CME 305: Discrete Mathematics and Algorithms

Instructor: Reza Zadeh (rezab@stanford.edu)

HW#2 – Due at the beginning of class Thursday 02/04/16

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

5. 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.

6. 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.

7. You are the sole proprietor of the company *Widgets Inc.*, a supplier of quality widgets to a large corporation in your country. You have just received a big order of w widgets from the large corporation, and are preparing a shipment from your facility, s, to their headquarters, t. Due to various idiosyncrasies in your country's postal service, the fastest way to deliver packages is to route them by hand and choose which postal hubs the package will stop in before it reaches its final destination. However, occasionally a hub's computer will crash and massively delay all shipments exiting the city. Since you want to reduce the effect of such a crash, you decide that you want to minimize the number of widgets that are sent through any given hub, while still sending all w widgets. Thus, if we model our country's postal service as a graph G with a set of hubs and cities V and a set of connections between hubs E, and assume that every edge in our graph has infinite capacity (in other words, we can send as many widgets across a given connection as we like), we would like to find the smallest number k such that we can route all w widgets from s to t while ensuring that no city handles more than k widgets. Find a polynomial time algorithm to do so.

8. Compute the cover time of a Hamiltonian cycle with n vertices.

9. Suppose we have a $2n \times 2n$ ($n \geq 2$) table where each cell is filled with an integer in $\{1, 2, 3, ..., 2n^2\}$. Moreover, each integer shows up exactly twice. Show that one can pick $2n$ cells that satisfy all the following conditions: (1). all the numbers written in these cells are distinct; (2). in each row exactly one cell is picked out; (3) in each

column exactly one cell is picked out.

10. After your success selling widgets, you have grown your business and now sell widgets of many different sizes and shapes. Shipping these widgets requires special boxes which you have custom built by a supplier. Unfortunately, having the supplier ship you these boxes is very expensive. You notice that some of the smaller of the boxes you need fit inside some of the larger ones, and seeing an opportunity to save money, you ask your supplier to ship smaller boxes inside of larger ones. He agrees, as long as no two boxes lie side-by-side inside of a larger one– in other words, as long as no two boxes A and B lie inside box C if neither A contains B nor B contains A. (Otherwise, the two inner boxes might damage each other during shipping.) Assuming you need n different boxes and you know which of the boxes you need can fit inside others, find a polynomial time algorithm to compute the minimum number of separate shipments needed, and compute the algorithm's running time.