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

Homework problems:

10.1 (**due Nov 16, 2006**) We are given a collection of n intervals $I_1, \ldots, I_n$. Interval $I_i = [a_i, b_i]$ is assigned weight w_i. We want to find a max-weight subset of disjoint intervals, i. e., we want to find $S \subseteq \{1, \ldots, n\}$ such that

- for any distinct $j, k \in S$ the intervals I_j and I_k are disjoint, and
- the $\sum_{j \in S} w_j$ is maximized.

Give an $O(n^2)$-time algorithm for this problem.

10.2 (**due Nov 16, 2006**) We are given a directed graph $G = (V, E)$. We want to find a subgraph H of G such that H is acyclic (i. e., H is a DAG), and H has at least $|E|/2$ edges. Give an $O(E + V)$-time algorithm which finds such H.

10.3 (**due Nov 16, 2006**) Let n be an integer and $p \in [0, 1]$ (i. e., p is a real number between 0 and 1). A $G(n, p)$ graph is a random undirected graph constructed as follows. We start with n vertices and for each pair of vertices u, v we put an edge between u, v, independently, with probability p. Implement the following:

- a procedure which generates a random $G(n, p)$ graph,
- the greedy algorithm for the Minimum-Vertex-Cover problem,
- the maximal-matching algorithm for the Minimum-Vertex-Cover problem.

Experimentally compare the performance of the the two algorithms for $n = 1000$ and $p = i/1000$ for $i = 0, ..., 100$ (note that $p \in [0, 1/10]$).

Draw (using a computer!) a plot which has two curves in it: 1) the expected number of vertices in the vertex cover found by the greedy algorithm, 2) the expected number of vertices in the vertex cover found by the maximal-matching algorithm. (To estimate the expected values take the average of, say, 100 experiments.)

Bonus problem:

10.4 (**due Nov 16, 2006**) Consider a function $f : \mathbb{Z}^+ \to \mathbb{Z}^+$. Let $\{a_i\}$ be the sequence defined by $a_1 = 1$ and $a_{i+1} = f(a_i)$ for $i \geq 1$. If there exist distinct i, j such that $a_i = a_j$ then we will say that f is *periodic*. From now on we assume that f is periodic.

Let i, j the lexicographically smallest pair of distinct indices such that $a_i = a_j$. The number $j - i$ is called the *period* of f. The number i is called the *pre-period* of f.

We would like to find the period and the pre-period of f. We only have black-box for f, and we are only allowed to store some small (constant) number of integers in our program.

- Give an algorithm which finds the period of f in time $O(\text{period}(f) + \text{pre-period}(f))$.
- Give a fast algorithm which finds the pre-period of f. Can you find an algorithm which runs in time $O(\text{period}(f) + \text{pre-period}(f))$?