CME 305: Discrete Mathematics and Algorithms

Instructor: Reza Zadeh (rezab@stanford.edu)

HW#3 – Due at the beginning of class Thursday 02/26/15

1. Consider a model of a nonbipartite *undirected* graph in which two particles (starting at arbitrary positions) follow a random walk i.e. with each time step both particles uniform randomly move to one of the neighbors. Prove that the expected time until they collide is $O(n^6)$. A collision is when both particles are on the same node at the same time step.

2. Let A be a $n \times n$ matrix, B a $n \times n$ matrix and C a $n \times n$ matrix. We want to design an algorithm that checks whether $AB = C$ without calculating the product AB. Provide a randomized algorithm that accomplishes this in $O(n^2)$ time with high probability.

3. Given a connected, undirected graph $G = (V, E)$ and a set of terminals $S = \{s_1, s_2, \ldots, s_k\} \subseteq V$, a multiway cut is a set of edges whose removal disconnects the terminals from each other. The multiway cut problem asks for the minimum weight such set. The problem of finding a minimum weight multiway cut is NP-hard for any fixed $k \geq 3$. Observe that the case $k = 2$ is precisely the minimum $s - t$ cut problem.

   Define an *isolating cut* for s_i to be a set of edges whose removal disconnects s_i from the rest of the terminals. Consider the following algorithm

   - For each $i = 1, \ldots, k$, compute a minimum weight isolating cut for s_i, say C_i.
   - Discard the heaviest of these cuts, and output the union of the rest, say C.

   Each computation in Step 1 can be accomplished by identifying the *terminals* in $S - \{s_i\}$ into a single node, and finding a minimum cut separating this node from s_i; this takes one max-flow computation. Clearly, removing C from the graph disconnects every pair of terminals, and so is a multiway cut.

   (a) Prove that this algorithm achieves a $2 - 2/k$ approximation.

   (b) Prove that this analysis is tight by providing an example graph where the approximation bound is exactly achieved.

4. Consider variants on the metric TSP problem in which the object is to find a simple path containing all the vertices of the graph. Three different problems arise, depending on the number (0, 1, or 2) of endpoints of the path that are specified. If zero or one endpoints are specified, obtain a 3/2 factor algorithm.

   Hint. Consider modifying Christofides algorithm for metric TSP.

5. Recall the *minimum vertex cover* problem: given a graph $G(V, E)$ find a subset $S^* \subseteq V$ with minimum cardinality such that every edge in E has at least one endpoint in S^*. Consider the following greedy algorithm. Find the highest degree vertex, add it to the vertex cover S and remove it along with all incident edges. Repeat iteratively. Prove that this algorithm has an unbounded approximation factor i.e. for any c there exists an input graph G such that $|S| \geq c$ OPT.

6. Recall that to show a problem X is NP-complete we must show that it is in NP and construct a polynomial-time computable function that maps inputs of a known NP-complete problem to inputs of X (e.g. 3-SAT $\leq_p X$) in a way that preserves problem satisfiability. The last statement would imply that all problems in NP are polynomial reducible to X, hence X is NP-complete.

   (a) Integer Programming (IP) decision problem asks whether there exists an integer solution x satisfying linear constraints $Ax \leq b$ and with objective value $c^T x$ at least k. Prove that IP is NP-complete.

   (b) The Clique decision problem asks whether a clique (a complete subgraph) of size k exists in given graph G. Prove that Clique is NP-complete.

   (c) Consider a modified version of Clique in which all vertices have degree at most 3. Is this problem NP-complete? Why or why not?

   (Hint: Use an NP-Complete problem from class)

7. An oriented incidence matrix B of a directed graph $G(V, E)$ is a matrix with $n = |V|$ rows and $m = |E|$ columns with entry B_{ve} equal to 1 if edge e enters vertex v and -1 if it leaves vertex v. Let $M = BB^T$.

   (a) Prove that $rank(M) = n - w$ where w is the number of connected components of G.

   (b) Show that for any $i \in \{1, \ldots, n\}$,

$$\det M_{ii} = \sum_N (\det N)^2,$$

   where $M_{ii} = M \backslash \{i^{th} \text{ row and column}\}$, and N runs over all $(n-1) \times (n-1)$ submatrices of $B \backslash \{i^{th} \text{ row}\}$. Note that each submatrix N corresponds to a choice of $n - 1$ edges of G.

   (c) Show that

$$\det N = \begin{cases} \pm 1 & \text{if edges form a tree} \\ 0 & \text{otherwise} \end{cases}$$

   This implies that $t(G) = \det M_{ii}$, where $t(G)$ is the number of spanning trees of G. In this definition of a tree, we treat a directed edge as an undirected one.

   (d) Show that for the complete graph on n vertices K_n,

$$\det M_{ii} = n^{n-2}.$$

8. Given a sequence p_i of stock prices on n days, we need to find the best pair of days to buy and sell. i.e. find i and j that maximizes $p_j - p_i$ subject to $j \geq i$. Give an $O(n)$ dynamic programming solution.