Due: Wed, November 30, beginning of lecture

NOTE: Each problem set only counts 5% of your mark, but it is important to do your own work (but see below). Similar questions will appear on the next term test. You may consult with others concerning the general approach for solving problems on assignments, but you must write up all solutions entirely on your own. Anything else is *plagiarism*, and is subject to the University's Code of Behavior. You will receive 1/5 points for any (non bonus) question/subquestion for which you say "I do not know how to answer this question". You will receive .5/5 points if you just leave the question blank.

1. (20 points)

   Show that each of the following problems are NP complete.

   (a) QUARTER-CLIQUE $= \{G = (V, E) | G$ has a clique of size at least $\lceil |V|/4 \rceil \}$

   (b) UNIT-THOUGHPUT $= \{(J_1, J_2, \ldots, J_n)|$ all n of these jobs can be scheduled without conflict$\}$. Here $J_i = (r_i, p_i, d_i)$ where r_i is the release time, p_i is the processing time, and d_i is the deadline. Each such job must be scheduled to start no sooner than r_i and must complete by time d_i. Two scheduled jobs J_i and J_k do not conflict if they do not intersect (except we will allow the starting time of one job to coincide with the finishing time of another job).

2. (10 points)

   Let L be a NP complete set and let one NP representation of L be $L = \{w | \exists y [R(w, y)$ is true and $|y| \leq q(|w|\}$. Here q is a polynomial and R s a polynomial time predicate. The associated search problem, $L(w, R, q)$, is "Given w, find a certificate y if it exists (i.e. if $w \in L$), else say that y does not exist". Show that $P = NP$ implies that $L(w, R, q)$ can be computed in polynomial time.
   Hint: you will need to define another NP language which will allow you to construct a certificate y (one symbol at a time) when it exists.
   A better formulation for the previous question is as follows:

   Let L be a NP complete set and let one NP representation of L be $L = \{w | \exists y [R(w, y)$ is true and $|y| \leq q(|w|\}$. Here q is a polynomial and R s a polynomial time predicate. The associated search problem, $L(w, R, q)$, is "Given w, find a certificate y if it exists (i.e. if $w \in L$), else say that y does not exist". Show that the search problem $L(w, R, q)$ can be polynomial time reduced to the decision problem for L. That is, $L(w, R, q) \leq_T^{poly} L$.

   In the original formulation, it is sufficient that L be in NP. But in the new formulation, it is important that L also be complete. In particular, we could formulate the set $COMPOSITE = \{w | \exists y : y$ is a proper divisor of $w\}$ where w and y say are decimal representations of positive integers. Whereas we know that $COMPOSITE$ is in P, the search problem essentially solves the integer factoring problem which we strongly believe cannot be computed in polynomial time.

3. (20 points)

   Consider the weighted exact max-3-Sat problem and the single flip local search algorithm ALG for this problem. That is, we let the neighbourhood of a solution be $N(\tau) = N_1(\tau) = \{\tau' : \tau' \text{ differs from } \tau \text{ on exactly one variable}\}$.

   (a) Prove that the the ratio $\frac{W(ALG)}{W} \geq 3/4$ where $W(ALG)$ is the weight of any local optimum and W is the sum of all clause weights.

   (b) Suppose we modify the local neighbourhood $N(\tau)$ of a solution so that now $N(\tau) = N_1(\tau) \cup \{\bar{\tau}\}$ where $\bar{\tau}$ is the complement of the assignment τ. Show that the ratio $\frac{W(ALG)}{W} > 3/4$.

4. (20 points)

   Consider the following call routing problem. There is an n node bi-directional ring network $G = (V, E)$ upon which calls must be routed. That is $V = \{0, 1, \ldots, n-1\}$ and $E = \{(i, i+1 \ mod \ n)\} \cup \{(i, i-1 \ mod \ n)\}$ and calls c_j are pairs (s_j, f_j) originating at node s_j and terminating at node f_j. Each call can be routed in a clockwise or counter-clockwise direction. The load L_e on any directed edge is the maximum number of calls routed on this edge. The goal is to minimize $max_{e \in E} L_e$.

   (a) Formulate this problem as an IP.
   Hint: Consider variables x_j and y_j that indicate the direction of call c_j. (You can also use just one indicator variable to represent the direction but I think it might be easier to think in terms of two such variables.)

   (b) Using an LP relaxation of this problem, show how to derive a 2-approximation algorithm.

5. Consider the following *3 from 4 frequency set cover problem*: We are given a collection of sets $\mathcal{S} = \{S_1, \ldots, S_m\}$ for $S_i \subseteq U$ with the property that every $u \in U$ occurs in exactly four different sets S_i in $\mathcal{S}$. There is also a cost function $c : \mathcal{S} \rightarrow \mathbb{R}^{\geq 0}$ and we let c_i denote the cost of set S_i. A feasible solution is a sub-collection $\mathcal{S}' \subseteq \mathcal{S}$ such that every $u \in U$ occurs in at least three different sets S_i in $\mathcal{S}'$. The goal is to find a feasible solution $\mathcal{S}'$ so as to minimize the cost $c(\mathcal{S}') = \sum_{i: S_i \in \mathcal{S}'} c_i$.

   (a) Formulate the *3 from 4 frequency set cover problem* as a $\{0, 1\}$ IP

(b) Show how to use LP relaxation + rounding to obtain a 2-approximation algorithm. Explain why your rounded solution is a feasible solution to the IP and why it provides a 2-approximation.

6. Consider the weighted max cut problem. That is, we are given a graph $G = (V, E)$ and edge weights $w : E \rightarrow \Re^{\geq 0}$.

   Consider the following randomized max cut algorithm that computes a cut (S, T)

   $S := \emptyset; T := \emptyset$
   While $V \neq \emptyset$
   Choose any $v \in V$
   With probability $= 1/2$, add v to S and otherwise add v to T
   $V := V - \{v\}$
   End While

   (a) Show $\mathcal{E}[w(S,T)] = \frac{\sum_{e \in E} w(e)}{2}$

   (b) Describe how to use the method of conditional expectations to derive a deterministic max cut algorithm producing a cut (S, T) satisfying $w(S,T) \geq \frac{\sum_{e \in E} w(e)}{2}$

7. Suppose we are given an algorithm that multiplies two degree n univariate polynomials $a(x)$ and $b(x)$ and returns what is supposed to be the degree $2n$ polynomial $c(x) = a(x) * b(x)$. Describe an $O(n)$ time randomized testing algorithm and explain the error bound in terms of the time bound. More precisely, your testing algorithm will always say "good" if indeed $c(x) = a(x) * b(x)$ and will say "bad" with some constant probability $\delta > 0$ if $c(x) \neq a(x) * b(x)$.