Practice problems (don't turn in):

1. [DPV] Problems 5.1, 5.2. Practice fundamentals of MST designs.
2. [DPV] Problem 5.9 (multiple statements about MST. We will provide the answer to a few, you are welcome to try them all.)

Instructions.

For the graded problems, you are allowed to use the algorithms from class as black-boxes without further explanation. These include

- DFS (outputs connected components, a path between two vertices, topological sort on a DAG. You also have access to the pre and post arrays!), BFS and the Explore subroutine.
- Dijkstra's algorithm to find the shortest distance from a source vertex to all other vertices and a path can be recovered backtracking over the pre labels.
- Bellman-Ford and Floyd-Warshall to compute the shortest path when weights are allowed to be negative.
- SCCs which outputs the strongly connected components, and the metagraph of connected components.
- Kruskal's and Prim's algorithms to find MST.
- Ford-Fulkerson and Edmonds-Karp to find max flow on networks.

When using a black-box, make sure you clearly describe which input you are passing to it and how you use the output from or take advantage of the data structures created by the algorithm. To receive full credit, your solution must:

- Include the description of your algorithm in words (no pseudocode!).
- Explain the correctness of your design.
- State and analyse the running time of your design (you can cite and use the running time of black-boxes without further explanations).

Unless otherwise indicated, black-box graph algorithms should be used without modification.

Example: I take the input graph G, I first find the vertex with largest degree, call it v^*. I take the complement of the graph G, call it $\overline{G}$. Run Dijkstra's algorithm on $\overline{G}$ with $s = v^*$ and then I get the array $dist[v]$ of the shortest path lengths from s to every other vertex in the graph $\overline{G}$. I square each of these distances and return this new array.

We don't want you to go into the details of these algorithms and tinker with it, just use it as a black-box as showed with Dijkstra's algorithm above. Make sure to explain your algorithm in words, no pseudocode.

Problem 1 (MCQs on MST and Ford-Fulkerson.)

For each part, choose the answer that is *always* true. You do not need to explain your choice. Enter your responses directly on Gradescope.

Part (a): (2.5 points) Let $\{G = (V, E), s, t \in V, \{c_e\}_{e \in E}\}$ be a network. Consider the flow $f = 0$ for all $e \in E$. The residual network $\{G^f = (V, E^f), s, t \in V, \{\hat{c}_e\}_{e \in E^f}\}$

A: has more edges than the original graph $G = (V, E)$.
B: has a unique strongly connected component.
C: satisfies $G = G^f$.
D: has at least one edge with capacity $\hat{c}_e \neq c_e$.

Part (b): (2.5 points) Let $\{G = (V, E), s, t \in V, \{c_e\}_{e \in E}\}$ be a network. You are told that c_e is a natural number, for all $e \in E$.

True or False: the flow has integer values after every round of Ford-Fulkerson.

A: True.
B: False.

Part (c): (2.5 points) Let $\{G = (V, E), s, t \in V, \{c_e\}_{e \in E}\}$ be a network and f^* a maximum flow. An edge $e \in E$ is called a *bottleneck edge* if increasing its capacity increases the size of the maximum flow f^*.

True or False: Every saturated edge (this is, $f^*(e) = c_e$) in the network is a bottleneck edge.

A: True.
B: False.

Part (d): (2.5 points) True or False: If the edge of minimum weight on a graph is unique, then it belongs to *any* MST of G.

A: True.
B: False.

Part (e): (2.5 points) True or False: If a connected, undirected, weighted graph $G = (V, E)$ has a unique cycle, there is a linear time algorithm to find a MST of G.

A: True.
B: False.

Part (f): (2.5 points) Let $G = (V, E)$ be a connected, undirected, weighted graph with weights $w(e) = 2022$ for all $e \in E$. True or False: there is a linear time algorithm to find a MST of G.

A: True.
B: False.

Part (g): (2.5 points) Let $G = (V, E)$ be a connected, undirected, weighted graph with all weights distinct. For every vertex $v \in V$, denote by e_v the edge connected to v of lightest weight. Note that this edge depends only on v, and we can have $e_v = e_{v'}$ for vertices $v \neq v'$.

True or False: The set $\{e_v,\ v \in V\}$ is part of every MST of G.

A: True.
B: False.

Part (h): (2.5 points) Let $G = (V, E)$ be a connected, undirected, weighted graph. Consider the following *Divide and Conquer Algorithm* to build a MST of G.

- If $V = 2$, return E. (In our class, unless otherwise stated, we assume all graphs are simple).
- Else, partition $V = V_1 \cup V_2$ into two disjoint sets of equal size, recursively find MSTs T_1 and T_2 on V_1 and V_2, respectively.
- Find an edge e of minimum weight connecting one vertex of V_1 to one vertex of V_2. Return $T_1 \cup T_2 \cup \{e\}$.

True or False: This algorithm outputs a MST of G.

A: True.
B: False.

Problem 2 (Edge on MST)

You are given a weighted graph $G = (V, E)$ with positive weights, c_i for all $i \in E$. Give a linear time ($O(|E| + |V|)$) algorithm to decide if an input edge $e = (u, v) \in E$ with weight c_e is part of some MST of G or not. You should describe your algorithm in words (a list is okay); no pseudocode. Explain why it is correct and justify its running time.