CS 331: Algorithms and Complexity (Spring 2024)
Unique Number: 50930, 50935 50940, 50945

Assignment 4

Due on Tuesday, 27 February, by 11.59pm

Problem 1

(10 points) You are given the following recurrences.

$$T_1(n) = 2T_1(n/4) + n^2, T_1(1) = 1$$
$$T_2(n) = 2T_2(4n/3) + n^{1.5}, T_2(1) = 1$$
$$T_3(n) = 2T_3(n/2) - 5n^3, T_3(1) = 1$$
$$T_4(n) = 2T_4(n/2) + n \log n, T_4(2) = 0$$
$$T_5(n) = \sqrt{n}T_5(\sqrt{n}) + n, n \geq 2$$

(a) (1 point each) While these are all fine as mathematical expressions, two of these recurrences make no sense when used to describe a program's runtime. Identify which ones they are and state why.

(b) (2 points each) Solve the three recurrences that aren't ridiculous to find asymptotic bounds.

(c) (2 points) Sort the three recurrences in increasing asymptotic growth rate.

Problem 2

(10 points)

Consider a modified mergesort algorithm so that it splits the input *not* into two sets of almost-equal sizes, but into **five** sets of sizes approximately one-fifth. You have to do the following.

(a) **(4 points)** Describe this **5-ary** sorting algorithm including both the process of dividing and combining. For describing your algorithm, you should specify the algorithm's input, output, and process. You must describe your algorithm from scratch. You should not call the original mergesort as a part of your algorithm.

(b) **(2 points)** When combining five sets of sizes one-fifth each to form a set of size n, what is the worst case number of comparisons that your algorithm makes in terms n. You should provide the exact number **not in big O notation**. Show your work by an example.

(c) **(2 points)** Write down the recurrence for this algorithm. (you can assume that $T(1) = 1$ and N is a power of 5.) Justify your answer.

(d) **(2 points)** Find the asymptotic complexity of this algorithm. Show your work.

Problem 3

(10 pts) As a programmer in a startup company, you are assigned a list of independent tasks, numbered $1, 2, \ldots, n$. Completing task i gets you p_i points. You have to complete the tasks in order and can work on one task at a time. Each task, i, has a distinct challenge factor, c_i. If you face challenging tasks, you may decided to skip them. You do this because if you work on a challenging task, i, you won't be able to solve any of the following next c_i tasks.

For example, let p_i={3,2,5,10,6} and let c_i={1,0,1,2,3}, then the maximum number of point you can obtain is 14, which resulted from choosing the tasks {3, 5, 6}.

Given p_i and c_i for the tasks, you have to do the following.

(a) **(3 points)** Give the recurrence that will enable you to maximize the total number of earned points.

(b) **(4 points)** Write the pseudo-code for an iterative algorithm that implements the recurrence. Clearly show the time complexity of your algorithm.

(c) **(3 points)** Prove the correctness of your algorithm.