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

Homework 4

Released 3/12/2019 Due 3/29/2019 11:59pm in Gradescope

Instructions. You may work in groups, but you must write solutions yourself. List collaborators on your submission.

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.

Problem 1. (10 points) Recurrence Consider the following recurrence:

$$T(n+1) = T(n) + \sqrt{n}$$

Show that $T(n)$ is $\Theta(n^d)$ for some $d > 0$.

Problem 2. (10 points) Fast Matrix Multiplication. Given two $n \times n$ matrices X, Y of integers, their product is another $n \times n$ matrix Z with,

$$Z_{ij} = \sum_{k=1}^{n} X_{ik} Y_{kj}.$$

Naively, computing Z seems to take $O(n^3)$ time since there are n^2 entries, and computing the value for a single entry involves n addition operations. In this problem, we'll develop a faster matrix multiplication algorithm. For simplicity, assume that n is a power of 2.

First, define eight $n/2 \times n/2$ matrices A, B, C, D, E, F, G, H so that,

$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix}, Y = \begin{bmatrix} E & F \\ G & H \end{bmatrix}$$

From basic linear algebra, we know that:

$$Z = XY = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

(a) The equation above suggests a divide and conquer algorithm for matrix multiplication. Describe the algorithm in words, write down the running time as a recurrence relation, and solve the recurrence.

(b) The above algorithm uses eight recursive calls, but a more clever decomposition due to Strassen can achieve the same result using only seven recursive calls of similar form. What is the recurrence for such an algorithm and what would the running time be?

Problem 3. (20 points) Butterflies (Again?). You have collected n butterfly specimens of different species. You want to know if there is a single species that accounts for more than half of the specimens. You don't know butterflies well enough to name the species of any single specimen, but you can carefully compare any two specimens and judge (correctly) if they are from the same species or not. Design an algorithm that returns "true" if there are more than $n/2$ specimens from one species, and returns "false" otherwise, using only $O(n \log n)$ pairwise comparisons.

Problem 4. (20 points) Pebbles Game. Two players play a game, starting with three piles of n pebbles each. Players alternate turns. On each turn, a player may do one of the following:

- take one pebble from any pile
- take two pebbles each from two different piles

The player who takes the last pebble(s) wins the game.

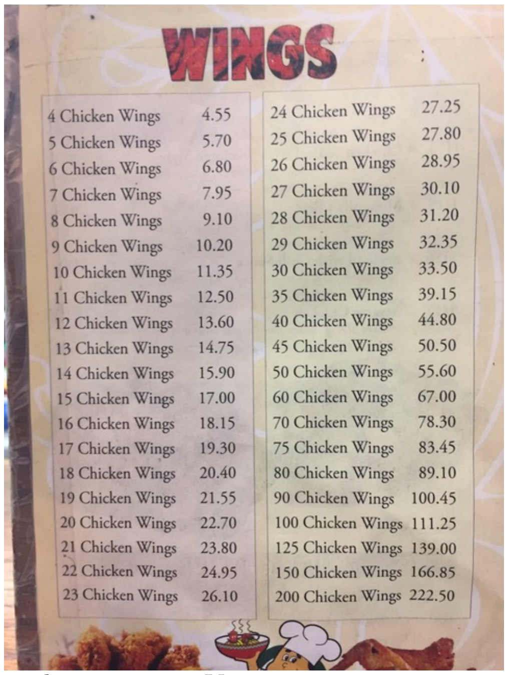
Write an algorithm that, given n, determines whether the first player can "force" a win. That is, can the first player choose moves such that, regardless of what the other player does, they will always win.

In addition, write a procedure that tells a player which move (if any) they should make to force a win given the current triple of pebbles in each pile.

What is the complexity of your solution?

Problem 5. (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:

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



WINGS

4 Chicken Wings	4.55	24 Chicken Wings	27.25
5 Chicken Wings	5.70	25 Chicken Wings	27.80
6 Chicken Wings	6.80	26 Chicken Wings	28.95
7 Chicken Wings	7.95	27 Chicken Wings	30.10
8 Chicken Wings	9.10	28 Chicken Wings	31.20
9 Chicken Wings	10.20	29 Chicken Wings	32.35
10 Chicken Wings	11.35	30 Chicken Wings	33.50
11 Chicken Wings	12.50	35 Chicken Wings	39.15
12 Chicken Wings	13.60	40 Chicken Wings	44.80
13 Chicken Wings	14.75	45 Chicken Wings	50.50
14 Chicken Wings	15.90	50 Chicken Wings	55.60
15 Chicken Wings	17.00	60 Chicken Wings	67.00
16 Chicken Wings	18.15	70 Chicken Wings	78.30
17 Chicken Wings	19.30	75 Chicken Wings	83.45
18 Chicken Wings	20.40	80 Chicken Wings	89.10
19 Chicken Wings	21.55	90 Chicken Wings	100.45
20 Chicken Wings	22.70	100 Chicken Wings	111.25
21 Chicken Wings	23.80	125 Chicken Wings	139.00
22 Chicken Wings	24.95	150 Chicken Wings	166.85
23 Chicken Wings	26.10	200 Chicken Wings	222.50

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) There is a natural greedy algorithm where you first buy the largest quantity v_i such that $v_i \leq V$, and then repeat on the remaining $V - v_i$ wings. Show that this algorithm 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.

Problem 6. (20 points) Texting. You are competing with your friends to type text messages on your smartphone as quickly as possible. Here are the rules: you use two thumbs for texting and they start out on the bottom left and bottom right keys of the keyboard. To type a character, you move either thumb from its current key to the key you need to press, and it takes time equal to the distance between the keys. You can assume the following:

- The keyboard has keys labeled $\{1, 2, \ldots, k\}$ and there is a function `dist`(i,j) to calculate the distance between two keys i and j. (To visualize this, you may want to imagine the digits 1 through 9 arranged on a standard numeric keypad).
- Your left thumb starts on key a, and your right thumb starts on key b. (For example, on the 9-digit numeric keypad, $a = 7$ is the bottom left key, and $b = 9$ is the bottom right key.)
- You can press any key with either thumb
- Both thumbs can rest on the same key if necessary
- The characters to by typed are $c_1 c_2 \cdots c_n$, where $c_i \in \{1, 2, \ldots, k\}$ is the ith key to push

Design an algorithm that finds the fastest way to type the message. In other words, your algorithm needs to decide which thumb to use to type each character, and it should minimize the total distance moved by your two thumbs. Try to use $O(nk^2)$ time.

Example. Imagine the 9-digit numeric keypad where your thumbs start at $a = 7$ and $b = 9$, with input message $c_1c_2c_3 = 589$. The solution "left, right, left" would look like this:

0 Left/right thumbs start at 7/9

1. Left thumb moves from 7 to $c_1 = 5$. Time = `dist`(7, 5). Thumbs end at 5/9.
2. Right thumb moves from 9 to $c_2 = 8$. Time = `dist`(9, 8). Thumbs end at 5/8.
3. Left thumb moves from 5 to $c_3 = 9$. Time = `dist`(5, 9). Thumbs end at 9/8.

Total time = `dist`(7, 5) + `dist`(9, 8) + `dist`(5, 9).

Problem 7. (0 points). How long did it take you to complete this assignment?