COMPSCI 311: Introduction to Algorithms **Fall 2019**

Homework 4

Released 10/22/2019 Due 11/4/2019 11:59pm in Gradescope

Instructions. You may work in groups, but you must write solutions yourself. List collaborators on your submission. Also list any sources of help (including online sources) other than the textbook and course staff.

If you are asked to design an algorithm, please provide: (a) the pseudocode or precise description in words of the algorithm, (b) an explanation of the intuition for the algorithm, (c) a proof of correctness, (d) the running time of your algorithm and (e) justification for your running time analysis.

Submissions. Please submit a PDF file. You may submit a scanned handwritten document, but a typed submission is preferred. Please assign pages to questions in Gradescope.

1. **(20 points) Recurrences** Find and prove Θ bounds for the following recurrences. Assume $T(1) = 1$.

a) $T(n) = 2T(n/2) + n^2$ for even n, $T(n) = T(n-1) + n$ for odd n.

b) $T(n) = 2T(n/2) + T(n-1)/\log n$ ($n/2$ is integer division)

2. **(20 points) Chicken Wings.**
The image to the right is a real restaurant menu. The goal of this problem is to find the cheapest way to buy V chicken wings for some integer V given a menu like this one. Assume the menu is given as a list of n menu items $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$, where w_i is the price to buy v_i wings and v_i and w_i are both integers. In the example, assuming costs are computed in cents, we would have:

$$(v_1, w_1) = (4, 455)$$
$$(v_2, w_2) = (5, 570)$$
$$(v_3, w_3) = (6, 680)$$
$$\ldots$$

You are allowed to choose any combination of orders whose quantities add up to V, including ordering the same quantity multiple times. You can assume there is always *some* combination of orders to buy exactly V wings.

(a) A natural greedy algorithm has you buy the largest quantity $v_i \leq V$, and then repeat on the remaining $V - v_i$ wings. Show that this is not optimal for the menu shown to the right.
(b) Write a dynamic programming algorithm to find the cheapest set of orders to buy exactly V wings.

3. **(20 points) Three-Choice Nim** Here is another variation of Nim, a two-player game with three piles of stones. On their move, each player has a choice of three possible types of moves:
a) remove one stone from each pile
b) remove two stones from each of two piles
c) remove three stones from any one pile

The first player who cannot move loses. (So as in ordinary Nim, taking the last stone is a winning move. But there are four other positions from which there is no legal move, and leaving your opponent in one of them is also a winning move.)

Give an algorithm that takes input n, a positive integer, and determines which player has a winning strategy if the game starts with three piles of n stones each.

Find out who has the winning strategy (first or second player) if n = 5.

4. **(20 points) Gerrymandering**. We are given a set of n precincts, each with a known number a_i of voters from Party A and a known number b_i from Party B. Each precinct has the same number m of voters, $a_i + b_i = m$. We must group these precincts into two districts with $n/2$ precincts each (assume n is even). Our goal is to do so in such a way that Party A has a majority of the voters in both districts.

Write an algorithm that takes the voter information as input and returns a division into districts where A has a majority in both, or says (correctly) that this is not possible). Prove your algorithm correct and analyze its running time (which should be polynomial in n and m).

5. **(20 points) Colored Heap**. Here our input is a full (but not necessarily balanced) binary tree. There is a root, some internal nodes, and some leaves, and every non-leaf node has exactly two children. Our output will be a three-coloring of the nodes, that is, an assignment of a label 1, 2, or 3, to each node such that no two nodes connected by an edge have the same label.

If possible we would like the labels to form a heap, so that every parent has a larger number than each of its children. Failing this, we would like to come as close to it as possible. We will pay a penalty of 1 for each child that has a label larger than that of its parent.

Give an algorithm that inputs a full binary tree and returns a three-coloring that is optimal, in that no other three-coloring of the same tree has a smaller penalty. Argue that your algorithm is correct and analyze its running time (which should be polynomial).

6. **(0 points).** How long did it take you to complete this assignment?