Network Flow
Network Flow terminology

Network flow is similar to finding how much water we can bring from a “source” to a “sink” (infinite) (intermediates cannot “hold” water)
Network Flow terminology

Definitions:
\( c(u,v) \): edge capacity, \( c(u,v) \geq 0 \)
\( f(u,v) \): flow from \( u \) to \( v \) s.t.
1. \( 0 \leq f(u,v) \leq c(u,v) \)
2. \( \sum_v f(u,v) = \sum_v f(v,u) \)
\( s \): a source, \( \sum_v f(s,v) \geq \sum_v f(v,s) \)
\( t \): a sink, \( \sum_v f(t,v) \leq \sum_v f(v,t) \)
Network Flow terminology

Definitions (part 2):

\[ |f| = \sum_v f(s,v) - \sum_v f(v,s) \]

\(^\wedge\) amount of flow from source

Want to maximize \(|f|\) for the maximum-flow problem
Network Flow terminology

Graph restrictions:
1. If there is an edge \((u,v)\), then there cannot be edge \((v,u)\)
2. Every edge is on a path from source to sink
3. One sink and one source

(None are really restrictions)
Network Flow terminology

1. If there is an edge \((u,v)\), then there cannot be edge \((v,u)\)
Network Flow terminology

2. Every edge is on a path from source to sink
flow in = flow out, only possible flow in is 0
(worthless edge)
Network Flow terminology

3. One sink and one source

\[
\begin{align*}
\text{sink: } & s_1, s_2 \\
\text{source: } & t_1, t_2 \\
\text{sink: } & s \\
\text{source: } & t
\end{align*}
\]
Ford-Fulkerson

Idea:
1. Find a path from source to sink
2. Add maximum flow along path (minimum capacity on path)
3. Repeat

Note: this path needs to be found in a "residual" graph
What is a residual graph?

Forward edges = capacity left
Back edges = flow

Original  Residual
Ford-Fulkerson

Idea: Find a way to add some flow, modify graph to show this flow reserved... repeat.

Augment
Ford-Fulkerson

\[ \text{Ford-Fulkerson}(G, s, t) \]

initialize network flow to 0

while (exists path from \( s \) to \( t \))

\hspace{1cm} \text{augment flow, } f, \text{ in } G \text{ along path}

return \( f \)

(Note: “augment flow” means add this flow to network)
Ford-Fulkerson

drawings of graphs with directed edges and capacities, showing the development of a flow through the network.
Subscript “f” denotes residual (or modified graph)
$G_f = \text{residual graph}$
$E_f = \text{residual edges}$
$c_f = \text{residual capacity}$
$c_f(u,v) = c(u,v) - f(u,v)$
$c_f(v,u) = f(v,u)$

“forward edge” capacity - flow
“back edge” just flow
Ford-Fulkerson

Ford-Fulkerson(G, s, t)
for: each edge (u,v) in G.E: (u,v).f=0
while: exists path from s to t in G_f
    find c_f(p) // minimum edge cap. on path
    for: each edge (u,v) in p
        if(u,v) in E: (u,v).f=(u,v).f + c_f(p)
        else: (u,v).f=(u,v).f - c_f(p)
Ford-Fulkerson

Runtime:

How hard is it to find a path?

How many possible paths could you find?
Ford-Fulkerson

Runtime:

How hard is it to find a path?
- $O(E)$ (via BFS or DFS)

How many possible paths could you find?
- $|f^*|$ (paths might use only 1 flow)

.... so, $O(E \ |f^*|)$
Ford-Fulkerson

\[(f \uparrow f')(u,v) = \text{flow } f \text{ augmented by } f'\]
\[(f \uparrow f')(u,v) = f(u,v) + f'(u,v) - f'(v,u)\]

Lemma 26.1: Let \( f \) be the flow in \( G \), and \( f' \) be a flow in \( G_f \), then \((f \uparrow f')\) is a flow in \( G \) with total amount:
\[|f \uparrow f'| = |f| + |f'|\]

Proof: pages 718-719
Ford-Fulkerson

For some path $p$:
$c_f(p) = \min(c_f(u,v) : (u,v) \text{ on } p)$
\(\wedge\wedge\) (capacity of path is smallest edge)

Claim 26.3:
Let $f_p = c_f(p)$, then
$|f \uparrow f_p| = |f| + |f_p|$
Ford-Fulkerson

More bad notation:
\[ c(u,v) = \text{capacity of an edge} \]
if u and v are single vertexes

\[ c(S,T) = \text{capacity across a cut} \]
if S and T are sets of vertexes

... Similarly for \( f(u,v) \) and \( f(S,T) \)
Max flow, min cut

Relationship between cuts and flows?

\[ c(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v) \]

\[ f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u} \sum_{v} f(v,u) \]
Max flow, min cut
Max flow, min cut

Relationship between cuts and flows?
\[ c(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v) \]
\[ f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u} \sum_{v} f(v,u) \]

cut capacity \geq \text{flows across cut}
Max flow, min cut

Lemma 26.4
Let (S,T) be any cut, then $f(S,T) = |f|$.

Proof:
Page 722
(Kinda long)
Corollary 26.5
Flow is not larger than cut capacity
Proof:
\[|f| = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u) \leq \sum_{u \in S} \sum_{v \in T} f(u,v) \leq \sum_{u \in S} \sum_{v \in T} c(u,v) = c(S,T)\]
Theorem 26.5
All 3 are equivalent:
1. $f$ is a max flow
2. Residual network has no aug. path
3. $|f| = c(S,T)$ for some cut $(S,T)$

Proof: = min cut (i.e. bottleneck)
Will show: $1 \implies 2$, $2 \implies 3$, $3 \implies 1$
Max flow, min cut

\[ f \text{ is a max flow} \implies \text{Residual network has no augmenting path} \]

**Proof:**
Assume there is a path \( p \)
\[ |f \uparrow f_p| = |f| + |f_p| > |f|, \text{ which is a} \]
contradiction to \( |f| \) being a max flow
Residual network has no aug. path => \(|f| = c(S,T)\) for some cut (S,T)

Proof:
Let \(S = \) all vertices reachable from \(s \in G_f\)

\(u \in S, v \in T \Rightarrow f(u,v) = c(u,v)\) else there would be path in \(G_f\)
Max flow, min cut

Also, \( f(v,u) = 0 \) else \( c_f(u,v) > 0 \) and again \( v \) would be reachable from \( s \)

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

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

\[= c(S, T)\]
$|f| = c(S,T)$ for some cut $(S,T)$
$\Rightarrow f$ is a max flow

Proof:
$|f| \leq c(S,T)$ for all cuts $(S,T)$

Thus trivially true, as $|f|$ cannot get larger than $C(S,T)$
Edmonds-Karp

exists shortest path (BFS)

Ford-Fulkerson(G, s, t)

for: each edge (u,v) in G.E: (u,v).f=0

while: exists path from s to t in G_f

find c_f(p) // minimum edge cap.

for: each edge (u,v) in p

if(u,v) in E: (u,v).f=(u,v).f + c_f(p)

else: (u,v).f=(u,v).f - c_f(p)
Edmonds-Karp

Lemma 26.7
Shortest path in $G_f$ is non-decreasing

Theorem 26.8
Number of flow augmentations by Edmonds-Karp is $O(|V||E|)$
So, total running time: $O(|V||E|^2)$
Matching

Another application of network flow is maximizing (number of) matchings in a bipartite graph.

Each node cannot be “used” twice.
Matching

Add “super sink” and “super source” (and direct edges source -> sink) capacity = 1 on all edges