http://www.cs.unh.edu/~ruml/cs758
Ford-Fulkerson
Given directed graph, source and sink, find flow of maximum value.

logistics

network design

tasking

flow constraints: edge capacity, conservation at vertices

\[
0 \leq f(u,v) \leq c(u,v)
\]

\[
\forall v \in V - \{s, t\}, \sum_{u \in V} f(v,u) = \sum_{u \in V} f(u,v)
\]

details: removing ‘anti-parallel’ edges, multiple sources or sinks
Iteratively augment flow until no augmenting path exists.

Find augmentation via ‘residual network’ $G_f$ with costs

$$c_f(u, v) = \begin{cases} 
  c(u, v) - f(u, v) & \text{if } (u, v) \in E \\
  f(v, u) & \text{if } (v, u) \in E \\
  0 & \text{otherwise}
\end{cases}$$

residual network has reverse flow edges: not a legal ‘flow network’

to augment $(u, v)$, add $f(u, v)$ and subtract $f(v, u)$
1. for each edge, \((u, v).f \leftarrow 0\)
2. while there exists an \(s \leadsto t\) path \(p\) in the residual network
3. \(c_f(p) \leftarrow \min\) capacity of edges along \(p\)
4. for each edge \((u, v)\) in \(p\)
5. if \((u, v) \in E\)
6. \((u, v).f \leftarrow (u, v).f + c_f(p)\)
7. else
8. \((v, u).f \leftarrow (v, u).f - c_f(p)\)
What is its running time?

Is resulting flow maximum? (ie, ‘no augmenting path’ suffices?)
If capacities are integer, converges in at most $|f^*|$ iterations. Each iteration is $O(V + E) = O(E)$. So $O(|f^*|E)$ overall. Is this polynomial time?
Augmentation

If capacities are integer, converges in at most $|f^*|$ iterations. Each iteration is $O(V + E) = O(E)$. So $O(|f^*|E)$ overall. Is this polynomial time?

Edmonds-Karp: find augmenting path via breadth-first search. Book has proof that this is $O(VE)$ iterations. Each iteration $O(E)$ so $O(VE^2)$ overall. (Fancy alg in book is $O(V^3)$.)

(correctness of FF requires talking about cuts!)
asst 11
Cuts and Flows
consider cuts that separate $s$ and $t$

$$\text{flow across cut } f(S, T) = \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{u \in S} \sum_{v \in T} f(v, u)$$

Theorem: for a given flow, flow across any cut $f(S, T) = \text{value of flow } |f|$ (a constant!)

Proof (sketch): flow comes out of $s$, goes into $t$, and is conserved everywhere else. As we ‘push out’ equality from $s$ towards cut, each vertex we cross conserves flow when we consider all its edges.
The value of any flow $f \leq$ capacity of every cut

Proof:

$$|f| = f(S, T) \text{ by previous theorem}$$

$$= \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{u \in S} \sum_{v \in T} f(v, u)$$

$$\leq \sum_{u \in S} \sum_{v \in T} f(u, v)$$

$$\leq \sum_{u \in S} \sum_{v \in T} c(u, v) \text{ by capacity constraint}$$

$$\leq c(S, T)$$
value of a flow $= \text{flow across any cut}$

any flow value $\leq$ capacity of cut

..now...
The Story So far

value of a flow = flow across any cut

any flow value \( \leq \) capacity of cut

..now...

Maximum flow value = minimum cut capacity
Theorem: these are the same:

1. \( f \) is a maximum flow
2. the residual network \( G_f \) contains no augmenting paths
3. there exists a cut whose capacity is the value of \( f \)

1+2 would mean FF is correct.

1+3 would mean we can find minimum cuts in graphs using FF!
Proof of Max-Flow Min-Cut Theorem

Proof: (1 ⇒ 2) if augmenting path exists, could increase value of flow.

(2 ⇒ 3) Define $S$ to contain all vertices reachable from $s$ in residual $G_f$, $T = V - S$. Consider vertices $u, v$ where $u \in S, v \in T$. If edge $(u, v) \in E$, $f(u, v) = c(u, v)$ (otherwise $v \in S$). If $(v, u) \in E$, $f(v, u) = 0$ (otherwise $v \in S$). If neither edge $\in E$, $f(u, v) = 0$. So

$$|f| = \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{u \in S} \sum_{v \in T} f(v, u)$$

$$= \sum_{u \in S} \sum_{v \in T} c(u, v) - \sum_{u \in S} \sum_{v \in T} 0$$

$$= c(S, T)$$

(3 ⇒ 1) $\forall c, |f| \leq c(S, T)$, so if $|f| = c(S, T)$ then it’s maximum.
all nodes reachable from $s$ in $G_f$ are on one side

edges crossing cut are at capacity, by definition

no flow back from $T$ to $S$, also by definition
For example:

- What’s still confusing?
- What question didn’t you get to ask today?
- What would you like to hear more about?

Please write down your most pressing question about algorithms and put it in the box on your way out.

*Thanks!*