Network Flows

\[ \text{flow} = \text{max} \]

**Goal:** maximize the "flow" from \( s \) to \( t \)

1) **flow along an edge** \( = f(e) \geq 0 \)

\[ c(e) \geq f(e) \geq 0 \]

\[ \text{capacity} \uparrow \text{flow} \uparrow \]

2) **flow must be conserved at each vertex** \( v \)

Incoming flow

\[ f^\text{in}(v) = \sum_{e=(v,u)} f(v,u) \]

Outgoing flow

\[ f^\text{out}(v) = \sum_{e=(u,v)} f(u,v) \]

\[ f^\text{in}(v) = f^\text{out}(v) \]

3) **value of the flow**

\[ |F| = \left| f^\text{out}(s) \right| = f^\text{in}(t) \]
\[|F| = \min \left( f^\text{out}(s), f^\text{in}(t) \right) = f^\text{in}(t) \]

**Problem/Task:** Maximize the flow (find value of \( f(e) \) for)

\[
\max \ f^\text{out}(s) \\
\text{such that } c(e) \geq f(e) \geq 0 \\
\text{and flow is conserved at } u, \ i.e. f^\text{in}(u) = f^\text{out}(u) \text{ (except for } s \text{ and } t) \\
\]

**Duality:** Maximum flow = minimum cut in the graph

\[
A = \{ s, u, v \} \\
B = \{ \ell, v' \}
\]

Capacity Cut \((A, B) = s + 10 + 6 = 21\)

Any flow must cross the cut

\(|F| \leq \text{capacity of cut} \leq 21\)

\[
\text{Cut}(A, B) = s + 2 = 3 \\
|F| \leq 3
\]

\(|F| = 3\)

\[\text{Maximum flow} = \text{min cut} \]
Ford-Fulkerson Algorithm

Given a graph $G$, and given some flow along the edge, we will define a residual graph $G_f$.

The idea finds paths from $s$ to $t$ in $G_f$ and tries to increase the flow along that path.

$G_f$

1) For any edge $(a,b) \in G$ such that $f(a,b) > 0$

In the residual graph $G_f$, add the edge $(a,b)$ with "residual" capacity

$C(e) - f(e)$

2) We also add a backword edge in $G_f$

$(b,a) = f(e)$

$G_f = G$

Find a path
Find a path
\[ S \rightarrow u \rightarrow t : 20 \]
Sending 20 units from \( u \rightarrow t \)

Find another path
\[ S \rightarrow v \rightarrow u \rightarrow t : 10 \]
Send 10 units from \( v \rightarrow u \)

\[ f = 30 \]
\[ f^i(u) = 20 \quad f^o(u) = 20 \]
\[ f^i(v) = 20 \quad f^o(v) = 20 \]

Find a path from \( s \rightarrow t \)

There is no such path!

Done!

\( \text{Ford-Fulkerson Algorithm} \)

\[ f(e) = 0 \quad \forall e \in G \]

While \( \exists \text{ a path from } s \rightarrow t \) in \( G \)

Use DFS/BFS \( \Theta (|V| + |E|) \)

Shortest path from \( s \rightarrow t \) using edge \( e \)
Computational complexity?

Each iteration $g$ while $O(|V| + |E|)$ time

how many iterations before we stop?

each time we find a path $p: s \rightarrow t$, the flow increase by $cc(p)$

In the worst case, the flow increase by $1$ unit

any flow $|f| \leq \sum_{e \in E} c(e) = C$

$O(C)$ iterations in the worst case! upper bound on the value of the flow

$O\left(C \cdot (|V| + |E|)\right)$

$= O\left(C \cdot |E|\right)$

$= O\left(|f^*| \cdot |E|\right)$

Pseudo-polynomial running time in worst case
Edmonds Karp Algorithm

In FF (Ford Fulkerson)

Choose the paths "wirely"

always find shortest path (in BFS), in terms of the number of edges, between s and t.

Then the # of iterations is bounded by $O(\sqrt{V} \cdot |E|)$

And therefore the total time

$O\left( V \cdot \sqrt{E} \cdot |E| \right) = O\left( V \cdot |E|^2 \right)$
\[ O(\frac{|V| \cdot |E|}{r}) = O(|V| \cdot |E|^2) \]

When FF/ER stops, we can extract both the flow and cut info.

- Maximum flow = minimum cut

\[ G \rightarrow G^f \]

In \( G^f \): find all vertices that are reachable from \( s \); they comprise one side of the cut.

\[ A \]

and the rest \( V - A = B \)

- Any forward edge \( e \) is an edge in \( G^f \)
- If an edge from \( B \) vertex to \( A \) vertex, then it must have flow 0.
\[ \Rightarrow \min \text{ cur}(A, B) = \max \text{ flow in } E \]