Spring 2013 **CMSC 351: Homework 3** Clyde Kruskal

Due Thursday, February 21. Write your NAME and SECTION on the top of your solution.

Problem 1. Consider the following recurrence.

$$T(n) = 5T(n/2) + 2n - 1, \qquad T(1) = 3 \ .$$

(a) Calculate $T(4)$ by hand. Show your work.

(b) Use the iteration method to solve the recurrence exactly, assuming $n$ is a power of 2.

Problem 2. Use the Master Theorem to solve the following recurrences.

(a) Assume that $n$ is a power of 3.

$$T(n) = 4T(n/3) + 2n, \qquad T(1) = 5 \ .$$

(b) Assume that $n$ is a power of 4.

$$T(n) = 3T(n/4) + 2n - 1, \qquad T(1) = 5 \ .$$

Problem 3. Use the iteration method to solve the following recurrence exactly.

$$T(n) = T(n-1) + 3n + 1, \qquad T(1) = 2 \ .$$

Problem 4. We would like generalize the fast integer multiplication algorithm that we did for two digit numbers to three digit numbers. Assume we multiply the numbers $abc$ and $def$ (or more formally $aB^2 + bB + c$ and $cB^2 + dB + e$ in base $B$).

(a) How many atomic multiplies does the standard algorithm use?

(b) Show how we can do this with only seven atomic multiplications by forming the product $(a + b + c)(d + e + f)$.

(c) Use this to develop a fast integer multiplication algorithm. You can describe it at a high level.

(d) Write a recurrence for the running time of your algorithm. You can assume that the number of digits is nice, and make the simplifying assumptions that we did in class. You can write $\Theta(n)$ for the linear work, rather than calculate it precisely.

(e) What is the order of the running time of your algorithm? You may use the Master Theorem.

(f) How does your value in part (e) compare with the standard algorithm and with the fast algorithm from class?

(g) How few atomic multiplications do we need to do for an algorithm based on multipliying three digit numbers to be better than the fast algorithm from class? You may use the Master Theorem.

"Master Theorem":

$$T(n) = \begin{cases} aT(n/b) + cn^d & n > 1 \\ f & n = 1 \end{cases}$$

implies

$$T(n) = \begin{cases} \left(f + \frac{c}{ab^{-d}-1}\right) n^{\log_b a} - \frac{cn^d}{ab^{-d}-1} = \begin{cases} \Theta(n^{\log_b a}) & a > b^d \\ \Theta(n^d) & a < b^d \end{cases} & \\ n^d(f + c\log_b n) = \Theta(n^d \log_b n) & a = b^d \end{cases}.$$

`Summing solutions:` If

$$T(n) = \begin{cases} aT(n/b) + \sum c_i n^{d_i} & n > 1 \\ f & n = 1 \end{cases}$$

then we can just sum the solutions of each recurrence:

$$T_i(n) = \begin{cases} aT_i(n/b) + c_i n^{d_i} & n > 1 \\ 0 & n = 1 \end{cases}$$

and add in $fn^{\log_b a}$ for the contribution from the leaves.