CS41 Homework 10

This homework is due at 11:59PM on Sunday, December 3. This is a 14-point homework. Write your solution using LaTeX. Submit this homework using **github** as **.tex** file. This is a **partnered homework**. You should primarily be discussing problems with your homework partner.

It's ok to discuss approaches at a high level with others. However, you should not reveal specific details of a solution, nor should you show your written solution to anyone else. The only exception to this rule is work you've done with a lab teammate *while in lab*. In this case, note (in your **homework submission poll**) who you've worked with and what parts were solved during lab.

1. Multiple-Interval-Scheduling (K&T 8.14)

   In this problem, there is a machine that is available to run jobs over some period of time, say 9AM to 5PM.

   People submit jobs to run on the processor; the processor can only work on one job at any single point in time. However, in this problem, each job requires a **set of intervals** of time during which it needs to use the machine. Thus, for example, one job could require the processor from 10AM to 11AM and again from 2PM to 3PM. If you accept this job, it ties up your machine during these two hours, but you could still accept jobs that need any other time periods (including the hours from 11AM to 2PM).

   Now, you're given an integer k and a set of n jobs, each specified by a set of time intervals, and you want to answer the following question: is it possible to accept at least k of the jobs so that no two of the accepted jobs have any overlap in time?

   In this problem, you are to show that Multiple-Interval-Scheduling $\in$ NP-complete. To assist you, we've broken down this problem into smaller parts:

   (a) First, show that Multiple-Interval-Scheduling $\in$ NP.

   (b) In the remaining two parts, you will reduce

   $$\text{Independent-Set} \leqslant_P \text{Multiple-Interval-Scheduling}\ .$$

   Given input $(G = (V, E), k)$ for Independent-Set, create a valid input for Multiple-Interval-Scheduling. First, divide the processor time window into m distinct and disjoint *intervals* $i_1, \ldots, i_m$. Associate each interval i_j with an edge e_j. Next, create a different job J_v for each vertex $v \in V$. What set of time intervals should you pick for job J_v?

   (c) Finally, run the Multiple-Interval-Scheduling algorithm on the input you create, and output yes iff the Multiple-Interval-Scheduling algorithm outputs yes. Argue that the answer to Multiple-Interval-Scheduling gives you a correct answer to Independent-Set.

2. **Clique is NP-complete.**

   A **clique** is a set of nodes $C \subseteq V$ such that every two vertices $u, v \in C$ are connected by an edge: $\{u, v\} \in E$.

   Consider the problem Clique: on input a graph G and integer k, return yes iff there is a clique in G of size k.

   Prove that Clique is NP-complete.

3. **Approximations via Reductions.**

   Recall that many problems have a decision version and an optimization version, so for example we can consider the problems

   - INDEPENDENT-SET(G, k) returns YES iff there is an independent set in G of size $\geqslant k$,
   - MAX-INDEPENDENT-SET(G) returns the size of the largest independent set in G,
   - VERTEX-COVER(G, k) returns YES iff there is a vertex cover of G of size at most k,
   - MIN-VERTEX-COVER(G) returns the size of the smallest vertex cover of G,
   - CLIQUE(G, k) returns YES iff there is a clique in G of size k, and
   - MAX-CLIQUE(G) returns the size of the largest clique in G.

   We know that all NP-COMPLETE problems reduce to each other. It would be nice if this meant that an approximation algorithm for one NP-COMPLETE problem can be adapted easily into an equally good approximation algorithm for any other NP-COMPLETE problem.

   (a) We proved that VERTEX-COVER $\leqslant_p$ INDEPENDENT-SET by giving an algorithm for VERTEX-COVER that used an algorithm for INDEPENDENT-SET as a black box.

   VERTEX-COVER (G = (V,E), k)
   1 $k' := n - k$.
   2 $z =$ INDEPENDENT-SET(G, k').
   3 **return** z.

   Suppose we wanted to solve the optimization versions of these problems, MIN-VERTEX-COVER and MAX-INDEPENDENT-SET. Can we do the same reduction idea with approximation algorithms? That is, suppose we have a black box 2-approximation algorithm IS-APPROX for MAX-INDEPENDENT-SET. The idea would be an approximation algorithm that works like this:

   VC-APPROX (G = (V,E))
   1 $z =$ IS-APPROX(G).
   2 **return** $n - z$.

   Assuming that IS-APPROX is a 2-approximation for MAX-INDEPENDENT-SET, what kind of approximation guarantee does this algorithm give for MIN-VERTEX-COVER? Either prove an approximation ratio for this algoritm, or explain why this ratio is hard to calculate / this is a bad idea.

   (b) Assume we have an k-approximation algorithm for MAX-CLIQUE where k is a constant. Can we use this to construct a decent approximation algorithm for MAX-INDEPENDENT-SET? Justify your answer by designing an approximation algorithm for MAX-INDEPENDENT-SET, and either proving an approximation ratio or explaining why this ratio is hard to calculate / this is a bad idea.

4. **Three-Coloring, approximately.**

Recall the THREE-COLORING problem: Given a graph $G = (V, E)$, output YES iff the vertices in G can be colored using only three colors such that the endpoints of any edge have different colors. In homework 9, you showed that THREE-COLORING is NP-COMPLETE.

Let THREE-COLORING-OPT be the following problem. Given a graph $G = (V, E)$ as input, find a coloring of the vertices in G, using at most three colors, that **maximizes** the number of *satisfied* edges, where an edge $e = (u, v)$ is satisfied if u and v have different colors.

Give a deterministic (not randomized), polynomial-time (3/2)-approximation algorithm for THREE-COLORING-OPT. Your algorithm must satisfy at least $2c^*/3$ edges, where for an arbitrary input $G = (V, E)$, the number c^* denotes the maximum number of satisfiable edges.

5. **(extra challenge)**

   We all know and love the decision problem 3-SAT. The optimization version MAX-3-SAT asks: given a set of n variables $\{x_1, \ldots, x_n\}$ and m clauses $\{c_1, \ldots, c_m\}$ where each clause contains exactly three literals[1], find a truth assignment that **maximizes** the number of satisfied clauses.

   Design and analyze an approximation algorithm for MAX-3-SAT using your (3/2)-approximation algorithm for THREE-COLORING-OPT.

[1] a literal is a variable or a negated variable