CS3230 Design and Analysis of Algorithms August 10, 2020

Problem Set 1

Lecturer: Divesh Aggarwal

Instruction

This tutorial contains two kinds of questions - questions marked with an asterisk (*) are part of the assignment, and you need to submit your solutions to these questions; solutions to unmarked questions need not be submitted, but you should still attempt them. Your assignment solution will be graded. Submit your assignment solution by uploading to the LumiNUS. Solutions typeset using LaTeXare encouraged but not necessary. You may also submit scanned handwritten solutions, but they must be clearly readable. Solutions that are not readable may receive only partial marks. Late submissions will receive no marks. This assignment solution will be due by 11:59 pm, August 22, 2020 (Saturday).

Problem 1-1 (Recurrence and Recurrence)

Give asymptotic upper and lower bounds for each $T(n)$. Assume that $T(n)$ is constant for $n \leq 10$. Make your bounds as tight as possible and justify your answers. In the following, we mean $\lfloor n/7 \rfloor$ and $\lfloor \sqrt{n} \rfloor$ when we write $n/7$ and $\sqrt{n}$. Similarly, for $n/2, n/4, n/8$.

(a) $T(n) = 7T(n/7) + \sqrt[3]{n}$.

(b) $T(n) = T(n/2) + T(n/4) + T(n/8) + n$.

(c) $T(n) = T(\sqrt{n}) + 3$.

Problem 1-2 (Tower of Hanoi)

The Tower of Hanoi is a well known mathematical puzzle. It consists of three rods, and a number n of disks of different sizes which can slide onto any rod. The puzzle starts with all disks stacked up on the 1st rod in order of increasing size with the smallest on top. The objective of the puzzle is to move all the disks to the 3rd rod, while obeying the following rules.

- Only one disk is moved at a time
- Each move consists of taking one disk from top of a rod, and moving it on top of the stack on another rod
- No disk may be placed on top of a smaller disk.

A recursive algorithm that solves this problem is as follows: We first move the top $n-1$ disks from rod 1 to rod 2. Then we move the largest disk from rod 1 to rod 3 and then move the $n-1$ smaller disks from rod 2 to rod 3. Using the symmetry between the rods, the number of steps that this algorithm takes is given by the recurrence

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

which can be solved to get $T(n) = 2^n - 1$.

(a) Show that the above algorithm is optimal, i.e., there does not exist a strategy that solves the Tower of Hanoi puzzle in less than $2^n - 1$ steps.

(b) Suppose the moves are restricted further such that you are only allowed to move disks to and from rod 2. Give an algorithm that solves the puzzle in $O(3^n)$ steps.

(c) Suppose the moves are restricted such that you are only allowed to move from rod 1 to rod 2, rod 2 to rod 3, and from rod 3 to rod 1. Give an algorithm that solves the puzzle in $O((1+\sqrt{3})^n)$ steps.

Problem 1-3 (*Counting Inversions) 10 Points

Let $A[1 \ldots n]$ be an array of pairwise different numbers. We call pair of indices $1 \leq i < j \leq n$ an *inversion* of A if $A[i] > A[j]$. The goal of this problem is to develop a divide-and-conquer based algorithm running in time $\Theta(n \log n)$ for computing the number of inversions in A.

(a) (5 Points) Suppose you are given a pair of *sorted* integer arrays A and B of length $n/2$ each. Let C an n-element array consisting of the concatenation of A followed by B. Give an algorithm for counting the number of inversions in C and analyze its runtime. Make sure you also argue why your algorithm is correct.

(b) (5 Points) Give an algorithm (in pseudocode) for counting the number of inversions in an n element array A that runs in time $\Theta(n \log n)$. Make sure you formally prove that your algorithm runs in time $\Theta(n \log n)$ (e.g., write the recurrence and solve it.)
(**Hint**: Combine Merge Sort with part (a).)

Problem 1-4 (Counting Intersections)

Suppose you are given two sets of n points, one set $p_1, p_2, ..., p_n$ on the line $y = 0$ and the other set $q_1, q_2, ..., q_n$ on the line $y = 1$. Create a set of n line segments by connecting each point p_i to the corresponding point q_i. Describe and analyze a divide-and-conquer algorithm to determine how many pairs of these line segments intersect, in $O(n \log n)$time. [Hint:See the previous problem.]

Problem 1-5 (*Recursive Squaring) 10 Points

Describe a recursive algorithm that squares any n-digit number in $O(n^{\log_3 5})$ time, by reducing to squaring only *five* $(n/3 + O(1))$-digit numbers.

(**Hint**: What is $(a+b+c)^2 + (a-b+c)^2$? Consider using a^2, c^2, $(a+b+c)^2$, $(a-b+c)^2$ and $(\cdots)^2$.)

For **generous** partial credit, describe a recursive algorithm that squares any n-digit number in $O(n^{\log_3 6})$ time, by reducing to squaring only *six* $(n/3 + O(1))$-digit numbers.