CSCI-GA.1170-003/004 Fundamental Algorithms	February 6, 2019
Problem Set 2	
Lecturer: Yevgeniy Dodis	*Due: Thursday, February 14*

Problem 2-1 (Sort recurrences) — 8 Points

Sort the following recurrences in increasing order of growth of the corresponding functions. Justify (very) briefly.[1]

(a) $T(n) = 16T(n/4) + n$;

(b) $T(n) = 2T(n/4) + n$;

(c) $T(n) = 3T(n/5) + \log n$;

(d) $T(n) = 9T(n/3) + n^2$;

(e) $T(n) = T(n/3) + 10$;

(f) $T(n) = 9T(n/3) + n^3$;

(g) $T(n) = 8T(n/2) + n^3$.

Problem 2-2 (Methods for Solving Recurrences) — 12 points

Consider the recurrence $T(n) = T(\lceil n/4 \rceil) + T(\lceil n/3 \rceil) + n$ with $T(1) = 1$.

(a) (4 Points) Using a recursion tree, determine a tight asymptotic upper bound on $T(n)$.

(b) (4 Points) Prove your upper bound using induction.

(c) (4 Points) Using a suitable variable change, solve the recurrence $U(n) = 3U(\lceil n^{1/3} \rceil) + 7$ with $U(2) = 1$.

Problem 2-3 (The Same Outcome in Different Ways) — 15 Points

Consider the following recurrence $T(n) = 4T(n/2) + n^2 \log n$, $T(1) = 1$.

(a) (2 points) Can the master's theorem, as stated in the book, be applied to solve this recurrence? If yes, apply it. If not, formally explain the reason why.

(b) (4 points) Solve the above recurrence using the recursion tree method.

(c) (4 points) Formally verify that your answer from part (b) is correct using induction.

~~(d) (5 points) Solve the above recurrence exactly using domain-range substitution.~~

[1]For this entire homework assignment, you may ignore the fact that the argument to T may not be an integer.

Problem 2-4 (Faster Mergesort) 10 points

(a) (6 points) Suppose you have some procedure FASTMERGE that given two sorted lists of length m each, merges them into one sorted list using m^c steps for some constant $c > 0$. Write a recursive algorithm using FASTMERGE to sort a list of length n and also calculate the run-time of this algorithm as a function of c. For what values of c does the algorithm perform better than $O(n \log n)$.

(b) (4 points) Let $A[1 \dots n]$ be an array such that the first $n - \sqrt{n}$ elements are already in sorted order. Write an algorithm that will sort A in substantially better than $O(n \log n)$ steps.