HW 7 (due Wednesday, at noon, October 24, 2018)

CS 473: Algorithms, Fall 2018 Version: **1.3**

Submission guidelines and policies as in homework 1.

1 (100 PTS.) More about coloring.

1.A. (20 PTS.) Prove that a graph G with a chromatic number k (i.e., k is the minimal number of colors needed to color G), must have $\Omega(k^2)$ edges.

1.B. (20 PTS.) Consider a graph G with n vertices and m edges. Consider the greedy algorithm, which orders the vertices in arbitrary order $v_1, \ldots, v_n$. Assume the algorithm colored the first $i-1$ vertices. In the ith iteration, the algorithm assign v_i the lowest color that is not assigned to any of its neighbors that are already colored (assume the colors used are the numbers $1, 2, 3, \ldots$). Provide an upper bound, as low as possible, on the number of colors used by your algorithm as a function of m. You can assume $m \geq n \geq 100$.

1.C. (20 PTS.) Prove that if a graph G is k-colorable, then for any vertex v, the graph $\mathsf{H} = \mathsf{G}_{N(v)}$ is $k-1$-colorable, where $N(v)$ is the set of vertices in G adjacent to v, and $\mathsf{G}_{N(v)}$ is the ***induced subgraph*** on $N(v)$. Formally, we have

$$\mathsf{G}_{N(v)} = (N(v), \{uv \in \mathsf{E}(\mathsf{G}) \mid u, v \in N(v)\}).$$

1.D. (20 PTS.) Describe a polynomial time algorithm that given a graph G, which is 3-colorable, it computes a coloring of G using, say, at most $O(\sqrt{n})$ colors, where n is the number of vertices in G.

Hint: First color the low degree vertices in the graph. If a vertex has a high degree, then use (C).

1.E. (20 PTS.) (Harder.) Describe a polynomial time algorithm that given a graph G, which is k-colorable, it computes a coloring of G using, say, at most $O(n^{1-2^{-(k-2)}})$ colors. Here k is a small **constant**. What is roughly the running time of your algorithm.

2 (100 PTS.) Greedy algorithm does not work for TSP with the triangle inequality.

In the greedy Traveling Salesman algorithm, the algorithm starts from a starting vertex $v_1 = s$, and in ith stage, it goes to the closest vertex to v_i that was not visited yet.

2.A. (20 PTS.) 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.

2.B. (80 PTS.) 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.)

3 (100 PTS.) Maximum Clique

The max-clique problem has the property that given a low quality approximation algorithm, one can get a better quality approximation algorithm – this question describe this amplification behavior.

Let $\mathsf{G} = (V, E)$ be an undirected graph. For any $k \geq 1$, define $\mathsf{G}^{(k)}$ 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 $v_i = w_i$.

3.A. (50 PTS.) Prove that the size of the maximum clique in $G^{(k)}$ is equal to the kth power of the size of the maximum clique in G.

3.B. (50 PTS.) Argue that if there is a c-approximation algorithm for finding a maximum-size clique in a graph, for some constant $c > 1$, then there is a polynomial time α-approximation algorithm for max-clique, where $\alpha = (c + 1)/2$.

(Hint: Use the first part.)