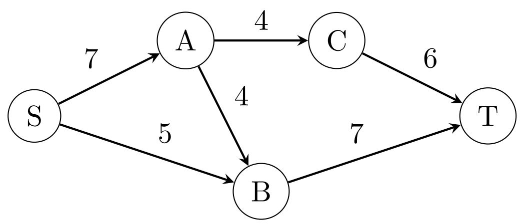
CS 170 DIS 08

Released on 2018-03-14

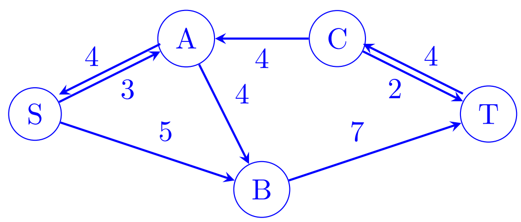
1 Residual in graphs

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:

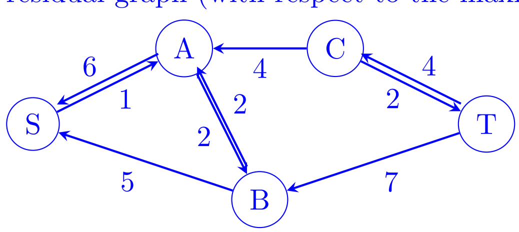


(b) Compute a maximum flow of the above graph. Find a minimum cut. Draw the residual graph of the maximum flow.

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 $B \to T$).

2 A cohort of spies

A cohort of k spies resident in a certain country needs escape routes in case of emergency. They will be travelling using the railway system which we can think of as a directed graph

$G = (V, E)$ with V being the cities. Each spy i has a starting point $s_i \in V$ and needs to reach the consulate of a friendly nation; these consulates are in a known set of cities $T \subseteq V$. In order to move undeteced, the spies agree that at most c of them should ever pass through any one city. Our goal is to find a set of paths for each of the spies (or detect that the requirements cannot be met).

Hint: Model this problem as a flow network.

Solution: We can think of each spy i as a unit of flow that we want to move from s_i to any vertex $\in T$. To do so, we can model the graph as a flow network by setting the capacity of each edge to ∞, adding a new vertex t and adding an edge (t', t) for each $t' \in T$. We can add a source s and edges of capacity 1 from s to s_i. By doing so, we restrict the maximum flow to be k.

Lastly, we need to ensure that no more than c spies are in a city. Add vertex capacities of c to each vertex – we can, as we saw in the previous problem set, adapt the graph to express this solely with edge capacities.

If the max flow is indeed k, then as every capacity is an integer, the Ford-Fulkerson algorithm for computing max flow will output an integral flow (one with all flows being integers). Therefore, we can incrementally starting at each spy i follow a path in the flow from s_i to any vertex in T. We iterate through all of spies to reach a solution.

3 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:

Note that the maximum flow in the new network will be either f or $f - 1$ (because changing one edge can change the capacity of the min-cut by at most 1). Now consider the residual graph of G, taking capacity of edge (u, v) as c_{uv}. Since there is a flow of at least 1 unit going from v to t, the residual graph must have a path from t to v (each edge along which there is a flow creates a backwards edge in the residual graph). Similarly, there must be a path in the residual graph from u to s, since at least 1 unit of flow reaches u from s.

Find the paths by doing a DFS from u and t, and send back 1 unit of flow through this path. This changes the flow through edge (u, v) to $c_{uv} - 1$. Notice that this is a valid flow even if we replace the capacity of edge (u, v) by $c_{uv} - 1$. The flow through the new graph is now $f - 1$, with the edge (u, v) having capacity $c_{uv} - 1$ and a flow of $c_{uv} - 1$ through it. This is just an intermediate stage of the Ford-Fulkerson algorithm. If it is possible to increase the flow, then there must be an $s - t$ path in the residual graph. This can be checked by a DFS (or BFS). Since the algorithm just involves calling DFS thrice, the running time is $O(|V| + |E|)$.