Spring 2009, CS 598CSC: Approximation Algorithms
Homework 0
Due: Never

Collaboration Policy: This homework is to test your knowledge and will not be graded. Try to work out the problems on your own but feel free to talk to other students.

Problem 1 We are given a set of n intervals on the real line, $I_1, I_2, \ldots, I_n$. Each interval I_j is specified as (ℓ_i, r_i) where ℓ_i is the left end point and r_i is the right end point. There is a positive weight w_j associated with I_j. Use dynamic programming to find the maximum weight subset of intervals that do not overlap. Suppose we allow intervals to overlap but we require that the maximum number of intervals that can contain any point x on the real line is B for some constant $B > 1$. Can you generalize your algorithm to find a maximum weight subset of intervals that satisfy this relaxed condition? What is the running time of your algorithm?

Problem 2 Lipton and Tarjan showed that for any n vertex planar graph there is a balanced separator of size $O(\sqrt{n})$. A balanced separator is set of vertices whose removal partitions the graph into two disconnected graphs each with no more than $2n/3$ vertices. They also show that such a separator can be found in polynomial time. Use these facts to show how to compute a maximum independent set in G in $2^{O(\sqrt{n})}$ time. An independent set in a graph is a set of vertices with no edge between any two vertices in the set (also called a stable set).
Hint: Use divide and conquer and dynamic programming.

Problem 3 Ball and bins. Consider throwing n balls into n bins where each ball is thrown independently and uniformly at random into a bin. What is the probability that a given bin (say the first bin) is empty? What is the probability that it contains exactly k balls? What is the expected number of bins that are empty? If you know Chernoff bounds prove that the maximum number of balls in any bin is $O(\log n / \log \log n)$ with high probability (with probability at least $1 - 1/\text{poly(n)}$).

Problem 4 The classical $0, 1$ knapsack problem is the following. We are given a set of n items. Item i has two positive integers associated with it: a size s_i and a profit p_i. We are also given a knapsack of integer capacity B. The goal is to find a maximum profit subset of items that can fit into the knapsack. (A set of items fits into the knapsack if their total size is less than the capacity B.) Use dynamic programming to obtain an exact algorithm for this problem that runs in $O(nB)$ time. Also obtain an algorithm with running time $O(nP)$ where $P = \sum_{i=1}^{n} p_i$. Note that both these algorithms are not polynomial time algorithms. Do you see why?

Problem 5 In the (maximum-cardinality) matching problem, given a graph $G(V, E)$, the goal is to find a largest subset of edges E' such that no two edges in E' share a common vertex. (Equivalently, each vertex must be adjacent to at most one edge in E'.)

1. Write a Linear Program (LP) for the matching problem in bipartite graphs.
2. Write a linear program for the matching problem in general graphs.
3. Write the duals to the primal linear programs from parts 1 and 2.
4. Give an example to show that there is a fractional solution to the LP of part 2 with value greater than that of an optimal integral solution. (That is, the *integrality gap* of this linear program is greater than 1.)
5. Prove that the optimal value to the LP of part 1 is an integer.
   Hint: You may use the fact that the in a network with integer capacities, the value of the maximum flow is integral.

Problem 6 Let G be a complete graph with non-negative edge weights. One can compute in polynomial time a minimum weight perfect matching in G (assuming G has an even number of vertices). We want to use the matching algorithm to solve a problem on directed graphs. Let $H = (V, E)$ be a *directed* graph with non-negative arc weights given by $w : E \to \mathcal{R}^+$. We wish to find a minimum weight collection of disjoint directed cycles in H such that every vertex is in exactly one of those cycles. Show that one can solve this problem by reducing it to the matching problem.
Hint: Split each vertex v in H into two vertices v^- and v^+ with v^- for incoming arcs into v and v^+ for outgoing arcs from v.