HW 7 (due Tuesday, at 11am, March 31, 2015)

OLD CS 473: Fundamental Algorithms, Spring 2015 Version: **1.07**

Collaboration Policy: For this homework, Problems 1–3 can be worked in groups of up to three students.

---

1. (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.

(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?

2. (50 PTS.) Great Hashing.

Here, we investigate the construction of hash table for a given set W, provided in advance. We care only but the time to do lookup in the resulting hash table.

Let $U = \{1, \ldots, m\}$, and $p = m + 1$ is a prime number (potentially large).

Let $W \subseteq U$, such that $n = |W|$, and s is an arbitrary number $\geq n$ (but smaller than p). Consider the hash function

$$h(x) = h_k(x) = (kx \mod p) \mod s.$$

We have two parameters $k \in U$ and s, and our purpose is to prove that one can always choose these parameters such that one can build a good hash table.

(A) (5 PTS.) Consider two distinct numbers $x, y \in W$, such that $h(x) = h(y)$. Prove that then

$$k(x - y) \bmod p \in \{\pm s, \pm 2s, \pm 3s, \ldots, \pm \lfloor (p-1)/s \rfloor s\}.$$

(B) (5 PTS.) Prove that for fixed x and y, there are at most $2(p-1)/s$ choices of k, such that $h(x) = h(y)$. (You can use here without proof that for any $\alpha, \beta \in U$ there is a unique $z \in U$ such that $\alpha z = \beta \bmod p$.)

(C) (5 PTS.) Consider the set of elements of W that get mapped to the value j, for a specific value of k. That is, the set

$$B_k(j) = \left\{ x \in W \;\middle|\; h_k(x) = j \right\}$$

of all the elements that get mapped to value j by the hash function h. In particular, let $V_k = \left\{ \{x, y\} \;\middle|\; x, y \in W, x \neq y, h_k(x) = h_k(y) \right\}$ be all the pairs that collide under h_k. Prove that $\sum_{k=1}^{p-1} |V_k| \leq \frac{2(p-1)}{s} \binom{n}{2}$.

(D) (5 PTS.) Let $\beta_k(j) = \beta_{k,s}(j) = |B_k(j)|$. Prove using (C) that $\displaystyle\sum_{k=1}^{p-1} \sum_{j=1}^{s} \binom{\beta_k(j)}{2} < \frac{(p-1)n^2}{s}$.

(E) (5 PTS.) Prove that there exists $k \in U$, such that $\displaystyle\sum_{j=1}^{s} \binom{\beta_k(j)}{2} < \frac{n^2}{s}$.

(F) (5 PTS.) Prove that $\sum_{j=1}^{s} \beta_k(j) = |W| = n$.

(G) (5 PTS.) Here, let set $s = n$. Prove that there exists a $k \in U$ such that $\sum_{j=1}^{s} \left(\beta_k(j)\right)^2 < 3n$

(H) (5 PTS.) Prove that there exists a $k' \in U$, such that the function $h_{k'}(x) = (k'x \mod p) \mod n^2$ is one-to-one when restricted to W.

(I) (5 PTS.) Conclude, that one can construct a hash-table for W, of size $O(n^2)$, such that there are no collisions, and a search operation can be performed in $O(1)$ time (note that the time here is worst case, also note that the construction time here is quite bad - ignore it).

(J) (5 PTS.) Using the above describe how to build a two-level hash-table that uses $O(n)$ space, stores the set W, and perform a lookup operation in $O(1)$ time (worst case).

(Hint: Use (G) for the top level hash table, and use (I) for the hash-table inside each bucket.)