(n) = 2T(n/5) + n$

   (b) $T(n) = 5T(n/2) + n$

   (c) $T(n) = 5T(n/5) + n$

5. Thomas thinks that he might improve on the binary search of a sorted array A: he will divide the array's elements into thirds instead of halves, recursing, when necessary, on the correct third. (a) Write down pseudocode for an algorithm `TS` that correctly captures the idea of this "trinary search." (b) Write a recurrence relation that gives, precisely, the worst-case number of comparissons used by `TS`. (c) Write a recurrence relation that gives, within a constant, the anticipated worst-case running time of `TS`. (d) Solve this recurrence relation to within $\Theta(\cdot)$.