NOTE TO STUDENTS: This file contains sample solutions to the term test together with the marking scheme for each question. Please read the solutions and the marking schemes 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. [20 MARKS]

Part (a) [13 MARKS]

Explain how to modify Dijkstra's algorithm so that if there is more than one minimum path from s to v, a path with the fewest number of edges is chosen.

SAMPLE SOLUTION:

Define a new array A of size $|V|$ such that $A[v]$ denotes the minimum number of edges on a path of weight $d[v]$ (recall $d[v]$ denotes the weight of a shortest path from s to v). This array A will be used to break ties when we find two competing shortest paths.


```
1:  procedure DIJKSTRA(G = (V, E, w), s)
2:      Define arrays d and A of size |V|
3:      Set d[v] = A[v] = ∞ for all v ∈ V
4:      Set d[s] = A[s] = 0
5:      Define array Π of size |V|
6:      Set Π[v] = NIL for all v ∈ V
7:      Q := V
8:      S := ∅
9:      while Q ≠ ∅ do
10:         u = Extract-Min(Q)
11:         S = S ∪ {u}
12:         for each vertex v ∈ Adj[u] do
13:             if d[v] > d[u] + w[u, v] then
14:                 d[v] = d[u] + w[u, v]
15:                 Π[v] = u
16:                 A[v] = A[u] + 1
17:                 Decrease-Key(Q, v)
18:             else if d[v] == d[u] + w[u, v] then
19:                 if A[u] + 1 < A[v] then
20:                     A[v] = A[u] + 1
21:                     Π[v] = u
22:     return d
```

MARKING SCHEME:

A. 1 mark: for writing a pseudocode
B. 2 marks: for correct explanation and definition of the array A
C. 6 marks: 3 marks for each of the two "if" blocks
D. 4 marks: for correct overall algorithm

Part (b) [7 MARKS]

Company A produces and sells two products P and Q. They have sixty workers. They believe they can make a profit of \$3,000 per worker producing P and \$5,000 per worker producing Q. From past experience, they feel that they cannot take care of more than fifty workers producing P or forty workers producing Q. Write a linear program to show how many workers they should assign to the two products to maximize profit.

SAMPLE SOLUTION:

Let x_P and x_Q denote the number of workers allotted to product P and product Q, respectively. Then the total profit is given by $3000x_P + 5000x_Q$.

Thus, the corresponding LPP is given by

$$\begin{aligned} \text{maximize} \quad & 3000x_P + 5000x_Q \\ \text{subject to} \quad & \\ & x_P \leqslant 50 \\ & x_Q \leqslant 40 \\ & x_P + x_Q \leqslant 60 \\ & x_P, x_Q \geqslant 0. \end{aligned}$$

MARKING SCHEME:

A. 1 mark: for the correct definition of variables
B. 2 marks: for the correct objective function
C. 4 marks: 1 mark for each of the constraints

Question 2. [20 MARKS]

You are given a railway network in the form of a weighted graph where vertices are rail stations, edges are rail connections between stations, and each edge has a weight/capacity which limits the number of trains that can travel on that rail connection daily. After the COVID-19 pandemic outbreak, the authorities have decided to reduce the number of trains running on the network to limit the spread of the virus. To that end, they have imposed restrictions on the stations whereby they have limited the number of trains that can travel through a given station (the number varies by station). You may assume all the trains in the network start from one source station, end at one terminal station, and for all other stations, if a train arrives at that station it must leave the station. The goal is to find the maximum number of trains that can run through the network under these restrictions.

Part (a) [7 MARKS]

Model this problem mathematically as a network flow problem.

SAMPLE SOLUTION:

Let $N = (V, E)$ denote the network, s denote the source station, t denote the terminal station, c_e denote the edge capacity for edge e (for $e \in E$), and c_v denote the node capacity for node v (for $v \in V$). The following linear program models the given problem.

Variables: one variable x_e for each edge $e \in E$

Objective function: maximize $\sum_{e=(s,u)\in E} x_e$

Constraints:

- $0 \leqslant x_e$ for each edge $e \in E$
- $x_e \leqslant c_e$ for each edge $e \in E$
- $\sum_{(u,v)\in E} x_{(u,v)} = \sum_{(v,u)\in E} x_{(v,u)}$ for each vertex $v \in V \setminus \{s, t\}$
- $\sum_{(u,v)\in E} x_{(u,v)} \leqslant c_v$ for each vertex $v \in V$
- $\sum_{(v,u)\in E} x_{(v,u)} \leqslant c_v$ for each vertex $v \in V$

MARKING SCHEME:

A. 1 mark: for the correct definition of variables
B. 1 mark: for the correct objective function
C. 5 marks: 1 mark for each of the constraints

Part (b) [7 MARKS]

Design an algorithm to solve this problem.

SAMPLE SOLUTION:

Given the network $N = (V, E)$ above with vertex and edge capacities, construct a new network $N' = (V', E')$ by replacing each vertex v with a pair of vertices v^{in}, v^{out} and adding an edge (v^{in}, v^{out}) with capacity c_v — original edges coming into v are assigned to go into v^{in}, while original edges going out of v are assigned to go out of v^{out}. Formally,

$$\begin{aligned} V' &= \{v^{in}, v^{out} \mid v \in V\} \\ E' &= \{(u^{out}, v^{in}) \text{ with capacity } c_{(u,v)} \mid (u, v) \in E\} \\ &\quad \cup \{(v^{in}, v^{out}) \text{ with capacity } c_v \mid v \in V\} \end{aligned}$$

Clearly, this construction can be carried out in polynomial time (in fact, in linear time).

```
1: procedure MaximumNumberOfTrainsInTheNetwork(N = (V, E), {c_e | e ∈ E}, {c_v | v ∈ V})
2:     construct the network N' = (V', E') as mentioned above
3:     use Edmonds-Karp algorithm to find a maximum flow f' in N'
4:     |f'| := Σ_{e=(s,u)∈E'} f'(e)
5:     return |f'|
```

MARKING SCHEME:

A. 3 marks: for the correct definition of the new modified network
B. 3 marks: for the correct algorithm
B. 1 mark: for writing a pseudocode

Part (c) [4 MARKS]

Prove the correctness of your algorithm.

SAMPLE SOLUTION:

Any flow f in N can be extended to a flow f' in N' by letting $f'(u^{out}, v^{in}) = f(u, v)$ and $f'(v^{in}, v^{out}) = \max\{f^{in}(v), f^{out}(v)\}$ (the max handles the cases of s and t as well), where

$$\begin{aligned} f^{in}(v) &= \sum_{(u,v)\in E} f(u, v) \\ f^{out}(v) &= \sum_{(v,u)\in E} f(v, u). \end{aligned}$$

So the maximum flow value in N' is at least as large as the maximum flow value in N.

Conversely, any flow f' in N' yields a valid flow f in N by letting $f(u, v) = f'(u^{out}, v^{in})$ and ignoring the flow values on the additional edges — the total flow into or out of v is limited by $c_{(v^{in},v^{out})} = c_v$, as required. So, the maximum flow value in N is at least as large as the maximum flow value in N'.

Hence, the maximum flow value in N' equals the maximum flow value in N.

MARKING SCHEME:

A. 4 marks: 2 marks for each direction of the proof

Part (d) [2 MARKS]

Compute the runtime of your algorithm.

SAMPLE SOLUTION:

Lines 2 and 4 take linear time $\mathcal{O}(m' + n')$, while line 3 takes $\mathcal{O}(n'm'^2)$ time, where n' is the number of nodes and m' is the number of edges in N'. In terms of the original network N with n nodes and m edges, we know $n' = 2n - 2$ and $m' = m + n - 2$.

Thus, ignoring constants, overall complexity $= \mathcal{O}(n(m + n)^2) = \mathcal{O}(m^2n + mn^2 + n^3) = \mathcal{O}(m^2n + n^3)$.

MARKING SCHEME:

A. 2 marks: for the correct runtime computation