1 Announcements

- HW4 due tomorrow!
- HW5 out now!
- Mary isn’t here today! (Back next week). Pras is in charge.

2 Warm-Up

Group Work

1. Show that, in any undirected, unweighted graph $G = (V, E)$ with no self-loops, there is a cut with at least $\frac{|E|}{2}$ edges that cross it. (Recall that a cut in $G$ is just a partition of the vertices $V = S \cup \bar{S}$, and that an edge $\{u, v\}$ crosses the cut if $u \in S$ and $v \in \bar{S}$ or the other way around).

2. Let $\varphi$ be a 3-CNF formula. That is, $\varphi$ is the AND of a bunch of clauses that look like $(x \lor y \lor z)$ (or $(x \lor \bar{y} \lor \bar{z})$, or ..., where $\bar{x}$ means “not $x$”). For example, maybe

$$\varphi = (x_1 \lor \bar{x}_2 \lor x_3) \land (x_2 \lor \bar{x}_4 \lor x_5) \land \cdots \land (x_{23} \lor \bar{x}_1 \lor \bar{x}_5).$$

Suppose that each clause has three distinct literals that appear in it. (e.g., $(x_1 \lor x_1 \lor x_1)$ is not allowed).

Given an assignment $\sigma$ to the variables $x_1, x_2, \ldots$ (eg, $x_1 = TRUE, x_2 = FALSE$, etc), we say that a clause of $\varphi$ is satisfied by $\sigma$ if that clause evaluates to TRUE.

Show that any 3-CNF formula $\varphi$ has an assignment $\sigma$ so that at least $7/8$ of the clauses are satisfied.

3 Recap/Questions

Any questions from the minilectures and/or the quiz? (The probabilistic method; Ramsey numbers; Independent sets)
4 Derandomization via conditional expectation

In class today, we’ll explore a general way to turn an existence proof—like the ones from your warm-up exercise—into an algorithm. This is called “Derandomization via conditional expectation.”

Group Work

Our goal in this group work is to find an efficient, deterministic algorithm to find a cut $(S, \bar{S})$ so that the number of edges crossing the cut is at least $|E|/2$. In general, finding a cut with the maximum number of edges crossing it is NP-hard; but this will at least find a large-ish cut.

Note: There is a straightforward deterministic greedy algorithm to do this. Here, we’ll see a way to derive a deterministic algorithm using conditional expectations.

1. Let $G = (V, E)$ be as in warm-up question 1. Suppose the vertices are ordered $V = \{v_1, v_2, \ldots, v_n\}$.
   Suppose that $S \subseteq V$ is chosen uniformly at random (that is, each $v_i$ is included in $S$ independently with probability $1/2$). Let $X$ be the number of edges crossing the cut $(S, \bar{S})$.
   Convince yourself that $\mathbb{E}[X|v_1 \in S] = |E|/2$.

2. Suppose that you have made some choices for $v_1, v_2, \ldots, v_{t-1}$ (e.g., $v_1 \in S, v_2 \notin S, v_3 \in S, \ldots, v_{t-1} \in S$), so that
   $$\mathbb{E}[X|\text{choices for } v_1, \ldots, v_{t-1}] \geq \frac{|E|}{2}.$$  
   Show that either
   $$\mathbb{E}[X|\text{choices for } v_1, \ldots, v_{t-1}; \text{ and } v_t \in S] \geq \frac{|E|}{2}$$  
or
   $$\mathbb{E}[X|\text{choices for } v_1, \ldots, v_{t-1}; \text{ and } v_t \notin S] \geq \frac{|E|}{2}$$

3. Again, suppose that you have made choices for $v_1, \ldots, v_{t-1}$ so that
   $$\mathbb{E}[X|\text{choices for } v_1, \ldots, v_{t-1}] \geq \frac{|E|}{2}.$$  
   Show how to deterministically, efficiently make a choice for $v_t$ so that
   $$\mathbb{E}[X|\text{choices for } v_1, \ldots, v_{t-1}; \text{ and } v_t] \geq \frac{|E|}{2}.$$
4. Building on your method above, design an algorithm to make a choice for $v_1$, and then $v_2$, and then $v_3$, and so on, so that eventually you have (efficiently, deterministically) found a set $S$ so that at least $|E|/2$ edges cross the cut $(S, \bar{S})$.

[Solutions and discussion of the general paradigm of derandomization via conditional expectation; see lecture notes and/or slides]

**Group Work**

1. Let $\varphi$ be a 3-CNF formula with $n$ variables and $m$ clauses, and 3 distinct variables in each clause. Use the method of derandomization via conditional expectation to give an efficient (polynomial in $n, m$) deterministic algorithm to find an assignment to $\varphi$ so that at least a $7/8$-fraction of the clauses are satisfied.

2. (If time) There is also a natural greedy algorithm for this problem:
   - For $i = 1, 2, \ldots, n$:
     - Assign $x_i$ to be whichever value makes the most currently unsatisfied clauses true (breaking ties arbitrarily).

   In the previous example (maximizing the size of a cut), the algorithm we came up with was secretly the natural greedy algorithm. Is your algorithm from the previous part the same as this natural greedy algorithm? Is it better or worse?