CS41 Lab 12

1. **Three-Coloring Revisited.** 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 8, you showed that THREE-COLORING is NP-COMPLETE. In this lab, we'll look at several approximation and randomized algorithms for the optimization version of THREE-COLORING.

   Let THREE-COLOR-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 *satisfied* edges, where an edge $e = (u, v)$ is satisfied if u and v have different colors.

   For a graph $G = (V, E)$, let c^* denote the maximum number of satisfiable edges.

   (a) **Hardness of Three-Color-OPT.** Show that if there is a polynomial-time algorithm for THREE-COLOR-OPT then P = NP.

   (b) **Randomized Algorithms.** Give randomized algorithms for THREE-COLOR-OPT with the following behavior:

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

      ii. 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 $2c^*/3$.

      iii. 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: $1 - x \leq e^{-x}$ for any $x > 0$.

   (c) **Approximation Algorithm.** Give a deterministic, polynomial-time (3/2)-approximation algorithm for THREE-COLOR-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.

2. **Chromatic Number.** Consider the optimization problem CHROMATICNUMBER, 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 CHROMATICNUMBER is NP-hard. (Bonus: prove that it's NP-COMPLETE.)

   (b) Prove that there is no efficient $\frac{4}{3}$-approximation to CHROMATICNUMBER unless P = NP.

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

3. **Max XOR SAT.** Suppose we are given a set of n variables $x_1, x_2, \ldots, x_n$ each of which can take one of the values in the set $\{0, 1\}$. We are also given a set of m equations; the r^{th} equation has the form

$$(x_i + x_j) \mod 2 = b_r$$

   for some choice of two distinct variables x_i, x_j, and some for some value b_r that is either 0 or 1. Thus each equation specifies whether the sum of two variables is even or odd.

Consider the problem of finding an assignment of values to variables that maximizes the number of equations that are satisfied (i.e., in which equality actually holds). This problem is NP-hard, though you don't have to prove this.

For example, suppose we are given the equations

$$\begin{aligned}(x_1 + x_2) &\mod 2 = 0\\ (x_1 + x_3) &\mod 2 = 0\\ (x_2 + x_4) &\mod 2 = 1\\ (x_3 + x_4) &\mod 2 = 0\end{aligned}$$

over the four variables $x_1, \ldots, x_4$. Then it is possible to show that no assignment of values to variables will satisfy all equations simultaneously, but setting all variables equal to 0 satisfies three of the four equations.

(a) Let c^* denote the maximum possible number of equations that can be satisfied by an assignment of values to variables. Give a polynomial-time algorithm that produces an assignment satisfying at least $\frac{c^*}{2}$ equations. If you want, your algorithm can be randomized; in this case, the *expected* number of equations it satisfies should be at least $\frac{c^*}{2}$. In either case, you should prove that your algorithm has the desired performance guarantee.

(b) Suppose we drop the condition that each equation must have exactly two variables; in other words, now each equation simply specifies that the sum of an arbitrary subset of the variables, mod 2, is equal to a particular value b_r.

Again, let c^* denote the maximum possible number of equations that can be satisfied by an assignment of values to variables, and give a polynomial-time algorithm that produces an assignment satisfying at least $\frac{c^*}{2}$ equations. (As before, your algorithm can be randomized.) If you believe that your algorithm from part (a) achieves this guarantee here as well, you can state this and justify it with a proof of the performance guarantee for this more general case.

4. **One-Pass Auction.** Suppose you are planning to sell an old textbook via a *one-pass auction*, in which buyers make sequential bids and each bid must be immediately (and irrevocably) accepted or refused. Specifically, the auction works as follows:

- n buyers arrive in a random order. You know n, but no other information about the buyers.
- When buyer i arrives, they make a bid $b_i > 0$. This bid is fixed, and you have no advance knowledge of the distribution of bids.
- You must immediately decide whether to accept the bid or not. If you accept the bid, you are done; all future buyers are turned away. If you reject b_i, buyer i departs and the bid is withdrawn; only then do you see bids from any future buyers.

Your goal is to design a strategy you can follow to ensure a reasonable chance of accepting the highest of the n bids. Here, a *strategy* is a rule by which you decide whether to accept each presented bid, based only on the value of n and the bids you've seen so far.

For example, one possible strategy would be to always accept the first bid presented. Following this strategy, you will accept the highest of the n bids with probability $\frac{1}{n}$, since you only succeed if the highest bid happens to be the first one presented.

Give a strategy under which you accept the highest of the n bids with probability at least $\frac{1}{4}$, regardless of the value of n. (For simplicity, you may assume that n is an even number.) Prove that your strategy achieves this probabilistic guarantee.