CS CS 473g: Algorithms, Fall 2007
Homework 2 (due Tuesday, October 9, 2007 at 11:59.99 p.m.)

Version 1.02

<table>
<thead>
<tr>
<th>Name:</th>
<th>Net ID:</th>
<th>Alias:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name:</th>
<th>Net ID:</th>
<th>Alias:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#</th>
<th>Score</th>
<th>Grader</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.
In the back of her mind, Sherri thought about death ceaselessly. Everything else, all people, objects and processes had become reduced to the status of shadows. Worse yet, when she contemplated other people she contemplated the injustice of the universe. They did not have cancer. This meant that, psychologically speaking, they were immortal. This was unfair. Everyone had conspired to rob her of her youth, her happiness and eventually her life; in place of those, everyone else had piled infinite pain on her, and probably they secretly enjoyed it. "Enjoying themselves" and "enjoying it" amounted to the same evil thing. Sherri, therefore, had motivation for wishing that the whole world would go to hell in a handbasket.

— Valis, Philip K. Dick

Required Problems

1. Greedy algorithm does not work for TSP with the triangle inequality. [20 Points]
   In the greedy Traveling Salesman algorithm, the algorithm starts from a starting vertex \( v_1 = s \), and in \( i \)-th stage, it goes to the closest vertex to \( v_i \) that was not visited yet.
   
   (a) [10 Points] Show an example that prove that the greedy traveling salesman does not provide any constant factor approximation to the TSP. Formally, for any constant \( c > 0 \), provide a complete graph \( G \) and positive weights on its edges, such that the length of the greedy TSP is by a factor of (at least) \( c \) longer than the length of the shortest TSP of \( G \).

   (b) [10 Points] Show an example, that prove that the greedy traveling salesman does not provide any constant factor approximation to the TSP with triangle inequality. Formally, for any constant \( c > 0 \), provide a complete graph \( G \), and positive weights on its edges, such that the weights obey the triangle inequality, and the length of the greedy TSP is by a factor of (at least) \( c \) longer than the length of the shortest TSP of \( G \). (In particular, prove that the triangle inequality holds for the weights you assign to the edges of \( G \).)

2. Maximum Clique [10 Points]
   Let \( G = (V, E) \) be an undirected graph. For any \( k \geq 1 \), define \( G^{(k)} \) to be the undirected graph \( (V^{(k)}, E^{(k)}) \), where \( V^{(k)} \) is the set of all ordered \( k \)-tuples of vertices from \( V \) and \( E^{(k)} \) is defined so that \( (v_1, v_2, ..., v_k) \) is adjacent to \( (w_1, w_2, ..., w_k) \) if and only if for each \( i \) (\( 1 \leq i \leq k \)) either vertex \( v_i \) is adjacent to \( w_i \) in \( G \), or else \( v_i = w_i \).

   (a) [5 Points] Prove that the size of the maximum clique in \( G^{(k)} \) is equal to the \( k \)-th power of the size of the maximum clique in \( G \).

   (b) [5 Points] Argue that if there is an approximation algorithm that has a constant approximation ratio for finding a maximum-size clique, then there is a polynomial time approximation scheme for the problem.

3. Greedy algorithm does not work for independent set. [20 Points]
   A natural algorithm, \texttt{GreedyIndependent}, for computing maximum independent set in a graph, is to repeatedly remove the vertex of lowest degree in the graph, and add it to the independent set, and remove all its neighbors.
(a) [5 Points] Show an example, where this algorithm fails to output the optimal solution.

(b) [5 Points] Let $G$ be a $(k, k+1)$-uniform graph (this is a graph where every vertex has degree either $k$ or $k+1$). Show that the above algorithm outputs an independent set of size $\Omega(n/k)$, where $n$ is the number of vertices in $G$.

(c) [5 Points] Let $G$ be a graph with average degree $\delta$ (i.e., $\delta = 2|E(G)|/|V(G)|$). Prove that the above algorithm outputs an independent set of size $\Omega(n/\delta)$.

(d) [5 Points] For any integer $k$, present an example of a graph $G_k$, such that \texttt{GreedyIndependent} outputs an independent set of size $\leq |OPT(G_k)|/k$, where $OPT(G_k)$ is the largest independent set in $G_k$. How many vertices and edges does $G_k$ have? What is the average degree of $G_k$?

4. PACK THESE SQUARES.

[10 Points]

Let $R$ be a set of squares. You need to pack them inside the unit square in the plane (i.e., place them inside the square, so that their sides are parallel to the sides of the unit square), such that all the squares are interior disjoint. Provide a polynomial time algorithm that outputs a packing that covers at least $OPT/4$ fraction of the unit square, where $OPT$ is the fraction of the unit square covered by the optimal solution.

5. GREEDY COLORING DOES NOT WORK EVEN IF YOU DO IT IN THE RIGHT ORDER. [20 Points]

Given a graph $G$, with $n$ vertices, let us define an ordering on the vertices of $G$ where the min degree vertex in the graph is last. Formally, we set $v_n$ to be a vertex of minimum degree in $G$ (breaking ties arbitrarily), define the ordering recursively, over the graph $G \setminus v_n$, which is the graph resulting from removing $v_n$ from $G$. Let $v_1, \ldots, v_n$ be the resulting ordering, which is known as \textsc{min last ordering}.

Greedy coloring works by coloring vertices according to their min last ordering. At the $i$th step, it assigns a color to $v_i$ that is consistent with the colors already assigned to $v_1, \ldots, v_{i-1}$. If there is a color that was already used but its not used by any of the neighbors of $v_i$ in $\{v_1, \ldots, v_{i-1}\}$ then we use it to color $v_i$. Otherwise, we introduce a new color and use it to color $v_i$.

(a) [10 Points] Prove that the greedy coloring algorithm, if applied to a planar graph $G$, which uses the min last ordering, outputs a coloring that uses 6 colors.\footnote{There is a quadratic time algorithm for coloring planar graphs using 4 colors (i.e., follows from a constructive proof of the four color theorem). Coloring with 5 colors requires slightly more cleverness.}

(b) [10 Points] Give an example of a graph $G_n$ with $O(n)$ vertices which is 3-colorable, but nevertheless, when colored by the greedy algorithm using min last ordering, the number of colors output is $n$. 