Problem 1: DPV 7.18 (a) and (b)

Solve the following problems by reducing to the original max-flow problem. In other words, explain how to solve the new flow variant problem using an algorithm for solving max-flow as a black-box. Explain how to take an input for the new problem and define an input for the original max-flow problem. Then given a max-flow f^* to this input you just defined, explain how to get the solution to the new problem.

(a) There are many sources and many sinks, and we wish to maximize the total flow from all sources to all sinks.

Solution.

Suppose we have as input to this problem a directed graph $G = (V, E)$, with a set of sources $S \subseteq V$ and a set of sinks $T \subseteq V$. We will construct a new graph $G' = (V', E')$ on which we can run the original max-flow problem, then use the flow f^* from the original problem to create a solution for this variant problem.

To construct G', we will add a new super-source node s' and a super-sink node t'. We will connect the super-source to each of the sources in the input graph with an edge of infinite capacity, and connect each sink in the input graph to the super-sink with an edge of infinite capacity. Then, $V' = V \cup \{s', t'\}$ and $E' = E \cup \{(s', s) : s \in S\} \cup \{(t, t') : t \in T\}$.

Now we can run the original max-flow algorithm on G'. Suppose this finds the flow f^*. To find the maximum flow from all of the input sources to all of the input sinks on G, simply ignore all flow from the super-source to the sources, and from the sinks to the super-sink (keep only the flows on edges in the input graph). Since the added edges could not have been bottlenecks, their addition could not have affected the flow.

*An infinite capacity edge can be implemented as an edge with sufficiently large capacity to never be saturated. For example, an edge with capacity $|E|w^*$, where w^* is the maximum weight of an edge in the graph, will never be saturated.

(b) Each vertex also has a capacity on the maximum flow that can enter it.

Solution.

Suppose we have as input to this problem a directed graph $G = (V, E)$, with a source $s \in V$, a sink $t \in V$, and a capacity c_v for each non-source vertex $v \in V \backslash \{s\}$. We will construct a new graph $G' = (V', E')$ on which we can run the original max-flow problem, then use the flow f^* from the original problem to create a solution for this variant problem.

To construct G', we will replace each vertex in the original graph with a pair of vertices connected by an edge with weight equal to the capacity of that vertex (with the exception of the source vertex). Then, $V' = \{s\} \cup \{v_{in} : v \in V \backslash \{s\}\} \cup \{v_{out} : v \in V \backslash \{s\}\}$ and $E' = E \cup \{(v_{in}, v_{out}) : v \in V \backslash \{s\}\}, w_{v_{in}, v_{out}} = c_v$.

Now we can run the original max-flow algorithm on G'. Suppose this finds the flow f^*. To find the maximum flow on the input graph with vertex capacities, combine each v_{in}, v_{out} pair and look at the flow on the original vertices. Since the flow from f^* going through a vertex cannot be more than the capacity of the edge “inside” that vertex, this flow will not exceed each vertex's capacity. The source does not have this constraint, since no flow can enter the source.

Problem 2: DPV 5.22 (a)
Prove the following property carefully:
Pick any cycle in the graph (denote it by C), and let e^* be the heaviest edge in that cycle C. Thus, $w(e^*) \geq w(e')$ for all $e' \in C$. Then there is a minimum spanning tree T' that does not contain e^*.

Hint: Take a MST T which contains e^*. Construct a new tree T' which does not contain e^* and $w(T') \leq w(T)$.

Solution.

Consider a cycle C; let e^* be the heaviest edge in C. Consider a MST T containing the edge e^*. If there is no such MST T then we are done, there is nothing to prove. The difficult case is when *every* MST contains e^*. Consider one such MST containing e^*, call this MST T. We will prove there exists a tree T' which does not contain e^* and where $w(T') \leq w(T)$, and thus T' is a MST which does not contain e^* which proves the property.

To create T', remove e^* from T. Since T is a spanning tree and $e^* \in T$, then $T - e^*$ consists of two disjoint trees, namely T_A and T_B. Say the endpoints of e^* are vertices $a \in T_A$ and $b \in T_B$. Note that $C - e^*$ is a path between vertex a in T_A and vertex b in T_B. Therefore, there must be at least one edge $e' \in C - e^*$ crossing between T_A and T_B. It follows that $T' = T \cup e' - e^*$ is a spanning tree that does not contain e^*. Moreover, since e^* is the heaviest edge in C we have $w(e') \leq w(e^*)$ and

$$w(T') = w(T) + w(e') - w(e^*) \leq w(T).$$

Therefore, T' is a MST which does not contain e^*.

Problem 3: DPV 5.9 (d)
If the lightest edge in a graph is unique, then it must be part of every MST.
Is the above statement True or False?
If True, then prove that is correct (explain why it always holds). Or if it is False then give a counterexample (show a graph where it does not hold).

Solution.

True.

Let $e^* = (u, v)$ be the unique lightest edge in the graph $G = (V, E)$ and thus $w(e^*) < w(e')$ for all $e' \in E - e^*$.

We will prove the claim by contradiction. Suppose there is a MST T which does not include edge e^*. We will construct a tree T' where $w(T') < w(T)$ and thus T is not a MST, which gives the contradiction and hence proves the claim.

So consider MST T where $e^* \notin T$. Add e^* to T. This creates a cycle C. Remove the heaviest edge e' in this cycle. Note e' is not e^* because e^* is strictly lighter than every other edge on the cycle C. Let $T' = T \cup e^* - e'$ denote this new set. Note T' is a tree and

$$w(T') = w(T) + w(e^*) - w(e') < w(T) \text{ since } w(e^*) < w(e').$$

Therefore we have constructed a new tree T' with smaller weight than T, contradicting the assumption that T was an MST.