Note: Your TA may not get to all the problems. This is totally fine, the discussion worksheets are not designed to be finished in an hour. The discussion worksheet is also a resource you can use to practice, reinforce, and build upon concepts discussed in lecture, readings, and the homework.

1 Basics

Flow. The *capacity* indicates how much flow can be allowed on an edge. Given a directed graph with edge capacity $c(u, v)$ and s, t, a flow is a mapping $f : E \to \mathbb{R}^+$ that satisfies

- Capacity constraint: $f(u, v) \leq c(u, v)$, the flow on an edge cannot exceed its capacity.
- Conservation of flows: $f^{\text{in}}(v) = f^{\text{out}}(v)$, flow in equals flow out for any $v \notin \{s, t\}$

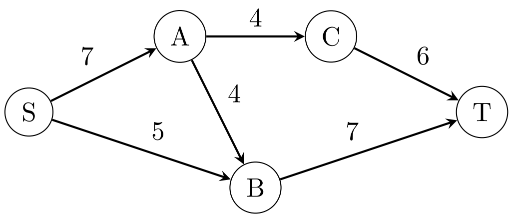
Here, we define $f^{\text{in}}(v) = \sum_{u:(u,v)\in E} f(u, v)$ and $f^{\text{out}}(v) = \sum_{u:(v,u)\in E} f(u, v)$. We also define $f(v, u) = -f(u, v)$, and this is called *skew-symmetry*.

Residual Graph. Given a flow network (G, s, t, c) and a flow f, the *residual capacity* (w.r.t. flow f) is denoted by $c_f(u, v) = c_{uv} - f_{uv}$. And the *residual network* $G_f = (V, E_f)$ where $E_f = \{(u, v) : c_f(u, v) > 0\}$.

Ford-Fulkerson. Keep pushing along s-t paths in the residual graph. Runs in time $O(mF)$.

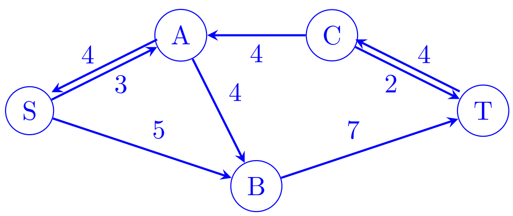
2 Residual in graphs

Consider the following graph with edge capacities as shown:



(a) Consider pushing 4 units of flow through $S \to A \to C \to T$. Draw the residual graph after this push.

Solution:

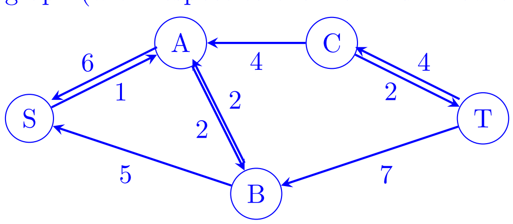


(b) Compute a maximum flow of the above graph. Find a minimum cut. Draw the residual graph of the maximum flow.

Solution: A maximum flow of value 11 results from pushing:

- 4 units of flow through $S \to A \to C \to T$;
- 5 units of flow through $S \to B \to T$; and
- 2 units of flow through $S \to A \to B \to T$.

(There are other maximum flows of the same value, can you find them?) The resulting residual graph (with respect to the maximum flow above) is:



A minimum cut of value 11 is between $\{S, A, B\}$ and $\{C, T\}$ (with cross edges $A \to C$ and $B \to T$).

3 Reductions Among Flows

Show how to reduce the following variants of Max-Flow to the regular Max-Flow problem, i.e. do the following steps for each variant: Given a directed graph G and the additional variant constraints, show how to construct a directed graph G' such that

(1) If F is a flow in G satisfying the additional constraints, there is a flow F' in G' of the same size,

(2) If F' is a flow in G', then there is a flow F in G satisfying the additional constraints with the same size.

Prove that properties (1) and (2) hold for your graph G'.

(a) **Max-Flow with Vertex Capacities:** In addition to edge capacities, every vertex $v \in G$ has a capacity c_v, and the flow must satisfy $\forall v : \sum_{u:(u,v)\in E} f_{uv} \le c_v$.

(b) **Max-Flow with Multiple Sources:** There are multiple source nodes $s_1, \ldots, s_k$, and the goal is to maximize the total flow coming out of all of these sources.

Solution:

(a) Split every vertex v into two vertices, v_{in} and v_{out}. For each edge (u, v) with capacity c_{uv} in the original graph, create an edge (u_{out}, v_{in}) with capacity c_{uv}. Finally, if v has capacity c_v, then create an edge (v_{in}, v_{out}) with capacity c_v. If F' is a flow in this graph, then setting $F(u, v) = F'(u_{out}, v_{in})$ gives a flow in the original graph. Moreover, since the only outgoing edge from v_{in} is (v_{in}, v_{out}), and incoming flow must be equal to outgoing flow, there can be at most c_v flow passing through v. Likewise, if F is a flow in the original graph, setting $F'(u_{out}, v_{in}) = F(u, v)$, and $F'(v_{in}, v_{out}) = \sum_u F(u, v)$ gives a flow in G'. One can easily see that these flows have the same size.

(b) Create one "supersource" S with edges (S, s_i) for each s_i, and set the capacity of these edges to be infinite. Then if F is a flow in G, set $F'(S, s_i) = \sum_u F(s_i, u)$. Conversely, if F' is a flow in G', just set $F(u, v) = F'(u, v)$ for $u \neq S$, and just forget about the edges from S. One can easily see that these flows have the same size.

4 Secret Santa

Imagine you are throwing a party and you want to play Secret Santa. Thus you would like to assign to every person at the party another partier to whom they must anonymously give a gift. However, there are some restrictions on who can give gifts to who. For instance, nobody should be assigned to give a gift to themselves or to their spouse. Since you are the host, you know all of these restrictions. Give an efficient algorithm that determines if you and your guests can play Secret Santa.
Solution: Let n be the number of guests. For guest i, make two vertices u_i and v_i. Let $U = \{u_i : i = 1, \ldots, n\}$ and $V = \{v_i : i = 1, \ldots, n\}$. Construct a graph $G = (U \cup V, E)$, where there is an edge between u_i and v_j if guest i can give a gift to guest j. You can play Secret Santa iff G has a perfect matching. Run max-flow on G with edge capacities $c_e = 1$ $\forall e \in E$ to get a flow f. size(f) is the size of the largest matching, so G has a perfect matching iff size(f) $= n$.