CPSC 420+500 Problem Set 5 6 Mar 2017
W. Evans Due: 13 Mar 2017 at 21.00 (9pm)

Submission Instructions

Handin your solutions using `handin`. You can write your solutions by hand and scan the pages or take pictures of them with your phone; or use a word processing package to typeset your solutions. Whatever you do, you should produce **pdf** files **assign5Q1.pdf**, **assign5Q2.pdf**, etc. containing your solutions for Question 1, Question 2, etc.

To handin: Copy your solution files to the directory `~/cs420/a5` in your home directory on a CS undergraduate machine. (You may have to create this directory using `mkdir ~/cs420/a5`.) Then run `handin cs420 a5` from your home directory.

If you do not have a CS department account, you can email your assign1.pdf to me `will@cs.ubc.ca`.

Late submissions are not accepted.

Grading Policy

We will grade a subset of these questions of size at least two. It's a good idea to do all of the questions because (1) you don't know which ones we'll grade and (2) answering these questions is good practice for the exams. We will email feedback to the email address associated with the account you used to handin the assignment.

Questions

Try to answer these on your own but if you work with someone or use an outside source you must acknowledge them in your write-up. Do not copy solutions from any source.

1. (from Exercise 6.1 Kleinberg & Tardos) *Max weight independent set in a path* In a graph, a subset of vertices is an *independent set* if no two are joined by an edge. Given a graph $G = (V, E)$ **that is a path** in which every vertex v has a weight $w(v)$, we want to find an independent set of maximum weight.

   - *Heaviest-First* Give an example to show that the algorithm that chooses the vertex of heaviest weight, removes it and its adjacent vertices from G, and repeats, does not choose an independent set of maximum weight.
   - *Even-Odd* Give an example to show that the algorithm that chooses the heavier of the set of all even vertices and the set of all odd vertices does not choose an independent set of maximum weight. Note: A vertex is *even* if it is the ith vertex from the start of the path (pick one end to be the start) and i is even, otherwise it is *odd*.
   - Describe an algorithm that runs in time polynomial in the number of path vertices, n, that *does* find an independent set of maximum weight.

2. (Problem 5.8.2 in Algorithmics by Brassard and Bratley) Consider the alphabet $\Sigma = \{a, b, c\}$. The elements of Σ have the following multiplication table:

		Right-hand symbol		
		a	b	c
Left-hand	a	b	b	a
symbol	b	c	b	a
	c	a	c	c

Thus $ab = b$, $ba = c$, and so on. Note that the multiplication defined by this table is neither commutative nor associative.

Find an efficient algorithm that examines a string $x = x_1 x_2 \ldots x_n$ of characters of Σ and decides whether or not it is possible to parenthesize x in such a way that the value of the resulting expression is a. For example, if $x = bbbba$, your algorithm should return "yes" because $(b(bb))(ba) = a$. (This is not the only parenthesization that works: $(b(b(b(ba)))) = a$.)

3. (from Exercise 16.2-2 from CLRS) A thief enters a store that has n items; the ith item is worth v_i dollars and weighs w_i pounds, where v_i and w_i are integers. The thief can carry at most W pounds in his knapsack for some integer W. The **knapsack problem** is to find the most valuable load the thief can take, subject to the weight capacity W of his knapsack. Give a dynamic-programming solution to the knapsack problem that runs in $O(nW)$ time.

   Note: As an interesting optional exercise, what if the thief could take a *fraction* f_i (between 0 and 1) of an item and obtain value $f_i v_i$ and weight $f_i w_i$? Can you model this problem as a linear program?

4. (Problem 15-3 from CLRS) *Printing neatly* Consider the problem of neatly printing a paragraph on a printer. The input text is a sequence of n words of lengths $\ell_1, \ell_2, \ldots, \ell_n$, measured in characters. We want to print this paragraph neatly on a number of lines that hold a maximum of M characters each. Our criterion of "neatness" is as follows. If a given line contains words i through j, where $i \leq j$, and we leave exactly one space between words, the number of extra space characters at the end of the line is $M - j + i - \sum_{k=i}^{j} \ell_k$, which must be nonnegative so that the words fit on the line. We wish to minimize the sum, over all lines except the last, of the *cubes* of the numbers of extra space characters at the ends of the lines. Give a dynamic-programming algorithm to print a paragraph of n words neatly on a printer. Analyze the running time and space requirements of your algorithm.