CME 305: Discrete Mathematics and Algorithms

Instructor: Reza Zadeh (rezab@stanford.edu)

HW#2 – Due at the beginning of class Thursday 02/05/15

1. **(Kleinberg Tardos 7.27)** Some of your friends with jobs out West decide they really need some extra time each day to sit in front of their laptops, and the morning commute from Woodside to Palo Alto seems like the only option. So they decide to carpool to work. Unfortunately, they all hate to drive, so they want to make sure that any carpool arrangement they agree upon is fair and doesn't overload any individual with too much driving. Some sort of simple round-robin scheme is out, because none of them goes to work every day, and so the subset of them in the car varies from day to day.

   Here's one way to define *fairness*. Let the people be labeled $S = \{p_1, \ldots, p_k\}$. We say that the *total driving obligation* of p_j over a set of days is the expected number of times that p_j would have driven, had a driver been chosen uniformly at random from among the people going to work each day. More concretely, suppose the carpool plan lasts for d days, and on the i^{th} day a subset $S_i \subseteq S$ of the people go to work. Then the above definition of the total driving obligation Δ_j for p_j can be written as $\Delta_j = \sum_{i:p_j \in S_i} \frac{1}{|S_i|}$. Ideally, we'd like to require that p_j drives at most Δ_j times; unfortunately, Δ_j may not be an integer.

   So let's say that a *driving schedule* is a choice of a driver for each day — that is, a sequence $p_{i_1}, p_{i_2}, \ldots, p_{i_d}$ with $p_{i_t} \in S_t$ — and that a *fair driving schedule* is one in which each p_j is chosen as the driver on at most $\lceil \Delta_j \rceil$ days.

   (a) Prove that for any sequence of sets $S_1, \ldots, S_d$, there exists a fair driving schedule.

   (b) Give an algorithm to compute a fair driving schedule with running time polynomial in k and d.

2. Recall Karger's algorithm for the global min-cut problem. In this problem we modify the algorithm to improve its running time.

   (a) Prove that if we stop the original Karger's algorithm when the remaining number of vertices is
   $$\max\left\{\lceil 1 + n/\sqrt{2} \rceil, 2\right\},$$
   the probability that we have contracted an edge in the min-cut is less than 1/2. Lets call this procedure *Partial Karger*.

   (b) Now suppose we apply *Partial Karger* to two copies of G to produce graphs G_1 and G_2. We then recursively apply these steps to G_1 and G_2 and so on until each recursive call returns a graph on two vertices. If $r(n)$ is the running time of this process as a function of the number of vertices n of G, derive a recursive equation for $r(n)$ and solve it to obtain an explicit expression for the running time (you may use $O(\cdot)$ notation to simplify your recursive equation).

   (c) Show that the algorithm in part (b) produces $O(n^2)$ contracted graphs on two vertices each. Prove that the probability that at least one of them contains a global min-cut is at least $1/\log(n)$ up to a multiplicative constant.

   Hint: Think of the recursion as a binary tree with paths leading to the $O(n^2)$ leaves representing the two-vertex contracted graphs.

(d) Compare the running time of the above algorithm to Karger's original given the same probability of failure.

3. An independent set in a graph is a set of vertices with no edges connecting them. Let G be a graph with $nd/2$ edges ($d > 1$), and consider the following probabilistic experiment for finding an independent set in G: delete each vertex of G (and all its incident edges) independently with probability $1 - 1/d$.

   (a) Compute the expected number of vertices and edges that remain after the deletion process. Now imagine deleting one endpoints of each remaining edge.

   (b) From this, infer that there is an independent set with at least $n/2d$ vertices in any graph with on n vertices with $nd/2$ edges.

4. Let $G = (V, E)$ be a bipartite graph on n vertices with a list $S(v)$ of at least $1 + \log_2 n$ colors associated with each vertex $v \in V$. A graph coloring is an assignment of colors to nodes such that no two adjacent nodes have the same color. Prove that there is a proper coloring of G assigning to each vertex of v a color from its list $S(v)$.

5. Prove that a graph can only have at most $\binom{n}{2}$ different cuts that realize the global minimum cut value.

6. Exhibit a graph $G = (V, E)$ where there are an exponential (in $|V| = n$, the number of nodes) number of minimum cuts between a particular pair of vertices. Do this by constructing a family of graphs parameterized on n and give a pair of vertices s, t such that there are exponentially many minimum cuts between s and t.

7. A *cycle cover* C of a directed graph $G(V, E)$ is a collection of vertex-disjoint directed cycles so that each vertex in V belongs to some cycle in C. Give a polynomial time algorithm to compute a cycle cover of a given directed graph, or correctly report that one does not exist.

8. Exhibit a directed graph that has cover time exponentially large in the number of nodes. Contrast this with the cover time of undirected graphs discussed in class.

9. A connected d-regular graph G is one in which all vertices have degree d. Prove the cover time of a d-regular graph with n nodes is $O(n^2 \log(n))$.