Homework set 3

Note: the homework sets are not for submission. They are designed to help you prepare for the quizzes. It is **highly recommended** that you solve all problems and write the solutions down.

1. We are given a directed graph $G = (V, E)$, with two special vertices s and t, and non-negative capacities $c(e)$ on edges $e \in E$. Assume that s has no incoming edges and t has no outgoing edges.

   (a) Show an efficient algorithm that finds a maximum s-t flow f in G, such that f is acyclic (A flow f is acyclic, if G contains no cycles, where every edge carries positive flow).

   (b) A collection $\mathcal{P}$ of paths connecting s to t, together with values $f'(P) \geq 0$ for each $P \in \mathcal{P}$ is called a *valid flow-paths solution*, iff for every edge $e \in E$, $\sum_{\substack{P\in\mathcal{P}:\\ e\in P}} f'(P) \leq c(e)$.

   Assume that we are given a valid acyclic s-t flow f in G. Show an efficient algorithm that finds a valid flow-paths solution $(\mathcal{P}, f')$, with $|\mathcal{P}| \leq |E|$, such that for each edge $e \in E$,

   $$\sum_{\substack{P\in\mathcal{P}:\\ e\in P}} f'(P) = f(e).$$

   Prove the algorithm's correctness.

   (c) Let OPT_f denote the value of the maximum flow in G. Given a valid flow-paths solution $(\mathcal{P}, f')$, its value is denoted by $v(\mathcal{P}, f') = \sum_{P\in\mathcal{P}} f'(P)$. Let v^* be the maximum value of any valid flow-paths solution. Prove that $v^* = \mathsf{OPT}_f$.

   (d) Assume now that all edge capacities are integral. Prove that there is an optimal flow-path solution, where the values $f'(P)$ for every path P are integral, and the number of paths with non-zero value $f'(P)$ is at most $|E|$.

2. In this question we study a variant of the Ford-Fulkerson algorithm. Recall that given a residual graph G_f and an s-t path P in G_f, we have denoted by $b_f(P) = \min_{e\in P}\{c_f(e)\}$ - the minimum residual capacity of any edge on P. We run the standard Ford-Fulkerson algorithm, except that we choose augmenting paths according to the following rule: select a path P with maximum value $b_f(P)$, breaking ties arbitrarily. For each iteration i of the algorithm, let b_i denote the value $b_f(P)$ of the path P selected in iteration i. Prove or disprove: The values b_i, for $i \geq 1$, always form a non-increasing sequence.

   Hint: the statement is false.

3. We are given a flow network $G = (V, E)$, with positive integral capacities $c(e)$ on edges $e \in E$, a source s and a sink t. Recall that an s-t cut in G is a partition (A, B) of the vertices of V, such that $s \in A$, $t \in B$. An s-t cut (A, B) is a minimum cut iff the value $C(A, B)$ is minimal among all s-t cuts. Notice that it is possible for a graph to contain several minimum cuts.

   - Show an example of a graph G, that contains $\Omega(n^2)$ minimum s-t cuts, where $n = |V|$.
   - Show an example of a graph G that contains a unique minimum s-t cut (that is, the number of minimum s-t cuts in G is 1).

- Show an efficient algorithm to determine whether G contains a unique minimum s-t cut, or the number of minimum cuts is greater than 1. Prove the algorithm's correctness.
- An s-t cut (A, B) in G is called the *best* minimum s-t cut iff it minimizes $|E(A, B)|$ among all s-t cuts. Show an efficient algorithm to compute the best minimum s-t cut in G.

4. Given a graph G (that can be directed or undirected), and two special vertices s and t, a collection of *node-disjoint s-t paths* is any set $\mathcal{P} = \{P_1, \ldots, P_k\}$ of paths, where each path $P_i \in \mathcal{P}$ connects s to t, and every vertex $v \in V(G) \setminus \{s, t\}$ appears on at most one path in $\mathcal{P}$.

   (a) Design an efficient algorithm, that, given a directed graph G, and two vertices $s, t \in V(G)$, computes a largest-cardinality set $\mathcal{P}$ of node-disjoint s-t paths in G.

   (b) Design an efficient algorithm, that, given an undirected graph G, and two vertices $s, t \in V(G)$, computes a largest-cardinality set $\mathcal{P}$ of node-disjoint s-t paths in G.

   (c) Suppose we are given an undirected graph G, and three distinct vertices $x, y, z \in V(G)$. We would like to know whether there is a simple path from x to z that contains y. Design an efficient algorithm that finds such a path in G if it exists. Prove the algorithm's correctness.

5. Suppose we are given an $n \times n$ square grid, some of whose squares are colored black, and the rest are white. We are also given n tokens. Describe and analyze an algorithm to determine whether tokens can be placed on the grid, so that:

   - Every token is on a distinct white square;
   - Every row of the grid contains exactly one token; and
   - Every column of the grid contains exactly one token.