CSC364 Summer 2002 – Homework 2

The following questions are assigned each week. You need only hand in those questions marked with a star * as part of your assignment, and only these questions will be marked. Make sure you include a completed and signed cover page as your first page.

Week 5

1. We tend to consider problems solved by polynomial time algorithms as tractable. What happens if we use such algorithms as procedures in loops, recursive calls, or different phases of our programs? Will such a program be tractable, or will it take more time to run? The fundamental question is whether our idea of tractable is closed under such operations.

   Prove that the class of polynomials over real numbers, $\mathbb{R}[x]$, is closed under

   (a) addition,
   (b) multiplication, and
   (c) composition.

   That is, for polynomials $p(x)$ and $g(x) \in \mathbb{R}[x]$, show that $p(x) + q(x)$, $p(x) \cdot q(x)$, and $p(q(x)) \in \mathbb{R}[x]$.

*2. (hand in) Let $S_1, S_2, \ldots, S_n$ be a collection of n MP3 songs. Song S_i requires m_i kilobytes of storage. We have a personal MP3 player with D kilobytes of memory, where $D < \sum_{i=1}^{n} m_i$. We want to load the maximum number of different songs onto our MP3 player.

   (a) Give a greedy algorithm that outputs a set of songs to load such that we load the maximum number of songs possible.
   (b) Prove that your algorithm correctly finds the maximum number of songs we can store on the player.
   (c) State and prove the running time of your algorithm.

3. Consider the same MP3 player as in question 2. We want to use as much of the player memory as possible. Prove or disprove: we can use a greedy algorithm to select songs leaving the minimum amount of unused storage space on the player.

Week 6

4. Let A be an $n \times n$ array with each row and column sorted. That is, for each row i, $a_{i1} \leq a_{i2} \leq \cdots \leq a_{in}$ and for each column j, $a_{1j} \leq a_{2j} \leq \cdots \leq a_{nj}$. Design an algorithm that determines whether a value d appears in the array. Give the running time of your algorithm and prove your algorithm correct.

*5. (hand in) In the great convergence of telecommunications, n companies want to merge. Assume that company i has c_i customers and the number of customers remains constant throughout the merger process: a merged company inherits all customers from the companies being merged. Assume that all customers are unique.

   The Canadian Radio-television and Telecommunications Commission (CRTC) regulates the industry and wants to ensure sufficient competition in the market and must approve each merger. Furthermore, the CRTC only approves mergers between exactly two companies at a time. The risk of a merger being denied is proportional to the number of customers the two companies have at the time of the merger. For a sequence of mergers to succeed, each merger must be approved, so the total risk is the sum of risks of individual mergers.

   Design an algorithm to determine an optimal order of mergers to minimize the total risk. Prove your algorithm correct and justify its running time. *Note:* Higher marks to more efficient algorithms.

6. Consider the problem of making change for n cents using as few coins as possible.

   (a) Suppose the available denominations of coins are powers of $d > 1$: there are $d^0, d^1, d^2, \ldots, d^k$ coins for some $k \geq 1$. Show that there is a greedy algorithm that always yields an optimal solution.

   (b) Give a set of coin denominations for which the greedy algorithm does not always yield an optimal solution. You should include the unit coin (penny) so there is a solution for every value n.

Week 7

*7. (hand in) Consider an automated manufacturing process where the product is formed by joining n pieces together. Assume the pieces are attached in a linear sequence and the order that the product is assembled is unimportant.

For example, the sequence $(p_1, p_2, p_3, \ldots, p_n)$ means that piece p_1 attaches to piece p_2, piece p_2 attaches to piece p_3, and so on. To form the product, we could first attach piece p_1 to p_2 and then attach this joined piece $p_{1\&2}$ to piece p_3, or we could first attach p_2 to p_3 and then attach p_1 to this conglomerate piece $p_{2\&3}$.

A robot attaches the pieces together by picking up 2 pieces and joining them. The cost of the combination is directly proportional to the weight of the 2 pieces. The cost of combining p_i and p_{i+1} is $w_i + w_{i+1}$ where w_i is the weight of piece p_i.

Design an algorithm that, given the weights $(w_1, w_2, \ldots, w_n)$ of the pieces to be joined in the sequence $(p_1, p_2, p_3, \ldots, p_n)$, determines the optimal order of combining the pieces yielding the minimum possible total cost. Give the running time of your algorithm and prove your algorithm correct. *Note:* Higher marks to more efficient algorithms.

8. Consider again the problem of making change for n cents using as few coins as possible. Design an efficient algorithm that always finds an optimal solution for any set of denominations. Prove your algorithm correct and justify its running time.

9. You are given a vector $V = [a_1, a_2, \ldots, a_n]$ of n integers, any of which may be positive, negative or zero. A subvector is a contiguous group of integers $[a_k, a_{k+1}, \ldots, a_l]$ from this vector. The sum of a vector is the sum of its elements, and the sum of an empty vector is 0.

   (a) Design an efficient algorithm to find the maximum sum of any subvector of V. Prove your algorithm correct and justify its running time.

   (b) Modify your algorithm above to find a subvector of V with maximum sum.

Week 8

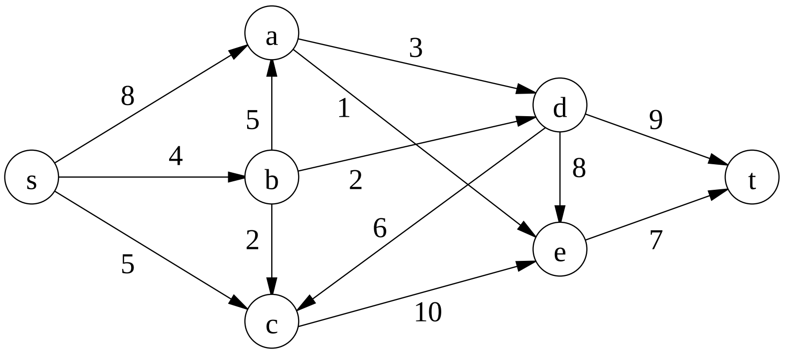
10. Consider the alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$. The elements of Σ have the following multiplication table, where the row is the left-hand symbol and the column is the right-hand symbol.

	a	b	c
a	c	b	a
b	a	c	a
c	b	b	c

For example, $\mathtt{ab} = \mathtt{b}$, $\mathtt{ba} = \mathtt{a}$, and so on. Note that the multiplication given by this table is neither associative nor commutative.

Design an efficient algorithm that, when given a string $w = w_1 w_2 \cdots w_n$ of characters in Σ^*, determines whether or not it is possible to parenthesize w such that the value of the resulting expression is $\mathtt{b}$. For example, if $w = \mathtt{abba}$, your algorithm should return "yes" since either $(\mathtt{a}((\mathtt{bb})\mathtt{a})) = \mathtt{b}$ or $(((\mathtt{ab})\mathtt{b})\mathtt{a}) = \mathtt{b}$. Prove your algorithm correct and justify its running time.

*11. (hand in) Consider the following network with source s, target t, and the given capacities for each arc.



Demonstrate Ford-Fulkerson's algorithm (CLRS page 658, first edition page 596) on this network. Show the augmenting path found at each step of the algorithm. Prove that the flow found by the algorithm is maximum in this network by demonstrating a cut of the same size.

12. Problem 26-1, CLRS page 692: The Escape Problem, parts a and b. (It is Problem 27-1, page 625 in the first edition of CLR.)