Max-Flow is a very general algorithms problem that can be nevertheless solved efficiently. The definition of the problem involves shipping stuff around a network, but it can be used to model & solve so many other unrelated-looking problems:

a paradigm for finding optimal
- matchings in bip. graphs
- shortest paths in graphs with negative edge costs
- schedules with precedence constraints
- sports playoff scenarios...

Input: 
- Directed graph \( G = (V, E) \)
- "source" \( s \in V \) (no in-edges)
- "target" \( t \in V \) (no out-edges)
- "positive capacities" \( c_e \in \mathbb{R}_+ \) on each edge \( e \in E \)

You are shipping "stuff" (e.g., tons of cement) along edges. Capacities (e.g., railroads) indicate max amount you can ship (e.g., per day). Fractional amounts OK.

Goal: ship as much from \( s \) to \( t \) as possible,

subject to "Flow constraint": \( \forall v \in \{s, t\}, \text{ incoming flow = outgoing flow} \)
Maximize the volume \( V(x) = f(x) \) (5) for all \( x \) in \( [a, b] \), for all \( \epsilon \) in \( E \), capped by \( \epsilon \) cubic centimeters.

**Formula:** Seek \( f: E \rightarrow R \) not allowing fractions, flow \( \epsilon \) ft.

For \( m \in \log u \), \( \log(20) \) can still be used. For small exercises, you lose a lot about time.

Is it true that even controlled, "polynomial"? Is an issue. Not even considered. No Reply. This is not a work.

Also, no good if \( \epsilon \leq 1000 \text{ kg} \).

\( \log(20) \) or \( \log(4) \) is the study.

"From Euler's formula, \( \Omega(n) \) or \( \Omega(2^n) \).

\\( \text{If} \ n \text{ is odd, then} \ L = 2, \text{ and if} \ n \text{ is even, then} \ L = \text{odd}. \)

How efficient? Say capabilities in 2, 3, ..., \( \Omega \), \( \text{ if not, we edge.} \)

How much people! Say capabilities in 2, 3, ..., \( \Omega \), \( \text{ if not, we edge.} \)

"This is really nice!" "Shh, it's two of (each), not of (each)."

\( \text{If all caps} \ (c) \ \text{ are integers,} \)

\( \text{We'll see how} \)
Stop

![Graph example]

Previous example:

authors! what's good on us. An

so no big deal if running time

edges summing correctly

undone. Added parallel

now much

new cut

left hand instance \((G', c')\) found as follows.

**Def**: Given instance \((G, c)\) \(f\) a flow, \(f\) the

Idea 2: Greedy, but allow undirected

But max-flow was at

Such: no more paths

Example 1:

Left with free capacities:

Can push as much flow as you can report.

E.g.:

Find a path simple. Push as much flow as you can report.

Try 

Greedy
In this eg, \( f_1 \cdot f_2 = \) is max flow.

[Not a coincidence, this alg works!]

**FF Alg on \((G,c)\)**

1. Initialize \( f(e) = 0 \) for.
2. While \( \exists \) s-t path in residual graph \( G_f \),
   - Let \( P \) be any such path
   - Let \( b = \min \{ c(e) : e \in P \} \) ("bottleneck")
   - For each \( e \in P \):
     - \( f(e) = b \)

// this creates a new residual inst. \( G_f, c_f \).

**Running time:**
- \( O(m) \) per iter (BFS)
- \# iters \( \leq \) value of max flow, \( F^* \)
- because flow goes up +1 each iter
- Time: \( O(F^* m) \)

**e.g.**

\[ G, c \]

Say \( f_1 = s \rightarrow a \rightarrow s \rightarrow d \rightarrow t \). Solvent is 1, push 1 unit up

New \( G_{f_1} \):
Total: 4
Sofar
If we know that's max, I told you so.

Termination
No more s->t paths. Final flow \( f^* = f_1 + f_2 + f_3 + f_4 \).

Say, let \( R = \{ v : v \text{ reachable from } s \} = \{ s, a \} \neq \emptyset \).

Claim: in original \( G \), each edge \( (u,v) \) out of \( R \) must be saturated by \( f^* \).

Else \( G_{f^*} \) would have some capacity on \( (u,v) \), and so \( v \) would be reachable from \( s \) too.

Claim: in original \( G \), each edge \( (u,v) \) into \( R \) must be unused by \( f^* \).

Else \( G_{f^*} \) would have some capacity on \( (u,v) \).

\[
f^* (R) = f^* (R) = \sum_{(u,v) \in R} f^* (u,v) - \sum_{(u,v) \in \partial^{-} R} f^* (u,v)
\]
def \( c^* = \sum_{(u,v) \in \partial^{+} R} c^* (u,v) - \sum_{(u,v) \in \partial^{-} R} c^* (u,v) \)

def \( \text{"Net flow of } f^* \text{ out of } R \" = \) net flow of \( f^* \) out of \( R \).

Summary: At end of F.F., you get a set \( R \subseteq V \) with \( s \in R, t \notin R \) (this is called an "s-t cut")
And net flow of \( f^* \) out of \( R = \text{cap}(R \rightarrow \overline{R} \text{ cut}) \).
This is amazing. Why?

1. Suppose \( G \) is any \( s-t \) cut (\( s \in S, t \in T \)).
   
   Then it's easy to see: \( \text{max-flow} \leq \text{cap}(G \rightarrow \overline{G}) \).
   
   The total amount of capacity getting out of \( G \) is the most you could ever ship from \( s \) to \( t \). \( G \) is like an upper limit.

2. And if \( G \) is any cut, \( f \) is any flow,
   
   \[ \text{value}(f) = \text{net flow of } f \text{ out of } G = f^\text{out}(G) - f^\text{in}(G) \]
   
   just because flow can't get stuck anywhere. Net flow out of \( G \) is \( f^\text{out}(G) \), out of \( \overline{G} \) is \( f^\text{in}(G) \), etc.
   
   So that equals \( \text{net flow out of } G = f^\text{out}(G) \).

Now look at (1):

(1) \( \implies \text{LHS} = \text{value}(f^*) \)

(1) \( \implies \text{RHS} \geq \text{max-flow} \).

So \( \text{value}(f^*) \geq \text{max-flow} \implies \text{must equal max-flow!} \)

(1) \( \implies \text{F.F. finds maximum flows} \)

Also finds \( R \) with minimum possible cut value

(1) \[ \text{Theorem: Max } s-t \text{ Flow value } = \text{Min } s-t \text{ Cut value.} \]

And F.F. finds the max flow \& the Min-Cut in time \( O(m \cdot \text{Val}(f^*)) \). Great if you have reason to believe \( \text{Val}(f^*) \) not too large!...