CSCI 570 - Fall 2022 - HW 5

September 16, 2022

Problem 1

Solve the following recurrences by giving tight Θ-notation bounds in terms of n for sufficiently large n. Assume that T(·) represents the running time of an algorithm, i.e. T(n) is a positive and non-decreasing function of n. For each part below, briefly describe the steps along with the final answer.

(a) $T(n) = 4T(n/2) + n^2 \log n$

(b) $T(n) = 8T(n/6) + n \log n$

(c) $T(n) = \sqrt{6000}\ T(n/2) + n^{\sqrt{6000}}$

(d) $T(n) = 10T(n/2) + 2^n$

(e) $T(n) = 2T(\sqrt{n}) + \log_2 n$

Problem 2

Solve Kleinberg and Tardos, Chapter 5, Exercise 3.

Problem 3

Solve Kleinberg and Tardos, Chapter 5, Exercise 5.

Problem 4

Assume that you have a blackbox that can multiply two integers. Describe an algorithm that when given an n-bit positive integer a and an integer x, computes x^a with at most $\mathcal{O}(n)$ calls to the blackbox.

Problem 5

Consider two strings a and b and we are interested in a special type of similarity called the "J-similarity". Two strings a and bare considered J-similar to each other in one of the following two cases: Case 1) a is equal to b, or Case 2) If we divide a into two substrings a_1 and a_2 of the same length, and divide bin the same way, then one of following holds: (a) a_1 is J-similar to b_1, and a_2 is J-similar to b^2 or (b) a_2 is J-similar to b_1, and a_1 is J-similar to b_2. Caution: the second case is not applied to strings of odd length.
Prove that only strings having the same length can be J-similar to each other. Further, design an algorithm to determine if two strings are J-similar within $O(\text{n logn})$ time (where n is the length of strings).

Problem 6

Given an array of n distinct integers sorted in ascending order, we are interested in finding out if there is a Fixed Point in the array. Fixed Point in an array is an index i such that arr[i] is equal to i. Note that integers in the array can be negative

Example: Input: arr[] = -10, -5, 0, 3, 7 Output: 3, since arr[3] is 3

a) Present an algorithm that returns a Fixed Point if there are any present in the array, else returns -1. Your algorithm should run in $O(\text{log n})$ in the worst case.

b) Use the Master Method to verify that your solutions to part a) runs in $O(\text{log n})$ time.

c) Let's say you have found a Fixed Point P. Provide an algorithm that determines whether P is a unique Fixed Point. Your algorithm should run in $O(1)$ in the worst case