CS 580-Spring 2019
Instructor: Jeremiah Blocki
TAs: Hamidreza Amini Khorasgani, Akash Kumar
I collaborated with (...). I affirm that I wrote the solutions in my own words and that I understand the solutions I am submitting.

Homework 4

Due Date: March 8, 2019 at 11:59PM on Gradescope.

Question 1

1. Use Dinic's algorithm to compute the maximum flow of the graph below. You should show the level graph/blocking flow after every step of the algorithm. If you find that it saves time you may scan handwritten pictures. (Alternatively, you may also copy-paste the LaTeX source for the graph below)

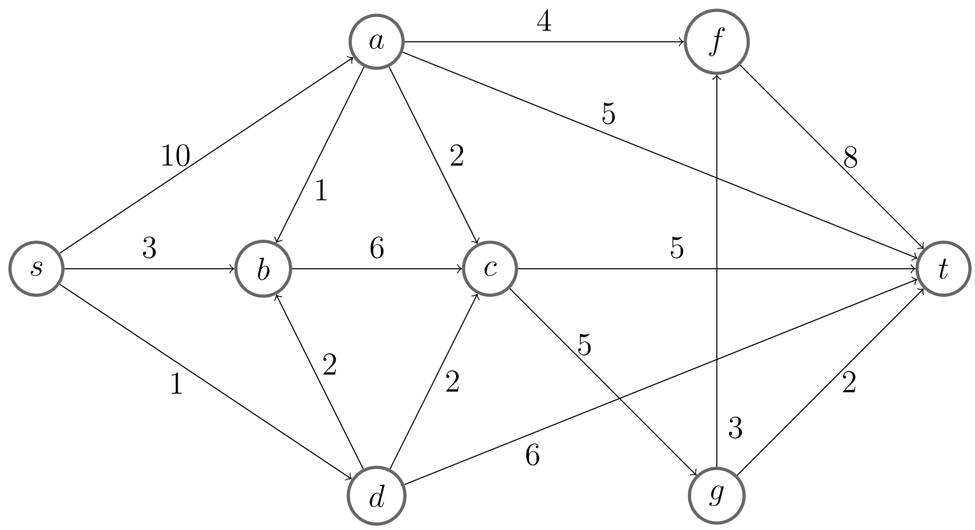

Figure 1: Flow Network

2. What is the minimum capacity s-t cut for the above flow network?

Question 2

1. Suppose we are given a bipartite graph $G = (V = (L, R), E)$ where the left nodes $L = \{u_1, \ldots, u_n\}$ and the right nodes are $R = \{v_1, \ldots, v_n\}$ we want to determine whether or not a perfect matching $M^* \subseteq E$ exists with $|M^*| = n$. Further suppose that G contains the edge $\{u_i, v_i\} \in E$ for each $i \leq n - \log n$, but for $i > n - \log n$ it may or may not be the case that $\{u_i, v_i\} \in E$. Develop an $O(|E| \log n)$ time algorithm to determine whether or not a perfect matching exists and explain why your algorithm is correct.

2. Suppose that G does not have a perfect matching. Extend your algorithm from the previous part to produce a short certificate C with $|C| = O(n)$ which proves that G

does not have a perfect matching. You should also develop a linear time algorithm $\mathcal{A}$ which validates the certificate in time $O(|E| + n)$. You should prove that $\mathcal{A}(C, G)$ always accepts honestly produced certificates (those output by your algorithm), but that if G does have a perfect matching then for *any* certificate C $\mathcal{A}(C, G)$ always rejects.

Question 3 (Network flow with taxation)

Recall that in the standard network flow problem we required that the for each vertex v (excluding the source s and sink t) the sum of the flow into that vertex is equal to the sum of flow out of that vertex. Suppose that we replace this *conservation constraint* with a *taxation constraint*. In particular, suppose each vertex represents a country and that each country $v \notin \{s, t\}$ has an associated tax-rate $0 < t_v < 1$ meaning that country v will keep t_v fraction of the goods flowing through node v. Given a flow network $G = (V, E)$ with maximum capacities $c(e)$ on each edge $e \in V$ and tax rates t_v for each node $v \notin \{s, t\}$ our goal is to find the maximum amount of goods that can be transported from s to t under these taxation constraints. Write down a LP to solve this problem. You should explain why your linear program is correct. (Note: A good explanation will always include a clear/intuitive description of each variable and each constraint.)

Question 4

Suppose that we order the edge relaxations in each pass of the Bellman-Ford algorithm as follows. Before the first pass, we assign an arbitrary linear order $v_1, v_2, \ldots, v_{|V|}$ to the vertices of the input graph $G = (V, E)$. Then, we partition the edge set E into $E_f \cup E_b$, where $E_f = \{(v_i, v_j) \in E : i < j\}$ and $E_b = \{(v_i, v_j) \in E : i > j\}$. (Assume that G contains no self-loops, so that every edge is in either E_b or E_f). Define $G_f = (V, E_f)$ and $G_b = (V, E_b)$.

- Prove that G_f is acyclic with topological sort $\langle v_1, v_2, \ldots, v_{|V|} \rangle$ and that G_b is acyclic with topological order $\langle v_{|V|}, v_{|V|-1}, \ldots, v_1 \rangle$.

- Suppose that we implement each pass of the Bellman-Ford algorithm in the following way. We visit each vertex in the order $v_1, v_2, \ldots, v_{|V|}$ relaxing edges of E_f that leave the vertex. We then visit each vertex in the order $v_{|V|}, v_{|V|-1}, \ldots, v_1$ relaxing edges of E_b that leave the vertex.

- Prove that with this scheme, if G contains no negative-weight cycles that are reachable from the source vertex s, then after only $\lceil |V|/2 \rceil$ passes over the edges, $v.d = \delta(s, v)$ for all vertices $v \in V$. (Here, $\delta(s, v)$ represents the length of the shortest path from s to v and $v.d$ represents the distance we have computed in the algorithm.)

- Does this scheme improve the asymptotic running time of the Bellman-Ford algorithm?

(Bonus) 10 points

A set of positive integers $P = \{a_1, a_2, \ldots, a_n\}$ is given. Give an algorithm which outputs two subsets of P, S_1 and S_2 such that $S_1 \cup S_2 = P$ and $S_1 \cap S_2 = \emptyset$ and $d := \left|\sum_{a_i \in S_1} a_i - \sum_{a_j \in S_2} a_j\right|$ is minimum (If $S = \emptyset$, we define $\sum_{a_i \in S} a_i = 0$). Your algorithm should run in time $O(nM)$ where $M = \sum_{i=1}^{n} a_i$ represents the summation of all numbers in the set P.