Advanced Algorithms — Problem set 3

Whenever you give an algorithm, also argue for its running time and correctness. Always feel free to ask in the forum or send a mail if you're stuck, need a clarification, or suspect a typo.

1. Given sets $S_1, \dots, S_m \subseteq U$, you would like to obtain an (ϵ, δ)-approximation to $|S|$, where $S = \bigcup_i S_i$. The following is available to you:

   - The numbers $|S_i|$ for $i = 1 \dots m$.
   - For each $x \in U$, the number of $i \in \{1, \dots, m\}$ with $x \in S_i$.
   - For each $i = 1 \dots m$, a procedure for uniformly randomly sampling some $x \in S_i$.

   Describe a randomized algorithm that yields an (ϵ, δ)-approximation to $|S|$.

2. Based on the previous exercise, we extend the algorithm for k-cycle detection from Sheet 2 to approximate counting. Let $G = (V, E)$ be a graph for which we wish to obtain an (ϵ, δ)-approximation to the number of k-cycles. We use without proof that there is an algorithm that outputs colorings $c_1, \dots, c_s : V \to \{1, \dots, k\}$ with $s \le e^k \mathrm{poly}(n)$ and running time $O(s)$ such that each k-cycle is colorful under *at least one* coloring. (In other words, the random choice of a coloring from Sheet 2 can be replaced by an algorithm that explicitly outputs some candidate colorings.) Show how to

   - count, for any fixed coloring c, the colorful k-cycles under c, (Hint: Matrix multiplication or DP.)
   - determine, for each k-cycle C in G, how many colorings make C colorful, and
   - uniformly sample, for any fixed coloring c, a k-cycle C that is colorful under c. (Hint: This is tricky. You can include vertices of C one by one, at each step randomly drawing the next vertex of C that should be included. But the probability of drawing vertices will not be uniform, as some vertices may be contained in more cycles than others. For example, isolated vertices will never be included in a cycle. You can determine the relevant probabilities by modifying the counting algorithm from part (a) to count the k-cycles *including a chosen set of vertices*.)

   Combine this to an algorithm for approximately counting k-cycles. Which running time do you get?

3. Let $n, N \in \mathbb{N}$ and $\mathcal{S} = \{S_1, \dots, S_m\}$ be a set family over $U = \{1, \dots, n\}$ such that $S_i \subseteq U$ for each $i = 1 \dots m$. A weight function is a mapping $w : U \to \{1, \dots, N\}$ and the weight of $S \subseteq U$ is defined as $w(S) = \sum_{x \in S} w(x)$. The *isolation lemma* states that for a uniformly random choice of weight function w, with probability $\ge 1 - \frac{n}{N}$, there is a *unique* set $S^* \in \mathcal{S}$ such that $w(S^*)$ is minimal among $S \in \mathcal{S}$.

   (a) Give an example of a set family $\mathcal{S}$ and a weight function w such that several minimum-weight sets of $\mathcal{S}$ exist under w.

   (b) For $x \in U$, define $\alpha(x) = \min_{S \in \mathcal{S} \text{ with } x \notin S} w(S) - \min_{S \in \mathcal{S} \text{ with } x \in S} w(S \setminus \{x\})$. What is the probability of $\alpha(x) = w(x)$ over the random choice of w? (Hint: Deferred decisions.)

   (c) Consider two distinct sets $S, S' \in \mathcal{S}$ that both have minimum weight under w, and let $x \in U$ be contained in S but not in S'. Show that $\alpha(x) = w(x)$ holds in this case. Use this with (b) to prove the isolation lemma.

   (d) We define "Numeric SAT": The inputs ψ are CNFs φ with an additional numeric clause $(a_1 x_1 + \dots + a_n x_n = b)$ for numbers $a_1, \dots, a_n, b \in \mathbb{N}$ that are "hard-coded" into the formula. For example:

   $$\psi := (x_1 \vee x_2 \vee \overline{x_6}) \wedge (x_3 \vee \overline{x_5}) \wedge (x_4 \vee x_5) \wedge (3x_1 + 5x_2 + 8x_3 + 4x_4 + 9x_5 = 20).$$

   The numeric clause is declared to be satisfied by an assignment if its equation holds, where we view $x_i = 1$ if x_i is set to true, and $x_i = 0$ otherwise. Assume you're given a black-box algorithm that decides whether a "Numeric SAT" formula ψ admits a *unique* satisfying assignment. Show how to wrap this into a randomized algorithm for deciding "vanilla" CNF-SAT.