Note to Students: This file contains sample solutions to the term test together with the marking scheme and comments for each question. **Please read the solutions and the marking schemes and comments carefully.** Make sure that you understand why the solutions given here are correct, that you understand the mistakes that you made (if any), and that you understand *why* your mistakes were mistakes.

Remember that although you may not agree completely with the marking scheme given here it was followed the same way for all students. We will remark your test only if you clearly demonstrate that the marking scheme was not followed correctly.

For all remarking requests, please submit your request **in writing** directly to your instructor. For all other questions, please don't hesitate to ask your instructor during office hours or by e-mail.

Question 1. [9 marks]

Recall the Maximum Bipartite Matching problem: given an undirected bipartite graph $G = (V_1, V_2, E)$, where $E \subseteq V_1 \times V_2$, find a subset of disjoint edges with maximum size (where two edges are *disjoint* if they have no common endpoint).

Give a linear (or integer) program that corresponds to this problem. Describe clearly every component of your answer. Then, justify the correctness of your linear program: explain clearly what each variable and constraint represents and how solutions to each problem correspond to each other (and what that tells you about the relative values of those solutions).

Hint: Use one variable for each edge. If you are unable to solve this problem, you can get more than 10% of the marks by explaining clearly what your solution should consist of (including how to argue its correctness).

Sample Solutions:

Let $V_1 = \{u_1, \ldots, u_k\}$ and $V_2 = \{w_1, \ldots, w_\ell\}$.

Variables: $x_{i,j}$ for all i, j with $(u_i, w_j) \in E$ — *intention: matching* $M = \left\{(u_i, w_j) : x_{i,j} = 1\right\}$

Objective Function: maximize $\sum_{(u_i,w_j)\in E} x_{i,j}$ — *equal to* $|M|$

Constraints: $x_{i,j} \in \{0, 1\}$, for all $(u_i, w_j) \in E$

$\left.\begin{array}{l} \sum_{w_j \in V_2 : (u_i,w_j)\in E} x_{i,j} \leqslant 1, \text{ for each } u_i \in V_1 \\ \sum_{u_i \in V_1 : (u_i,w_j)\in E} x_{i,j} \leqslant 1, \text{ for each } w_j \in V_2 \end{array}\right\}$ *no two edges in M have a common endpoint*

Every matching M over G gives rise to a feasible solution by setting $x_{i,j} = 1$ iff $(u_i, w_j) \in M$: since no two edges share an endpoint, every constraint is satisfied. Moreover, the size of M is equal to the value of the objective function, so the maximum value of the objective function is at least as large as the size of the maximum matching.

Every feasible solution to the integer program yields a matching M by selecting every edge (u_i, w_j) such that $x_{i,j} = 1$: the contraints guarantee that no two edges share an endpoint and the value of the objective is equal to the size of M. So the size of the maximum matching is at least as large as the maximum value of the objective function.

Alternative Solution:

Variables: $x_{i,j}$ for all $1 \leqslant i \leqslant k = |V_1|$, $1 \leqslant j \leqslant \ell = |V_2|$

Objective Function: maximize $\sum_{i=1}^{k} \sum_{j=1}^{\ell} x_{i,j}$

Constraints: $x_{i,j} \in \{0, 1\}$ for all i, j

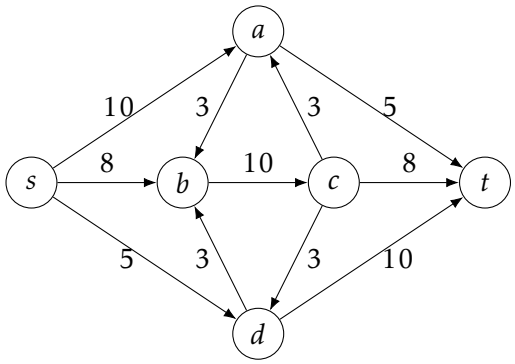
$x_{i,j} = 0$ for all i, j such that $(u_i, w_j) \notin E$

$\sum_{j=1}^{\ell} x_{i,j} \leqslant 1$ for $i = 1, \ldots, k$

$\sum_{i=1}^{k} x_{i,j} \leqslant 1$ for $j = 1, \ldots, \ell$

Question 2. [8 MARKS]

Consider the network N_1 pictured on the right.



Part (a) [3 MARKS]

Compute the *capacity* of the cut $X_1 = \big(\{s,b,c\},\{a,d,t\}\big)$. **Show your work:** list the components of the network used to obtain your answer.

SAMPLE SOLUTIONS:

$$\begin{aligned} c(X_1) &= c(s,a) + c(c,a) + c(c,t) + c(c,d) + c(s,d) \\ &= 10 + 3 + 8 + 3 + 5 = 29 \end{aligned}$$

Part (b) [1 MARK]

Let C_1 represent your answer to Part (a). What can you conclude about $|f|$, for *all* valid flows f over N_1?

SAMPLE SOLUTIONS:

$|f| \leqslant C_1$ (or $|f| \leqslant 29$)

Part (c) [4 MARKS]

Given one network N and one cut $X = (S,T)$ in N, suppose that we **reduce** the capacity of every *forward* edge across X. Does the maximum flow value in N necessarily decrease? Justify your answer.

SAMPLE SOLUTIONS:

No: consider network $N = s \xrightarrow{2} a \xrightarrow{4} t$ and cut $X = \big(\{s,a\},\{t\}\big)$. If we reduce the capacity of every forward edge across X by 1, the resulting network $N' = s \xrightarrow{2} a \xrightarrow{3} t$ still has the same maximum flow value as N.

Question 3. [8 MARKS]

For each of the following decision problems D, state whether $D \in P$, $D \in NP$ or $D \in coNP$—make the strongest claim that you can. Then, justify your answer by writing down an explicit algorithm for D and explaining why it satisfies the properties required for your answer. *Do **not** attempt to justify that other complexity classes are incorrect; focus on providing evidence that your choice is correct.*

Part (a) [4 MARKS]

SOMESHORTPATHS ("SSP" for short)

Input: Undirected graph $G = (V,E)$, vertices $s,t \in V$, non-negative integer $k \leqslant |V|$.
Output: Does G contain *some* simple path from s to t with no more than k edges on the path?

SAMPLE SOLUTIONS:

SSP $\in P$. On input $G = (V,E)$, $s,t \in V$, $k \leqslant |V|$:

> run BFS in G starting from s
> let ℓ be the number of edges on the path from s to t found by BFS
> **return** $\ell \leqslant k$

Because BFS is known to find a path with the minimum number of edges, the algorithm returns TRUE iff G contains a simple path from s to t with at most k edges. Moreover, BFS runs in polytime so the entire algorithm is polytime.

Part (b) [4 marks]

AllShortPaths ("ASP" for short)

Input: Undirected graph $G = (V, E)$, vertices $s, t \in V$, non-negative integer $k \leqslant |V|$.
Output: Does *every* simple path in G from s to t contain no more than k edges?

Sample Solutions:

ASP $\in$ *coNP*. On input $G = (V, E)$, $s, t \in V$, $k \leqslant |V|$, c:

```
# where c is a sequence of edges from G
check that c represents a simple path in G from s to t
return len(c) > k
```

Clearly, the algorithm runs in polytime. Moreover, it outputs True for some value of certificate c iff there is some simple path in G from s to t with more than k edges, iff G, s, t, k is a no-instance.

(Note: It's acceptable to invert the output of the verifier.)