Flow network \((D, c, s, t)\)

directed graph \(D = (V, E)\); \(|V| = n, |E| = m\),
capacity \(c : V \times V \to \mathbb{IR}^+ \cup \{0\}\),
such that \(c(uv) = 0\) if \(uv \notin E\)
source \(s\), sink \(t\)

Notation: ordered pair \((u, v)\) is denoted by \(uv\).
Assumption: for every vertex \(v \in V\) there is an \(st\)-path through \(v\). Thus \(m \geq n - 1\).

\(f : V \times V \to \mathbb{IR}\) is a flow on the network \((D, c, s, t)\) if it satisfies

1. Capacity constraint:
   \[ f(uv) \leq c(uv) \text{ for every } u, v \in V \]
2. Skew symmetry:
   \[ f(uv) = -f(vu) \text{ for every } u, v \in V \]
3. Flow conservation:
   \[ \sum_{v \in V} f(uv) = 0 \text{ for every } u \in V \setminus \{s, t\} \]
The MaxFlow problem

value of flow \( f \): \( |f| := \sum_{v \in V} f(sv) \).

maximum flow of a network: a flow whose value is maximum over all flows of the network

The Problem
Given a flow network \((D, c, s, t)\), find a maximum flow.
Further notation and basic properties

\[ f(X, Y) := \sum_{x \in X} \sum_{y \in Y} f(xy). \]

**Examples:**

1. Flow conservation property:
   \[ f(\{u\}, V) = 0 \text{ for every } u \in V \setminus \{s, t\}. \]

2. Value of flow \( f \):
   \[ |f| = f(\{s\}, V) \]

**Lemma** Let \( f \) be a skew-symmetric function. Then

(i) \( f(X, X) = 0 \) for all \( X \subseteq V \)

(ii) \( f(X, Y) = -f(Y, X) \) for all \( X, Y \subseteq V \)

(iii) \( f(X \cup Y, Z) = f(X, Z) + f(Y, Z) \) and
      \[ f(Z, X \cup Y) = f(X, Z) + f(Y, Z) \]
      for all \( X, Y, Z \subseteq V \) with \( X \cap Y = \emptyset \)

**Claim** \( |f| = f(V, t) \)

**Proof:** Use Lemma and flow conservation.
Residual network

residual network \((D_f, c_f, s, t)\)

residual capacity \(c_f(\text{uv}) = c(\text{uv}) - f(\text{uv})\)

residual digraph \(D_f = (V, E_f)\)

where \(E_f = \{uv \in V \times V : c_f(\text{uv}) > 0\}\)

**Remark** Edges in \(E_f\) are either edges in \(E\) or their reversals. Hence \(|E_f| \leq 2m\).

**Lemma** Let \(f\) be a flow in the flow network \(D\) and let \(f'\) be a flow in the residual flow network \(D_f\).

Then \(f + f'\) is a flow in \(D\) with value \(|f| + |f'|\).
Augmenting paths

An $st$-path $P$ in $D_f$ is called an augmenting path.

residual capacity of $P$:

$$c_f(P) = \min\{c_f(uv) : uv \text{ is on } P\}$$

Define

$$f_P(uv) := \begin{cases} 
  c_f(P) & \text{if } uv \text{ is on } P \\
  -c_f(P) & \text{if } vu \text{ is on } P \\
  0 & \text{otherwise}
\end{cases}$$

Claim $f_P$ is a flow in $D_f$.

Corollary

$f + f_P$ is a flow in $D$ with value larger than $|f|$. 
Ford-Fulkerson Method

**Initialization** $f \equiv 0$

WHILE there exists an augmenting path $P$
    DO augment flow $f$ along $P$

return $f$

Running times:

- Basic (careless) Ford-Fulkerson: might not even terminate, flow value might not converge to maximum;
  when capacities are integers, it terminates in time $O(m |f^*|)$, where $f^*$ is a maximum flow.

- Edmonds-Karp: chooses a *shortest* augmenting path; runs in $O(nm^2)$
Example of Zwick (1995)

**Remark.** The max fbw is 199. There is such an unfortunate choice of a sequence of augmenting paths, by which the fbw value tends to 3. Even our implementation can be cheated to do this by introducing an extra vertex in the middle of each edge.
Cuts of flow networks

Cut $[S, T]$ of a flow network $(D, c, s, t)$ is a partition of $V$ into $S$ and $T$ such that $s \in S$ and $t \in T$

capacity of cut $[S, T]$: $c(S, T) = \sum_{x \in S} \sum_{y \in T} c(xy)$

minimum cut of a network: the one whose capacity is minimum over all cuts of the network.

Lemma $f(S, T) = |f|$ for any cut $[S, T]$.

Corollary For any flow $f$ and any cut $[S, T]$

$$|f| \leq c(S, T).$$

Max-flow/min-cut Theorem The following are equivalent.

(i) $f$ is a maximum flow

(ii) $D_f$ contains no augmenting paths.

(iii) $|f| = c(S, T')$ for some cut $[S, T]$. 
Preflow

$f$ is a preflow if the following hold.

1. Capacity constraint:
   \[ f(uv) \leq c(uv) \text{ for every } u, v \in V \]

2. Skew symmetry:
   \[ f(uv) = -f(vu) \text{ for every } u, v \in V \]

3. Relaxed “flow conservation”:
   \[ f(V, u) = \sum_{v \in V} f(uv) \geq 0 \text{ for every } u \in V \setminus \{s\} \]

excess flow into $u$: \[ e(u) := f(V, u) \]

vertex $u \in V \setminus \{s, t\}$ is overflowing if $e(u) > 0$

**Lemma** For any overflowing vertex $u$ there is a $us$-path in the residual network $D_f$.
In particular, there is a residual outgoing edge from $u$. 

9
Height function

$h : V \to \mathbb{N}$ is a height function for a preflow $f$ if

- $h(s) = n$,
- $h(t) = 0$,
- $h(u) \leq h(v) + 1$ for every residual edge $uv \in E_f$

**Lemma** If $f$ is a preflow which has a height function then there is no augmenting path in the residual network $D_f$.

**Corollary** If $f$ is a flow which has a height function, then $f$ is a maximum flow.

**Lemma** If $f$ is a preflow with a height function $h$, then for any overflowing vertex $u$ we have $h(u) \leq 2n - 1$. 
Initialization

Every flow network \((D, c, s, t)\) has a preflow with a height function:

\[
\text{INITIALIZE-PREFLOW}(D, c, s, t)
\]

FOR each pair \(uv \in V \times V\)

DO \(f(uv) := 0\)

FOR each vertex \(u \in N^+(s)\)

DO \(f(su) := c(su)\)

\(f(us) := -c(su)\)

FOR each vertex \(u \in V\)

DO \(h(u) := 0\)

\(h(s) := n\)

Claim \text{INITIALIZE-PREFLOW} outputs a preflow \(f\) of \((D, c, s, t)\) with a height function \(h\).

The \text{GENERIC-PUSH-RELABEL} algorithm maintains a preflow with a height function while performing a series of basic operations (\text{PUSHes} and \text{RELABELs}) and eventually outputing a flow with a height function.
The **PUSH** operation

**PUSH**(u, v) is applicable if

- u is overflowing,
- \( c_f(uv) > 0 \) (that is, \( uv \in E_f \)), and
- \( h(u) = h(v) + 1 \)

**Action:** \( d_f(uv) := \min\{e(u), c_f(uv)\} \) amount of flow is “pushed from u to v”:

\[
\begin{align*}
f(uv) & := f(uv) + d_f(uv) \\
f(vu) & := -f(uv)
\end{align*}
\]

**Remark:** Preflow changes, height function does not.

1. **saturating push:** if \( d_f(uv) = c_f(uv) \).

After a saturating push \( uv \) becomes “saturated”, i.e., \( c_f(uv) \) becomes 0.

2. **nonsaturating push:** if \( d_f(uv) = e(v) \).

After a nonsaturating push u is no longer overflowing.
The RELABEL operation

RELABEL\((u)\) applies if

- \(u\) is overflowing and
- \(h(u) \leq h(v)\) for all residual edges \(uv \in E_f\).

**Action:** Define new height for \(u\)
\[
h(u) := 1 + \min\{h(v) : uv \in E_f\}
\]

**Remark:** Minimum is well-defined, because \(u\) is overflowing, so there is an outgoing residual edge.

**Remark:** Height function changes, preflow does not.

**Remark:** \(s\) and \(t\) cannot be relabeled
The **GENERIC-PUSH-RELABEL Algorithm**

**INITIALIZE-PREFLOW** \((D, c, s, t)\)

WHILE there exists an applicable push or relabel operation

DO select an applicable push or relabel operation

and perform it
Correctness of the push-relabel method

**Theorem** If \texttt{GENERIC-PUSH-RELABEL} algorithm terminates then the preflow it computes is a maximum flow of the network $D$.

*Proof:*

**Lemma** If $u$ is an overflowing vertex then either a push or a relabel operation applies to it.

**Corollary** At termination $f$ is a flow.

**Lemma** $h$ is maintained as a height function.

*Proof:*

**Lemma** During execution $h(u)$ never decreases
Termination and running time analysis

Lemma (Bound on \texttt{RELABEL} operations) The number of relabel operations is at most $2n - 1$ per vertex and at most $2n^2$ overall.

Proof: Any time during execution $h(u) \leq 2n - 1$ for each vertex $u \in V$.

Lemma (Bound on saturating \texttt{PUSH}es) The number of saturating pushes is at most $2nm$.

Proof: Between two saturating pushes from $u$ to $v$ the height of $v$ increases by at least 2.

Lemma (Bound on non-saturating \texttt{PUSH}es) The number of non-saturating pushes is at most $4n^2(n + m)$.

Proof: Estimate the change of the potential function $\Phi = \sum_{v \in V, e(v) > 0} h(v)$ during the three basic operations.

Theorem The number of basic operations for the \texttt{GENERIC-PUSH-RELABEL} algorithm is at most $O(n^2m)$.

Corollary There is an implementation of the \texttt{GENERIC-PUSH-RELABEL} algorithm which runs in $O(n^2m)$ on any flow network.
Admissable edges and admissable digraph

$uv$ is an admissable edge if

\begin{itemize}
  \item $c_f(uv) > 0$ and
  \item $h(u) = h(v) + 1$
\end{itemize}

Admissable digraph: $D_{f,h} = (V, E_{f,h})$, where $E_{f,h}$ is the set of admissable edges.

**Lemma** The admissable digraph is acyclic.

**Observation** If $u$ is overflowing and $uv$ is an admissable edge then $\text{PUSH}(u, v)$ applies. $\text{PUSH}(u, v)$ does not create any new admissable edges, but it may cause $uv$ to become inadmissable.

**Observation** If $u$ is overflowing and there are no admissable edges leaving $u$, then $\text{RELABEL}(u)$ applies. After $\text{RELABEL}(u)$ there is at least one admissable edge leaving $u$ and there are no admissable edges entering $u$. 

17
Notation

$D$ is given by (non-cyclic) neighbor lists:

- $N(u)$ is the neighbor list of $u$
- $v$ is on $N(u)$ if $uv$ or $vu \in E$

- $\text{head}(N(u))$ points to the first vertex in $N(u)$
- $\text{next-neighbor}(v)$ points to the vertex following $v$ in $N(u)$
- $\text{next-neighbor}(v) = \text{NIL}$ if $v$ is the last vertex of $N(u)$
- $\text{current}(u)$ points to the neighbor $u$ currently under consideration. Initially $\text{current}(u)$ points to $\text{head}(N(u))$. 
Recall – Basic operations

**PUSH** \((u, v)\) applies if

\(u\) is overflowing and \(uv \in E_{f,h}\). Then

\[
d_f(\uv) := \min\{e(u), c_f(\uv)\}
\]
\[
f(\uv) := f(\uv) + d_f(\uv)
\]
\[
f(\vu) := -f(\uv)
\]
\[
e(u) := e(u) - d_f(\uv)
\]
\[
e(v) := e(v) + d_f(\uv)
\]

**RELABEL** \((u)\) applies if \(u\) is overflowing and \(uv \not\in E_{f,h}\) for all \(v \in V\). Then

\[
h(u) := 1 + \min\{h(v) : uv \in E_f\}
\]
Discharging a vertex

**DISCHARGE**($u$)

1. WHILE $e(u) > 0$
2. DO $v := \text{current}(u)$
3. IF $v = \text{NIL}$
   4. THEN RELABEL($u$)
   5. $\text{current}(u) := \text{head}(N(u))$
4. ELSEIF $c_f(u, v) > 0$ and $h(u) = h(v) + 1$
5. THEN PUSH($u, v$)
6. ELSE $\text{current}(u) := \text{next-neighbor}(v)$

**Lemma** (Algorithm **DISCHARGE** is well-defined)
When **DISCHARGE** calls **PUSH**($u, v$) then a push operation applies to $uv$.
When **DISCHARGE** calls **RELABEL**($u$) then a relabel operation applies to $u$. 
The RELABEL-TO-FRONT($D, c, s, t$) algorithm

1. $f := \text{INITIALIZE-PREFLOW}(D, c, s, t)$
2. $L := \text{any order of } V \setminus \{s, t\}$
3. FOR each $u \in V \setminus \{s, t\}$ DO current($u$) := head($N(u)$)
4. $u := \text{head}(L)$
5. WHILE $u \neq \text{NIL}$ DO old-height := $h(u)$
6. DISCHARGE($u$)
7. IF $h(u) > \text{old-height}$ THEN move $u$ to the front of the list $L$
8. $u := \text{next}(u)$

**Theorem** (Correctness)
The RELABEL-TO-FRONT algorithm is an implementation of the GENERIC-PUSH-RELABEL algorithm.

**Proof.** At each test in line 6 of the algorithm the list $L$ is a topological sort of the vertices of the admissable digraph $D_{f,h}$ and no vertex in the list before $u$ has excess flow.
Running time analysis

**Theorem** The running time of RELABEL-TO-FRONT on any flow network is $O(n^3)$

*Proof*

“Phase”: time between two relabel operations.

There are at most $O(n^2)$ relabel operations. Hence there are at most $O(n^2)$ phases.

Each phase consists of $\leq n$ calls to DISCHARGE. Hence the total time of the WHILE loop excluding the work DISCHARGE does is $O(n^3)$. 
Time spent during DISCHARGE

The $O(n^2)$ relabel operations can be done in $O(nm)$ (Homework)

Updating $\text{current}(u)$ in line 8 of DISCHARGE occurs $O(\text{deg}(u))$ times each time a vertex $u$ is relabeled and $O(n \text{deg}(u))$ times over for the vertex. All together the time is $O(nm)$ (Handshaking Lemma).

The overall number of saturating pushes is $O(nm)$

There is at most one non-saturating push per call to DISCHARGE

Hence the number of nonsaturating pushes is at most $O(n^3)$. 

23