HW 6 (due Monday, 6pm, October 19, 2015)

NEW CS 473: Theory II, Fall 2015 Version: **1.11**

Collaboration Policy: This homework can be worked in groups of up to three students.
Submission is online on moodle.

1. (50 PTS.) Collect the min-cut.

Consider the algorithm that given a graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ with n vertices and m edges, starts with the empty forest $\mathsf{H}_0 = (\mathsf{V}, \emptyset)$ where every vertex is a single node tree.

In the ith epoch, the algorithm picks an edge e randomly and uniformly from the original set of edges E, and check:

- If the vertices of e belong to different trees of the forest H_{i-1}, then it is ***useful***. If so the algorithm adds e to this graph, to form the new forest H_i, and continues to the next epoch.
- If the edge connects two vertices that are already in the same tree of H_{i-1}, then this edge is ***useless***.

The algorithm continues in this fashion, till it computes H_{n-2}, which has exactly two trees.

(A) (5 PTS.) Let H_{i-1} be as above. Assume that the minimum cut in G is of size k, and H_{i-1} does not contain any edges of this min-cut. Let U_i be the set of edges of G that are useful as far as H_{i-1} is concerned. Prove, a lower bound, as large as possible, on U_i as a function of i, k, n and m.

(B) (5 PTS.) Provide a lower bound, as large as possible, on the probability that a random edge of G is useful for H_{i-1}. (Use (A).)

(C) (10 PTS.) Let N_i be the number of edges sampled in the ith epoch till a useful edge for H_{i-1} is found. Prove an upper bound, as small as possible, on $\mathbf{E}[N_i]$. (Use (B).)

(D) (10 PTS.) Consider the forest H_{n-2}. Prove a lower bound as large as possible, on the probability that H_{n-2} (with its two trees), represents a min-cut of G. Provide a full and self contained proof of this lower bound (i.e., you can not refer to the class notes/web/the universe for a proof of this).

(E) (10 PTS.) The expected number of edges inspected by the above algorithm to construct H_{n-2} is

$$\alpha = \mathbf{E}[N_1 + N_2 + \ldots + N_{n-2}] .$$

Provide and prove an upper bound, as small as possible, on the value of α (as a function of k, n and m.

(F) (10 PTS.) (Hard.) Provide a randomized algorithm, as fast as possible, that computes the connected components of H_{n-2} (and its associated vertex cut [i.e., it does not compute the edges in the cut themselves]). Prove a bound on the expected running time of your algorithm.
For full credit, the expected running time of your algorithm should be $O(\alpha)$, and should not use union-find and hashing[1]. A slower algorithm would get half the points (and then this part is not that hard).
(Computing the edges of the cut themselves, onces the vertex cut is known, can be easily done in $O(m)$ time.)
Note: You do not know what the value of k is!

2. (50 PTS.) Disjoint paths.

Let $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ be a directed graph, with n vertices and m edges. Let s and t be two vertices in G. For the sake of simplicity, assume that there are no u, v such that (u, v) and (v, u) are in G.

A set of paths $\mathcal{P}$ in G is ***edge disjoint*** if no two paths in $\mathcal{P}$ share an edge.

(A) (10 PTS.) Let $\mathcal{P}$ be a set of k edge disjoint paths from s to t. Let π be a path from s to t (which is not in $\mathcal{P}$). Prove or disprove: There is a set $\mathcal{P}'$ of k edge disjoint paths from s to t in G that contains π as one of the paths.

[1] Note that I see how hashing helps here.

(B) (10 PTS.) Let $\mathcal{P}$ be a given set of edge disjoint paths from s to t. Let $\mathsf{E}(\mathcal{P})$ be the set of edges used by the paths of $\mathcal{P}$. The ***leftover graph*** $\mathsf{G}_{\mathcal{P}}$ is the graph where $(u, v) \in \mathsf{E}(\mathsf{G}_{\mathcal{P}})$ if $(u, v) \in \mathsf{E}(\mathsf{G}) \setminus \mathsf{E}(\mathcal{P})$ or $(v, u) \in \mathsf{E}(\mathcal{P})$ (note that the edge (u, v) is the reverse edge of (v, u)).

Describe how to compute the leftover graph in $O(m)$ time (no hashing please).

(C) (5 PTS.) Let $\mathcal{P}$ be a set of k edge disjoint paths from s to t. Let π be a path in $\mathsf{G}_{\mathcal{P}}$ from s to t. Prove that there is a set of $\mathcal{P}'$ of $k+1$ edge disjoint paths from s to t in G. In particular, show how to compute $\mathcal{P}'$ given $\mathcal{P}$ and π in $O(m)$ time. (For credit, your solution should be self contained and not use min-cut max-flow theorem or network flow algorithms.)

(D) (5 PTS.) The natural greedy algorithm for computing the maximum number of edge disjoint paths in G, works by starting from an empty set of paths $\mathcal{P}_0$, then in the ith iteration, it finds a path π_i in the leftover graph $\mathsf{G}_{\mathcal{P}_{i-1}}$ from s to t, and then compute a set of i edge-disjoint paths $\mathcal{P}_i$, by using the algorithm of (C) on $\mathcal{P}_{i-1}$ and π_i.

Assume the algorithm stops in the $(k+1)$th iteration, because there is not path from s to t in $\mathsf{G}_{\mathcal{P}_k}$. We want to prove that the k edge-disjoint paths computed (i.e., $\mathcal{P}_k$) is optimal, in the sense that there is no larger set of edge-disjoint paths from s to t in G.

To this end, let S be the set of vertices that are reachable from s in $\mathsf{G}_{\mathcal{P}_k}$. Let $T = \mathsf{V}(\mathsf{G}) \setminus S$ (observe that $t \in T$). Prove, that every path in $\mathcal{P}_k$ contains exactly one edge of

$$(S, T) = \{(u, v) \in \mathsf{E}(\mathsf{G}) \mid u \in S, v \in T\}.$$

(Hint: Prove first that no path of $\mathcal{P}_k$ can use an edge of the "reverse" set (T, S).)

(E) (5 PTS.) Consider the setting of (D). Prove that $k = |\mathcal{P}_k| = |(S, T)|$.

(F) (5 PTS.) Consider any set X of edge-disjoint paths in G from s to t. Prove that any path π of X must contain at least one edge of (S, T).

(G) (5 PTS.) Prove that the greedy algorithm described in (D) indeed computes the largest possible set of edge-disjoint paths from s to t in G.

(H) (5 PTS.) What is the running time of the algorithm in (D), if there are at most k edge-disjoint path in G?