Maximum Flow
Flow Network

- The following figure shows an example of a flow network:

- A flow network $G = (V, E)$ is a directed graph. Each edge $(u, v) \in E$ has a nonnegative capacity $c(u, v) \geq 0$. $c(u, v)$ is possibly not equal to $c(v, u)$. By convention, we say $c(u, v) = 0$ if $(u, v) \notin E$.

- There is one source vertex and one sink vertex in a flow network. We denote them by $s$ and $t$, respectively.
• We want to find a “flow” with maximum value that flows from the source to the target.

• Maximum Flow is a very practical problem.

• Many computational problems can be reduced to a Maximum Flow problem.
A Flow

• For any vertex $v$, we assume that there is a path from $s$ to $v$ and a path from $v$ to $t$.

• A flow in $G$ is a function $f : V \times V \to \mathbb{R}$ that specifies the direct flow value between every two nodes.

• $f$ should satisfy the following three properties before it can be called as a flow.

  • **Capacity constraint:** For all $u, v \in V$, $f(u, v) \leq c(u, v)$.
  • **Skew symmetry:** For all $u, v \in V$, $f(u, v) = -f(v, u)$.
  • **Flow conservation:** For all $u \in V - \{s, t\}$, $\sum_{v \in V} f(u, v) = 0$. 
If \((u, v) \notin E\) and \((v, u) \notin E\), then \(c(u, v) = c(v, u) = 0\).

By capacity constraint, \(f(u, v) \leq 0\) and \(f(v, u) \leq 0\).

By skey symmetry, \(f(u, v) \geq 0\) and \(f(v, u) \geq 0\).

Therefore \(f(u, v) = f(v, u) = 0\).

If there is no edge between \(u\) and \(v\), then there is no flow between \(u\) and \(v\).

**Note:** Our definition of a flow is slightly different from the one in the textbook.
• The value of the flow $f$, denoted by $|f|$, is defined by

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

• $|f|$ is the total flow out of the source.

Lemma 1.

$$|f| = \sum_{u \in V} f(u, t).$$

That is, the flow out of the source is equal to the flow into the sink.

Proof.

(1) $\sum_{u \in V} \sum_{v \in V} f(u, v) = 0.$ (Skew symmetry)

(2) $\sum_{u \in V \setminus \{s, t\}} \sum_{v \in V} f(u, v) = 0.$ (Flow conservation)

(3) $\sum_{u \in \{s, t\}} \sum_{v \in V} f(u, v) = 0.$

(4) $\sum_{v \in V} f(s, v) = -\sum_{v \in V} f(t, v) = \sum_{v \in V} f(v, t).$
The Ford-Fulkerson method is the standard method for solving a maximum-flow problem. The idea of the method is “iterative improvement”. We start with an arbitrary flow. Then we check whether an improvement is possible. Suppose we start with an empty flow. The improvement is a path from the source to the sink. What if the current flow is not empty?
• We need to examine the “residual capacity” for each edge.
• We check whether there is a path \( s \rightarrow t \) such that all edges on the path have a positive “residual capacity”.
• If so, we increase the flow. If not, we have got a maximal solution.
• Given a flow network \( G \). Let \( f \) be a flow. The residual capacity of \( (u, v) \) is given by \( c_f(u, v) = c(u, v) - f(u, v) \).
• The residual network induced by \( f \) is \( G_f = (V, E_f) \), where \( E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\} \).
• If there is a path from \( s \) to \( t \) in the residual network, then there is room to improve the current flow.
A flow in a flow network and its residual network.
• Note that if both \((u, v)\) and \((v, u)\) are not in the original flow network \(G\), neither \((u, v)\) nor \((v, u)\) can appear in the residual network. Therefore, \(|E_f| \leq 2|E|\).

• Let \(f'\) be a flow in the residual network \(G_f\). We can define a new flow \((f + f')\) in \(G\), as follows

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

• Lemma 2. \(f + f'\) is a flow in \(G\).

Proof.

We need to verify the three constraints:

1. Capacity constraint: \((f + f')(u, v) \leq c(u, v)\).
   Since \(f'(u, v) \leq c_f(u, v) = c(u, v) - f(u, v)\).

2. Skew symmetry: \((f + f')(u, v) = -(f + f')(v, u)\).
   Since \(f(u, v) = -f(v, u)\) and \(f'(u, v) = -f'(v, u)\).

3. Flow conservation: For all \(u \in V - \{s, t\}\), \(\sum_{v \in V} (f + f')(u, v) = 0\).
   Since for all \(u \in V - \{s, t\}\), \(\sum_{v \in V} f(u, v) = 0\) and \(\sum_{v \in V} f'(u, v) = 0\).
**Lemma 3.** The value of the new flow $f + f'$ is equal to total values of $f$ and $f'$. I.e., $|f + f'| = |f| + |f'|$.

**Proof.**

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

- Given a flow network $G = (V, E)$ and a flow $f$ in $G$, an augmenting path is a simple path from $s$ to $t$ in the residual graph $G_f$.
- An augmenting path admits some additional positive flow for each edge on the path.
- The residual capacity of an augmenting path $p$ is defined as
  \[ c_f(p) = \min\{c_f(u, v) : (u, v) \text{ is in } p\} \]
- $c_f(p)$ is the maximum amount of additional flow we can increase through path $p$.

**Lemma 4.** Let $G = (V, E)$ be a flow network, let $f$ be a flow in $G$, and let $p$ be an augmenting path in $G_f$. Define a function $f_p : V \times V \rightarrow \mathbb{R}$ by

\[
f_p(u, v) = \begin{cases} 
  c_f(p) & \text{if } (u, v) \text{ is on } p, \\
  -c_f(p) & \text{if } (v, u) \text{ is on } p, \\
  0 & \text{otherwise.}
\end{cases}
\]

Then, $f_p$ is a flow in $G_f$ with value $|f_p| = c_f(p) > 0$.  

A flow in a flow network and its residual network.

A new flow from the augmenting path and its residual network.
The basic Ford-Fulkerson algorithm

- Ford-Fulkerson($G,s,t$)
  1. for each edge $(u, v) \in E$
  2. $f[u, v] \leftarrow 0, f[v, u] \leftarrow 0.$
  3. while there exists a path $p$ from $s$ to $t$ in the residual network $G_f$
  4. $c_f(p) \leftarrow \min\{c_f(u, v) : (u, v) \text{ is in } p\}.$
  5. for each edge $(u, v)$ in $p$
  6. $f[u, v] \leftarrow f[u, v] + c_f(p)$
  7. $f[v, u] \leftarrow -f[u, v]$

- The path $p$ from $s$ to $t$ in the residual network $G_f$ is called the augmenting path.
- The augmenting path $p$ defines a flow in $G_f$. By adding this flow $f_p$ to the current flow $f$, we get a better flow $f + f_p$ with value $|f| + |f_p|$.
- Figure 26.6 on p.726-627 of the textbook shows an example.
Is the solution optimal?

• We have found an intuitive algorithm to provide a maximal flow. But is this flow maximum?

• Although we cannot increase the current flow by augmenting paths, is it possible that we find a completely different flow which has a better value?

• It turns out that the solution found by the Ford-Fulkerson algorithm is the maximum one.

• But we want to prove it.
Working with flows

• Let \( f \) be a flow. The flow from one set of vertices, \( X \), to another set \( Y \), is defined by
  \[
  f(X, Y) = \sum_{x \in X} \sum_{y \in Y} f(x, y).
  \]

• **Lemma 5.** Let \( G = (V, E) \) be a flow network and let \( f \) be a flow on \( G \), then:
  1. For all \( X \subseteq V \), \( f(X, X) = 0 \).
  2. For all \( X, Y \subseteq V \), \( f(X, Y) = -f(Y, X) \).
  3. For all \( X, Y, Z \subseteq V \) with \( X \cap Y = \emptyset \),
     \( f(X \cup Y, Z) = f(X, Z) + f(Y, Z) \) and
     \( f(Z, X \cup Y) = f(Z, X) + f(Z, Y) \).

• **Proof.**
Cuts of flow networks

- A cut \((S, T)\) in the flow network \(G = (V, E)\) is a partition of \(V\) into \(S\) and \(T = V - S\) such that \(s \in S\) and \(t \in T\).
- The net flow across the cut \((S, T)\) is defined to be
  \[
  f(S, T) = \sum_{u \in S} \sum_{v \in T} f(u, v).
  \]
- The capacity of the cut \((S, T)\) is defined to be
  \[
  c(S, T) = \sum_{u \in S} \sum_{v \in T} c(u, v).
  \]
- Obviously, \(f(S, T) \leq c(S, T)\).
Lemma 6. Let $f$ be a flow in flow network $G$. Let $(S, T)$ be any cut of $G$. Then the net flow across $(S, T)$ is $f(S, T) = |f|$. 

Proof.
By flow conservation, we have $f(S - \{s\}, V) = 0$.
Also, $f(S, V) = f(S, S) + f(S, T) = f(S, T)$. Therefore, $f(S, T) = f(S, V) = f(S - \{s\}, V) + f(\{s\}, V) = f(\{s\}, V) = |f|$. 

Therefore, the maximum flow is bounded by the capacity of the “minimum” cut.
**Theorem 1.** If \( f \) is a flow in a flow network \( G = (V, E) \) with source \( s \) and sink \( t \), then the following conditions are equivalent:

1. \( f \) is a maximum flow in \( G \).
2. The residual network \( G_f \) contains no augmenting paths.

**Proof.** (1) \( \Rightarrow \) (2): Obvious, because the existence of augmenting paths means a better flow exists.

(2) \( \Rightarrow \) (1): \( G_f \) has no path from \( s \) to \( t \). Let \( S \) be all the vertices that can be reached from \( s \), and \( T = V - S \). Then \( (S, T) \) is a cut.

For each \( u \in S \) and \( v \in T \), \( f(u, v) = c(u, v) \). Therefore, \( f(S, T) = c(S, T) \). But we know that \( f^*(S, T) \leq c(S, T) \) for any flow \( f^* \). Hence we conclude that \( f \) is the maximum.

**Exercise:** Read the proof of Theorem 26.6 at p.723 of the textbook. The proof there is essentially the same but in a different form.

**Corollary 1.** The Ford-Fulkerson algorithm gives the maximum flow of a flow network.
Complexity

- Assuming that the capacities are integers.
- Every augmenting path will increase the flow by at least 1. So, the while loop will be repeated $O(|f^*|)$ time, where $f^*$ is the maximum flow.
- The time complexity is $O(|E| \times |f^*|)$.
- Figure 26.7 on p.728 of textbook shows a worst case example.
Edmonds-Karp algorithm

- The Edmonds-Karp algorithm is almost the same as the Ford-Fulkerson algorithm.
- The difference is that we find the shortest path (in terms of number of edges) from $s$ to $t$ in the residual graph, and use the shortest path as the augmenting path.
- The worst case running time is reduced to $O(|V| \times |E|^2)$.
- Proof is omitted. See p.729 of textbook if you are interested to know.
Applications

- The maximum-bipartite-matching problem.
  Example: $m$ boys and $n$ girls are attending a dance party. Some of them can be matched. Find a solution so that you have maximum number of matches.

- The multiple-source max-flow problem.
  Example: A supermarket has several vendors for the same merchandise. It wants to transport the maximum number of merchandise to the market through its own transportation network.

- The multiple-sink max-flow problem.
  Example: A factory wants to send the maximum number of products to several countries through its own transportation network.

- The multiple-source multiple-sink max-flow problem.

- Maximum bipartite matching.

- Many other applications.