- This exam has 2 questions for a total of 100 marks.
- Go through all the questions once before you start writing your answers.
- Please follow that instructions that I sent you by Moodle.
- Warning: CMI's academic policy regarding cheating applies to this exam.

Unstated assumptions and lack of clarity in solutions can and will be used against you during evaluation. Please ask the invigilator if you have questions about the questions.

You may assume that each array access and numerical operation takes constant time.

1. Let $G = (V, E)$ be a simple undirected graph.

   (a) Define a *spanning tree* of a simple undirected graph. [5]

   (b) Prove that G has a spanning tree if and only if G is connected. [5]

   (c) Write the pseudocode for an algorithm IsConnected which: [10]

   1. Takes an adjacency list of G as input;
   2. Runs in time $\mathcal{O}(|V(G)| + |E(G)|)$ in the worst case, and;
   3. Figures out whether G is connected, or not.

   You will get the credit for this part only if your algorithm (i) is correct, and (ii) runs within the required time bound.

   (d) Prove that your algorithm of part (c) is correct. You will get the credit for this part only if you provide an *explicit* argument for the correctness; merely referring to the correctness of algorithms that we saw in class, will not get you the credit. [20]

   (e) Prove that your algorithm of part (c) runs in time $\mathcal{O}(|V(G)|+|E(G)|)$, in the worst case. You will get the credit for this part only if you provide an *explicit* argument for the running time; merely referring to the running time of algorithms that we saw in class, will not get you the credit. [10]

2. Let $G = (V = \{v_1, v_2, \ldots, v_n\}, E)$ be an edge-weighted *directed* graph with weight function $w : E \to \mathbb{R}$. Recall that a *path* in G has no repeated vertices. For vertices x, y in G and path P from x to y—also called an $x \rightsquigarrow y$ path—the *internal vertices* of P are those vertices in P which are not one of $\{x, y\}$.

   The weight $w(p)$ of a path P in G is the sum of the weights of all the edges in P. For any two vertices x, y in G the *shortest-path weight* from x to y, denoted $\delta(x, y)$, is the minimum weight of an $x \rightsquigarrow y$ path, if it exists. If there is no $x \rightsquigarrow y$ path then $\delta(x, y) = \infty$. A *shortest path* from x to y is any $x \rightsquigarrow y$ path P with $w(P) = \delta(x, y)$.

   For vertices $x, y \in V$ and integer $0 \leq r \leq n$ let $S[x, y, r]$ denote the weight of a shortest $x \rightsquigarrow y$ path whose *internal vertices*—if any—are all from the set $\{v_1, v_2, \ldots, v_r\}$.

(a) For two vertices x_1, x_k in G let $P = \langle x_1, x_2, \ldots, x_k \rangle$ be a shortest $x_1 \rightsquigarrow x_k$ path in G. For any $1 \leq i \leq j \leq k$ let $P_{ij} = \langle x_i, x_{i+1}, \ldots, x_j \rangle$ be the sub-path of P from x_i to x_j. Prove that P_{ij} is a shortest $x_i \rightsquigarrow x_j$ path in G. [10]

(b) Write a *recurrence relation* for $S[x, y, r]$ indexed on r. That is: express $S[x, y, r]$ *recursively* in terms of one or more values $S[x', y', r']$ for some $r' \neq r$. Your recurrence must have at least one base case, and at least one inductive case. **Explain why your recurrence is correct.** [10]

This part does *not* ask for pseudocode, so your answer should *not* be pseudocode. You will get the credit for this part only if *both* your recurrence *and* your explanation are correct.

(c) Write the pseudocode for an algorithm that converts the recurrence from part (b) to a *dynamic programming* solution that runs in time $\mathcal{O}(|V|^3)$ and computes, for *every* pair of vertices $x, y \in V$, the value $\delta(x, y)$. Your algorithm should take $(G = (V, E), w)$ as input and compute a $|V| \times |V|$ array D such that $D[i][j] = \delta(v_i, v_j)$. [10]

You will get the credit for this part only if your algorithm (i) is a DP based on the recurrence of part (b), (ii) is correct, and (iii) runs in $\mathcal{O}(|V|^3)$ time.

(d) Prove that your algorithm of part (c) correctly computes the required table D. [10]

(e) Prove that your algorithm of part (c) runs in $\mathcal{O}(|V|^3)$ time. [10]