CS 573: Algorithms, Fall 2009

Homework 1, due Monday, September 14, 23:59:59, 2009

Version 1.0

Name:	
Net ID:	Alias:

#	Score	Grader
1.		
2.		
3.		
4.		
Total		

Neatly print your name(s), NetID(s), and the alias(es) you used for Homework 0 in the boxes above. Staple this sheet to the top of your homework. If you are on campus, submit the homework by submitting it in SC 3306 (or sliding it under the door).

Note: You will be held accountable for the appropriate responses for answers (e.g. give models, proofs, analysis, etc). For NP-Complete problems you should prove everything rigorously, i.e. for showing that it is in NP, give a description of a certificate and a polynomial time algorithm to verify it, and for showing problems are NP-Hard, you must show that your reduction is polynomial time (by similarly proving something about the algorithm that does the transformation) and proving both directions of the 'if and only if' (a solution of one is a solution of the other) of the many-one reduction.

This homework should be easier than hw0. You are encouraged to discuss problems in this homework with people, but should submit your homework on your own.

Only of myself I know how to tell,
my world is as narrow as an ant's.
like an ant too my burden I carry,
too great and heavy for my frail shoulder.

My way too - like the ant's to the treetop -
is a way of pain and toil;
a gigantic hand, assured and malicious,
a mocking hand hinders

All my paths are made bleak and tearful
by the constant dread of this giant hand.

Why do you call to me, wondrous shores?
Why do you lie to me, distant lights?
– Only of Myself, Rachel

Required Problems

1. Traveling in elephants.
   [20 Points]

   The TSP faun, the stepfather of the deduction fairy, had visited you before school started and gave you, as a token of appreciation, a black-box that can solve TSP in polynomial time (note that this black box solves the decision problem). Let us refer to this black box **decider_TSP**. Given as input a complete graph G over n vertices, a cost function $c(\cdot)$ over the edges (the costs are non-negative integers), and a target t, **decider_TSP** can in, say linear time, decide if G has a TSP tour of cost at most t in it.

   Show, how to build a fast algorithm (that uses **decider_TSP**) that given a complete graph over n vertices and a cost function $c(\cdot)$ computes the cheapest TSP in G; namely, it outputs the edges used by the shortest TSP. What is the running time of your algorithm as a function of n, and $W = \max_{e \in E(G)} c(e)$. Naturally, the faster your algorithm the better it is.

   (It seems unlikely that there is a polynomial time algorithm for this problem with running time that is independent of W. Such algorithms are referred to as being strongly polynomial. Naturally, extra credit would be given to such a solution, or a proof that such an algorithm does not exist unless $P = NP$.)

2. NP-Completeness collection
   [30 Points]

   Prove that the following problems are NP-Complete.

   A. **[6 Points]**

   Problem: MINIMUM SET COVER

Instance: Collection C of subsets of a finite set S and an integer k. *Question:* Are there k sets $S_1, \ldots, S_k$ in C such that $S \subseteq \cup_{i=1}^{k} S_i$?

   B. **[6 Points]**

   Problem: BIN PACKING

Instance: Finite set U of items, a size $s(u) \in \mathbb{Z}^+$ for each $u \in U$, an integer bin capacity B, and a positive integer K. *Question:* Is there a partition of U into disjoint sets $U_1, \ldots, U_K$ such that the sum of the sizes of the items inside each U_i is B or less?

   C. **[6 Points]**

   Problem: TILING

Instance: Finite set $\mathcal{R}$ of rectangles and a rectangle R in the plane. *Question:* Is there a way of placing the rectangles of $\mathcal{R}$ inside R, so that no pair of the rectangles intersect, and all the rectangles have their edges parallel of the edges of R?

   D. **[6 Points]**

Problem: HITTING SET

Instance: A collection C of subsets of a set S, a positive integer K. *Question:* Does S contain a *hitting set* for C of size K or less, that is, a subset $S' \subseteq S$ with $\|S'\| \leq K$ and such that S' contains at least one element from each subset in C.

E. **[6 Points]**

Problem: Max Degree Spanning Tree

Instance: Graph $G = (V, E)$ and integer k *Question:* Does G contains a spanning tree T where every node in T is of degree at most k?

3. k-LINE GRAPH.
[30 Points]

A graph G is a k-line graph if it vertices are

$$V(G) = \left\{ i \;\middle|\; i = 1, \ldots, n \right\},$$

and two vertices i and j can be connected only if $|i - j| \leq k$. Here n is the number of vertices of G.

(A) **[15 Points]** Let k be a constant.
Provide an algorithm (as fast as possible) that in polynomial time computes the *maximum* size Independent Set for G, where G is a k-line graph. What is the running time of your algorithm (explicitly specify the dependency on k)?

(B) **[15 Points]** Let k be a constant.
Provide an algorithm (as fast as possible) that in polynomial time computes the minimum COLORING for G. That is, it computes a valid coloring of the vertices of G that uses the minimum number of colors. What is the running time of your algorithm (explicitly specify the dependency on k)?

4. POLY TIME SUBROUTINES CAN LEAD TO EXPONENTIAL ALGORITHMS
[20 Points]

Show that an algorithm that makes at most a constant number of calls to polynomial-time subroutines runs in polynomial time, but that a polynomial number of calls to polynomial-time subroutines may result in an exponential-time algorithm.