University of Houston

Homework 0

COSC 6320
Algorithms and Data Structures

Gopal Pandurangan

NAME

September 03, 2020

This page intentionally left blank.

Exercise A.6: Consider the set of the first $2n$ positive integers, i.e., $A = \{1, 2, \ldots, 2n\}$. Take any subset S of $n + 1$ distinct numbers from set A. Show that there are two numbers in S such that one divides the other.

Solution. TYPE SOLUTION HERE. □

Exercise 2.3: Rank the following functions by order of growth; that is, find an arrangment $g_1, g_2, \ldots$ of the functions satisfying $g_1 = \mathcal{O}(g_2)$, $g_2 = \mathcal{O}(g_3)$, ...

$$n^2, \frac{n}{\log n}, n \log n, 1.001^n, \frac{1}{n^2}, \log^{100} n, \frac{1}{\log n}, 4^{\lg n}, n!, n^{\lg \lg n}, n^{1.6}, 2^{\sqrt{\log n}}$$

Note: lg or log denotes the base-2 logarithm and that $\log^k n$ denotes $(\log n)^k$.

Solution. TYPE SOLUTION HERE. □

Exercise 2.11: You have the task of heating up n buns in a pan. A bun has two sides and each side has to be heated up separately in the pan. The pan is small and can hold only (at most) two buns at a time. Heating one side of a bun takes 1 minute, regardless of whether you heat up one or two buns at the same time. The goal is to heat up both sides of all n buns in the minimum amount of time. Suppose you use the following recursive algorithm for heating up (both sides) of all n buns. If $n = 1$, then heat up the bun on both sides; if $n = 2$, then heat the two buns together on each side; if $n > 2$, then heat up any two buns together on each side and recursively apply the algorithm to the remaining $n - 2$ buns.

- Set up a recurrence for the amount of time needed by the above algorithm. Solve the recurrence.
- Show that the above algorithm does not solve the problem in the minimum amount of time for all $n > 0$.
- Give a correct recursive algorithm that solves the problem in the minimum amount of time.
- Prove the correctness of your algorithm (use induction) and also find the time taken by the algorithm.

Solution. TYPE SOLUTION HERE □

Exercise 4.1(c): Prove that the following recurrence is $\mathcal{O}(n \log n)$. Assume that the base cases of the recurrence is constant, i.e., $T(n) = \Theta(1)$ for $n < c$ for some constant c.

$$T(n) = T\left(\frac{5n}{6}\right) + T\left(\frac{n}{6}\right) + n$$

Solution. TYPE SOLUTION HERE □

Exercise 4.8: Give a recursive algorithm to compute 2^n (in decimal) for a given integer $n > 0$. Your algorithm should perform only $\mathcal{O}(\log n)$ integer multiplications.

Solution. TYPE SOLUTION HERE □

Exercise B.6: A strange number is one whose only prime factors are in the set $\{2, 3, 5\}$. Give an efficient algorithm (give pseudocode) that uses a binary heap data structure to output the n^{th} strange number. Explain why your algorithm is correct and analyze the run time of your algorithm. (Hint: Consider generating the strange numbers in increasing order, i.e., $2, 3, 4, 5, 6, 8, 9, 10, 12, 15$, etc,. Show how to efficiently generate the next strange number using a heap).

Solution. TYPE SOLUTION HERE □