Algorithms and Data Structures: Network Flows
Flow Networks

Definition 1
A flow network consists of

- A directed graph $G = (V, E)$.
- A capacity function $c : V \times V \rightarrow \mathbb{R}$ such that $c(u, v) \geq 0$ if $(u, v) \in E$ and $c(u, v) = 0$ for all $(u, v) \notin E$.
- Two distinguished vertices $s, t \in V$ called the source and the sink, respectively.

We read $(u, v)$ to mean $u \rightarrow v$.

Assumption
Each vertex $v \in V$ is on some directed path from $s$ to $t$. This implies that $G$ is connected (but not necessarily strongly connected), and that $|E| \geq |V| - 1$. 

ADS: lectures 10 & 11 – slide 2 –
For this graph, $V = \{s, r, u, v, w, x, y, z, t\}$. The edge set is

$$E = \{(s, u), (s, r), (s, x), (u, v), (u, x), (v, x), (v, w), (r, w), (r, y), (x, y), (y, r), (y, z), (z, w), (z, t), (w, t)\}.$$

Some examples of capacities are $c(s, x) = 10$, $c(r, y) = 5$, $c(v, x) = 20$ and $c(v, r) = 0$ (since there is no arc from $v$ to $r$).
Network Flows

**Definition 2**

Let \( N = (G = (V, E), c, s, t) \) be a flow network.

A flow in \( N \) is a function

\[
f : V \times V \rightarrow \mathbb{R}
\]

satisfying the following conditions:

- **Capacity constraint:** \( f(u, v) \leq c(u, v) \) for all \( u, v \in V \).
- **Skew symmetry:** \( f(u, v) = -f(v, u) \) for all \( u, v \in V \).
- **Flow conservation:** For all \( u \in V \setminus \{s, t\} \),

\[
\sum_{v \in V} f(u, v) = 0.
\]
Network Flows (cont’d)

\[ \mathcal{N} = (G = (V, E), c, s, t) \] flow network, \( f : V \times V \rightarrow \mathbb{R} \) flow in \( \mathcal{N} \).

- For \( u, v \in V \) we call \( f(u, v) \) the net flow at \((u, v)\).
- The value of the flow \( f \) is the number

\[ |f| = \sum_{v \in V} f(s, v). \]

Notice that our particular defn. of flow (the “skew-symmetry” constraint) ensures that \( f(u, v) \) is truly the “net flow” in the usual sense of the word (e.g. if \((r, y)\) on slide 2 was to carry flow 3, and \((y, r)\) to carry flow 4, we will have \( f(r, y) = -1 \)).
Example

A flow of value 18.

Only positive net flows are shown.
The Maximum-Flow Problem

Input: Network $\mathcal{N}$
Output: Flow of maximum value in $\mathcal{N}$

The problem is to find the flow $f$ such that $|f| = \sum_{v \in V} f(s, v)$ is the largest possible (over all “legal” flows).
The Ford-Fulkerson Algorithm

Published in 1956 by Delbert Fulkerson and Lester Randolph Ford Jr.

**Algorithm** \( \text{FORD-FULKERSON}(\mathcal{N}) \)

1. \( f \leftarrow \text{flow of value 0} \)
2. \( \textbf{while} \) there exists an \( s \rightarrow t \) path \( \mathcal{P} \) in the “residual network” \( \textbf{do} \)
3. \( f \leftarrow f + f_\mathcal{P} ; \)
4. \( \text{Update the “residual network”} . \)
5. \( \textbf{return} \ f \)

The “residual network” is \( \mathcal{N} \) with the “used-up” capacity removed.

To make this precise, we need notation, and proofs - this lecture.
Some Technical Observations

\[ \mathcal{N} = (\mathcal{G} = (V, E), c, s, t) \] flow network, \( f : V \times V \to \mathbb{R} \) flow in \( \mathcal{N} \), \( u, v \in V \).

1. \( f(u, u) = 0 \) for all \( u \in V \).
   
   "Proof": \( f(u, u) = -f(u, u) \) by skew symmetry.

2. For any \( v \in V \setminus \{s, t\} \),
   \[
   \sum_{u \in V} f(u, v) = 0.
   \]

   Proof: \( \sum_{u \in V} f(u, v) = -\sum_{u \in V} f(v, u) = 0 \) by skew symmetry and flow conservation.

3. If \( (u, v) \notin E \) and \( (v, u) \notin E \) then \( f(u, v) = f(v, u) = 0 \).

   Proof: Either \( f(u, v) \) or \( f(v, u) \geq 0 \) by skew symmetry. Say, \( f(u, v) \geq 0 \).
   Then \( 0 \leq f(u, v) \leq c(u, v) = 0 \) by the capacity constraint. So \( f(u, v) = 0 \).
   By skew symmetry, this shows \( f(v, u) = 0 \).
One More Technical Observation

4. The *positive net flow entering* \( v \) is:

\[
\sum_{u \in V} f(u, v) \cdot \frac{f(u, v)}{f(u, v) > 0}
\]

The *positive net flow leaving* \( v \) is defined symmetrically.

Flow conservation now says:

“positive net flow in = positive net flow out”.

All these observations are just to make it easy for us to talk about flows.
Working with Flows

*Implicit summation notation:* For $X, Y \subseteq V$ put

$$f(X, Y) = \sum_{u \in X} \sum_{v \in Y} f(u, v) = \sum_{(u, v) \in X \times Y} f(u, v).$$

*Abbreviations:*

- $f(u, Y)$ stands for $f(\{u\}, Y)$ and
- $f(X, v)$ stands for $f(X, \{v\})$.

Conservation of flow is now:

$$f(u, V) = 0 \text{ for all } u \in V \setminus \{s, t\}.$$
Lemma 3

\( \mathcal{N} = (G = (V, E), c, s, t) \) flow network, \( f \) flow in \( \mathcal{N} \).

Then for all \( X, Y, Z \subseteq V \),

1. \( f(X, X) = 0 \).
2. \( f(X, Y) = -f(Y, X) \).
3. If \( X \cap Y = \emptyset \) then

\[
\begin{align*}
  f(X \cup Y, Z) &= f(X, Z) + f(Y, Z), \\
  f(Z, X \cup Y) &= f(Z, X) + f(Z, Y).
\end{align*}
\]

Lemma “lifts” Network flow properties to sets-of-vertices.
Proof of Lemma 3

1. \[ f(X, X) = \sum_{(u,v) \in X \times X} f(u, v) \quad \text{by defn. of } f(X, X) \]
   \[ = \sum_{\{u,v\} \subseteq X} (f(u, v) + f(v, u)) \quad \text{take } (u, v), (v, u) \text{ together} \]
   \[ = 0. \quad \text{by skew-symm} \]

2. \[ f(X, Y) = \sum_{(u,v) \in X \times Y} f(u, v) \quad \text{by defn of } f(X, Y) \]
   \[ = \sum_{(u,v) \in X \times Y} -f(v, u) \quad \text{by skew-symmetry} \]
   \[ = - \sum_{(v,u) \in Y \times X} f(v, u) \quad \text{take } - \text{ outside the summation} \]
   \[ = -f(Y, X). \quad \text{by defn of } f(Y, X) \]
Proof of Lemma 3 (cont’d)

3. 

\[ f(X \cup Y, Z) = \sum_{u \in X \cup Y} \sum_{v \in Z} f(u, v) \]
\[ = \sum_{u \in X} \sum_{v \in Z} f(u, v) + \sum_{u \in Y} \sum_{v \in Z} f(u, v) - \sum_{u \in X \cap Y} \sum_{v \in Z} f(u, v) \]

(expand sum into X and Y, subtract duplicates in X \cap Y)

\[ = \sum_{u \in X} \sum_{v \in Z} f(u, v) + \sum_{u \in Y} \sum_{v \in Z} f(u, v) \]

(but X \cap Y = \emptyset, so third term disappears)

\[ = f(X, Z) + f(Y, Z). \]

Moreover,

\[ f(Z, X \cup Y) = -f(X \cup Y, Z) = -(f(X, Z) + f(Y, Z)) = f(Z, X) + f(Z, Y). \]
Corollary 4

\( \mathcal{N} = (G = (V, E), c, s, t) \) flow network, \( f \) flow in \( \mathcal{N} \). Then

\[ |f| = f(V, t). \]

Proof:

\[
\begin{align*}
|f| &= f(s, V) \quad \text{(by definition)} \\
    &= f(V, V) - f(V \setminus \{s\}, V) \quad \text{(by Lemma 3 (3.))} \\
    &= -f(V \setminus \{s\}, V) \quad \text{(by Lemma 3 (1.))} \\
    &= f(V, V \setminus \{s\}) \quad \text{(by Lemma 3 (2.))} \\
    &= f(V, t) + f(V, V \setminus \{s, t\}) \quad \text{(by Lemma 3 (3.))} \\
    &= f(V, t) + \sum_{v \in V \setminus \{s, t\}} f(V, v) \quad \text{(by Definition)} \\
    &= f(V, t) \quad \text{(by flow conservation)}
\end{align*}
\]
Residual Networks

Idea is to capture possible extra flow given current flow.

**Definition 5**

\[ \mathcal{N} = (\mathcal{G} = (V, E), c, s, t) \] is a flow network, \( f \) flow in \( \mathcal{N} \).

1. For all \( u, v \in V \times V \), the *residual capacity* of \( (u, v) \) is

\[ c_f(u, v) = c(u, v) - f(u, v). \]

2. The *residual network* of \( \mathcal{N} \) induced by \( f \) is

\[ \mathcal{N}_f((V, E_f), c_f, s, t), \]

where

\[ E_f = \{(u, v) \in V \times V \mid c_f(u, v) > 0\} \]

Notice that \( E_f \) may contain edges not originally in \( E \) ("back-edges").
Example

A flow and the corresponding residual network
Lemma 6

Let $\mathcal{N} = (G = (V, E), c, s, t)$ be a flow network.
Let $f$ be a flow in $\mathcal{N}$.
Let $g : V \times V \to \mathbb{R}$ be a flow in the residual network $\mathcal{N}_f$.
Then the function $f + g : V \times V \to \mathbb{R}$ defined by

$$(f + g)(u, v) = f(u, v) + g(u, v)$$

is a flow of value $|f| + |g|$ in $\mathcal{N}$. 
Proof of Lemma 6

First we have to check that \( f + g \) is actually a flow in \( N \).

**Capacity constraints:**

\[
(f + g)(u, v) = f(u, v) + g(u, v) \\
\leq f(u, v) + c_f(u, v) \\
= f(u, v) + c(u, v) - f(u, v) \\
= c(u, v).
\]

**Skew symmetry:**

\[
(f + g)(u, v) = f(u, v) + g(u, v) = -f(v, u) - g(v, u) = -(f + g)(v, u).
\]

**Flow Conservation:** For every \( u \in V \setminus \{s, t\} \):

\[
\sum_{v \in V} (f + g)(u, v) = \sum_{v \in V} f(u, v) + \sum_{v \in V} g(u, v) = 0 + 0 = 0.
\]
Proof of Lemma 6 (cont’d)

Next we have to check that $f + g$ does have the value that we claimed for it.

**Value:**

$$|f + g| = \sum_{v \in V} (f + g)(s, v)$$

$$= \sum_{v \in V} f(s, v) + \sum_{v \in V} g(s, v)$$

$$= |f| + |g|.$$
Augmenting Paths

Definition 7

\[ \mathcal{N} = (\mathcal{G} = (V, E), c, s, t) \] flow network, \( f \) flow in \( \mathcal{N} \).

Then an **augmenting path** for \( f \) is a path \( \mathcal{P} \) from \( s \) to \( t \) in the residual network \( \mathcal{N}_f \).

The **residual capacity** of \( \mathcal{P} \) is

\[ c_f(\mathcal{P}) = \min \{ c_f(u, v) \mid (u, v) \text{ edge on } \mathcal{P} \}. \]

Note that \( c_f(\mathcal{P}) > 0 \), by definition of \( E_f \) (recall that we only keep edges in \( E_f \) if their residual capacity is strictly positive).
An augmenting path of residual capacity 10
Pushing Flow through an Augmenting Path

Lemma 8

\( \mathcal{N} = (\mathcal{G} = (V, E), c, s, t) \) flow network, \( f \) flow in \( \mathcal{N} \).

\( \mathcal{P} \) augmenting path. Then \( f_{\mathcal{P}} : V \times V \rightarrow \mathbb{R} \) defined by

\[
f_{\mathcal{P}}(u, v) = \begin{cases} 
c_f(\mathcal{P}) & \text{if } (u, v) \text{ is an edge of } \mathcal{P}, \\
-c_f(\mathcal{P}) & \text{if } (v, u) \text{ is an edge of } \mathcal{P}, \\
0 & \text{otherwise}
\end{cases}
\]

is a flow in \( \mathcal{N}_f \) of value \( c_f(\mathcal{P}) \).

Proof left as an exercise. It is not too difficult - just have to check that the three conditions of a flow are satisfied (and that the value is \( c_f(\mathcal{P}) \)). Similar to Lemma 6.

ADS: lectures 10 & 11 – slide 23 –
Corollary 9

\( \mathcal{N} = (\mathcal{G} = (V, E), c, s, t) \) flow network, \( f \) flow in \( \mathcal{N} \). Let \( \mathcal{P} \) be an augmenting path. Then \( f + f_\mathcal{P} \) is a flow in \( \mathcal{N} \) of value

\[ |f| + c_f(\mathcal{P}) > |f|. \]

Proof: Follows from Lemma 6 and Lemma 8.
The Ford-Fulkerson Algorithm

Algorithm $\text{FORD-FULKERSON}(\mathcal{N})$

1. $f \leftarrow$ flow of value 0
2. while there exists an augmenting path $P$ in $\mathcal{N}_f$ do
3. \hspace{1em} $f \leftarrow f + f_P$
4. return $f$

To prove that $\text{FORD-FULKERSON}$ correctly solves the Maximum Flow problem, we have to prove that:

1. The algorithm terminates.
2. After termination, $f$ is a maximum flow.
Cuts

Definition 10

$\mathcal{N} = (G = (V, E), c, s, t)$ flow network.

A cut of $\mathcal{N}$ is a pair $(S, T)$ such that:

1. $s \in S$ and $t \in T$,
2. $V = S \cup T$ and $S \cap T = \emptyset$.

The capacity of the cut $(S, T)$ is

$$c(S, T) = \sum_{u \in S, v \in T} c(u, v).$$
Example

A cut of capacity 45.
Example

A cut of capacity 25.
Lemma 11

\( \mathcal{N} = (G = (V, E), c, s, t) \) flow network, \( f \) flow in \( \mathcal{N} \), \( (S, T) \) cut of \( \mathcal{N} \). Then

\[ |f| = f(S, T). \]

Proof: We apply Lemma 3:

\[
|f| &= f(s, V) \\
&= f(s, V) + f(S - \{s\}, V) \quad [t \not\in S \Rightarrow f(S - \{s\}, V) = 0] \\
&= f(S, V) \\
&= f(S, T) + f(S, S) \\
&= f(S, T).
\]
Corollary 12

The value of any flow in a network is bounded from above by the capacity of any cut.

Proof: Let $f$ be a flow and $(S, T)$ a cut. Then

$$|f| = f(S, T) \leq c(S, T).$$
The Max-Flow Min-Cut Theorem

Theorem 13

Let $\mathcal{N} = (\mathcal{G} = (V, E), c, s, t)$ be a flow network.

Then the maximum value of a flow in $\mathcal{N}$ is equal to the minimum capacity of a cut in $\mathcal{N}$. 
Proof of the Max-Flow Min-Cut Theorem

Let $f$ be a flow of maximum value and $(S, T)$ a cut of minimum capacity in $\mathcal{N}$. We shall prove that

$$|f| = c(S, T).$$

1. $|f| \leq c(S, T)$ follows from Corollary 12. So all we have to prove is that there is a cut $(S, T)$ such that

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

2. First remember that $|f|$ has no augmenting path. 

Proof: If $P$ was an augmenting path, then $f + f_P$ would be a flow of larger value (because by definition of $\mathcal{N}_f$, all edges in $\mathcal{N}_f$ have strictly positive weights).

3. Thus there is no path from $s$ to $t$ in $\mathcal{N}_f$. Let

$$S = \{v \mid \text{there is a path from } s \text{ to } v \text{ in } \mathcal{N}_f\}$$

and $T = V \setminus S$. Then $(S, T)$ is a cut.
4. By definition of $S$, and because reachability in graphs is a transitive relation, there cannot be any edge from $S$ to $T$ in $\mathcal{N}_f$. Thus for all $u \in S$, $v \in T$ we have $c(u, v) - f(u, v) = 0$.

5. Thus

$$c(S, T) = \sum_{u \in S} \sum_{v \in T} c(u, v) = \sum_{u \in S} \sum_{v \in T} f(u, v) = f(S, T) = |f|$$

(by Lemma 11).
Corollaries

Corollary 14
A flow is maximum if, and only if, it has no augmenting path.

Proof: This follows from the proof of the Max-Flow Min-Cut theorem.

Corollary 15
If the Ford-Fulkerson algorithm terminates, then it returns a maximum flow.

Proof: The flow returned by Ford-Fulkerson has no augmenting path.
Termination

Let $f^*$ be a maximum flow in a network $\mathcal{N}$.

- If all capacities are integers, then \textsc{Ford-Fulkerson} stops after at most
  
  $$|f^*|$$

  iterations of the main loop.

- If all capacities are rationals, then \textsc{Ford-Fulkerson} stops after at most
  
  $$q \cdot |f^*|$$

  iterations of the main loop, where $q$ is the least common multiple of the denominators of all the capacities.

- For arbitrary real capacities, it may happen that \textsc{Ford-Fulkerson} does not stop.
A Nasty Example

ADS: lectures 10 & 11 – slide 36 –
The Edmonds-Karp Heuristic

Idea
Always choose a shortest augmenting path.

$n$ number of vertices, $m$ number of edges. Recall that $n \leq m + 1$
A shortest augmenting path can be found by Breadth-First-Search (reading assignment) in time $O(n + m) = O(m)$.

Theorem 16
The Ford-Fulkerson algorithm with the Edmonds-Karp heuristic stops after at most $O(nm)$ iterations of the main loop.
Thus the running time is $O(nm^2)$. 

ADS: lectures 10 & 11 – slide 37 –
We will run Ford-Fulkerson (with the Edmonds-Karp heuristic) on this network. This is interesting because we will see the “back-edges” being used to “undo” part of an previous augmenting path.
1st augmenting path: $s \rightarrow r \rightarrow w \rightarrow t$.

Length is 3 (so we satisfy Edmonds-Karp rule to take a shortest possible path). Min capacity is 10, so we push flow of 10 along the path. Starting flow becomes 10.
Residual network after adding first flow of value 10 along $s \rightarrow r \rightarrow w \rightarrow t$.

The newly-created “back-edges” are shown in red.
There is no longer any augmenting path of length \( \leq 3 \), and the only one of length 4 is \( s \rightarrow x \rightarrow y \rightarrow z \rightarrow t \), which has a minimum capacity \( \min\{10, 10, 15, 15\} \), ie 10.

We push this extra flow of value 10 along \( s \rightarrow x \rightarrow y \rightarrow z \rightarrow t \), bringing overall flow to 20.
Residual network after adding flow from second augmenting path $s \rightarrow x \rightarrow y \rightarrow z \rightarrow t$, overall flow now 20.
Now there is only one simple augmenting path - $s \rightarrow u \rightarrow v \rightarrow w \rightarrow r \rightarrow y \rightarrow z \rightarrow t$, with minimum residual capacity 5.

Notice we use the “back-edge” $w \rightarrow r$ in our path. This is essentially “re-shipping” 5 units from the first flow-path away from $r \rightarrow w \rightarrow t$ and along $r \rightarrow y \rightarrow z \rightarrow t$ instead.
Residual network after adding 3rd flow, of value 5 \( \Rightarrow \) total flow 25.

There is no longer any augmenting path in our residual network (set of vertices “reachable” from \( s \) is \( \{s, u, v, x, w, r\} \)).
Reading and Problems

[CLRS] Chapter 26
For breadth-first search: [CLRS], Section 22.2.

Problems

   
   Not in [CLRS] (ed 3). Question is: consider Figure 26.1(b) and find a pair of subsets $X, Y \subseteq V$ such that $f(X, Y) = -f(V \setminus X, Y)$. After that, find a pair of subsets $X', Y' \subseteq V$ for which $f(X', Y') \neq -f(V \setminus X', Y')$.


4. Problem 26-4 of [CLRS].