Flows and cuts:

we have a graph \( V = \{ \text{vertices} \} \)
\( E = \{ \text{edges} \} \)

make this graph directed

there is a unique vertex \( s \) (the source): this has only outgoing edges

and \( d \) (the drain): this has only incoming edges

- each edge has an associated capacity (which is integer, \( > 0 \))
- flow through the edge may not exceed its capacity
- count incoming flows +ve, outgoing -ve
- Kirchhoff's law applies for all but \( s \) and \( d \)

\[
\sum_{e \in E(V)} f_e = 0
\]
Problem: What is the maximum flow from $s$ to $d$?

- many practical applications
  - many algorithms follow

**Augmenting path:**

- a path from $s$ to $d$
  - undirected
    - all forward $(s \rightarrow d)$ arrows below capacity
  - all backward $(d \rightarrow s)$ arrows greater than zero

1) if an augmenting path exists, flow is not maximal
   Because we can increase flow along this path.

2) if flow is maximal, there is no AP?
   Because if there were, we could augment
Algorithm outline

- Find augmenting path
- Max flow on this path

Efficiency depends on details of how path is found.

Now, consider a disconnection cut:
- Set of edges that disconnects $s, d$
- Divide verts into $A, B$ s.t. $A \cup B = V$ and $A \cap B = \emptyset$ and $s \in A$, $d \in B$

```
Diagram:
```

- Total flow $A \rightarrow B = \sum_{\text{forward}} - \sum_{\text{backward}}$
Value of the cut

\[ = \sum \text{capacities forward} \quad \sum \text{capacities backward} \]

⇒ If flow is maximal, there is at least 1 cut so that

- all forward are at capacity
- all backward are zero

⇐ if such a cut exists, flow is maximal.

(Easy; there can be no augmenting path)

⇒ consider all paths, s → d
Each has, at least, either an edge which is forward and at capacity or backward and at zero.

(There might be more than one)

Now cut each path at some such edge.

→ Is this a cut?

No: Because some verts could appear on both s-side and d-side.

But: We can get a cut out of it.

Assume \( v \in V(s) \), and \( v \in V(d) \)

Cut a move from \( V(d) \) to \( V(s) \)

because otherwise there would be an augmenting path.
Notice

Value of this cut

= Max flow.

But there cannot be a cut with lower value, because then flow would be smaller.

(Getting a min-cut from a max-flow:

= wasn't constructive (cycles)

cut all capacity, 0 edges.

= if this is a cut, stop

= otherwise, put other CC's into V(s), V(d) at will

= restore edges in V(s), V(d)

= this is a cut)
Min-cut = easy 0-1 QP

- Consequence - if we have an easy 0-1 QP, we can do it with any fast min-cut alg we have.

Have \( x_i \), 1 per vertex:

\[
\begin{align*}
U: & \quad x_i = 0 \\
V: & \quad x_i = 1
\end{align*}
\]

Value of cut = \( \sum_{a \in \text{edges from } u \to v} c_a \)

= \( \sum_{i} (1 - x_i)(x_j). C_{i \rightarrow j} \)

& all edges
So we have

\[ \min \sum_{i,j} (1-x_i)(x_j) \quad C_{i \rightarrow j} \]

\[ \text{st.} \quad x \in \{0,1\} \]

But this is

\[ \max \sum_{i,j} (x_i - 1)x_j \quad C_{i \rightarrow j} \]

\[ \text{st.} \quad x \in \{0,1\} \]

And we have

\[ \max \frac{1}{2}x^T A x + b^T x \]

\[ \text{non neg} \]

\[ \text{st.} \quad x \in \{0,1\} \]
Goldberg - Tarjan Push-Relabel:

- one of many alg's, r: good complexity properties

- idea: use infeasible flows ("preflows")
  - keep approximate track of path length to \( t \)
  - make flows "more feasible"

- def's:
  - work on \( (V, V \times V) \)
  - \( i.e. \) all directed edges
  - any edge not in \( G \) gets \( c = 0 \)
  - flow is

\[
\begin{align*}
\mathbf{f} : V \times V & \rightarrow \mathbb{R} \\
\mathbf{f}(u \rightarrow v) & \leq \mathbf{c}(v \rightarrow w) \\
\mathbf{f}(v \rightarrow w) & = -\mathbf{f}(w \rightarrow v)
\end{align*}
\]

\[
\sum_{u} \mathbf{f}(u \rightarrow v) = 0 \quad \text{for all } v \in V - \{s,t\}
\]
The value of a flow is

$$\sum_{v} f(v \rightarrow t)$$

A preflow is $f : V \times V \rightarrow \mathbb{R}$ such that

$$f(v \rightarrow w) \leq c(v \rightarrow w)$$

$$f(v \rightarrow w) = -f(w \rightarrow v)$$

$$\sum_{u} f(u \rightarrow v) \geq 0$$

$$\sum_{u} f(u \rightarrow v) = e(v)$$ is the excess

- We must push excess to the target
- Residual capacity for some preflow $f$ is

$$f_{r}(v \rightarrow w) = c(v \rightarrow w) - f(v \rightarrow w)$$

If $f_{r}(v \rightarrow w) > 0$, can move this edge.
Label function.

\[ d: V \rightarrow N \cup \{0, \infty\} \]

is a valid labelling for \( f \) if

\[ d(s) = |V|, \quad d(t) = 0 \]

\[ d(v) \leq d(w) + 1 \quad \text{for} \; v \rightarrow w \quad \text{st} \]

\[ f(v \rightarrow w) > 0 \]

(Notice gives an approx. of "dist" to t along paths where we can push)

Algorithm:

Initialize all product and value function:

\[ f(s) = (\infty, 0) \quad \text{for all} \; v \]

\[ f(u \rightarrow v) = 0 \quad \text{for all} \; u \in V, \; v \]

\[ d(s) = |V|, \quad d(t) = 0, \quad d(v) = 0, \; \forall v \in V \setminus \{s, t\} \]
Algorithm:

1. Initialize with preflow and valid labelling
   - \( f(s,v) = c(s,v) \) for all \( v \)
   - \( f(u,v) = 0 \) for all \( u \neq s, v \)

   \[ d(s) = |V|, \quad d(t) = 0, \quad d(v) = 0, \quad \forall v \in V - \{s,t\} \]

2. While there is an active vertex

   \[ \text{active is } (v \neq s,t, \quad e(v) > 0, \quad d(v) < \infty) \]

3. Choose one and execute an admissible operation for that vertex.
Operations are:

Push: (on an edge \( v \rightarrow w \))
- admissible if \( v \) is active
  and \( f(v) > 0 \)
  and: \( d(v) = d(w) + 1 \)

\[ S = \min \{ e(v), f(v \rightarrow w) \} \]

\[ f(v \rightarrow w) = f(w \rightarrow v) + S \]
\[ f(w \rightarrow v) = f(w \rightarrow v) - S \]

\[ f(v \rightarrow w) = f(v \rightarrow w) / S \]
\[ f(w \rightarrow v) = f(w \rightarrow v) + S \]

- admissible if \( v \) is active

\[ e(v) \rightarrow e(v) - S \]
\[ e(w) \rightarrow e(w) + S \]

always implies \( e(v) < e(w) \)

\[ d(v) = \min \{ d(w) + 1 \} \text{ over edges} \]
\[ f(v \rightarrow w) > 0 \]
Relabel:

- admissible if \( v \) is active, and if \( f(v \to w) > 0 \)

always an edge \( d(v) \leq d(w) \)

\[
d(v) = \min \{ d(w) + 1 \mid \text{over edges} \}
\]

Facts:

1. Let \( f \) be a preflow on graph \( G \), and \( d \) an arbitrary valid labeling.
2. If \( f \) is a preflow and \( d \) a valid labeling, then either a spash or a relabel is admissible for \( v \).

Then \( f \) is not accessible from \( s \).

- \( f \) is always a preflow
- if \( f \) starts a preflow \( d \) a valid labeling

always remain so during alg.

(Induction)
Let $f$ be a preflow on graph $G$ and $f$ an arbitrary valid labelling on $V$. Write $G_f$ for $(V, E_f)$

$$E_f = \{ v \to w \in E : f(v \to w) > 0 \}$$

If the graph $G$ contains a non-admissible path $S$ then there is a not admissible labelling $f$ such that $f$ is a flow, and $f$ is maximising.

**Proof:**

Assume there is no non-admissible path $S = v_0 \to v_1 \to v_2 \ldots \to v_m = t$.

Now, $d(v_i) \leq d(v_{i+1}) + 1$ for each path.

So, $d(v_0) = d(s) \leq d(t) + m < |V|$

This because $m \leq |V| - 1$, and $d(t) = 0$.

But $d(s) = 1 \to 0$ (for valid labelling).

Admissible operations.
If the algorithm terminates and all labels are finite, then at term $f$ is a flow, and is maximal.

**Proof:** Terminates because no admissible vert $\Rightarrow$ flow

maximal, because no augmenting path.

**Theorem:** Alg terminates after at most $O(|V|^2|E|)$ admissible operations
Flows, cuts and linear programs.

For a directed graph, write an incidence matrix $A$ between vertices and edges:

$$a_{ij} = \begin{cases} 
0 & \text{if } e_j \text{ not incident on } v_i \\
1 & \text{if } e_j \rightarrow v_i \text{ (i.e. } e_j \text{ enters } v_i) \\
-1 & \text{if } e_j \leftarrow v_i \text{ (i.e. } e_j \text{ leaves } v_i) 
\end{cases}$$

Make row 1 correspond to $s$ (source), row 2 correspond to $d$ (sink).

**Variations:**

$$\max \quad \sum_v v$$

subject to

$$A \mathbf{f} + v \begin{bmatrix} -1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = 0$$

$$\mathbf{f} \geq 0, \quad \mathbf{f} \leq c$$

Flow
Another version:

\[
A_R = \begin{bmatrix}
\text{all rows of } A \\
\text{but the first two}
\end{bmatrix}
\]

\[
\begin{align*}
\mathbf{w}^T &= \text{first row of } A \\
&\quad \text{second row of } A \\
&\quad \text{(which CSP target)}
\end{align*}
\]

\[
\max \mathbf{w}^T \mathbf{f}
\]

\[
\begin{align*}
\text{st. } A_f &= 0 \\
A_{R_f} &= 0
\end{align*}
\]

(Kirchhoff's laws: \( f \geq 0 \), \( f \leq c \))

We will do max-flow, min-cut by LP-duality

\text{Step 1:} \quad A, A_R \text{ are TUM}
Proof: (Induction)

- Consider a minor $B$, rank $k$
- 3 cases
  - Some column is all zeros, $\det(B) = 0$
  - $B$ has 1 column that has exactly 1 non-zero
    $\det(B) = \det\begin{bmatrix} 1 & b^T \\ 0 & B \end{bmatrix}_{(i,i)}$ by induction
    $\det B = \pm \det B$
  - $B$ has all columns with 2 non-zeros
    then $1^r B = 0$, $\det B = 0$
Duals of linear programs

1) Assume we have a linear program

\[ \begin{align*}
\text{min} & \quad -\omega^T f \\
\text{st.} & \quad A_k f = 0 \\
& \quad f \geq 0 \\
& \quad c - f \geq 0
\end{align*} \]

(This isn't the standard form — just convenient for flow)

Its Lagrangian is

\[ L(f, \lambda_e, \lambda_i^1, \lambda_i^2) = -\omega^T f + \lambda_e^T (A f) - \lambda_i^1 f - \lambda_i^2 (c - f) \]
we want the Lagrange dual.

\[
\inf_{\mathcal{F}} \mathcal{L}(\mathbf{f}, \lambda_e, \lambda_i^1, \lambda_i^2)
\]

\[
= \begin{cases} 
-\chi_i^2 \mathbf{c} & \text{if } \begin{bmatrix} -\mathbf{w}^T + \lambda_e A_e - \lambda_i^T + \lambda_i^2 \end{bmatrix} = 0 \\
-\infty & \text{otherwise}
\end{cases}
\]

Now the dual problem is to maximize this

i.e.

\[
\max \quad -\chi_i^2 \mathbf{c}
\]

s.t.

\[
-\mathbf{w}^T + \lambda_e^T A_e - \lambda_i^1 + \lambda_i^2 = 0
\]

\[
\lambda_i^1 \geq 0
\]

\[
\lambda_i^2 \geq 0
\]

We can clean this up a bit.
\[
\begin{align*}
\max & \quad -\lambda_i^T C \\
\text{s.t.} & \quad A_T \lambda_e + \lambda_i^2 \geq \omega^T \\
\lambda_i^2 & \geq 0
\end{align*}
\]
Step 2:

- Set up Max Flow as L.P.

$$\max \ w^T f$$

$$st \ A_R f = 0 \quad f \geq 0 \quad f \leq c$$

Step 3:

- Take the dual: because feasible set has interior point, max flow = min dual

- Dual is:

$$\min \ y^T c$$

$$st \ y \geq 0 \quad y^T + z^T A_R \geq w^T$$

Step 4:

- at solution, y and z are INTEGER.
to see this, write constraints as
\[
\begin{bmatrix}
1 & A_r
\end{bmatrix}
\begin{bmatrix}
y \\
\ast
\end{bmatrix} \succeq w
\]
\[
\begin{bmatrix}
y \\
\ast
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & A_r
\end{bmatrix}
\text{ is TUM}
\]

(Because minors are minors of } A_r \text{)

step 5:

\cdot write \quad \hat{z} = \begin{bmatrix} 0 & -1 & \ast \end{bmatrix}

\quad \uparrow \quad \uparrow \quad \text{source} \quad \text{drain}

\cdot then constraint is
\[
y^T + \hat{z}^T A \geq 0
\]
Step 6: A solution to the dual yields a cut

\[ U = \{ v \in \text{Verts} \mid \hat{z}_v > 0 \} \]

so \( U \) contains \( S \), not \( \{|\} \).

Step 7: Value of cut

\[ \sum_{i \in \text{edges leaving } U} c_i = C(S^{\text{out}}(U)) \]

it is enough to show

\[ C(S^{\text{out}}(U)) \leq y^T c = \max \text{ flow value, from 3} \]

because we must have \( C(S^{\text{out}}(U)) \geq \) flow

for any \( \hat{c} \).
Step 8:

- To show this, it is enough to show that for each edge \( a \) in the cut, \( y_a \geq 1 \)

  (because then \( y^T c \geq \sum_{i \text{ edges in cut}} c_i \frac{y_i}{y} = c^\text{out}(u) \))

- \( a = u \rightarrow v \)

  So \( \hat{z}_u > 0 \) (by defn of cut)

  \( \hat{z}_v \leq -1 \) (integer, less than 0)

  \( y^T + \hat{z}^T A > 0 \) implies \( y_a + \hat{z}_v - \hat{z}_u > 0 \)

  So \( y_a > \hat{z}_u - \hat{z}_v > 1 \)

  Done
Hence, similarly to Corollary 10.6a one has:

**Corollary 10.11a.** For integer capacities, a maximum flow can be found in time $O(n(\phi + m))$, where $\phi$ is the maximum flow value.

**Proof.** Similar to the proof of Corollary 10.6a.

Therefore,

**Corollary 10.11b.** For integer capacities, a maximum flow can be found in time $O(nm \log C)$.

**Proof.** In the proof of Theorem 10.10, a maximum flow with respect to $c'$ can be obtained from 2$\beta''$ in time $O(nm)$ (by Corollary 10.11a), since the maximum flow value in the residual graph $D_{\beta''}$ is at most $m$.

### 10.8b. Complexity survey for the maximum flow problem

Complexity survey (* indicates an asymptotically best bound in the table):

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(n^2mC)$</td>
<td>Dantzig [1951a] simplex method</td>
</tr>
<tr>
<td>$O(nmC)$</td>
<td>Ford and Fulkerson [1955,1957b] augmenting path</td>
</tr>
<tr>
<td>$O(nm^2)$</td>
<td>Dinitz [1970], Edmonds and Karp [1972] shortest augmenting path</td>
</tr>
<tr>
<td>$O(n^2m \log nC)$</td>
<td>Edmonds and Karp [1972] fattest augmenting path</td>
</tr>
<tr>
<td>$O(n^2m)$</td>
<td>Dinitz [1970] shortest augmenting path, layered network</td>
</tr>
<tr>
<td>$O(m^2 \log C)$</td>
<td>Edmonds and Karp [1970,1972] capacity-scaling</td>
</tr>
<tr>
<td>$O(nm \log C)$</td>
<td>Dinitz [1973a], Gabow [1983b,1985b] capacity-scaling</td>
</tr>
<tr>
<td>$O(n^3)$</td>
<td>Karzanov [1974] (preflow push); cf. Malhotra, Kumar, and Maheshwari [1978], Tarjan [1984]</td>
</tr>
<tr>
<td>$O(n^2 \sqrt{m})$</td>
<td>Cherkasskii [1977a] blocking preflow with long pushes</td>
</tr>
<tr>
<td>$O(nm \log^2 n)$</td>
<td>Shiloach [1978], Galil and Naamad [1979,1980]</td>
</tr>
<tr>
<td>$O(n^{5/3}m^{2/3})$</td>
<td>Galil [1978,1980a]</td>
</tr>
</tbody>
</table>


$O(nm \log(n^2/m))$ Goldberg and Tarjan [1986,1988a] push-relabel + dynamic trees

$O(nm + n^2 \log C)$ Ahuja and Orlin [1989] push-relabel + excess scaling

$O(nm + n^2 \sqrt{\log C})$ Ahuja, Orlin, and Tarjan [1989] Ahuja-Orlin improved

$O(nm \log((n/m)\sqrt{\log C} + 2))$ Ahuja, Orlin, and Tarjan [1989] Ahuja-Orlin improved + dynamic trees

$O(n^3/\log n)$ Cheriyian, Hagerup, and Mehlhorn [1990,1996]

$O(n(m + n^{5/3} \log n))$ Alon [1990] (derandomization of Cheriyian and Hagerup [1989,1995])

$O(nm + n^{2+\epsilon})$ (for each $\epsilon > 0$) King, Rao, and Tarjan [1992]

$O(nm \log_{n/\log n}(n^{2+\epsilon})$ (for each $\epsilon > 0$) Phillips and Westbrook [1993,1998]

$O(n^3/\log n)$ King, Rao, and Tarjan [1994]

$O(m^{3/2} \log(n^2/m) \log C)$ Goldberg and Rao [1997a,1998]

$O(n^{2/3} \log(n^2/m) \log C)$ Goldberg and Rao [1997a,1998]

Here $C := \|c\|_\infty$ for integer capacity function $c$. For a complexity survey for unit capacities, see Section 9.6a.

**Research problem:** Is there an $O(nm)$-time maximum flow algorithm?

For the special case of planar undirected graphs:

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(n^2 \log n)$</td>
<td>Itai and Shiloach [1979]</td>
</tr>
<tr>
<td>$O(n \log^2 n)$</td>
<td>Reif [1983] (minimum cut), Hassin and Johnson [1985] (maximum flow)</td>
</tr>
<tr>
<td>$O(n \log n \log^* n)$</td>
<td>Frederickson [1983b]</td>
</tr>
<tr>
<td>$O(n \log n)$</td>
<td>Frederickson [1987b]</td>
</tr>
</tbody>
</table>

For directed planar graphs:

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(n^{3/2} \log n)$</td>
<td>Johnson and Venkatesan [1982]</td>
</tr>
<tr>
<td>$O(n^{4/3} \log^2 n \log C)$</td>
<td>Klein, Rao, Rauch, and Subramanian [1994], Henschiger, Klein, Rao, and Subramanian [1997]</td>
</tr>
<tr>
<td>$O(n \log n)$</td>
<td>Weih [1994b,1997b]</td>
</tr>
</tbody>
</table>
Now, notice:

- in an augmenting path, we always had a flow

- is this a cut? not necessarily

- e.g.

\[
\begin{array}{c}
\text{S} \\
1 \rightarrow 1 \rightarrow 1 \\
2 \rightarrow 3 \rightarrow 3 \\
\text{d}
\end{array}
\]

- flow that isn't a cut, \textit{because} there is an augmenting path.

- cf. discussion of aug paths.

- a flow is \textit{only} a cut if there is no augmenting path, i.e. if its maximal.
In push-relabel, we have manipulating cuts (by them, above), and ONLY had a flow at optimality.

These two are dual

Max-flow, min-cut are DUAL

\[
\begin{align*}
\text{MAX FLOW} \\
\text{max} & \quad -\lambda^T C \\
\text{s.t.} & \quad -\omega + A^T e - \lambda^1 - \lambda^2 = 0 \\
& \quad \lambda^1 > 0 \\
& \quad \lambda^2 > 0
\end{align*}
\]

\[
\begin{align*}
\text{MIN CUT} \\
\text{min} & \quad -w^T f \\
\text{st.} & \quad A_R f = 0 \\
& \quad f \geq 0 \\
& \quad c - f \geq 0
\end{align*}
\]
These problems are coupled by complementarity conditions

\[
\begin{align*}
(\lambda_i^1) & \quad f_i = 0 \quad k = 1 \ldots \#E \\
(\lambda_i^2) & \quad \frac{1}{k} \left( c - f_i \right) \geq 0 \quad k = 1 \ldots \#E
\end{align*}
\]

(i.e. either the inequality is active or its Lagrange multiplier is 0)

Flow's
Now \[
(\lambda_i^1) = 0 \quad \text{implies that flow is } > 0
\]

\[
(\lambda_i^2) = 0 \quad " \quad < c
\]

- The non-zero \( \lambda_i \)'s identify edges that could disconnect (i.e. any path containing one is not augmenting)

- If the non-zero \( \lambda_i \)'s meet the equality, we have a cut
There is a general point here

consider \( (f, \lambda^1, \lambda^2, \lambda^e) \Rightarrow (p, d) \)

primal vars \( \Rightarrow \) dual vars.

1) at a solution if \( p \) is soln to Primal, \( d \) is soln to dual

\( p, d \) are complementary

(follows from KKT).

2) if \( (p, d) \) are complementary and

\( p \) is primal feasible and

\( d \) is dual feasible

\( \Leftrightarrow \)

\( p \) solves primal \( \Rightarrow \) \( d \) solves dual.
(This follows from KKT conditions).

Hence two kinds of strategy

primal-feasible, dual-infeasible (Aug path)

primal-infeasible, dual-feasible (Push relabel)