Computer Science 364 Fall, 2003
St. George Campus University of Toronto

Homework Assignment #1
Due: Oct 1, 2003, by 2pm
(in the lecture)

On the cover page of your assignment, you must write **and sign** *the following statement: "I have read and understood the homework collaboration policy described on the course webpage." Without such a signed statement your homework will not be marked.*

Greedy algorithms

[20] 1. **Weighted Independed Set**
Suppose we are given an undirected graph $G = (V, E)$, with $V = \{1, 2, ..., n\}$. An *independent set* of G is a subset $X \subseteq V$ of nodes such that no two nodes in X are connected by an edge in E.

Suppose that the nodes in G are weighted, so that node j gets weight 2^j. We define the total weight of a set $X \subseteq V$ of nodes in as a sum of weights of its nodes: $w(X) = \sum_{j \in X} 2^j$ The goal of this problem is to find, given a graph and weights of nodes, an independent set $X \subseteq V$ such that $w(X)$ is as large as possible.

[3] (a) Give an example of a graph in which the independed set with the largest weight (call it X) does not have the largest number of nodes (that is, in which there exist an independent set Y with more nodes than X has, but with a smaller total sum of weights).

[5] (b) Give a greedy algorithm which solves this problem.

[12] (c) Prove the correctness of your algorithm. That is, give a definition of "promising" appropriate for this problem, and give an induction proof that partial solutions produced by your algorithm are promising.

[20] 2. **2-processor activity selection**
The following is a special case of multiple processor activity selection, as described in the "Greedy Algorithms" notes on page 19.

The input is a list of activities $(s_1, f_1) \ldots (s_n, f_n)$, where s_i is a starting time of activity i and f_i is a finishing time of activity i. The output schedule is a pair (A_1, A_2) such that two conditions hold:
a) If for activities i and j both $i \in A_1$ and $j \in A_1$ or both $i \in A_2$ and $j \in A_2$, then activity i does not overlap activity j (that is, $f_i \leq s_j$ or $f_j \leq s_i$).
b) $A_1 \cap A_2 = \emptyset$, that is, no activity is scheduled in both A_1 and A_2.

[3] (a) Give an example of a set of activities that gives rise to at least two possible optimal schedules, and that has an activity that does not get scheduled in either case.

[5] (b) Give a polynomial-time greedy algorithm to solve this problem.

[12] (c) Prove that your algorithm always returns an optimal schedule. That is, give a definition of "promising" appropriate for this problem, and give an induction proof that partial solutions produced by your algorithm are promising.

[20] 3. **Number of subsets**

Suppose you are given a set of numbers $\{a_1, \ldots, a_k\}$ and a number t, where t is small (polynomial in k). Your goal is to find what is the number of different subsets S of $\{a_1, \ldots, a_k\}$ such that $\Sigma_{a_i \in S} a_i = t$. For example, if you have input $\{1, 2, 3, 4\}$ and $t = 7$, then you answer should be 2, since there are two subsets $S = \{1, 2, 4\}$ and $S = \{3, 4\}$ with sum of elements equal $t = 7$.

[5] (a) Define a suitable array A to store partial solutions to the problem and state how to find from this table the total number of subsets summing up to t .

[12] (b) Give a recurrence to compute elements of A, including initialization. Justify your answer.

[3] (c) Show how your algorithm works (i.e., fill the array) for a small example.

[20] 4. **Longest Palindrome**

The input is a string $\bar{s} = s_1 \ldots s_n$. Your goal is to find a maximum-length palindrome in s, that is, a subsequence of s that reads the same forward and backward.

For example, if your input string is "initialization", then a possible longest palindrome is "n i t a z a t i n", another "n i t i a i t i n", both with length 9.

[5] (a) Define a suitable array A to store partial solutions to the problem and state how to find from this table the *length* of the longest palindrome.

[10] (b) Give a recurrence to compute elements of A, including initialization.

[5] (c) Explain how to find the actual longest palindrome given A.