CSC473 W18: Homework Assignment #3

Due: April 4, by midnight

Guidelines: (read fully!!)

- Your assignment solution must be submitted as a *typed* PDF document. Scanned handwritten solutions, solutions in any other format, or unreadable solutions will **not** be accepted or marked. You are encouraged to learn the $\LaTeX$ typesetting system and use it to type your solution.
- To submit this assignment, use the MarkUs system, at URL `https://markus.teach.cs.toronto.edu/csc473-2018-01`
- This is a *group assignment*. This means that you can work on this assignment with *at most one other* student. You are *strongly encouraged* to work with a partner. Both partners in the group should work on and arrive at the solution together. Both partners receive the same mark on this assignment.
- Work on all problems together. For each problem, one of you should write the solution, and one should proof-read and revise it. The first page of your submission must list the *name*, *student ID*, and *UTOR email address* of both group members. It should also list, for each problem, which group member wrote the problem, and which group member proof-read and revised it.
- You **may not** consult any other resources except: your partner; your class notes; your textbook and assigned readings. *Consulting any other resource, or collaborating with students other than your group partner, is a violation of the academic integrity policy!*
- You may use any data structure, algorithm, or theorem previously studied in class, or in one of the prerequisites of this course, by just referring to it, and without describing it. This includes any data structure, algorithm, or theorem we covered in lecture, in a tutorial, or in any of the assigned readings. Be sure to give a *precise reference* for the data structure/algorithm/result you are using.
- Unless stated otherwise, you should justify all your answers using rigorous arguments. Your solution will be marked based both on its completeness and correctness, and also on the clarity and precision of your explanation.

Question 1. (16 marks) In this question you will fill in some of the gaps in the linear programming lecture notes.

a. Using Lemma 5 from the Linear Programming lecture notes, prove Lemma 7, i.e. prove that for any $m \times n$ matrix A and any $m \times 1$ vector b, exactly one of the following two statements is true:

i. There exists a $x \in \mathbb{R}^n$ such that $x \geq 0$ and $Ax \leq b$.

ii. There exists a $y \in \mathbb{R}^m$ such that $y \geq 0$, $A^\intercal y \geq 0$ and $b^\intercal y < 0$.

b. Using what you proved in the subproblem above (i.e. Lemma 7 from the Linear Programming notes), show that for any $m \times n$ matrix A and any $m \times 1$ vector b such that $\{x : Ax \leq b, x \geq 0\} \neq \emptyset$, and any $n \times 1$ vector c, the system of inequalities

$$\begin{aligned} c^\intercal x &\geq 1 \\ Ax &\leq b \\ x &\geq 0 \end{aligned}$$

is infeasible if and only if there exists a $y \in \mathbb{R}^m$, $y \geq 0$, such that $A^\intercal y \geq c$ and $b^\intercal y < 1$.

Question 2. (25 marks) Let $G = (V, E)$ be a ***directed*** graph where each edge e has weight $w_e > 0$. (Note that edges do ***not*** have capacities.) Let s and t be two nodes of G. Given a flow f from s to t, the weight $w(f)$ of the flow is defined as $w(f) = \sum_{e \in E} w_e f_e$. Consider a flow problem in which the objective is to send one unit of flow from s to t (i.e. the total flow leaving s must equal 1, and, equivalently, the total flow entering t must equal 1) so that the weight of the flow is minimized.

Note: Observe that this flow problem is feasible if and only if there exists some path from node s to node t in G.

In the following questions you can assume that the flow problem above is feasible, i.e. node t is reachable from node s.

a. Write a linear program equivalent to this flow problem. Derive the dual of this linear program, and give it in a form that is simplified as much as possible. The dual program must have one variable for each node of G, and one constraint for each edge of G.

b. Given a feasible solution y to the dual linear program, characterize when there exists a feasible flow f such that y and f satisfy the complementary slackness conditions.

c. Give a *primal-dual* algorithm which finds a pair of optimal feasible solutions to the primal and dual linear programs. Your algorithm should run in time polynomial in $n = |V|$ and $m = |E|$. At any time step, it should keep a *feasible* solution y to the dual linear program and try to find a feasible flow f such that y and f satisfy the complementary slackness conditions; if such a flow exists, the algorithm should terminate, and if it doesn't, the algorithm should modify y while keeping it feasible. Describe your algorithm in detail, and argue why it correctly solves the problem and runs in time polynomial in m and n.

Hint: Start with $y_u = 0$ for all $u \in V$. At the first step of the algorithm, modify y_s so that the objective function value of y increases as much as possible while keeping y feasible. Argue that the dual constraint for at least one edge going out of s becomes tight. Then, at every step of the algorithm, for every vertex u reachable from s along edges for which the dual constraints are currently tight, change the dual varible y_u to $y_u + \delta$ where $\delta \in \mathbb{R}$ is chosen so that y remains feasible and the objective function value increases as much as possible.

Question 3. (9 marks) Consider the problem in which we are given a ***directed*** graph $G = (V, E)$ with positive weights w_e on each edge $e \in E$, and our goal is to partition the vertices of G into k disjoint sets

$V_1, \ldots, V_k$ so that we maximize the objective function:

$$\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \sum_{u \in V_i, v \in V_j, (u,v) \in E} w_{uv}.$$

I.e. we maximize the total weight of the edges that go from some part V_i to another part V_j such that $i < j$. Give a simple randomized polynomial time approximation algorithm for this problem which achieves an approximation ratio of $\frac{k-1}{2k}$ in expectation. Justify the algorithm's approximation ratio and running time.