CS 573: Algorithms, Fall 2012

Homework 1, due Monday, September 24, 23:59:59, 2012

Version 1.1

Name:	
Net ID:	Alias:

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. 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.

2. Vogsphere's children hate poetry
[30 Points]
In Vogsphere there are n children waiting to participate in a poetry competition. You know all the pairs of children that are friends, and you want to order the children in line, such that no two friends are more than k places apart from each other. Formally, if C is the set of n children, then you need to compute a bijection between C and $\{1, \ldots, n\}$, such that if c and c' are friends, then $|f(c) - f(c')| \leq k$. Such an ordering is called a ***k-compliant ordering***, Finding the minimum k for which this is possible is of course NP-Hard. For the sake of simplicity, you can assume that for any two children, there is a sequence of friends that (indirectly) connect them.

(A) **[5 Points]** Let k be a small positive integer constant. Show an algorithm such that given an ordered list $L = \ell_1, \ldots, \ell_{3k}$ of $3k$ children (supposedly in their consecutive ordering along the line), the algorithm either:

  (I) Outputs that there is no valid k-compliant ordering of the n children, with the children of L appearing consecutively in line (with the order specified by L).

  (II) Output two disjoint sets of children B And F, such that in any ordering compliant with L, all the children of B must appear before L, and all the children of F appear after L.
  (Note, that this option is a bit strange – the algorithm might output sets B and F even if there is no k-compliant ordering of the whole list. All it is saying is that if there is a k-compliant ordering then B and F must be before and after L, respectively.)

  What is the running time of your algorithm?

(B) **[13 Points]** Given k, the list of children, and the friendship information. Provide an algorithm that in $n^{O(k)}$ time decides if there is a k-compliant ordering of the children, and if so it outputs it. Prove the correctness and the running time of your algorithm.
Hints:

  (I) The plan is to try and use (A).
  (II) First, how many different lists are there (like the one used in (A))?
  (III) Show how to compute for two disjoint lists L_1 and L_2 all the children that must be in between L_1 and L_2 if there is a k-compliant ordering of having L_1 somewhere before L_2 in the ordering.
  (IV) Compute, recursively, for any such pair of lists L_1 and L_2 whether or not there exists a k-compliant ordering having L_1 and L_2 in the ordering with L_1 to the left L_2 in the resulting ordering.
  (V) Victory! (But do provide the details!)

(C) **[12 Points]** Let k be a constant. Given a k-compliant ordering of the children, provide an algorithm, as fast as possible, that computes the largest group of children, such that no two children that are friends are in the group. What is exactly the running time of your algorithm? For full credit, your algorithm should have running time $O(f(k)n)$, where $f(k)$ is a function that depends only on k. Partial credit would be given to slower algorithms.

3. BEWARE OF GREEKS BEARING GIFTS

[20 Points]

(The expression "beware of Greeks bearing gifts" is Based on Virgil's Aeneid: "Quidquid id est, timeo Danaos et dona ferentes", which means literally "Whatever it is, I fear Greeks even when they bring gifts.".)

The reduction faun, the brother of the Partition satyr, came to visit you on labor day, and left you with two black boxes.

(A) **[10 Points]** The first black box, was a black box that can solves the following decision problem in polynomial time:

Minimum Test Collection

> **Instance**: A finite set A of "possible diagnoses," a collection C of subsets of A, representing binary "tests," and a positive integer $J \leq |C|$.
> **Question:** Is there a subcollection $C' \subseteq C$ with $|C'| \leq J$ such that, for every pair a_i, a_j of possible diagnoses from A, there is some test $c \in C'$ for which $|\{a_i, a_j\} \cap c| = 1$ (that is, a test c that "distinguishes" between a_i and a_j)?

Show how to use this black box, how to solve in polynomial time the optimization version of this problem (i.e., finding and outputting the smallest possible set C').

(B) **[10 Points]**

The second box was a black box for solving Subgraph Isomorphism.

Subgraph Isomorphism

> **Instance**: Two graphs, $G = (V_1, E_1)$ and $H = (V_2, E_2)$.
> **Question:** Does G contain a subgraph *isomorphic* to H, that is, a subset $V \subseteq V_1$ and a subset $E \subseteq E_1$ such that $|V| = |V_2|$, $|e| = |E_2|$, and there exists a one-to-one function $f : V_2 \to V$ satisfying $\{u, v\} \in E_2$ if and only if $\{f(u), f(v)\} \in E$?

Show how to use this black box, to compute the subgraph isomorphism (i.e., you are given G and H, and you have to output f) in polynomial time.

4. NP-COMPLETENESS COLLECTION.

[30 Points]

Prove that the following problems are NP-COMPLETE.

A. **[6 Points]**

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]**

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**]

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**]

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**]

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?