COMPSCI 311: Introduction to Algorithms **Spring 2020**

Homework 4

Released 3/15/2020 Due 4/1/2020 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.

Efficiency. The algorithms you design should be polynomial-time, unless something else is stated.

1. **(20 points) Longest Weighted Paths** Consider a DAG where each edge has a positive integer weight.

(a) Design an algorithm that finds a path of maximum weight.

(b) Design an algorithm that computes the *number of* paths with maximum weight.

Hint: Think first of the simpler problem with unweighted edges (equivalently, weight 1).

2. **(20 points) Maximum Parenthesized Sum** You are given a set of positive integers separated by + or − signs. Design an algorithm that inserts parentheses into this expression so the result has maximum value.

For example, given $5+4-3-7+2$, some possible expressions are $5+4-(3-7)+2$, and $5+4-(3-(7+2))$. Assume numbers are given in array $v[0..n]$ and operators in array $op[1..n]$. Indicate parentheses in pairs: (i, j) will include numbers $v[i]$ through $v[j]$.

3. **(20 points) Splitting into Words** Consider an array of letters that you want to split into words. Assume a constant-time function that looks up a string in a dictionary and says if it's a word or not.

a) Design an algorithm that computes the number of ways in which an array $S[1..n]$ can be split into words. For example, INTOMORROW might be split as IN-TOMORROW, IN-TO-MORROW, and INTO-MORROW (3 ways).

b) Design an algorithm that determines whether two arrays $S[1..n]$ and $T[1..n]$ can be split into words at the same positions. For instance, THISNOGOOD and THATISEASY can both be split with lengths 4-2-4.

4. **(20 points) 2D Sheet Cutting** Consider an $m \times n$ sheet of metal with m, n positive integers. You are given an array P with a real-valued price $P(i, j)$ for any integer-dimension rectangle of size $i \times j$. Assume $P(i, j) = P(j, i)$. You want to cut the sheet maximizing the total price of the resulting pieces. You can cut any piece, horizontally or vertically, with the cut going completely across.

Design an algorithm that indicates how to cut for maximum total price, producing instructions of the form "Cut a $i \times j$ piece at $x = k$ (or at $y = k$)".

5. **(20 points) Longest Fluctuating Sequence** We call a sequence of real numbers *fluctuating* if it does not have more than three consecutive elements that form an increasing or decreasing subsequence. That is, there is no index i such that $a_i \le a_{i+1} \le a_{i+2} \le a_{i+3}$ or $a_i \ge a_{i+1} \ge a_{i+2} \ge a_{i+3}$.

Design an algorithm that, given a sequence $a[1..n]$, finds a longest fluctuating subsequence of it.

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