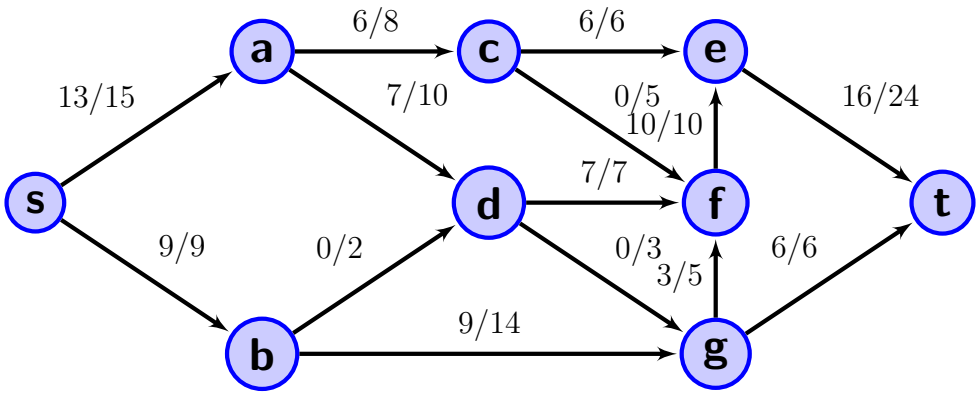
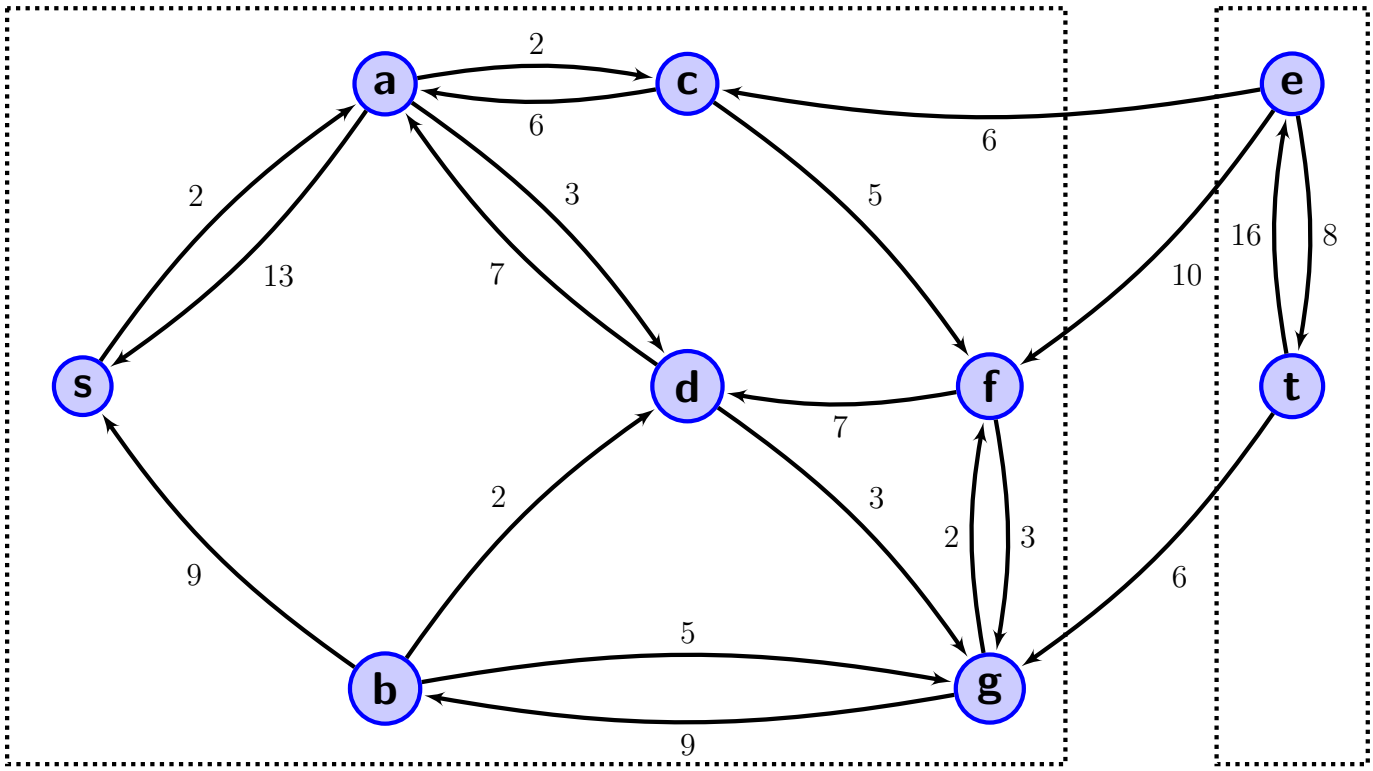
1. Compute maximum flow f and minimum cut (S, T).



Max flow f with value $|f| = 22$. Min cut (S, T) where $S = \{s, a, b, c, d, f, g\}$ and $T = \{e, t\}$ with capacity $c(S, T) = 6 + 10 + 6 = 22$.

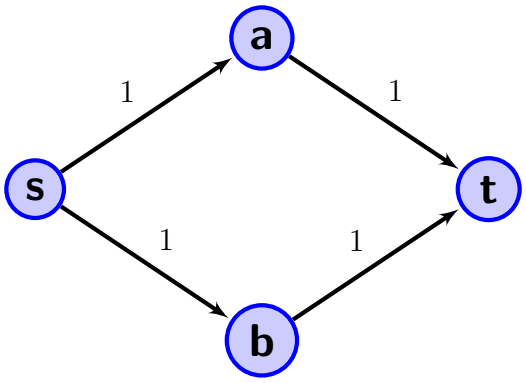
2. Draw the residual graph G_f - don't forget to state the capacities. Indicate the minimum cut (S, T) in the residual graph by circling S and T.



3. An edge is called *constricting* if increasing its capacity leads to an increase in the value of maximum flow. List all constricting edges in the above network.

Answer: $(c, e), (f, e), (g, t)$.

4. Find a small (at most 4 nodes) example of a network graph that has no constricting edges.



5. Describe in plain English an efficient algorithm to find all constricting edges. Argue correctness by using results from lectures/textbook. State the running time of your algorithm.

   Algorithm: compute max flow f. Run BFS from s in the residual graph G_f to find the set of nodes U_s that are reachable from s. Compute the transpose of G_f that is G_f^T by reversing all edges. Run BFS from t in the transpose of the residual graph G_f^T to compute the set U_t such that t is reachable from each node in U_t in G_f. Declare all edges $(u, v) \in E$ such that $u \in U_s$ and $v \in U_t$ to be constricting.

   Argument of correctness: first note that since f is maximum $U_s \cap U_t = \emptyset$. By Max-Flow Min-Cut theorem an edge (u, v) is constricting if and only if it has residual capacity 0 and increasing its capacity by any nonnegative amount creates a new augmenting path (independent of which maximum flow f we use to define residual capacity). This happens if and only if prior to the increase of the capacity u was reachable from s and t was reachable from v in the residual graph. The above algorithm computes exactly the set of such edges.

   Running time: first step is to run a max-flow algorithm. This could be, for instance, Edmonds-Karp, which takes $O(|V||E|^2)$ time. After that, constructing the residual graph, its transpose, running BFS - all take linear time, i.e., $O(|V| + |E|)$, assuming adjacency lists representation.