Homework 3: Recurrences and Sorting

Handed out Wednesday, October 19. Due at the start of class Tuesday, November 1.

Problem 1. Rank the following functions in increasing order of asymptotic growth rate. For functions that are asymptotically equivalent, group them together. You do not need to prove your ordering, but you may supply explanations for the purpose of getting partial credit. (Note that all logs with unspecified bases are assumed to be a log base 2.)

$$\begin{aligned}
f_1(n) &= 500n^3, \\
f_2(n) &= 17n + (2/n^2), \\
f_3(n) &= 7n \log\log n, \\
f_4(n) &= 20n \log^3 n + 5n^2, \\
f_5(n) &= 4\sqrt{n} + 3\log(n^2), \\
f_6(n) &= 2^{\sqrt{n}}, \\
f_7(n) &= 2^{(3\log n)}, \\
f_8(n) &= 5\log^2 n + 10\log n, \\
f_9(n) &= 20\log(n^2), \\
f_{10}(n) &= 4\log_3 n
\end{aligned}$$

Problem 2. A QM Sort is based on the Merge Sort alogorithm discussed in class and in Lecture 6 of Dr. Mount's notes. The difference between QM Sort and Merge Sort is that we split the current list into 5 pieces during the divide stage.

a. Write the pseudocode for QM sort, using the code in lecture 6 as a model.

b. Derive a recurrence equation for the time required to sort n elements.

c. Solve the recurrence for QM sort using the iteration method or the rrecurrence tree table, assuming that n is a power of 5.

d. Is QM sort stable? If not, give an example to illustrate your answer. If so, explain why stability is guaranteed.

Problem 3. Selection Sort can be thought of as a recursive alogorithm as follows: Find the largest element and put it at the end of the list (to be sorted). Recursively sort the remaining elements.

a. Write down the recursive version of Selection Sort in psuedocode.

b. Derive a recurrence for the exact number of comparisons that your alogorithm uses.

c. Use the iteration method or a recurrence tree to solve the recurrence. Simplify as much as possible.

Problem 4. This problem uses a more **A**mbitious version of **M**aster **T**heorem (**AMT**):

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

implies

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

Please solve the following recurrences *exactly* using the above version of the theorem, or if you cannot use the Ambitious Master Theorem (AMT) to do so, explain why. You should assume that $T(1) = 1$ unless otherwise stated.

(a) $T(n) = 4T(n/7) + 3^{\log n + 1}$.

(b) $T(n) = 3T(2n/3) + 2n$.

(c) $T(n) = T(\sqrt{n}) + 1$, and $T(2) = 1$.

(d) $T(n) = 4T(n/2) + 5n^2$.

(e) $T(n) = 4T(n/8) + n\log_8 n$.

Problem 5. Use iteration or a recurrence tree to solve (exactly) all of the recurrences of the previous problem which you could not solve by the **AMT**. In all cases you may make whatever simplifying assumptions you want about n. You may also assume that $T(1) = 1$. (In part (c) it is convenient to assume that n is of the form $2^{(2^k)}$ for some k, and that $T(2) = 1$.)