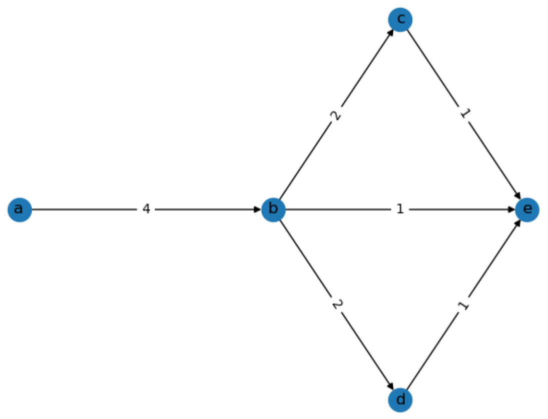
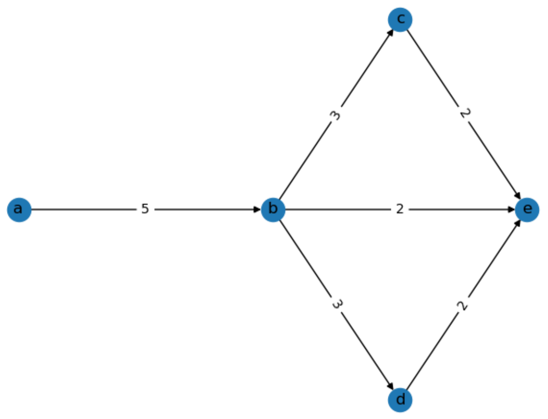
Homework 5

Liwen Ouyang

April 5 2020

Problem 1

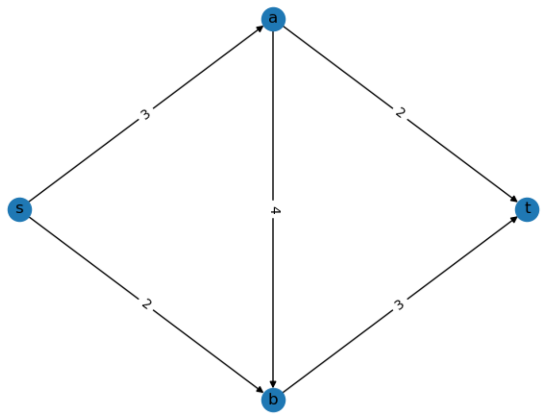


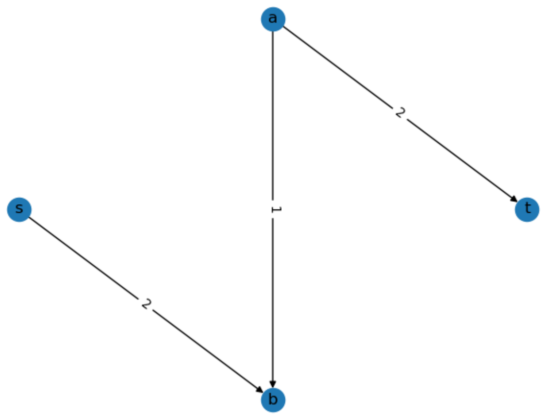


Consider the original network on the left, and the new network we get by increment the capacity of every edge by 1. In the original network, it is clear the the minimum cut is $\{a, b, c, d\}, \{e\}$, with a cut capacity of $1+1+1 = 3$. However, in the new network on the left, the cut capacity of $\{a, b, c, d\}, \{e\}$ cut is $2+2+2 = 6$, which is more than another cut $\{a\}, \{b, c, d, e\}$, which has a capacity of 5. Therefore we disproved the claim by a counterexample.

Problem 2

Consider the original network on the left and the network after one iteration of the described algorithm on the right:





If we choose the augmenting path in the first iteration of the algorithm to be the path $s \to a \to b \to t$, which has a bottleneck capacity of 3, and then subtract 3 from edges $(s, a), (a, b), (b, t)$, after which we delete the edges that are left with 0 capacity: $(s, a), (b, t)$. This lead us to the network on the right, which has no path from s to t. Therefore the algorithm will terminate and report a maximum flow of 3, which is not true. Clearly we can get a larger flow by sending 2 unit of flow on path $s \to a \to t$, 1 unit of flow on path $s \to a \to b \to t$, and 2 unit of flow on path $s \to b \to t$, which gives us a flow of 5.

Problem 3

To begin with let's prove a lemma:

Lemma 1. *Given a min-cut of a network, whose every edge has capacity c, which partitions the vertices of the network into two disjoint sets; $s \in S, t \in T$ with a capacity $C(S, T)$, after deleting a saturated edges from S to T, the maximum flow becomes $C(S, T) - c$, decreasing by c.*

Proof. if S/T is a minimum cut, as proved in class, all edges going from S to T will be saturated, so we can find such an edge. Now we need to show that after deleting such an edge, S/T still remains a minimum cut, with a capacity of $C(S, T) - c$. Let there be an arbitrary cut with capacity C' before we delete the edge. By definition of min-cut, $C(S, T) \leq C'$. Now as we delete an edge, there can be two cases: if the edge also contributes to the capacity of the arbitrary cut, we have $C(S, T) - c \leq C' - c$, which makes S/T a min-cut. If the edge doesn't contribute to the arbitrary cut, we get $C(S, T) - c < C(S, T) \leq C'$, which proves that indeed after we delete an saturated edge from S to T, the cut is still a min-cut with capacity $C(S, T) - c$. This means, by max-flow-min-cut theorem, that by deleting the edge, the maximum flow decreases by c, which is the maximum decrease we can get by deleting one edge because all edges have the same capacity c. □

Now we need to find the min-cut. Note that FF finds the maximum flow, which we can use to compute the min cut. Also note that FF in this case will have a running time of $O(m^2)$. Since every edge has the same capacity c, the augmenting path will always have a capacity of c. That means every time we increase the flow by c. Besides, we know that the flow out of s is bounded by cm. This gives us an upper-bound for the number of iterations: $O(m)$. For each iteration since we are doing $O(m)$ work in FF, we can conclude that FF in this case has a running time of $O(m^2)$

Now we can give an algorithm. We will use FF to find a maximum flow. With that flow, we can compute a minimum cut by taking the residual graph of the maximum flow and run a BFS on s to find S in $O(m)$ time as discussed in book. Now that we have a min-cut S/T, we can just repeatedly delete saturated edges from S to T k times, or less if the number of such edges is less than k, in which case we will be left with a 0 max flow. The running time of the algorithm is dominated by FF, which is $O(m^2)$. Its correctness follows from the proof of the lemma.

Problem 4

(a) To begin with it is useful for us to prove the following proposition: For a cyclic flow f, there is an acyclic flow $\tilde{f}$ with same value. Without loss of generality, let's assume there's 1 cycle in f. If we subtract the minimum value of flow on the edges that belong to that cycle from every edge in the cycle, it is clear that the resulting flow $\tilde{f}$ has no more cycle, because the edge that carries the minimum value of flow will have a value of 0. Now we need to show that $\tilde{f}$ preserves the flow property and also it has the same value as f. Since every edge in the cycle has a constant reduction in the value of flow, for every node in the cylce, the value of the flow going into the node and out of the flow both decreases by a constant. That means the flow conservation is satisfied, i.e. $\tilde{f}$ is a valid flow. It is easy to see that $v(f) = v(\tilde{f})$, as we don't change the value of the flow on any of the edges out of the source s. Therefore we proved the proposition for f with 1 cycle, and it is clear that we can generalize this to f with any number of cycles.

Now we can give a simple algorithm for this problem. First we fun FF on the given network, and we will get a maximum flow f. Now we check wether there's cycle in f by running a BFS or DFS from

s and only discover a node from an edge that carries positive flow. If there's no cycle, we simple return f. If there are cycles, we will mark them. Specifically, this can be done using any algorithm for finding strongly connected component (e.g. Tarjan's algorithm), which has a running time of $O(m+n)$. Then we will find the edges that carry the minimum value of flow in these cycles. We will subtract the corresponding value of flow from edges in every cycle as described in the proof of the proposition to get a flow $\tilde{f}$. Then we will return $\tilde{f}$.

The correctness of the algorithm follows from the proof of the proposition. The running time of the algorithm will be dominated by the initial computation of the maximum flow f using FF, so it will have the same running time as FF.

(b) Using similar arguments we made in (a), we can show that for a given acyclic flow f on network N, if there's a cycle in N such that no edge in the cycle is saturated, we can construct a cyclic flow f^* from f by adding 1 unit of flow to every edge in the cycle in N, where $v(f^*) = v(f)$. First we want to show that f^* is still a valid flow: As we increment the flow value on every edge in the cycle by 1, since no edge is saturated, the capacity constraint on every edge will not be violated. Besides that, for every node on the cycle, as we increase the value of flow into the node and the value of flow out of the node both by 1, the conservation of flow is not violated either. Therefore, f^* is still a valid flow, and it is cyclic because now every edge on the cycle must carry at least 1 unit of flow. It is easy to see $v(f^*) = v(f)$ as we won't change the amount of flow out of the source node s.

Now based on what we proved above, we know that for an acyclic maximum flow of a network, if there's a cycle in the network where no edge is saturated, there must be a cyclic maximum flow for the network. Exploiting this we can devise an algorithm.

First, we use FF to get a maximum flow f of the given network. Now we check whether f is acyclic or not. This can be done in $O(m+n)$ using DFS/BFS as described in (a). If f is cyclic, the algorithm will terminate and return false. If f is acyclic, we will mark all the edges in the network that belong to a cycle by finding the strongly connected components in $O(m+n)$. Now for each cycle we check wether there exists at least an edge that is saturated, this can be done in $O(m)$. If there exists a cycle where no edge is saturated, the algorithm will return false. Otherwise the algorithm returns true.

The algorithm is correct as if there's a cycle in the network where no edge is saturated after we compute a maximum flow, based on what we proved, we can just add 1 to every edge in that cycle to get a cyclic maximum flow, in which case the algorithm will return false. Otherwise, theres no way a cyclic maximum flow can exist when every cycle in the network has at least one edge that is saturated, in which case our algorithm will return true. The running time of the algorithm, as shown, is dominated by FF, therefore it should have the same running time as FF.

Problem 5

(a) Consider an arbitrary directed path P from s to t. Seeking for contradiction let's assume P doesn't contain any edge $e \in \delta S$. This means all edges in P can only connect two nodes that are both in S, or both in T. If all edges in P connect two nodes in S, then P cannot possibly reach $t \in T$ because T and S are disjoint, contradiction. Similarly, if all edges in P connect two nodes in T, then P can never start from $s \in S$, which is also a contradiction. If some of edges in P connect two nodes in S, and the rest of the edges in P connect two nodes in T, then there must be two consecutive edges e_1, e_2 such that e_1 connects two nodes in S and e_2 connects two nodes in T. Then the node they share is both in S and T, which is also a contradiction. Therefore, we proved that any path that connects s to t must contain an edge $e \in \delta S$, which makes δS, by definition, separate S and T.

(b) Consider the set of nodes S that are reachable from s with a path that does not contain any edge from X. By definition of X that separates s and t, $t \notin S$. Therefore S defines a $s-t$ cut. Now consider an arbitrary edge $(u,v) \in \delta S$. We know $u \in S \land v \notin S$. Therefore $(u,v) \in X$, because otherwise $v \in S$ by our construct of S, which is a contradiction. Hence we showed that an arbitrary edge in δS is also in X, which is equivalently $\delta S \subseteq X$, which proves the existence of such $s-t$ cut.

(c) According to the result we proved in (b), we know $\exists S : \delta S \subseteq X$. Now consider the $s-t$ cut such that $\delta S \subseteq X$. By the result we proved in (a), δS also separates s and t. Since X is minimal, the only subset of X that also separates s and t is X itself. Therefore we showed that $X = \delta S$

Problem 6

To begin with it's useful to prove a lemma:

Lemma 2. *Given a network N and its maximum flow f, let $\tilde{f}$ denote the maximum flow of a new network we get by decreasing or increasing the capacity of an arbitrary edge e in N by 1, then $|v(\tilde{f}) - v(f)| \leq 1$*

Proof. Denoting the minimum $s-t$ cut defined by f as S/T and its capacity as $C(S,T) = v(f)$, it's easy to see that if e is not saturated, increasing or decreasing its capacity by 1 will not affect the maximum flow of the network, because if e is not saturated, it is not an edge from S to T, and therefore changing its capacity will not affect the capacity of the min-cut, which means it will not affect max-flow, i.e. $v(\tilde{f}) - v(f) = 0$. Now if e is saturated, there can be two cases:

1 If e in an edge from S to T, decreasing its capacity by 1 will still left S/T a minimum cut with a capacity $C(S,T) - 1$. This is easy to see: let there be an arbitrary cut with capacity C, then we know $C(S,T) \leq C$ because S/T is a min-cut. If e contributes to that cut, after decreasing its capacity by 1, we have $C(S,T) - 1 \leq C - 1$, which tells us S/T is still a min-cut. If e doesn't contribute to the capacity of the arbitrary cut, we have $C(S,T) - 1 < C(S,T) \leq C$, which shows S/T is indeed a min-cut after we decrease e's capacity by 1. This tells us $v(\tilde{f}) = C(S,T) - 1$, and hence $|v(\tilde{f}) - v(f)| = 1$. If we increase e's capacity by 1, then similarly we can deduce that if S/T still remains a minimum cut, $v(\tilde{f}) = C(S,T) + 1$, and therefore $|v(\tilde{f}) - v(f)| = 1$. Now if S/T is no more a min-cut, that means there exists some cut after we change the capacity of e with capacity C such that $C < C(S,T) + 1$. Then $v(\tilde{f}) - v(f) = C - C(S,T) < 1$.

2 If e is not an edge from S to T. Increasing its capacity will not affect the capacity of the min-cut therefore we have $|v(\tilde{f}) - v(f)| = 0$. If we decrease e's capacity by 1, there can be two possible situations: The capacity of the min-cut is still $C(S,T)$, in which case we have $|v(\tilde{f}) - v(f)| = 0$. If the capacity of the min-cut is no more $C(S,T)$, then there is some cut with capacity C before we change e's capacity such that $C - 1 < C(S,T)$. Note $C \geq C(S,T)$, which tells us $C(S,T) - 1 \leq C - 1 \leq C(S,T)$. Therefore $-1 \leq v(\tilde{f}) - v(f) = C - 1 - C(S,T) \leq 0$. This proves $|v(\tilde{f}) - v(f)| \leq 1$.

□

Now that we have the lemma, it's easy to give an algorithm and justify its running time and correctness.

(a) We can construct the residual graph G for the original maximum flow f , adding 1 positive capacity to the corresponding edge of e in G in $O(m + n)$. Then we perform a BFS on G starting from s and update f by augmenting the path we get from BFS into f. Note that by the lemma we proved, the maximum flow can either increase by 1, in this case we will perform BFS and augmentation once, or 0, in which case we don't need augmentation. That means the algorithm has a running time of $O(m+n)$. Its correctness follows from the correctness of FF and the proof above.

(b) Similar to (a), we will construct the residual graph G for the original maximum flow. If the edge with decreasing capacity is not saturated, as discussed in the proof of the lemma, it is not going to affect the maximum flow, and therefore there is no need to update. If it's saturated, we can change the residual graph. Let the edge be (u, v). In the residual graph, we can make 1 as the capacity of the edge (v, u). Then we perform BFS on the residual graph to find a path from t to s that contains this edge, and then we will augment this path into the maximum flow we have by subtracting the bottleneck capacity of the path, i.e. 1, from every edge in the original max flow. We will at most need to do the augmentation once because of the lemma we proved. Therefore the running time is $O(m + n)$. The correctness follows from the proof of the lemma.