Problem 1

To find the best minimum (s,t)-cut, we need to find a minimum capacity (s,t)-cut, but out of all (s,t)-cuts of minimum capacity, we prefer the ones with fewer edges going across. Can we somehow combine these two objectives (capacity and number of edges) into a single objective in such a way that we prioritize minimizing capacity, but break ties using the number of edges? Here is a way to do so. Let $C(S, V-S)$ denote the capacity of the cut $(S, V-S)$, and let $N(S, V-S)$ denote the number of edges in this cut. Consider minimizing the objective $W(S, V-S) = M \cdot C(S, V-S) + N(S, V-S)$ where M is some large constant. If M is large enough, then the second term $N(S, V-S)$ in the sum will become unimportant, and cuts with small C will also have small W. On the other hand, if there are multiple cuts with the smallest capacity C, the extra term N will break ties in favor of those with fewer edges crossing the cut. In order for this approach to work, M needs to be larger than any value N could take on. So, we set M to n^2.

Next we observe that W is in fact the min cut objective over a different flow network. In particular, if we change the capacity of every edge e to $Mc_e + 1$, where c_e is the given capacity, then the capacity of a cut $(S, V-S)$ is precisely $M \cdot C(S, V-S) + N(S, V-S)$. Finding the min (s,t)-cut with these new capacities then returns the best minimum (s,t)-cut over the original capacities.

We now describe the details.

Algorithm:

1. Set $M = n^2$.
2. Construct a graph $G^{'}$, where $G^{'}$ contains all the vertices in G, and for all edges (u,v) in G, add an edge of capacity $c_{u,v} \cdot M + 1$ to $G^{'}$.
3. Find and return the minimum (s,t)-cut in $G^{'}$ using an efficient network flow algorithm.

Correctness: We will show the correctness of the algorithm by proving the following two claims.

Claim 1. *The minimum (s,t)-cut of $G^{'}$ is also the minimum (s,t)-cut of G.*

Proof. Suppose that we have a minimum (s,t)-cut of $G^{'}$, $(S, V-S)$. In original graph G, the total capacity of this cut is $C_G = \sum_{u \in S, v \in V-S, e(u,v) \in E}(c_{u,v})$. In graph $G^{'}$, the total capacity of the cut is $C_{G'} = \sum_{u \in S, v \in V-S, e(u,v) \in E}(c_{u,v} \cdot M + 1) = M \cdot \sum_{u \in S, v \in V-S, e(u,v) \in E}(c_{u,v}) + n_c = M \cdot C_G + n_c$, where n_c is the number of the edges in the cut.

Suppose for the sake of contradiction that the cut $(S, V-S)$ is not the minimum cut in G. Then there must exist some other cut $(S^*, V-S^*)$ that has capacity $C_G^* = \sum_{u \in S^*, v \in V-S^*, e(u,v) \in E}(c_{u,v})$, where $C_G^* < C_G$. Then this cut will have the total capacity of $C_{G'}^* = M \cdot C_G^* + n_c^*$ in $G^{'}$. Then we observe that:

$$\begin{aligned}
C_{G'}^* &= M \cdot C_G^* + n_c^* \\
&\leq M \cdot (C_G - 1) + n_c^* \\
&= M \cdot C_G - M + n_c^* \\
&< M \cdot C_G + n_c \\
&= C_{G'}.
\end{aligned}$$

Here the first inequality follows by recalling that $C_G^* < C_G$, and so, $C_G^* \leq C_G - 1$. The second follows by noting that $M = n^2$, and so, $-M + n_c^* < 0 \leq n_C$. This contradicts the minimality of the cut $(S, V-S)$ in $G^{'}$ and proves the claim. □

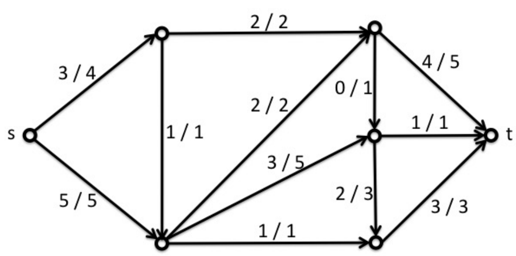
Claim 2. *The minimum (s,t)-cut of $G^{'}$ is the best minimum (s,t)-cut of G.*

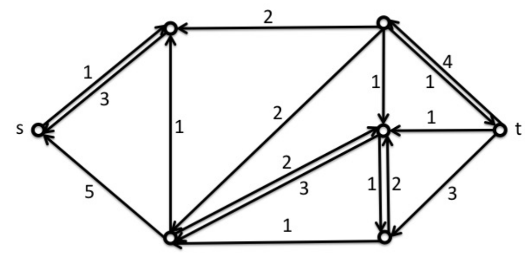
Proof. Suppose that $(S, V - S)$ is a minimum (s,t)-cut of $G^{'}$, it is also a minimum (s,t)-cut of G according to Claim 1. If the cut is not the best minimum (s,t)-cut of G, there must exist some other minimum cut $(S^*, V - S^*)$ of G that has the same capacity and less edges. The capacity of the two cuts in $G^{'}$ are $C_{G'} = M * C_G + n_c$ and $C^*_{G'} = M * C^*_G + n^*_c$, as shown in the proof of Claim 1. Note that both cuts are minimum cut of G, which means $C_G = C^*_G$, and $n^*_c < n_c$ since cut $(S^*, V - S^*)$ has less edges. Thus, the $C_{G'} > C^*_{G'}$, which contradicts the fact that $(S, V - S)$ is the minimum (s,t)-cut of $G^{'}$. □

Running time: Steps 1 and 2 take $O(m+n)$ time. Step 3 takes time equal to the running time of an efficient maximum flow algorithm, which as we discussed in class is polynomial in m and n.

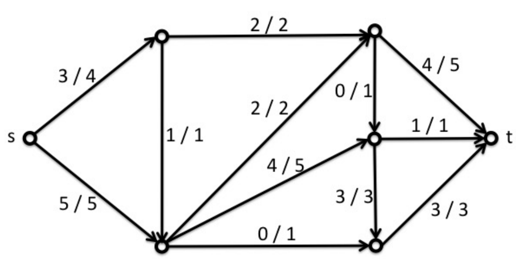
Problem 2

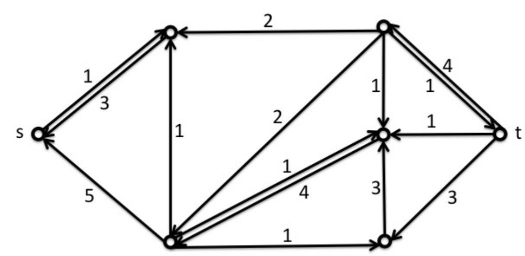
1. The following are a maximum flow and its residual graph.



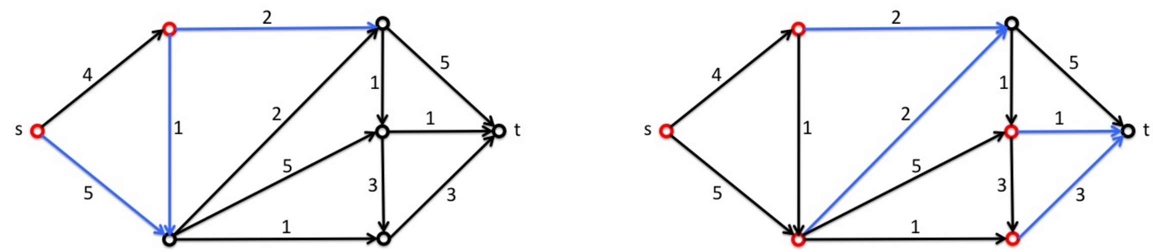


Here is another maximum flow and its residual graph.

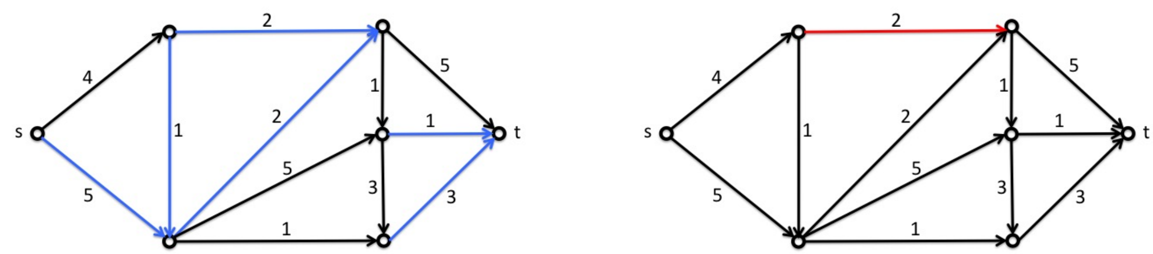




Below are two minimum cuts. The red vertices are in S, and the black ones are in T. The blue edges indicate the edges in the cut going from the s side to the t side.



2. The lower-binding edges are colored blue, the upper-binding edge is colored red.



3. We assume all the edge capacities are integral and nonzero. And we observe that if an edge e' is not used to its capacity, i.e. $f^*(e') < c(e')$, then it cannot be upper-binding. What happens when $f^*(e) = c(e)$ for an edge $e = (u, v)$? It means there is an edge going from v to u in the residual graph with residual capacity $c(e)$, and there is no edge going from u to v (in other words the residual capacity is 0). What happens in the residual graph if the edge capacity of e is increased by 1? There will be an edge going from u to v with residual capacity 1.

   Since f^* is a maximum flow in the original graph, there is no path from s to t in G_{f^*} according to the *Ford-Fulkerson Algorithm*. However, after the edge capacity increases by 1, the edge from u to v is available, which could potentially create a new augmenting path from s to t. If there is such a path going through (u, v), then we know there is 1 more unit of flow going from s to t, i.e. the flow f^* is not maximum anymore. This is to say, e is upper-binding. If there is not such a path, then we know that with the increased capacity of e, f^* is still the maximum flow. Because if there is a flow with higher value, we must be able to find an augmenting path from s to t.

   Therefore, in order to determine whether (u, v) is upper-binding, we need to determine whether there is an $s-t$ path in $G_{f^*} \cup \{(u, v)\}$, in other words whether there is an $s-u$ path as well as a $v-t$ path in G_{f^*}.

   Alternate characterization: A different way of thinking about lower binding and upper binding edges is that the former is the set of all edges that belong to **some** minimum s-t cut. The latter is the set of all edges that belong to **every** minimum s-t cut.

   Algorithm: With the help of the above analysis, we give the algorithm as follows:

(a) Construct G_{f^*}. Use BFS/DFS to find all the vertices reachable from s in G_{f^*}. Denote the set by S.

(b) Let G' denote the graph G_{f^*} with all edge directions reversed.[1] Use BFS/DFS to find all the vertices reachable from t in G'. This is the set of all the vertices that have a path going to t in G_{f^*}. Denote the set by T.

(c) Find all the pairs in $S \times T$ which are edges of G and the capacity of those edges are fully used. This step can be implemented by scanning through the list of all edges (u, v) in G and checking whether $u \in S$ and $v \in T$. These are all the upper-binding edges in G.

Correctness: We already know from the above analysis that $e = (u, v)$ is upper-binding if and only if there is an augmenting path from s to t in G_{f^*} after we add the edge (u, v). Our algorithm find exactly these edges. Note that without any modification, there is no path from s to t in G_{f^*}. Adding (u, v) creates a path if and only if u is reachable from s and t is reachable from v. Step 1 finds all the vertices S which are reachable from s; step 2 finds all the vertices T which have paths to t; step 3 finds all the edges (u, v) satisfying $u \in S$ and $v \in T$. These are exactly the edges whose increased capacity would facilitate a new augmenting path from s to t.

Running time: Step 1 and step 2 takes $O(m + n)$, and step 3 takes $O(m)$. The total running time is linear.

[1]That is, if G_{f^*} contains a directed edge from u to v, then G' contains a directed edge from v to u.