Computer Science C73
Scarborough Campus

October 21, 2019
University of Toronto

Homework Assignment #6
(worth 6% of the course grade)
Due: November 4 6, 2019, by ~~10 am~~ 8:35am

- ***You must submit your assignment through the Crowdmark system.*** *You will receive by email an invitation through which you can submit your work in the form of separate PDF docucments with your answers to each question of the assignment. To work with a partner, you and your partner must form a group on Crowdmark. Crowdmark does not enforce a limit on the size of groups.* ***The course policy that limits the size of each group to at most two remains in effect:*** *submissions by groups of more than two persons will not be graded.*
- *It is your responsibility to ensure that the PDF files you submit are legible. To this end, I encourage you to learn and use the LaTex typesetting system, which is designed to produce high-quality documents that contain mathematical notation. You are not required to produce the PDF files you submit using LaTex; you may produce it any way you wish, as long as the resulting document is legible.*
- *By virtue of submitting this assignment you (and your partner, if you have one) acknowledge that you are aware of the policy on homework collaboration for this course.*[a]
- *For any question, you may use data structures and algorithms previously described in class, or in prerequisites of this course, without describing them. You may also use any result that we covered in class, or is in the assigned sections of the official course textbooks, by referring to it.*
- *Unless we explicitly state otherwise, you may describe algorithms in high-level pseudocode or point-form English, whichever leads to a clearer and simpler description.*
- *Unless we explicitly state otherwise, you should justify your answers. Your paper will be graded based on the correctness and efficiency of your answers, and the clarity, precision, and conciseness of your presentation.*

[a] "In each homework assignment you may collaborate with at most one other student who is currently taking CSCC73. If you collaborate with another student on an assignment, you and your partner must submit only one copy of your solution, with both of your names. The solution will be graded in the usual way and both partners will receive the same mark. Collaboration involving more than two students is not allowed. **For help with your homework you may consult only the instructor, TAs, your homework partner (if you have one), your textbook, and your class notes. You may not consult any other source.**"

Recall that a dynamic programming algorithm to solve a given problem P involves the following elements:

(a) *A definition of a polynomial number of subproblems that will be solved (and from whose solution we will compute the solution to P — see (c) below).*
(b) *A recursive formula to compute the solution to each subproblem from the solutions to smaller subproblems. This induces a partial order on the subproblems defined in (a).*
(c) *A way to compute the solution to P from the solutions to the subproblems computed in (b).*

Proving the correctness of a dynamic programming algorithm amounts to justifying (i) why the recursive formula in step (b) correctly computes the subproblems defined in step (a), and (ii) why the computation in (c) yields a solution to the given problem. Part (ii) is often immediate from the definition of the subproblems.

Question 1. (10 marks) Let $S[1..n]$ be a string that may have been produced when all punctuation marks and spaces disappeared from an English text, and all letters became lower-case. (So, it might be

"manyyearslaterashefacedthefiringsquad...") Your programming language has access to a Boolean function DICT(w) which, given a string w returns true if and only if w is a valid English word.

a. (7 marks) Give a dynamic programming algorithm that determines whether $S[1..n]$ can be reconstructed as a sequence of valid English words. Briefly explain why your algorithm is correct. Your algorithm's complexity should be $O(n^2)$, assuming each call to DICT takes $O(1)$ time.

b. (3 marks) If S can be reconstructed as a sequence of valid English words, make your algorithm print that sequence of words, one per line.

Question 2. (10 marks) A certain string-processing language offers a primitive operation SPLIT(S, i, S_1, S_2) which, given a string S of length n and a positive integer $i \leq n$, copies $S[1..i]$ into S_1 and $S[i+1..n]$ into S_2. Because SPLIT involves copying, it takes n units of time to apply on a string of length n regardless of the position i of the cut.

If a string is to be cut into several pieces, the order of the cuts matters. For example, if we want to cut a 20-character string at positions 3 and 10, then making the first cut at position 3 incurs a total cost of 37, while making the first cut at position 10 incurs a total cost of 30.

a. Give a polynomial-time dynamic programming algorithm which, given the positions of m cuts $c_1 < c_2 < \ldots < c_m$, in a string of length n, finds the minimum cost of cutting the string into the specified $m+1$ pieces. (The c_j's, being positions of a string of length n, are positive integers between 1 and n.) Explain why your algorithm is correct, and analyze its running time.

b. Make your algorithm also print the order in which the cuts should be applied so as to incur the minimum cost.

Question 3. (10 marks) Consider the following problem. The input is a set A of n non-negative integers (given in an array $A[1..n]$) and a non-negative integer k. The output is **true** if we can partition A into two subsets whose sums differ by k, and **false** otherwise. More precisely, the output is **true** if there is some $B \subseteq A$ such that

$$\sum_{x \in B} x - \sum_{y \in A-B} y = k$$

i.e., the difference between the sum of the integers in B and the sum of the integers in $A - B$ is exactly k; and the output is **false** if no such subset of A exists. (The sum of "the integers" in the empty set is, by definition, zero.)

Give a pseudopolynomial dynamic programming algorithm for this problem. Explain why your algorithm is correct, and analyze its running time. Why is your algorithm ***pseudo***polynomial time and not polynomial time?