CS 761: Randomized Algorithms Fall 2019

Problem Set 3

Prof. Gautam Kamath *Deadline: 11:59 PM on November 22, 2019*

You are allowed to discuss the problems in small groups (2-4 people). List your collaborators for each problem. Every person must write up and submit their own solutions.

1. Generalize the randomized algorithm 3-SAT algorithm discussed in class to k-SAT. What is the expected running time of the algorithm? You can assume that the number of clauses m is polynomial in the number of variables n.

2. Suppose we have a Markov chain on the numbers 0 through $n-1$. At each step, if the chain is at number i, it stays where it is with probability 1/2, and moves to $i+1 \pmod n$ with probability 1/2 (that is, it moves to the next number, but "loops around"). Find the stationary distribution and show that for any $\varepsilon > 0$, the mixing time is $O(n^2 \log(1/\varepsilon))$.

3. This problem is due to Ilya Razenshteyn (kind of). Suppose we have a directed graph G on n nodes and m edges. Let R_v be the set of nodes which can be reached from v. In other words, R_v is the set of nodes u, such that there exists a (directed) path from v to u in G. We could compute this set for all nodes simultanously in $O(nm)$ time by using depth-first search, or in $O(n^{2.37\cdots})$ time using fast matrix multiplication. We can improve these bounds if we are satified by just approximating the *size* of R_v for all nodes, using Monte Carlo techniques. Specifically, we will derive a randomized algorithm which computes an (ε, δ)-approximation to $|R_v|$ for all v simultaneously, with running time $O(m \cdot \text{poly}(\log n)/\varepsilon^2)$.

   (a) Suppose we map each vertex to a uniformly random number between 0 and 1, and let f be the function which stores these values. We define $g(v)$ to be $\min_{w \in R_v} f(w)$. Show how to compute $g(v)$ for all vertices v at the same time in time $O(m \cdot \text{poly}(\log n))$.

   (b) Let X_t be the distribution of the minimum of t uniformly random numbers in $[0, 1]$. Show that if we have $O(\log(1/\delta)/\varepsilon^2)$ samples from X_t (where t is unknown), we can get a (ε, δ)-approximation of t.

   (c) Put together the previous two parts to derive an algorithm with the desired guarantees.

4. This problem is due to David Karger. This problem has to do with the Goemans-Williamson (GW) algorithm for MAX-CUT.

   (a) Show that GW can also be used to approximate the s-t MAX-CUT problem, where two specified vertices s and t must be separated, to within 0.878 as well.

   (b) Prove that if a graph is bipartite, then GW will find the optimal solution. Note that the maximum cut in a bipartite graph cuts all edges.

   (c) Generalize the previous part: prove that for any $\varepsilon > 0$, there exists a $\delta > 0$ such that for any graph that has a max-cut of value at least $(1-\varepsilon)m$, then GW algorithm will find a cut of value at least $(1-\delta)m$. How small of a δ can you get in terms of ε? Hint: consider the value of $\arccos(x)$ near $x = -1$.