CS41 Homework 11

This homework is due at 11:59PM on Sunday, December 10. 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. **Maximum feasible sum** (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 *feasible* if the sum of the numbers in S does not exceed B:

   $$\sum_{a_i \in S} a_i \leqslant B.$$

   The sum of the numbers in S will be called the *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.

   ```
   notQuiteRight(A = {a_1, ..., a_n}, B)
   initialize S = ∅
   define T = 0
   for i = 1 to n:
       if T + a_i ⩽ B
           S ← S ∪ {a_i}
           T ← T + a_i
   return S
   ```

   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 *some* feasible subset.)

   (b) Give a worst-case-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. **Three-Coloring, approximated.**

   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. The optimization version THREE-COLORING-OPT is the 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 *satisfied* edges, where an edge $e = (u, v)$ is satisfied if u and v have different colors.

Describe and analyze *randomized* algorithms for THREE-COLORING-OPT with the following behavior:

(a) An algorithm that runs in worst-case (i.e., not expected) polynomial time and produces a three-coloring such that the expected number of satisfied edges is at least $2c^*/3$.

(b) An algorithm with expected polynomial runtime that always outputs a three-coloring that satisfies at least $2c^*/3$ edges.

(c) An algorithm that runs in worst-case polynomial time, and with probability at least 99% outputs a three-coloring which satisfies at least $2c^*/3$ edges. What is the running time of your algorithm?

The following inequality might be helpful: $1 - x \leqslant e^{-x}$ for any $x > 0$.

(Hints: start with a very basic idea. Don't overthink the algorithm design! The challenge is the analysis.)

3. **Chromatic Number.**

Consider the optimization problem CHROMATIC-NUMBER: 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 CHROMATIC-NUMBER is NP-hard.

(b) Prove that there is no efficient $\frac{5}{4}$-approximation to CHROMATIC-NUMBER unless P = NP.

(c) Prove that for any ε where $0 < \varepsilon < 1/3$, there is no efficient $(1 + \varepsilon)$-approximation to CHROMATIC-NUMBER unless P = NP. (Hint: recall that $\forall k > 2$, k-coloring is NP-COMPLETE.)

4. **(extra challenge) Even *more* coloring!** ...with not *too* many colors.

Suppose we're somehow told that a graph is three-colorable. Could that help us color the graph? In this problem, you'll shoot for a different kind of approximation.

Give a polynomial-time deterministic algorithm that, given any *three-colorable* graph $G = (V, E)$, colors the graph using $O(\sqrt{n})$ colors. Note that the endpoints of each edge *must* be different colors, and you're given that it is *possible* to color the graph using just three colors, but you don't know what the coloring is.

Here are a few hints to help you along:

(a) First, give a simple greedy algorithm that, given a graph $G = (V, E)$ such that each vertex has at most d neighbors, colors G using only $d + 1$ colors.

(b) Second, recall the algorithm for deciding if a graph is *bipartite*.

(c) Third, start coloring the three-colorable graph taking the vertex with the most neighbors, and coloring those neighbors using just two colors.

5. **(extra challenge)** Does P = NP? Answer YES or NO.

Justify your response with a formal proof.