McGill

Fall 2019

Midterm Examination

Algorithm Design

COMP 360 SEC 001

4:05-5:20 PM Oct 16, 2019

EXAMINER:	Hamed Hatami	ASSOC. EXAMINER:	

STUDENT NAME:		**McGILL ID:**									

INSTRUCTIONS

EXAM:	CLOSED BOOK ☒ OPEN BOOK ☐
	SINGLE-SIDED ☒ PRINTED ON BOTH SIDES OF THE PAGE ☐
	EXTRA BOOKLETS PERMITTED: YES ☒ NO ☐ ANSWER ON EXAM ☒
	SHOULD THE EXAM BE: RETURNED ☒ KEPT BY STUDENT ☐
CRIB SHEETS:	NOT PERMITTED ☒ PERMITTED ☐ e.g. one 8 1/2X11 handwritten double-sided sheet Specifications:
DICTIONARIES:	TRANSLATION ONLY ☐ REGULAR ☒ NONE ☐
CALCULATORS:	NOT PERMITTED ☒ PERMITTED (Non-Programmable) ☐
ANY SPECIAL INSTRUCTIONS:	• Pick **three** problems and answer. • **Do not answer all the four problems** (otherwise only the first three questions will be graded). • Your grade is considered out of 20, but you can potentially acquire a higher grade by answering the last question. This will carry over towards your final grade in the course.

1	2	3	4	**Total**
/6	/6	/8	/10	**/20**

1. A farmer can choose from three feeds for his milk cows. The nutritional facts and costs of these feeds are shown in the following table. The minimum daily requirements of nutrients A, B, and C are 65, 82, 70 units, respectively. Write a linear program to determine the mixture of feeds that will supply the minimum nutritional requirement at smallest possible cost. (You do not need to solve the linear program).

Feed	A (units/lb)	B (units/lb)	C (units/lb)	Cost /lb
Feed 1	4	7	3	0.10
Feed 2	2	3	4	0.07
Feed 3	5	5	3	0.06

Solution: Variables are x_1, x_2, x_3 are the amounts of Feed 1, Feed 2, and Feed 3 (in lb) respectively.

$$
\begin{array}{lll}
\text{min} & 0.1x_1 + 0.07x_2 + 0.06x_3 & \\
\text{s.t.} & 4x_1 + 2x_2 + 5x_3 & \geq 65 \\
& 7x_1 + 3x_2 + 5x_3 & \geq 82 \\
& 3x_1 + 4x_2 + 3x_3 & \geq 70 \\
& x_1, x_2, x_3 & \geq 0
\end{array}
$$

2. We are given the locations of n propaganda broadcast stations, the radius they cover, and the type of the propaganda that they broadcast (which is either Red or Blue). More precisely, we are given n quadruples (x_i, y_i, r_i, T_i), where x_i, y_i, r_i are integers, and T_i is either equal to B or to R. This means that there is a station at coordinate (x_i, y_i) broadcasting propaganda of type T_i and it covers the circle of radius r_i centered at (x_i, y_i).

   We want to remove the smallest number of stations so that the area that is covered by Blue propaganda is disjoint from the area that is covered by Red propaganda. Show that this problem can be solved efficiently.

   Solution: This is essentially the minimum vertex cover in bipartite graph problem, and we have seen in the class that it can be solved efficiently.

   We put a node for each Red station in one part, and a node for each Blue station in the other part. Two nodes i and j are adjacent if and only if they are of different colours, and the area that they cover intersect (that is $r_i + r_j$ is at least as large as the distance between the two stations).

3. Consider the variant of the maximum flow problem where every node v also has an integer capacity $c_v \geq 0$. We are interested in finding the maximum flow as before, but now with the extra restriction that $f^{\text{in}}(v) \leq c_v$ for every node v. Solve this problem using the original maximum flow problem. (You just need to explain how to set up the flow network, and explain how its solution corresponds to the solution of this problem. You are not required to give a detailed proof).

   Solution: We construct a new flow network by replacing every node v with two nodes v^{in} and v^{out}. We add an edge of capacity c_v from v^{in} to v^{out}.

   Moreover, we want all the incoming edges of v to go to v^{in} and all the outgoing edges to leave from v^{out}. To this end, for each original edge uv, we connect u^{out} to v^{in} with an edge of capacity c_{uv}. The max flow in this constructed network is the desired solution.

   Common Errors: Many students tried to modify the FF algorithm to account for vertex capacities. One attempt was that whenever we augment a path we limit the amount of the bottleneck so that it will not exceed the capacity of any vertex. However this might mean that the bottleneck is zero (if there is a vertex that is fully saturated on the path), and thus we do not make any progress.

   Some students tried to get around this by avoiding the saturated vertices when they search for st-paths. However this does not consider the possibility of decreasing (i.e. pushing back) the flow on the vertices, and we run into similar problems that we have seen in the class if we do not decrease the flow in the FF.

   Unfortunately some students wrote statements such as, "try one augmenting path, and if it doesn't work, use the next augmenting path", or "consider all the possible augmenting paths". As it has been emphasized several times, the number of paths can be exponential and any such algorithm is going to be extremely slow in the worst case. Not to mention that use the next augmenting path is extremely vague (next in what order?).

   Another common attempt was to adjust the edge capacities. However this does not work because the vertex capacity is a cap on the total amount of flow coming into a vertex and cannot be emulated by putting caps on the edge capacities.

10

4. We are given a flow network G and our goal is to improve this network by increasing the capacities of some of the edges. More precisely, we want to find the smallest number of edges such that if we increase the capacities of all of them by 1 (simultaneously), then the value of the max-flow will increase (we do not care by how much). Design an efficient algorithm for this task based on FF.

   Solution: Run the efficient version of FF to find the max flow. Consider the residual graph G_f. Obviously there are no st-paths in G_f, and our goal is to create one by increasing the capacities of the fewest possible edges in the original network. To this end, assign a cost 0 to every edge in G_f, and for every saturated edge e in G, add the corresponding edge e to G_f but with cost 1. (Note that increasing the capacity of the edge in G will result in the addition of the edge e to G_f). Call this new graph (which is a copy of G_f with some newly added edges) H. Now our goal clearly is to find an st-path with smallest cost in H. This is just the shortest path problem and can be solved using for example the Dijeskra algorithm (It is also possible to use some modified version of BFS to solve this shortest path problem because all the costs are 0's and 1's).