

CS 573: Algorithms, Fall 2013

Homework 1, due Monday, September 16, 23:59:59, 2013

Version 1.2

Name:	
Net ID:	Alias:

Neatly print your name(s), NetID(s). Staple this sheet to the top of your homework. If you are on campus, submit the homework by submitting it in the homework boxes in the basement of SC.

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.** Poly time subroutines can lead to exponential algorithms. (20 PTS.)
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.
- 2.** Beware of algorithms carrying oracles. (50 PTS.)
Consider the following optimization problems, and for each one of them do the following:
 - (I) (2 PTS.) State the natural decision problem corresponding to this optimization problem.
 - (II) (3 PTS.) Prove that this decision problem is **NP-COMPLETE** by showing a reduction from one of the **NP-COMPLETE** problems seen in class. If you already seen this problem in class state “seen in class” and move on with your life.

- (III) (5 PTS.) Assume that you are given an algorithm that can solve the decision problem in polynomial time. Show how to solve the original optimization problem using this algorithm in polynomial time.

Prove that the following problems are NP-COMPLETE.

- (A) (10 PTS.)

SET COVER

Instance: Collection \mathcal{C} of subsets of a finite set S .

Target: Compute the minimum k , and the sets S_1, \dots, S_k in \mathcal{C} , such that $S \subseteq \cup_{i=1}^k S_i$.

- (B) (10 PTS.)

MIN 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

Target: Compute the minimum k , and a partition of U into disjoint sets U_1, \dots, U_k , such that the sum of the sizes of the items inside each U_i is B or less.

- (C) (10 PTS.)

HITTING SET

Instance: A *ground set* $U = \{1, \dots, n\}$, and a set $\mathcal{F} = \{U_1, \dots, U_m\}$ of subsets of U .

Target: Find the smallest set $S' \subseteq U$, such that S' hits all the sets of \mathcal{F} . Specifically, $S' \subseteq U$ is a *hitting set* if for all $U_i \in \mathcal{F}$, we have that S' contains at least one element of U_i .

- (D) (10 PTS.)

Max Degree Spanning Tree

Instance: Graph $G = (V, E)$.

Target: Compute the spanning tree T in G where the maximum degree of a vertex in T is minimized.

- (E) (10 PTS.)

Partition into independent sets.

Instance: Graph $G = (V, E)$.

Target: Compute the minimum k , and the partition of V into k sets V_1, \dots, V_k , such that for every edge uv of G , there exists two distinct indices i and j , such that $u \in V_i$, and $v \in V_j$.

3. Dominating set. (30 PTS.)

Consider the following problem (which is NP-COMPLETE):

Dominating set

Instance: Graph $G = (V, E)$ and an integer k .

Target: Is there a subset $X \subseteq V$ of size k , such that each vertex of $V \setminus X$ is adjacent to some vertex of X .

Consider the variant where $V = \{1, \dots, n\}$, and $ij \in E$, implies that $|i - j| \leq 8$. We refer to this variant as the **8Dominating set** problem.

Either prove that **8Dominating set** is NP-COMPLETE, or alternatively, provide a polynomial time algorithm for solving it.