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]

You have a business with several offices, and you want to lease phone lines to connect them up with each other. The phone company charges different amounts of money to connect different pairs of offices. You want a set of lines that connects all your offices and has the minimum total cost.

Part (a) [8 MARKS]

Write a polynomial-time algorithm to solve the above problem. Justify the correctness of your algorithm and compute its runtime.

SAMPLE SOLUTION:

Model the problem as a weighted undirected graph $G = (V, E, w)$, where each office represents a node in V, there is an edge between every pair of nodes (i.e., G is a complete graph), and the weight on an edge $e = (u, v)$ is the amount of money that the phone company charges to service a phone line between the offices u and v. Finding a set of phone lines that connects all the offices and has the minimum total cost then boils down to simply finding an MST T for G.

Algorithm:

```
1: procedure ComputeCheapestSetOfLines()
2:     Construct the graph G = (V, E, w) as mentioned above
3:     Compute an MST T for G using Kruskal's algorithm
4:     return T
```

Correctness: Let A be some set of lines that connects all the offices. If A contains a cycle, we can remove one of the edges in the cycle and still be connected. This will give us a better/cheaper solution. So, we can assume without loss of generality that A does not contain any cycle. Since A is connected, it must be a tree. Since an MST is also a tree with the least possible total weight, it provides an optimal solution that we are looking for.

Runtime: Let n be the number of offices. Then the graph G has n nodes and n^2 edges. Thus, constructing the network takes time $\Theta(n^2)$. Running Kruskal's algorithm on G takes time $\Theta(n^2 \lg n)$. Thus, total runtime is $\Theta(n^2 \lg n)$.

MARKING SCHEME:

- 2 marks: for correct idea and for thinking in the right direction
- 2 marks: for correct algorithm
- 2 marks: for correct justification why the algorithm works
- 2 marks: for correct runtime computation

Part (b) [12 MARKS]

The phone company informed you that one of the phone lines that you are using (from your solution in part (a)) has some technical problem and they won't be able to service that line. Write an algorithm to modify the existing solution to come up with a new solution, i.e., a new set of lines that connects all your offices and has the minimum total cost given the restriction. Justify the correctness of your algorithm and compute its runtime.

Your algorithm for part (b) must modify the existing solution, and not rerun the algorithm from part (a). For full marks, your algorithm for part (b) must be more efficient than your algorithm from part (a).

SAMPLE SOLUTION:

Let e_0 be the edge in G that cannot be serviced. Since e_0 belongs to T, its removal results in two connected components that are disconnected from each other. We create a new MST T' by finding a minimum weight edge e_1 (other than e_0) that connects these two components, and replacing e_0 by e_1 in T.

Algorithm:

1: **procedure** COMPUTEMODIFIEDCHEAPESTSETOFLINES($G, T, e_0 = (u, v)$)
2: Run BFS on the edges of $T - \{e_0\}$ starting from u; assign colour *white* to every vertex encountered
3: Run BFS on the edges of $T - \{e_0\}$ starting from v; assign colour *black* to every vertex encountered
4: Loop over every edge in $E - \{e_0\}$ to find a minimum-weight edge e_1 with one *white* endpoint and one *black* endpoint
5: **if** there is no such edge e_1 **then**
6: **return** NIL
7: **else**:
8: **return** $T - \{e_0\} + \{e_1\}$

Correctness: From the proof of correctness of Kruskal's algorithm, we know that it is always "safe" to add an edge of minimum weight between two connected components (while constructing an MST), exactly what the algorithm is doing.

Runtime: BFS takes time $\Theta(|V| + |E|)$ and finding e_1 takes time $\Theta(|E|)$. Thus, total runtime is $\Theta(|V| + |E|)$.

MARKING SCHEME:

- 4 marks: for correct idea and for thinking in the right direction
- 4 marks: for correct algorithm (the algorithm must be more efficient than the one in part a and must not in essence simply rerun the algorithm from part a on a modified graph, otherwise 0 marks)
- 2 marks: for correct justificaction why the algorithm works
- 2 marks: for correct runtime computation

Question 2. [20 MARKS]

The University of Toronto assigns Chief Presiding Officers (CPOs) to invigilate final exams. The office in charge of organizing final exams has the following information about exams and CPOs:

- Each exam has a date and a time, where time is one of "morning", "afternoon" or "evening". Each combination of dates and times (eg. "the evening of July 22") is called an *examination period*.
- Each CPO has a list of examination periods when the CPO is available. CPOs cannot be assigned to slots when they are not available.

- Each CPO has a maximum number of exam periods that the CPO can invigilate, which is usually (but not necessarily) less than or equal to the number of examination periods when the CPO is available. This number can be different for different CPOs.
- Each CPO can invigilate at most two exams on a single day.
- If an examination period has ℓ exams, there should be $\lceil 1.1\ell \rceil$ CPOs available for that period (10% extra), in case some CPOs cannot make their shifts due to last minute emergencies.

Part (a) [10 MARKS]

You are provided with a list of the CPOs along with their availabilities and work preferences, the examination days, the examination periods, and the number of exams in each slot. Design a polynomial-time algorithm using network flow that either assigns each CPO to examination slots satisfying the constraints listed above, or reports that an assignment is not possible.

SAMPLE SOLUTION:

From the input, extract the following information:

- List of CPOs: $[c_1, c_2, \ldots, c_m]$
- Availabilities of CPOs: $[A_1, A_2, \ldots, A_m]$
- Maximum number of exams for each CPO: $[e_1, e_2, \ldots, e_m]$
- List of exam periods: $[p_1, p_2, \ldots, p_n]$
- Size of each exam pool (already rounded up by a factor of 10%): $[\ell_1, \ell_2, \ldots, \ell_n]$
- List of exam days: $[d_1, d_2, \ldots, d_h]$.

Construct a network $N = (V, E)$ where V contains the following vertices:

- source s and sink t
- one vertex for each CPO: $c_1, c_2, \ldots, c_m$
- one vertex for each exam period: $p_1, p_2, \ldots, p_n$
- a vertex $a_{i,k}$ for each CPO c_i and exam day d_k

and E contains the following edges:

- (s, c_i) for each c_i, with capacity $c(s, c_i) = e_i$
- $(c_i, a_{i,k})$ for each c_i and $a_{i,k}$, with capacity $c(c_i, a_{i,k}) = 2$
- $(a_{i,k}, p_j)$ for each $a_{i,k}$ and p_j such that CPO c_i is available during exam period p_j (i.e., $p_j \in A_i$) and exam period p_j is on day d_k, with capacity $c(a_{i,k}, p_j) = 1$
- (p_j, t) for each p_j, with capacity $c(p_j, t) = \ell_j$.

1: **procedure** SCHEDULECPO($c_1, \ldots, c_m, A_1, \ldots, A_m, e_1, \ldots, e_m, p_1, \ldots, p_n, \ell_1, \ldots, \ell_n, d_1, \ldots, d_h$)
2: construct the network $N = (V, E)$ as mentioned above
3: use Edmonds-Karp algorithm to find a maximum flow f in N
4: **if** $|f| < \ell_1 + \cdots + \ell_n$ **then**
5: output "No assignment possible"
6: **else**
7: $S_i = \{p_j \mid f(a_{ik}, p_j) = 1 \text{ for some } k\}$ for $i = 1, \ldots, m$
8: **return** $S_1, \ldots, S_m$

Marking Scheme:

- 3 marks: for clearly mentioning the inputs and nodes
- 4 marks: for correctly defining the edges with capacities
- 3 marks: for correct algorithm

Part (b) [10 marks]

Briefly justify the correctness and runtime of the algorithm designed in part (a) using properties of network flow.

Sample Solution:

Correctness: Any valid assignment of CPOs to exam periods yields a valid flow in N by setting

$$\begin{aligned} f(s, c_i) &= \text{number of exam periods assigned to CPO } c_i \text{ (guaranteed } \leqslant e_i) \\ f(c_i, a_{i,k}) &= \text{number of exam periods assigned to CPO } c_i \text{ on day } d_k \text{ (guaranteed } \leqslant 2) \\ f(a_{i,k}, p_j) &= 1 \text{ if CPO } c_i \text{ is assigned to exam period } p_j \text{ (0 otherwise)} \\ f(p_j, t) &= \text{size of the pool for exam period } p_j \text{ (guaranteed } \leqslant \ell_j). \end{aligned}$$

Conversely, any valid integer flow in N yields a valid assignment of CPOs to exam periods by assigning CPO c_i to exam period p_j iff $f(a_{i,k}, p_j) = 1$ (where p_j is on day d_k). This assignment satisfies each of the problem constraints:

no CPO c_i is assigned to more than e_i exam periods because $c(s, c_i) = e_i$

no CPO c_i is assigned to more than two exam periods on the same day because $c(c_i, a_{i,k}) = 2$

no exam period p_j is assigned more than ℓ_j CPOs because $c(p_j, t) = \ell_j$.

Consequently, a schedule is possible if and only if there is a flow in N with $|f| = \ell_1 + \cdots + \ell_n$, in which case a feasible schedule is given by $\{S_1, \ldots, S_m\}$ as defined above. Also, observe that $|f| \leqslant \ell_1 + \cdots + \ell_n$ for any flow f in N (since $\ell_1 + \cdots + \ell_n$ is the total capacity into t). Hence, $|f| = \ell_1 + \cdots + \ell_n$ implies f is a max flow in N. Thus, if f is a max flow in N with $|f| < \ell_1 + \cdots + \ell_n$, then there is no feasible schedule.

Runtime: Note that $\frac{n}{3} \leqslant h \leqslant n$ (there are at most three exam periods on each exam day). Creating the network takes time $\Theta(m)$ (for vertices c_i and edges (s, c_i)) + $\Theta(n)$ (for vertices p_j and edges (p_j, t)) + $\Theta(mh) = \Theta(mn)$ (for vertices $a_{i,k}$ and all related edges). This yields a network with $\Theta(mn)$ vertices and $\Theta(mn)$ edges. Running the Edmonds-Karp algorithm takes time $\mathcal{O}((mn)(mn)^2) = \mathcal{O}((mn)^3)$. Generating the assignment of CPOs to exam periods takes time $\Theta(mn)$ (each edge $(a_{i,k}, p_j)$ is examined once). The total runtime is therefore $\mathcal{O}((mn)^3)$.

Marking Scheme:

- 6 marks: for correct explanation (3 marks for each direction)
- 2 marks: for correctly proving when a schedule is infeasible
- 2 marks: for correct runtime computation