CS41 Homework 11

This homework is due at 10:00PM on Sunday, December 8. Write your solution using \LaTeX. Submit this homework using github as a .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 partner while in lab. In this case, note (in your homework poll file) who you’ve worked with and what parts were solved during lab.

1. \textbf{(K&T 11.3)} Suppose you are given a set of positive integers \(A = \{a_1, a_2, \ldots, a_n\}\) and a positive integer \(B\). A subset \(S \subseteq A\) is called \textit{feasible} if the sum of the numbers in \(S\) does not exceed \(B\):

\[
\sum_{a_i \in S} a_i \leq B.
\]

The sum of the numbers in \(S\) will be called the \textit{total sum} of \(S\).

You would like to select a feasible subset \(S\) of \(A\) whose total sum is as large as possible.

For example, if \(A = \{8, 2, 4\}\) and \(B = 11\) then the optimal solution is the subset \(S = \{8, 2\}\).

(a) Here is an algorithm for this problem.

\begin{verbatim}
notQuiteRight(A = \{a_1, \ldots, a_n\}, B)
1 initialize \(S = \emptyset\)
2 define \(T = 0\)
3 for \(i = 1\) to \(n\):
4    if \(T + a_i \leq B\)
5        \(S = S \cup \{a_i\}\)
6        \(T = T + a_i\)
7 return \(S\)
\end{verbatim}

Give an input for which the total sum of the set \(S\) returned by this algorithm is less than half the total sum of some other feasible subset of \(A\). (You don’t necessarily have to find the optimal subset, just \textit{some} feasible subset.)

(b) Give a polynomial-time approximation algorithm for this problem with the following guarantee: It returns a feasible set \(S \subseteq A\) whose total sum is at least half as large as the maximum total sum of any feasible set \(S' \subseteq A\). Your algorithm should run asymptotically faster than \(O(n^2)\).

2. \textbf{Reductions.} Recall that many problems have a decision version and an optimization version, so for example we can consider the problems

\begin{itemize}
\item \textsc{Independent-Set}(\(G, k\)) returns \textsc{yes} iff there is an independent set in \(G\) of size \(\geq k\),
\item \textsc{Max-Independent-Set}(\(G\)) returns the size of the largest independent set in \(G\),
\item \textsc{Vertex-Cover}(\(G, k\)) returns \textsc{yes} iff there is a vertex cover of \(G\) of size at most \(k\),
\item \textsc{Min-Vertex-Cover}(\(G\)) returns the size of the smallest vertex cover of \(G\),
\item \textsc{Clique}(\(G, k\)) returns \textsc{yes} iff there is a clique in \(G\) of size \(k\), and
\item \textsc{Max-Clique}(\(G\)) returns the size of the largest clique\(^1\) in \(G\).
\end{itemize}

\(^1\)A \textbf{clique} is a set of nodes \(C \subseteq V\) such that every two vertices \(u, v \in C\) are connected by an edge: \(\{u, v\} \in E\).
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 \text{Vertex-Cover} \leq_p \text{Independent-Set} by giving an algorithm for \text{Vertex-Cover} that used an algorithm for \text{Independent-Set} as a black box.

Suppose we wanted to solve the optimization versions of these problems, \text{Min-Vertex-Cover} and \text{Max-Independent-Set}. Can we do the same thing with approximation algorithms? That is, suppose we have a black box 2-approximation algorithm for \text{Max-Independent-Set}. Design an approximation algorithm for \text{Min-Vertex-Cover} using (as a black box) a 2-approximation algorithm for \text{Max-Independent-Set}. (Your algorithm should be based on the reduction.)

What kind of approximation guarantee can you give? Either prove an approximation ratio of your algorithm, or explain why this ratio is hard to calculate.

(b) Assume we have an $k$-approximation algorithm for \text{Max-Clique} where $k$ is a constant. Can we use this to construct a decent approximation algorithm for \text{Max-Independent-Set}? Justify your answer by designing an approximation algorithm for \text{Max-Independent-Set}, and either proving an approximation ratio or explaining why this ratio is hard to calculate.

3. \textbf{Three-Coloring Revisited.} Recall the \textsc{Three-Coloring} problem: Given a graph $G = (V, E)$, output \textsc{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 8, you showed that \textsc{Three-Coloring} is NP-Complete. In this lab, we’ll look at several approximation and randomized algorithms for the optimization version of \textsc{Three-Coloring}.

Let \textsc{Three-Coloring-OPT} be the following problem. Given a graph $G = (V, E)$ as input, color the vertices in $G$ using at most three colors in a way that maximizes the number of \textit{satisfied} edges, where an edge $e = (u, v)$ is satisfied if $u$ and $v$ have different colors.

\textbf{Approximation Algorithm.} Give a deterministic, polynomial-time $(3/2)$-approximation algorithm for \textsc{Three-Coloring-OPT}. Your algorithm must satisfy at least $2c^*/3$ edges, where for an arbitrary input $G = (V, E)$, $c^*$ denotes the maximum number of satisfiable edges.

4. \textbf{Chromatic Number.} Consider the optimization problem \textsc{Chromatic-Number}, defined as follows. Given a graph $G = (V, E)$ as input, determine the smallest number $k$ such that it is possible to $k$-color the graph.

(a) Prove that \textsc{Chromatic-Number} is NP-hard.

(b) Prove that there is no efficient $4/3$-approximation to \textsc{Chromatic-Number} unless P = NP.

(c) Prove that for any $\epsilon > 0$ there is no efficient $1 + \epsilon$-approximation to \textsc{Chromatic-Number} unless $P = NP$. Hint: recall that $\forall k > 2$, $k$-coloring is NP-Complete.