Instructor: Daniel Štefankovič, CSB 620, e-mail: stefanko@cs.rochester.edu
www.cs.rochester.edu/~stefanko/Teaching/06CS282

Homework problems:

For some of the problems below you will need to use dynamic programming. If you use dynamic programming, your solution must start with a concise description of the "heart of the algorithm". For example, for the knapsack problem you would write:

We will have an array $K[0..n, 0..W]$ where $K[i, j]$ is the value of the optimal solution for knapsack size j and items $1, \ldots, i$. We have

$$K[0, j] = 0, \tag{1}$$

and

$$K[i, j] = \max \begin{cases} K[i-1, j] \\ K[i-1, j-w_i] + v_i \quad \text{if } j \geq w_i. \end{cases} \tag{2}$$

Then, of course, you would write an explanation (proof) why (1) and (2) are true and conclude with a pseudocode of your algorithm.

5.1 (**due Oct 12, 2006**) We are given an $n \times n$ array A of zeros and ones. Give an algorithm to find the size of the largest contiguous all-ones square. Your algorithm must run in time $O(n^2)$.

5.2 (**due Oct 12, 2006**) We are given n positive numbers $a_1, \ldots, a_n$ (the numbers are not necessarily integers). The goal is to select a subset of the numbers with maximal sum and such that no three consecutive numbers are selected. Here are three example inputs together with optimal solutions (the numbers in boxes are selected):

$$\boxed{5}\ \boxed{5}\ 8\ \boxed{5}\ \boxed{5}$$
$$\boxed{5}\ 5\ \boxed{12}\ 5\ \boxed{5}$$
$$1\ \boxed{2}\ \boxed{2}\ 1\ \boxed{2}\ \boxed{1}\ 2\ \boxed{5}\ \boxed{5}$$

Give an $O(n)$-time algorithm for the problem.

5.3 (**due Oct 12, 2006**) We are given n positive integers $a_1, \ldots, a_n$ and another positive integer M. We want to figure out if we can select a subset of the integers which sums to M. Give an $O(Mn)$-time algorithm for the problem.

5.4 (**due Oct 12, 2006**) We are given n coin values $c_1, c_2, \ldots, c_n$ and an amount P (the c_i and P are positive integers). Unlike in the original coin change problem (where we had an unlimited supply of each coin value) we now have only 2 of each coin value. We would like to figure out whether we can pay P, and if we can, what is the minimal number of coins that can be used to pay P. Give an efficient algorithm for the problem.

For example if the coin values are $1, 2, 5, 6$ and $P = 15$ then the answer is yes - use 5 coins (since $15 = 6 + 6 + 2 + 1$ or $15 = 6 + 5 + 2 + 2$). (Note that we cannot pay $15 = 5 + 5 + 5$, since we have only 2 coins of value 5.)

5.5 (**due Oct 12, 2006**) We are given n positive numbers $a_1, \ldots, a_n$ (the numbers are not necessarily integers (and hence you cannot use counting sort, radix sort, etc.)). We are given another positive number B. We want to find the size of the largest subset of $a_1, \ldots, a_n$ whose average is at least B. Give an $O(n)$ algorithm for the problem. For example if the numbers are $4, 1, 2, 4$ and $B = 3$ then we can select $4, 1, 4$ (their average is $(4 + 1 + 4)/3 = 3 \geq 3$) but we cannot select $4, 1, 2, 4$ (their average is $(4 + 1 + 2 + 4)/4 = 11/4 < 3$). (Thus the answer for this example is 3.)