CSC373 Fall'20

Midterm 2

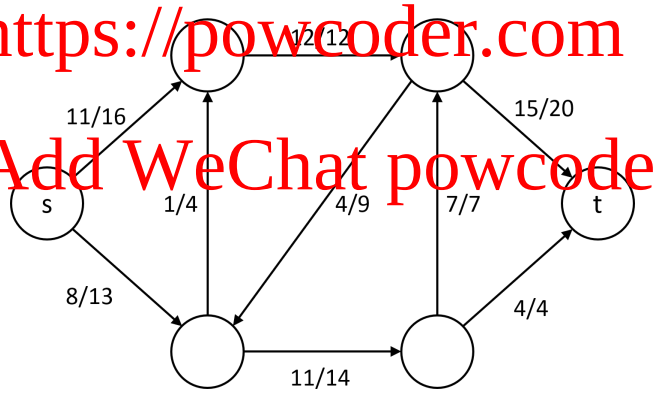
17th November 2020

Instructions

1. Upload a single PDF with your solutions to MarkUs at https://markus.teach.cs.toronto.edu/csc373-2020-09
2. There are *four* (4) questions with subquestions, each on a different page.
3. For handwritten solutions, please make sure that your handwriting is legible, and your scan is high-quality and not distorted.
4. **Remember: You receive 20% for any (sub)problem if you leave it blank (or cross off any written solution) and write "I do not know how to approach this problem.", and 10% if you leave it blank but do not write this or a similar statement.**

Q1 [20 Points] Ford-Fulkerson

The following figure shows a network G along with a valid flow f in the network. Each edge is labeled "x/y", where y is the capacity of the edge and x is the amount of flow it carries under f.



(a) [5 Points] Describe the residual graph G_f and clearly indicate the capacity of each of its edges. You can either draw it or list all the edges with their capacities. [There is no need to distinguish between forward and reverse edges.]

(b) [5 Points] If you run the Ford-Fulkerson algorithm starting from the given flow f, what augmenting paths can it choose in the very next iteration? [List *all* augmenting paths in G_f.]

(c) [5 Points] Among the augmenting paths identified in part (b), suppose we select the one with the smallest number of edges. What is its bottleneck capacity? What will be the value of the flow after augmenting along this path?

(d) [5 Points] After the augmentation step in part (c), is the resulting flow a max flow? Why?

Q2 [10 Points + 10 BONUS Points] Network Flow

(a) [10 Points] Consider the following variant of the network flow problem. We are given a network G with a capacity $c(e)$ for each edge e *as well as* a capacity $c(v)$ for each intermediate node v (i.e. for every $v \neq s, t$). The source and sink nodes do not have any capacities. We want to find a maximum flow in G, but with the *additional* constraint that the total incoming flow to each intermediate node v must be at most $c(v)$ (i.e. $f^{\text{in}}(v) = f^{\text{out}}(v) \leq c(v)$). Note that the flow must still satisfy flow conservation and edge capacity constraints as before.

Reduce this problem to the standard maximum flow problem without node capacities. You need to construct a new graph G' with *only* edge capacities such that there is a 1-1 correspondence between maximum flows in G and maximum flows in G'. Describe the construction of G' and briefly argue the 1-1 correspondence. You do not need to provide a detailed proof.

[HINT: Before the incoming flow to v can leave through its outgoing edges, can you force it to go through an edge with capacity $c(v)$?]

(b) [**BONUS**: 10 Points] Suppose we are given a network G with *integer* edge capacities and a maximum flow f in the network. We want to strictly increase the maximum flow value of the network by increasing the capacities of some of its edges by 1. Similarly to before, note that we do not care by how much the maximum flow value increases; just increasing it by 1 is sufficient.

Design a linear time algorithm to find the *minimum number of edges* whose capacities we need to increase by 1 to increase the maximum flow value by at least 1.

[HINT: Consider the residual graph G_f. Since f is a maximum flow in G, there is no s-t path in G_f. By increasing the capacities of some edges of G, we might add their corresponding forward edges to G_f, thus creating an s-t path in the residual graph and increasing the maximum flow value. That is, we want to find the minimum number of *missing forward edges* that must be added to G_f so that an s-t path is created. How do you find this?]

Q3 [20 Points] Linear Programs

(a) [10 Points] Convert to the following linear program to the standard form (maximize $c^T x$ subject to $Ax \leq b$ and $x \geq 0$). You only need to write the LP in the standard form; there is no need to show any intermediate steps. You also do not need to explicitly construct A, b, and c.

$$\begin{aligned}
&\text{minimize} \quad 5x_2 - 3x_1 \\
&\text{subject to} \\
&x_1 + 1 = 6 - x_2 \\
&2x_2 \geq x_1 - 5 \\
&x_1 \geq 0
\end{aligned}$$

(b) [10 Points] Julie, a gig worker, is considering gigs labeled $1, \ldots, n$. Each gig i has an associated revenue r_i and cost c_i. That is, if Julie accepts this gig, her net profit from it will be $r_i - c_i$, which can be positive, negative, or zero. Julie wants to accept a subset of gigs to maximize her total net profit.

However, there are some constraints on which gigs she can accept. She is given a set E of triplets such that for each $(i, j, k) \in E$, if she accepts gig k, then she must accept both gigs i and j, and conversely, if she accepts both gigs i and j, then she must accept gig k (think of this as a Boolean AND).

Write a Boolean integer linear program (with variables taking values in the set $\{0, 1\}$) for Julie's problem. Clearly define what your variables mean and briefly argue the correctness of your program.

Q4 [15 Points] Complexity

The *buyer-fulfillment problem* is as follows. Suppose you are an auctioneer in an auction with a set B of n buyers and a set C of m objects. Each buyer $b \in B$ desires a set of objects $C_b \subseteq C$: either the buyer receives exactly this set of objects and is considered "fulfilled", or she goes home with nothing. Each object must go to at most one buyer; it is okay to leave some objects unallocated.

Buyer-Fulfillment:

Input: Set of buyers B; set of objects C; the desired set $C_b \subseteq C$ of each buyer b; and an integer k.

Question: Is it possible to fulfill at least k buyers?

(a) [5 Points] Prove that the buyer-fulfillment problem is in NP.

(b) [10 Points] Prove that the buyer-fulfillment problem is NP-hard using a reduction from the 3D-Matching problem given below.

3D-Matching:

Input: Three disjoint and finite sets of elements X, Y, Z; a set $T \subseteq X \times Y \times Z$ (i.e. T consists of some triplets (i, j, k) where $i \in X$, $j \in Y$, and $k \in Z$); and an integer $k \geq 1$.

Question: Does there exist $M \subseteq T$ with $|M| \geq k$ such that each element in $X \cup Y \cup Z$ appears in at most one triplet in M?

[HINT: In the 3D-Matching problem, we want to select at least k triplets from T. In the buyer-fulfillment problem, we want to fulfill at least k buyers. This hints at what should map to what in your reduction.]