The edge or vertex connectivity of a pair of vertices is number of edge- or vertex-independent paths between them.

Show that edge/vertex connectivity is equal to the minimum edge/vertex cut set.

Max Flow/Min Cut then! edge-connectivity is equal to the maximum flow between pair vertices over unit-flow pipes.

Can we develop efficient algorithms for computing the maximum flow?

Ford-Fulkerson algorithm ( augmenting path algorithm).

average time $O((mn)^{1/2})$, (but potentially bad worst-case parameter).

Once one has computed maximum flow, of course also know # independent paths, and size of minimum edge cut set.

An extension can also find the actual edges that constitute the minimum edge cut set.

Restrict attention to the unit-pipe case, can be extended to the general case.

thoughts on algorithm?

Basic idea: run BFS from $s$ but to get one path.

this uses up some edges.

run again to get another independent path

\ldots
This greedy approach can get stuck in local optima:

using paths incorrectly since blocking things that shouldn't be.

```
S     t
```

Look from the back, but also look from the front.

Send fluid both ways down edges of network.

If fluid flowing both ways our pipe, this means there is no net flow.

so paths we find will no longer necessarily be independent paths since two can share edge as long as pass in opposite directions.

But key insight: flows that paths represent are still allowable flows, since no pipe is required to carry more than one unit of flow. [can have 3 farms, 4 barns, or 2 farm, 1 barn, etc.]

In the end, max flow = # independent paths, even though algorithm is counting something else.

⇒ count independent paths by counting special class of non-independent paths.

   Basically augment paths with backward paths.
To implant use a residual graph.

- directed graph in which edges connect pairs of vertices from original graph that still have capacity in given direction.

Initial construction:
- replace each undirected edge in initial network with 2 directed edges in each direction.

Basic Iteration:
- perform BFS on residual graph
- every time algorithm finds new path, update residual graph by adding directed edge in opposite direction to path between every pair of vertices along path [when no edge already there]

[if already have backword-pointing edge, throw away a forward-pointing one].

Largest number of edges which is on, so this takes $O(m)$.

- no difference is $O(mn)$ of BFS.

How many iterations, i.e. how many paths?

Dennis' bound: # independent paths can be no greater than smaller of degrees of source and target, $k_s, k_t$:

so $O\left(m \cdot \min(k_s, k_t) \cdot \min(n, m)\right)$.

Looking at average, $\left< \min(k_s, k_t) \right> \leq \left< k \right>$, and $\left< k \right> \geq \frac{2m}{n}$. 
We've analyzed complexity of algorithm and it works for one example, but is it generally correct?

How to prove? Essentially need to worry about Stepping stones for Circuits.

Suppose at some point in algorithm, we have found some or no paths for flow, but any paths we have found do not yet constitute maximum possible flow yet. Still room. Then we need to show there must still exist at least one augmenting path, so only stop when actually done.

Then: if at some point, if flow from s to t less than maximum possible flow, there must be at least one augmenting path on current residual graph.

Proof idea: if we subtract from the maximum flow any submaximal flow, the resulting difference flow necessarily contains at least one path from s to t, and that path is necessarily augmenting.

\[
\begin{align*}
\text{flow} & \quad - \quad \text{f} \quad = \quad \text{diff.}
\end{align*}
\]
To set independent paths themselves,
take final residual graph and remove every matching bidirectional link (but copy no flow).

What remains is set of independent paths that is max. (one of them).