University of Toronto, Department of Computer Science
CSC 373 Midterm Exam II, July 14, 2016

Lalla Mouatadid

Duration: 60 minutes

No Aids Allowed

PLEASE COMPLETE THE SECTION BELOW:

First Name: ______________________________

Last Name: ______________________________

Exam Instructions

- **Check that your exam book has 10 pages** (including this cover page and 2 blank pages at the end). The last 2 pages are for rough work only, *they will* **not** *be marked*. Please bring any discrepancy to the attention of an invigilator.
- There are 4 questions worth a total of 40 points. Answer all questions on the question booklet.
- For question 4, if you do not know how to answer the question, you can leave it blank or write "I DON'T KNOW." to receive 20 percent of the points of the question.

Course Specific Notes

- Unless stated otherwise, you can use the standard data structures and algorithms discussed in CSC263 and in the lectures without describing their implementation by simply stating their standard name (e.g. min-heap, merge- sort, DFS, Dijkstra). You do not need to provide any explanation or pseudo-code for their implementation. You can also use their running time without proof. For example, if you are using the merge-sort in your algorithm you can simply state that merge-sort's worst-case running time is $\mathcal{O}(n \log n)$. If you modify a data structure or an algorithm from class, you must describe the modification and its effects.
- In some questions you will be given a computational problem and then asked to design an efficient algorithm for it. Unless stated otherwise, for data structures and algorithms that you design you should provide a short high-level explanation of how your algorithm works in plain English, and the pseudo-code for your algorithm in a style similar to those we have seen in the lectures. If you miss any of these the answer might not be marked. Your answers will be marked based on the efficiency of your algorithms and the clarity of your explanations. State the running time of your algorithm with a brief argument supporting your claim and prove that your algorithm works correctly (e.g. finds an optimal solution).

PLEASE PRINT YOUR STUDENT NUMBER AND YOUR NAME - Yes, again!

Student Number: ______________________________

First Name: ______________________________

Last Name: ______________________________

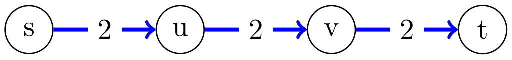
The section below is for marker's use only. Do NOT use it for answering or as scratch paper.

Questions	**Points**
1	/8
2	/8
3	/9
4	/15
Total	/40

1 . Prove or Disprove. [8]

- Let $(S, V\backslash S)$ be a minimum (s,t)-cut in a network flow graph G. let (u,v) be an edge that crosses the cut in the forward direction, i.e. $u \in S, v \in V\backslash S$. Then increasing the capacity of the edge (u,v) necessarily increases the maximum flow of G.

  False. In the graph below, $S = \{s,u\}$ forms a minimum (s,t)-cut, but increasing the capacity of the edge (u,v) does not increase the maximum flow of G.

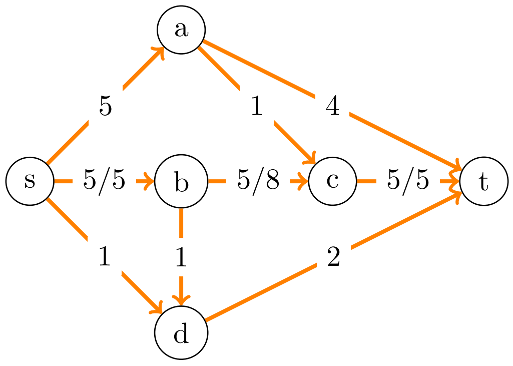
- If all directed edges in a network have distinct capacities, then there is a unique maximum flow.

  False. Consider the graph with

$$V = \{s,x,y,z,t\}$$
$$E = \{(s,x),(s,y),(x,z),(y,z),(z,t)\}$$
$$c(s,x) = 2, c(s,y) = 3, c(x,z) = 4, c(y,z) = 5, c(z,t) = 1$$

  Essentially, the edge (z,t) is a bottleneck, but you can choose whether to go through node x or node y to get the maximum flow. Thus there is no unique max flow even though all directed edges have distinct capacities.

2. Consider the flow network G below, with an initial flow on it: [8]



Find a maximum (s, t)-flow for G, using the initial flow as a starting point. Label each edge with the amount of flow sent along that edge, in your flow, and draw a minimum (s, t)-cut for the graph. Clearly state what your S, T sets for the cut are. Show your final answers only.

Let $val(f)$ denote the value of the flow. Starting with the initial flow, $val(f) = 5$. We augment along the following paths:

$s - a - t : 4$

$s - a - c - b - d - t : 1$ Notice that we redirect 1 unit of flow along the edge (b, c)

$s - d - t : 1$

$val(f) = 11$, notice that we have saturated the capacities out of s.

One possible cut is $S = \{s\}, T = V \backslash \{s\}$.

3 . Give necessary and sufficient conditions on the numbers a_1 and a_2 so that the following LP: [9]

$$\begin{aligned} \text{maximize} \quad & x_1 + x_2 \\ \text{subj. to} \quad & a_1 x_1 + a_2 x_2 \leq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

- is infeasible,

  This LP is **never** infeasible. To see this, note that $(x_1, x_2) = (0, 0)$ is feasible no matter what a_1, a_2 are.

- has a bounded optimal solution,

  The LP has a bounded optimal solution if both $a_1 > 0$ and $a_2 > 0$. In this case, the constraint implies that $x_1 \leq \frac{1}{a_1}$ and $x_2 \leq \frac{1}{a_2}$. Thus, the objective value of $x_1 + x_2$ is at most $\frac{1}{a_1} + \frac{1}{a_2}$. Since $(0, 0)$ is feasible, and gives value 0, the optimal value is finite and between 0 and $\frac{1}{a_1} + \frac{1}{a_2}$.

- is unbounded.

  The LP is unbounded if either $a_1 \leq 0$ or $a_2 \leq 0$. In the former case, $(x_1, x_2) = (K, 0)$ is feasible for any choice of $K \geq 0$. This gives an objective value of $x_1 + x_2 = K$. Setting K as large as we like shows the objective is unbounded.

4. Independent Set [15]

Recall a 0,1 Integer Linear Program is a special case of linear programming where the variables are $\{0, 1\}$.

· Formulate the Maximum Independent Set as a 0,1 integer linear program. [5]

For every vertex $v \in V$, we introduce a binary variable x_v, where $x_v = 1$ if v is in a maximum independent set of G and 0 otherwise.

$$\begin{aligned} \text{maximize} \quad & \sum_{v \in V} x_v \\ \text{subj. to} \quad & x_u + x_v \leq 1 \qquad \forall (u,v) \in E \\ & x_v \geq 0 \qquad \forall v \in V \\ & x_v \in \{0,1\} \end{aligned}$$

The first constraint guarantees that no two adjacent vertices can both be in the set.

· Ignoring the integrality constraint, this problem becomes a linear program. Write down the dual of this linear program. [5]

For every constraint in the primal, we introduce a dual variable. In this case, constraints are on the edges, so for every edge (u, v) we introduce a variable y_{uv}

$$\begin{aligned} \text{minimize} \quad & \sum_{(u,v) \in E} y_{uv} \\ \text{subj. to} \quad & \sum_{v \in V : (u,v) \in E} y_{uv} \geq 1 \qquad \forall u \in V : \exists v \in V \text{ with } (u,v) \in E \\ & y_{uv} \geq 0 \qquad \forall (u,v) \in E \end{aligned}$$

· Now constrain the solution to the dual problem to be an integer, producing another special case of Integer Programming. In terms of the graph G, this integer program is asking for a subset of edges of G, which is optimal in some sense. What is the subset and how is it optimal? [5]

This linear program asks for a set of edges of minimum size, such that every vertex with outgoing edges from the set is incident in at lease one edge in the set. This is an *edge cover*, similar to a vertex cover.