MAX-FLOW/MIN-CUT

ECE 1762 Algorithms and Data Structures
Fall Semester, 2013

1. [**CLRS Problem 26.2-11, page 731, Solution**]

For any two vertices u and v in G, we can define a flow network G_{uv} consisting of the directed version of G with all edge capacities 1, $s = u$, and $t = v$. Let f_{uv} denote a maximum flow in G_{uv}.

Claim: For any $u \in V$, the edge connectivity $k = min_{v \in V-\{u\}} |f_{uv}|$.

The claim follows from the max–flow min–cut theorem and the fact that we chose capacities so that the capacity of a cut is the number of edges crossing it. Here is the proof of why $k = min_{v \in V-\{u\}} |f_{uv}|$, for any $u \in V$:

- Proof that $k \geq min_{v \in V-\{u\}} |f_{uv}|$:
  Let $m = min_{v \in V-\{u\}} |f_{uv}|$,. Suppose we remove only $m - 1$ edges from G. For any vertex v, by the max–flow min–cut theorem, u and v are still connected. The max flow from u to v is at least m, hence any cut separating u from v has capacity at least m, which means at least m edges cross any such cut. Thus at least 1 edge is left crossing the cut when we remove $m - 1$ edges.
  Therefore, every node is connected to u, which implies that the graph is still connected. So at least m edges must be removed to disconnect the graph, *i.e.* $k \geq min_{v \in V-\{u\}} |f_{uv}|$.
- Proof that $k \leq min_{v \in V-\{u\}} |f_{uv}|$:
  Consider a v with the minimum $|f_{uv}|$. By the max–flow min–cut theorem, there is a cut of capacity $|f_{uv}|$ separating u and v. Since all edge capacities are 1, exactly $|f_{uv}|$ edges cross this cut. If these edges are removed, there is no path from u to v, and so our graph becomes disconnected. Hence $k \leq min_{v \in V-\{u\}} |f_{uv}|$.

We can find k as follows:

EDGE_CONNECTIVITY(G)
  Select any vertex $v \in V$
  for each vertex $v \in V - \{u\}$ **do** (* $|V| - 1$ iterations *)
    set up the flow network G_{uv} as described above
      find the maximum flow f_{uv} on G_{uv}
  return the minimum of the $|V| - 1$ max–flow values: $min_{v \in V-\{u\}} |f_{uv}|$

2. [**CLRS Problem 26-1, page 760, Solution**]

   (a) Assume given a directed graph $G = (V, E)$ with the vertex and edge capacities constraints. We, now, construct an equivalent directed graph $G' = (V', E')$, such that $V' = \{v_{in}, v_{out} :$ for all $v \in V\}$, $E' = \{(u_{out}, v_{in})$: for all $(u, v) \in E\} \bigcup \{(v_{in}, v_{out}) : v \in V\}$. And $capacity((v_{in}, v_{out})) = capacity(v)$, $capacity((u_{out}, v_{in})) = capacity((u, v))$.

We can see that $cost(v_{in}, v_{out})$ presents the vertex capacity of vertex v in V, because when we enter v (from v_{in}), the only way out is to follow edge (v_{in}, v_{out}). Therefore, we can take care the vertex capacity as well as edge capacity.

For a given undirected graph G, the edges have to duplicate to two copies, one from each direction. Therefore, (u_{out}, v_{in}) and (v_{out}, u_{in}) are both in E' if and only if (u, v) is in E.

(b) Essentially, this is a multiple-source, multiple-sink maximum flow problem. First, we take the starting points ($s'_{in}s$) as sources and take the boundary points ($t'_{out}s$) as sinks. Then we set all of the edge capacity to be 1. After we solve the maximum flow problem for G', if the flow is m, then the final residual network gives us the paths; if the flow is less than m, there is no solution. Now, $V \in \Theta(n^2)$ and $E \in \Theta(n^2)$ and $|f^*| \in O(n)$. Depending on which algorithm you use, you can fill up the time complexcity.