Below is a set of practice problems on NP-completeness and approximation to help you check your understanding of the approaches we’ve discussed in class.

You can find solutions to these problems on the course website. We’re also willing to listen to you describe your solutions or to look at your writeups during office hours.

Note: If you want to discuss these problems, I expect you to give correctness and complexity proofs for your algorithms here, just as on the actual homework. Efficiency matters!

Practice Problems

1. An independent set of a graph $G = (V, E)$ is a subset $V' \subseteq V$ of vertices such that each edge in $E$ is incident on at most one vertex in $V'$. The independent set problem is to find a maximum-size independent set in $G$. Recall that a clique of a graph $G = (V, E)$ is a subset $V' \subseteq V$ of vertices, each pair of which is connected by an edge in $E$. The clique problem is to find a maximum-size clique in $G$.

Suppose you have an approximation algorithm $A$ for the independent set problem whose competitive ratio is $f(n)$ on a graph of size $n$. Use $A$ to construct an $f(n)$-approximation algorithm for clique finding.

2. Way back on the Homework 1 practice problems, we saw the following preemptive scheduling problem. We are given $n$ jobs, where each job $i$ has an earliest start time $s_i$ and length $\ell_i$. Only one job can run at a time; however, a running job may be suspended at any time while other jobs run. Suppose that for a schedule $S$, job $i$ finishes at time $f_i$; the cost of schedule $S$ is then $\sum_i f_i$, and the goal is to minimize this sum. We came up with a polynomial-time greedy algorithm for this problem.

Consider the non-preemptive version of this problem. The input is the same, but now jobs cannot be suspended: once a job is started, it must run to completion.

One possible heuristic for non-preemptive scheduling is as follows. First, compute an optimal preemptive schedule $S'$ for the jobs. Then, schedule the jobs non-preemptively in the order in which they finish in $S'$, starting each job as soon as possible. Show that this heuristic is in fact a 2-approximation.

3. In class, we showed that $\text{INDEPENDENT-SET} \leq_p \text{VERTEX-COVER}$. Recall that a graph $G$ with $n$ vertices has an independent set of size at least $k$ iff it has a vertex cover of size at most $n - k$; in particular, the vertices not in the cover form an independent set.

In class, we saw a 2-approximation algorithm $A$ for finding the minimal vertex cover. Professor Ptolemy suggests that one can derive a constant-factor approximation algorithm for independent set by first reducing it to a vertex cover problem as above, then applying algorithm $A$. Is the professor correct? Justify your answer, either by proving that there exists a constant $c$ for which the above approach gives a $c$-approximation, or by showing that for any constant $c$, there exist graphs $G$ (with arbitrarily many vertices) for which the algorithm is not a $c$ approximation.