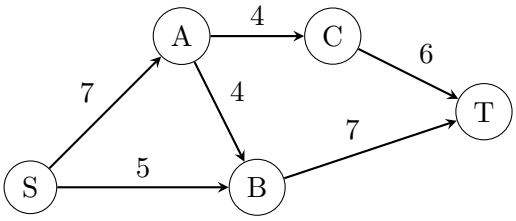
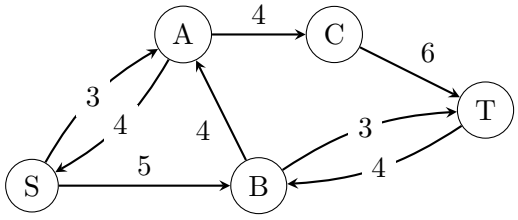
CS170 Discussion Section 9: *3/22*

Network Flows

Consider the following graph with edge capacities as shown:

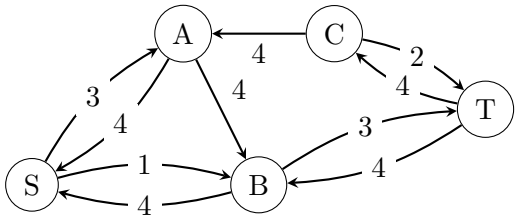


1. Draw the residual graph after pushing 4 units of flow through $S \to A \to B \to T$.
   Solution:



2. Starting from that residual graph, push four units of flow through $S \to B \to A \to C \to T$, and draw the new residual graph.

   Solution:



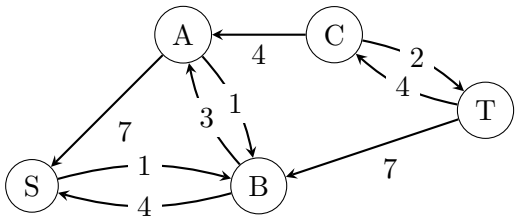
3. Compute a maximum flow of the above graph. Find a minimum cut. Draw the residual graph of the maximum flow.

   Solution:

   A maximum flow of value 11 results from pushing:

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

   (There are other maximum flows of the same value) The resulting residual graph for this particular max flow is:



The minimum cut can be found by looking at the nodes reachable from S in the residual graph above. The cut is between $\{S, A, B\}$ and $\{C, T\}$. Its cross edges (from the original graph) are $A \to C$ (weight 4) and $B \to T$ (weight 7), and the size of the cut is $4+7 = 11$.

Verifying a max-flow

Suppose someone presents you with a solution to a max-flow problem on some network. Give a *linear* time algorithm to determine whether the solution does indeed give a maximum flow.

Solution:

The max-flow algorithm has found the maximum flow when there is no $s - t$ path in the residual graph. Therefore, we just search for an $s - t$ path in the residual graph of the given flow to see if the given flow is maximal.

---

procedure CHECKFLOW(G, f)
Check that $\forall v \in V, v \neq s, t$, $\sum_{(u,v)\in E} f_{uv} = \sum_{(v,w)\in E} f_{vw}$
Compute G^f, the residual flow network of f.
Run BFS(G^f, s)
If BFS finds an s-t path, return false, otherwise return true.

Checking that f is a valid flow takes $O(|V| + |E|)$ time. Constructing G^f takes $O(|V| + |E|)$ time. Running BFS on G^f takes $O(|V| + |E|)$ time since G^f has $|V|$ vertices and $\leq 2\,|E|$ edges. Therefore, the algorithm is $O(|V| + |E|)$, which is linear.

Repairing a Flow

In a particular network $G = (V, E)$ whose edges have integer capacities c_e, we have already found the maximum flow f from node s to node t. However, we now find out that one of the capacity values we used was wrong: for edge (u, v) we used c_{uv} whereas it should have been $c_{uv} - 1$. This is unfortunate because the flow f uses that particular edge at full capacity: $f_{uv} = c_{uv}$. We could redo the flow computation from scratch, but there's a faster way. Show how a new optimal flow can be computed in $O(|V| + |E|)$ time.

Solution:

The maximum flow in the new network will be either $|f|$ or $|f| - 1$, because changing the capacity of 1 edge can change the capacity of the min-cut by at most 1. Now, look at the residual graph of G, taking capacity of edge (u, v) as c_{uv}. Since $f_{uv} = c_{uv} \geq 1$, there is a flow of at least 1 unit going from v to t. Therefore, the residual graph must have a path from t to v. Similarly, there must be a path in the residual graph from u to s, since at least 1 unit of flow from s enters u.

Find these paths by doing DFS from u and t in the residual graph, and send back 1 unit of flow through this path. This gives us a new flow f' where $f'_{uv} = c_{uv} - 1$. Notice that f' is a valid flow even if we replace the capacity of edge (u, v) by $c_{uv} - 1$. Now let's update the capacity to $c_{uv} - 1$. The flow through the new graph is f' with size $|f| - 1$. The edge (u, v) has capacity $c_{uv} - 1$ and a flow of $c_{uv} - 1$ through it. We are now at an intermediate stage of the Ford-Fulkerson algorithm. If f' is not a max-flow, then there must be an $s - t$ path in the residual graph. This can be checked by a DFS (or BFS) in the residual graph. Since the edges have integer capacities, if there is a path then the size of the flow is increased by exactly 1. Therefore, after checking for an s-t path in the residual graph (and updating the flow if necessary), we must have a max-flow. Since the algorithm simply calls DFS three times, the running time is $O(|V| + |E|)$.

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) $= n$.