SECTION 6.3

Max-flow min-cut
Fig. 7 — Traffic pattern: entire network available

Legend:
- — international boundary
- Railway operating division
- Capacity: 12 each way per day.
- Required flow of 9 per day toward destinations (in direction of arrow) with equivalent number of returning trains in opposite direction.

All capacities in (1000’s of tons) each way per day.

Origins: Divisions 2, 3, 8, 12, 13, 13Z, 20, 30 (USH), and Romania.

Destinations: Divisions 3, 6, 9 (Poland); B (Czechoslovakia); and 2, 5 (Austria).

Alternative destinations: Germany or East Germany.

Note: IK at Division 9, Poland.
Fig. 7 — Traffic pattern: entire network available

Legend:
1. International boundary
2. Railway operating division
3. Capacity: 12 each way per day. Required flow of 5 per day toward destinations (in direction of arrow) with equivalent number of returning trains in opposite direction.

Origins: Divisions 2, 3W, 3C, 23, 13N, 13S, 12, 52 (USN), and Roumania

Destinations: Divisions 3, 6, 9 (Poland); B (Czechoslovakia); and 2, 5 (Austria)

Alternative destinations: Germany or East Germany

Note 11K at Division 9, Poland
Fig. 7 — Traffic pattern: entire network available

Legend:
- International boundary
- Railway operating division
- Capacity: 12 each way per day
- Required flow of 9 per day toward destinations (in direction of arrow) with equivalent number of returning trains in opposite direction

All capacities in trains (1/100 of a ton) each way per day

Origins: Divisions 2, 3W, 3E, 23, 13N, 13S, 12, 52 (USN), and Rumania

Destinations: Divisions B, 6, 9 (Poland); Czechoslovakia; and 2, 3 (Austria)

Alternative destinations: Germany or East Germany

Note 11K at Division 9, Poland
**ORIGINS**

The Bottleneck

Total capacity 228 trains/day

Total capacity 163 trains/day

Total capacity 276 trains/day

---

Fig. 7 — Traffic pattern: entire network available

Legend:
- **—** International boundary
- **---** Railway operating division

**Capacity:** 16 each way per day. Required flow of 9 per day toward destinations (in direction of arrow) with equivalent number of returning trains in opposite direction

*All capacities in *1000's of tons* each way per day.*

**Origins:** Divisions 2, 3, 4, 5, 6, 7, 8, 9 (Poland); B (Czechoslovakia); and 1 (Austria)

**Destinations:** Divisions 3, 6, 9 (Poland); 11 (Czechoslovakia); and 2, 3 (Austria)

**Alternative destinations:** Germany or East Germany

Note 11K at Division 9, Poland
A **cut** is a partition of the vertices into two sets, \( V = S \cup \bar{S} \), with the source vertex \( s \in S \) and the sink vertex \( t \in \bar{S} \).

The **capacity** of the cut is
\[
\text{capacity}(S, \bar{S}) = \sum_{u \in S, v \in \bar{S}: u \to v} c(u \to v)
\]

**MAX-FLOW MIN-CUT THEOREM**
For any flow \( f \) and any cut \((S, \bar{S})\),
\[\text{value}(f) \leq \text{capacity}(S, \bar{S})\]
MAX-FLOW MIN-CUT THEOREM

For any flow $f$ and any cut $(S, \bar{S})$,

$$\text{value}(f) \leq \text{capacity}(S, \bar{S})$$
The Ford-Fulkerson algorithm, if it terminates, finds a flow $f^*$ and a cut $(S^*, \overline{S}^*)$ such that $\text{value}(f^*) = \text{capacity}(S^*, \overline{S}^*)$.

**Proof**

Let $g$ be any other flow.  

\[
\text{value}(g) \leq \text{capacity}(S^*, \overline{S}^*) \quad \text{by MFMC}
\]

\[
= \text{value}(f^*) \quad \text{by assumption}
\]

\[
\therefore f^* \text{ is a maximum flow}
\]

**FORD-FULKERSON CLAIM**

The Ford-Fulkerson algorithm, if it terminates, finds a flow $f^*$ and a cut $(S^*, \overline{S}^*)$ such that $\text{value}(f^*) = \text{capacity}(S^*, \overline{S}^*)$.
PROOF STRATEGY
Write out the flow conservation equations for each vertex in $S \setminus \{s\}$, and sum them. Then use
$0 \leq \text{flow} \leq \text{capacity}$

Flow conservation at vertices other than $s,t$:

\[
\begin{align*}
\text{a} &= \text{d} + \text{c} \\
\text{c} + \text{f} &= \text{b} + \text{e}
\end{align*}
\]

\[
\Rightarrow \quad \begin{cases}
\text{a} - \text{c} - \text{d} = 0 \\
-\text{b} + \text{c} - \text{e} + \text{f} = 0
\end{cases}
\]

\[
\Rightarrow \quad \text{a} - \text{b} = \text{d} + \text{e} - \text{f}
\]

\[
\Rightarrow \quad \text{a} - \text{b} \leq \text{d} + \text{e} \leq \text{capacity of cut}
\]

net flow out of $S$
MAX-FLOW MIN-CUT THEOREM. For any flow \( f \) and any cut \((S, \tilde{S})\), \( \text{value}(f) \leq \text{capacity}(S, \tilde{S}) \)

\[
\text{value}(f) = \sum_{u:s \to u} f(s \to u) - \sum_{u:s \to u} f(u \to s) \quad \text{by definition of flow value = net flow out - net flow in}\n\]

\[
= \sum_{u \in V} f(s \to u) - \sum_{u \in V} f(u \to s) \quad \text{where we've extended } f \text{ to all pairs of vertices, and set } f(v \to w) = 0 \text{ if there's no edge } v \to w\n\]

\[
= \sum_{v \in S} \left[ \sum_{u \in V} f(v \to u) - \sum_{u \in V} f(u \to v) \right] \quad \text{Flow conservation says that the term } [ \cdot ] \text{ is 0 for all vertices in } V \setminus \{s, \tilde{s}\}.\n\]

\[
= \sum_{v \in S} \sum_{u \in S} f(v \to u) + \sum_{u \in S} \sum_{v \in S} f(u \to v) - \sum_{v \in S} \sum_{u \in S} f(u \to v) - \sum_{u \in S} \sum_{v \in S} f(u \to v) \quad \text{expanding } \sum_{u \in V} f(u) = \sum_{v \in S} f(v) + \sum_{u \in \tilde{S}} f(u)\n\]

\[
= \sum_{a \in S} \sum_{b \in S} f(a \to b) + \sum_{u \in \tilde{S}} \sum_{v \in S} f(v \to u) - \sum_{b \in S} \sum_{a \in S} f(a \to b) - \sum_{v \in S} \sum_{u \in \tilde{S}} f(u \to v) \quad \text{relabelling the "loop" variables}\n\]

\[\text{These two terms are identical, so they cancel out.} \quad [\text{Similar to the "telescopic sum" in Johnson's alg.]}
\]

\[
\geq \sum_{v \in S} \sum_{u \in S} f(v \to u) - \sum_{u \in S} \sum_{v \in S} f(u \to v)\n\]

\[
\geq \sum_{v \in S} \sum_{u \in \tilde{S}} f(v \to u) \quad \text{since } f \geq 0 \text{ on every edge}\n\]

\[
\geq \sum_{v \in S} \sum_{u \in \tilde{S}} c(v \to u) \quad \text{since } f \leq c \text{ on every edge}\n\]

\[
= \text{capacity } (S, \tilde{S}) \quad \text{by definition of cut capacity}\n\]
Important bits of the proof:

\[
\text{value}(f) \leq \sum_{v \in S} \sum_{u \in S} f(v \rightarrow u) \quad \text{since } f = 0 \text{ on every edge}
\]

\[
= \sum_{v \in S} \sum_{u \in S} c(v \rightarrow u) \quad \text{since } f = c \text{ on every edge}
\]

\[
= \text{capacity } (S, \overline{S}) \quad \text{by definition of cut capacity}
\]

**Remark.**

If \( f = 0 \) on every edge from \( S \) to \( \overline{S} \) then the first inequality is an equality.

If \( f = c \) on every edge from \( S \) to \( \overline{S} \) then the second inequality is an equality.

And if both conditions are met, then

\[
\text{value}(f) = \text{capacity } (S, \overline{S})
\]

This is how we’ll prove the Ford-Fulkerson claim: We’ll demonstrate a flow \( f^* \) and a cut \((S^*, \overline{S}^*)\) such that all edges \( \overline{S}^* \rightarrow S^* \) have zero flow, and all edges \( S^* \rightarrow \overline{S}^* \) are at capacity.
A flow network

The residual graph

An augmenting path

\[ \delta = 4 \]
WALKTHROUGH OF FORD-FULKERSON
WALKTHROUGH OF FORD-FULKERSON

\[ \delta = 2 \]
WALKTHROUGH OF FORD-FULKERSON

We cannot find an augmenting path in the residual graph. So, terminate.
```python
def ford_fulkerson(g, s, t):
    # Let f be a flow, initially empty
    for u → v in g.edges:
        f(u → v) = 0

    # Define a helper function for finding an augmenting path
    def find_augmenting_path():
        # Define the residual graph h on the same vertices as g
        for u → v in g.edges:
            if f(u → v) < c(u → v): give h an edge u → v labelled “inc u → v”
            if f(u → v) > 0: give h an edge v → u labelled “dec u → v”

            if h has a path from s to t:
                return some such path, together with the labels of its edges
            else:
                # Let S be the set of vertices reachable from s (used in the proof)
                return None

        # Repeatedly find an augmenting path and add flow to it
        while True:
            p = find_augmenting_path()
            if p is None:
                break
            else:
                compute δ, the amount of flow to apply along p, and apply it
                # Assert: δ > 0
                # Assert: f is still a valid flow
```

```
vertices reachable from s
```

```
a
```

```
b
c
```

```
s
t
```

```
vertices reachable from s
```
The Ford-Fulkerson algorithm, if it terminates, finds a flow $f^*$ and a cut $(S^*, \bar{S}^*)$ such that $\text{value}(f^*) = \text{capacity}(S^*, \bar{S}^*)$

\textbf{Proof} Suppose it terminates, let $f^*$ be the final flow it produces, and let $S^* = \{v \in V | \text{there exists a path from } s \text{ to } v \text{ in the residual graph} \}$ at termination.

Then $(S^*, \bar{S}^*)$ is a cut.

(Recall the definition of a cut: we need $s \in S^*$ and $t \in \bar{S}^*$. This is so because, at termination, we can't reach $t$.)

And the residual graph has no edges $S^* \rightarrow \bar{S}^*$ (because vertices in $\bar{S}^*$ are unreachable, by defn. of $S^*$)

- If the capacity graph has an edge $\overline{v \rightarrow u}$ then $f^*(v \rightarrow u) = c(v \rightarrow u)$
  (otherwise the residual graph would have an edge $\overline{v \rightarrow u}$)

- If the capacity graph has an edge $u \rightarrow \overline{v}$ then $f^*(u \rightarrow v) = 0$
  (otherwise the residual graph would have an edge $u \rightarrow v$)

By the remark at the end of the proof of \textit{Max-Flow Min-Cut Theorem},

$\text{value}(f^*) = \text{capacity}(S^*, \bar{S}^*)$. 
The Ford-Fulkerson algorithm produces both a flow and a cut; and the cut acts as a *certificate of optimality* for the flow.

Many other optimization algorithms also produce a (solution, certificate) pair. The certificate corresponds to the dual variables in Lagrangian optimization.
A latent generative model is a neural network that has been trained to map a random noise vector into something that resembles items from the training dataset.

\[ X = f_\theta(Z) \]
A **latent generative model** is a neural network that has been trained to map a random noise vector into something that resembles items from the training dataset.

\[
\text{random noise } Z \xrightarrow{\text{edge weights } \theta} X = f_\theta(Z)
\]

An **adversary** is a neural network that guesses whether an input \( x \) is real (i.e. from the training dataset) or fake (i.e. generated by us).

\[
x \xrightarrow{\text{edge weights } \phi} Y = g_\phi(x) \in \{\text{real, fake}\}
\]

We can train a good generator by simultaneously training an adversary. When we’ve finished training, the adversary should be unable to detect whether a given \( x \) is real or fake. The adversary is a **certificate** that our generator is good.
Algorithms assignment grade-gpt: Grading ChatGPT’s proof

Can ChatGPT be persuaded to give a proper proof of correctness of an algorithm? Here are three attempts, for an algorithm that solves the bfs-all tick:

- Catley, Prynn, and Huang.

Please mark these attempts, on a scale of 0–20. Your mark should be for the final proof, not for how well it was elicited. Please submit your grades on Moodle. I’ll pick the most controversially-marked answer and go through it in lectures. Please use the following marking scheme:

<table>
<thead>
<tr>
<th>mark</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Coherent fragments</td>
</tr>
<tr>
<td>9</td>
<td>Coherent in parts, but with serious gaps</td>
</tr>
<tr>
<td>13</td>
<td>A basically correct argument but with some signs of confusion</td>
</tr>
<tr>
<td>17</td>
<td>Essentially correct, but not fully rigorous</td>
</tr>
<tr>
<td>19</td>
<td>Nearly all correct, only minor technical holes</td>
</tr>
</tbody>
</table>

How well does ChatGPT generate algorithms? For interest, here are the attempts to get ChatGPT to design the algorithm:

- Catley, Shen, Chen, Prynn, and Huang.