CS 584/684 Spring 2017 Homework 3 – due noon, Wednesday, April 26 2017

Your solutions to problems 1-4 should be type-set in LaTeX and submitted in both `.tex` and `.pdf` form, with file names `hw3.tex` and `hw3.pdf`. These two files, plus any additional source files invoked from your `.tex` file (such as pictures), should be bundled together into a single `.zip` file named *your-last-name*`-tex-hw3.zip`.

Submit by emailing to `hamialex@pdx.edu` including the zip file as a separate attachment and including "`CS584 HW3`" in the subject line.

All algorithms must be accompanied by proofs of correctness and of running time.

1. Find tight asymptotic solutions $\Theta(...)$ for the following recurrences. In all cases, assume $T(1) = 1$. Justify the correctness of your solutions using the "recursion tree" method. You do not have to be extremely formal, but be careful to justify how you compute the sum of the levels.

(a) $T(n) = 2T(n/2) + \Theta(1)$

(b) $T(n) = T(n/2) + \Theta(1)$

(c) $T(n) = 2T(n/2) + \Theta(n)$

(d) $T(n) = T(n/2) + \Theta(n)$

(e) $T(n) = T(n/2) + \Theta(\lg^p n)$, for constant $p \geq 1$. You may assume the fact that $\sum_{i=0}^{n} i^p = \Theta(n^{p+1})$, which can easily be proven using the method of approximation by integrals (CLRS p. 1154).

(f) $T(n) = 2T(n/2) + \Theta(\lg n)$. You may assume the fact that $\sum_{i=0}^{n} i/2^i = 2 - \frac{n+2}{2^n}$, which can easily be proven by induction.

Parts (e) and (f) are significantly harder than the others. Hints: use the assumed facts; use standard properties of logs; exploit a change of variables in summations (for example, $\sum_{i=0}^{k} f(k-i) = \sum_{j=0}^{k} f(j)$). Latex hint: see the lecture notes from week 1 for one fairly simple way to typeset recursion trees using the `qtree` package.

2. Do CLRS 27.3-3.

3. The *parentheses matching problem* is this: given a string (array of characters) consisting of open and close parentheses '`(`' and '`)`', answer whether the parentheses are properly matched, i.e. (a) the total number of left and right parentheses is equal; and (b) no prefix of the string has more right parentheses than left parentheses.

Give a parallel algorithm for this problem with work $\Theta(n)$ and span $\Theta(\lg n)$ by using map, scan and/or reduce operations on the array, but no other parallel operators. (A *map* operation applies a function independently to each element of the array, producing an array of the same length but perhaps with elements of a different type; it can obviously be parallelized by divide and conquer to run with work $\Theta(n)$ and span $\Theta(\lg n)$.)

4. Do CLRS problem 27-4.