Bipartite Matching

**Given:** A bipartite graph $G=(V,E)$
- $M \subseteq E$ is a matching in $G$ iff no two edges in $M$ share a vertex

**Goal:** Find a matching $M$ in $G$ of maximum possible size
**The Network Flow Problem**

- How much stuff can flow from \( s \) to \( t \)?

**Net Flow: Formal Definition**

Given:
- A digraph \( G = (V, E) \)
- Two vertices \( s, t \) in \( V \)
- A capacity \( c(u,v) \geq 0 \)
- for each \( (u,v) \in E \)
- (and \( c(u,v) = 0 \) for all non-edges \( (u,v) \))

Find:
- A flow function \( f: E \rightarrow \mathbb{R} \) s.t., for all \( u,v \):
  - \( 0 \leq f(u,v) \leq c(u,v) \)  \[\text{[Capacity Constraint]}\]
  - \( f(u,v) = \) if \( u \neq s,t \), i.e. \( f^{\text{out}}(u) = f^{\text{in}}(u) \) \[\text{[Flow Conservation]}\]

Maximizing total flow \( \nu(f) = f^{\text{out}}(s) \)

**Example: A Flow Function**

- Notation:
  - \( f^{\text{in}}(v) = \sum_{e = (u,v) \in E} f(u,v) \)
  - \( f^{\text{out}}(v) = \sum_{e = (v,w) \in E} f(v,w) \)

- Example: A Flow Function

\[
\begin{align*}
\text{flow/capacity, not .66...} \\
\nu(f) &= f^{\text{out}}(s) = 2 \\
\nu(f) &= f^{\text{out}}(u) = f^{\text{in}}(u) = 2 \\
\nu(f) &= f^{\text{out}}(t) = f^{\text{in}}(t) = 2
\end{align*}
\]
Example: A Flow Function

- Not shown: $f(u,v)$ if $= 0$
- Note: $\text{max flow } \geq 4$ since $f$ is a flow function, with $\nu(f) = 4$

Max Flow via a Greedy Alg?

While there is an $s \rightarrow t$ path in $G$
Pick such a path, $p$
Find $c$, the min capacity of any edge in $p$
Subtract $c$ from all capacities on $p$
Delete edges of capacity $0$

This does NOT always find a max flow:

![Diagram of a flow network with labels and capacities, showing a path from $s$ to $t$.]

If pick $s \rightarrow b \rightarrow a \rightarrow t$ first, flow stuck at $2$. But flow 3 possible.

A Brief History of Flow

<table>
<thead>
<tr>
<th>#</th>
<th>year</th>
<th>discoverer(s)</th>
<th>bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1993</td>
<td>Dantzig</td>
<td>$O(n^3 m/f^2)$</td>
</tr>
<tr>
<td>2</td>
<td>1955</td>
<td>Ford &amp; Fulkerson</td>
<td>$O(nmU)$</td>
</tr>
<tr>
<td>3</td>
<td>1970</td>
<td>Edmonds &amp; Karp</td>
<td>$O(nm^2)$</td>
</tr>
<tr>
<td>4</td>
<td>1970</td>
<td>Dijkstra</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>5</td>
<td>1972</td>
<td>Edmonds &amp; Karp</td>
<td>$O(m^2 \log U)$</td>
</tr>
<tr>
<td>6</td>
<td>1973</td>
<td>Dijkstra</td>
<td>$O(nm \log U)$</td>
</tr>
<tr>
<td>7</td>
<td>1974</td>
<td>Karp &amp; Knight</td>
<td>$O(n^3)$</td>
</tr>
<tr>
<td>8</td>
<td>1977</td>
<td>Christo &amp; Lawler</td>
<td>$O(n^3 \log^2 n)$</td>
</tr>
<tr>
<td>9</td>
<td>1980</td>
<td>Goldstone &amp; Lawler</td>
<td>$O(nm \log^2 n)$</td>
</tr>
<tr>
<td>10</td>
<td>1983</td>
<td>Megyeri &amp; Tarjan</td>
<td>$O(nm)</td>
</tr>
<tr>
<td>11</td>
<td>1986</td>
<td>Goldstein &amp; Tarjan</td>
<td>$O(nm \log^6 n)$</td>
</tr>
<tr>
<td>12</td>
<td>1987</td>
<td>Ahuja &amp; Orlin</td>
<td>$O(nm + m^2 \log^2 n)$</td>
</tr>
<tr>
<td>13</td>
<td>1987</td>
<td>Ahuja et al.</td>
<td>$O(nm \log^2 n)$</td>
</tr>
<tr>
<td>14</td>
<td>1989</td>
<td>Christo &amp; Hagerup</td>
<td>$O(nm \log^3 n)$</td>
</tr>
<tr>
<td>15</td>
<td>1990</td>
<td>Christo &amp; Hagerup</td>
<td>$O(nm \log^3 n)$</td>
</tr>
<tr>
<td>16</td>
<td>1990</td>
<td>Orlin</td>
<td>$O(nm \log^3 n)$</td>
</tr>
<tr>
<td>17</td>
<td>1992</td>
<td>Orlin &amp; Thangavel</td>
<td>$O(nm \log^3 n)$</td>
</tr>
<tr>
<td>18</td>
<td>1993</td>
<td>Phillips &amp; Wheel</td>
<td>$O(nm \log^3 n)$</td>
</tr>
<tr>
<td>19</td>
<td>1994</td>
<td>King et al.</td>
<td>$O(nm \log^3 n)$</td>
</tr>
<tr>
<td>20</td>
<td>1997</td>
<td>Goldberg &amp; Rao</td>
<td>$O(nm \log^3 n)$</td>
</tr>
</tbody>
</table>

Greed Revisited: Residual Graph & Augmenting Path

- $n$ = # of vertices
- $m$ = # of edges
- $U$ = Max capacity

Residual Graph
Greed Revisited: An Augmenting Path

Residual Capacity
- The residual capacity (w.r.t. \( f \)) of \((u,v)\) is 
  \[ c_f(u,v) = c(u,v) - f(u,v) \] if \( f(u,v) \leq c(u,v) \) 
  and 
  \[ c_f(u,v) = f(v,u) \] if \( f(v,u) > 0 \)
- \text{e.g. } c_f(s,b) = 7; c_f(a,x) = 1; c_f(x,a) = 3

Residual Graph & Augmenting Paths
- The residual graph (w.r.t. \( f \)) is the graph 
  \( G_f = (V,E_f) \), where 
  \( E_f = \{ (u,v) \mid c_f(u,v) > 0 \} \)
- Two kinds of edges
  - Forward edges 
    - \( f(u,v) < c(u,v) \) so \( c_f(u,v) = c(u,v) - f(u,v) > 0 \)
  - Backward edges 
    - \( f(u,v) > 0 \) so \( c_f(v,u) = f(v,u) > 0 \)
- An augmenting path (w.r.t. \( f \)) is a simple \( s \rightarrow t \) path in \( G_f \).

A Residual Network
Augmenting A Flow

\[ \text{augment}(f, P) \]

\[ c_P \leftarrow \min_{(u,v) \in P} c_f(u,v) \]

“bottleneck(P)"

for each \( e \in P \)

if \( e \) is a forward edge then

increase \( f(e) \) by \( c_P \)

else (\( e \) is a backward edge)

decrease \( f(e) \) by \( c_P \)

endif

endfor

return(\( f \) )

Claim 7.1

If \( G_f \) has an augmenting path \( P \), then the function \( f' = \text{augment}(f, P) \) is a legal flow.

Proof:

- \( f' \) and \( f \) differ only on the edges of \( P \) so only need to consider such edges \((u,v)\)
Proof of Claim 7.1

- If \((u,v)\) is a forward edge then
  \[ f'(u,v) = f(u,v) + c_p \leq f(u,v) + c_f(u,v) \]
  \[ = f(u,v) + c(u,v) - f(u,v) \]
  \[ = c(u,v) \]

- If \((u,v)\) is a backward edge then \(f\) and \(f'\) differ on flow along \((v,u)\) instead of \((u,v)\)
  \[ f'(v,u) = f(v,u) - c_p \geq f(v,u) - c_f(u,v) \]
  \[ = f(v,u) - f(v,u) = 0 \]

- Other conditions like flow conservation still met

Ford-Fulkerson Method

Start with \(f=0\) for every edge

While \(G_f\) has an augmenting path, augment

- Questions:
  - Does it halt?
  - Does it find a maximum flow?
  - How fast?

Observations about Ford-Fulkerson Algorithm

- At every stage the capacities and flow values are always integers (if they start that way)
- The flow value \(v(f') = v(f) + c_p > v(f)\) for \(f' = \text{augment}(f, P)\)
  - Since edges of residual capacity 0 do not appear in the residual graph
- Let \(C = \sum_{(s,u) \in E} c(s,u)\)
  - \(v(f) \leq C\)
  - F-F does at most \(C\) rounds of augmentation since flows are integers and increase by at least 1 per step

Running Time of Ford-Fulkerson

- For \(f=0\), \(G_f = G\)
- Finding an augmenting path in \(G_f\) is graph search \(O(n+m) = O(m)\) time
- Augmenting and updating \(G_f\) is \(O(n)\) time
- Total \(O(mC)\) time
- Does is find a maximum flow?
  - Need to show that for every flow \(f\) that isn’t maximum \(G_f\) contains an \(s-t\)-path
Cuts

- A partition \((A, B)\) of \(V\) is an \(s-t\)-cut if
  - \(s \in A, t \in B\)
- **Capacity** of cut \((A, B)\) is \(c(A, B) = \sum_{u \in A, v \in B} c(u, v)\)

\[
\text{Capacity of cut } (A,B) = \sum_{u \in A, v \in B} c(u, v)
\]

Convenient Definition

- \(f_{\text{out}}(A) = \sum_{v \in A, w \in A} f(v, w)\)
- \(f_{\text{in}}(A) = \sum_{v \in A, u \in A} f(u, v)\)

Claims 7.6 and 7.8

- For any flow \(f\) and any cut \((A, B)\),
  - the net flow across the cut equals the total flow, i.e., \(\nu(f) = f_{\text{out}}(A) - f_{\text{in}}(A)\), and
  - the net flow across the cut cannot exceed the capacity of the cut, i.e. \(f_{\text{out}}(A) - f_{\text{in}}(A) \leq c(A, B)\)
- Corollary:
  - Max flow \(\leq\) Min cut

Proof of Claim 7.6

- Consider a set \(A\) with \(s \in A, t \in A\)
- \(f_{\text{out}}(A) - f_{\text{in}}(A) = \sum_{v \in A, w \in A} f(v, w) - \sum_{v \in A, u \in A} f(u, v)\)
- We can add flow values for edges with both endpoints in \(A\) to both sums and they would cancel out so
  - \(f_{\text{out}}(A) - f_{\text{in}}(A) = \sum_{v \in A, w \in V} f(v, w) - \sum_{u \in V} f(u, v)\)
  - \(f_{\text{out}}(A) = \sum_{v \in A} f_{\text{out}}(v) - f_{\text{in}}(v)\)
  - \(f_{\text{in}}(A) = \sum_{v \in V} f_{\text{out}}(v) - f_{\text{in}}(s)\)

since all other vertices have \(f_{\text{out}}(v) = f_{\text{in}}(v)\)
- \(\nu(f) = f_{\text{out}}(s)\) and \(f_{\text{in}}(s) = 0\)
Proof of Claim 7.8

\[ \nu(f) = f^{\text{out}}(A) - f^{\text{in}}(A) \leq f^{\text{out}}(A) \]
\[ = \sum_{v \in A, w \in A} f(v,w) \leq \sum_{v \in A, w \in A} c(v,w) \leq \sum_{v \in A, w \in B} c(v,w) = c(A,B) \]

Claim 7.9

Let \( A = \{ u \mid \exists \text{an path in } G_f \text{ from } s \text{ to } u \} \)
\( B = V - A; \ s \in A, t \in B \)

This is true for every edge crossing the cut, i.e.
\[ f^{\text{out}}(A) = \sum_{u \in A, v \in B} f(u,v) = \sum_{u \in A, v \in B} c(A,B) \text{ and } f^{\text{in}}(A) = 0 \text{ so } \nu(f) = f^{\text{out}}(A) - f^{\text{in}}(A) = c(A,B) \]

Max Flow / Min Cut Theorem

Claim 7.9 For any flow \( f \), if \( G_f \) has no augmenting path then there is some \( s-t \)-cut \((A,B)\) such that \( \nu(f) = c(A,B) \) (proof on next slide)

- We know by Claims 7.6 & 7.8 that any flow \( f' \) satisfies \( \nu(f') \leq c(A,B) \) and we know that F-F runs for finite time until it finds a flow \( f \) satisfying conditions of Claim 7.9
- Therefore by 7.9 for any flow \( f' \), \( \nu(f') \leq \nu(f) \)
- Corollary (1) F-F computes a maximum flow in \( G \)

Flow Integrality Theorem

If all capacities are integers
- The max flow has an integer value
- Ford-Fulkerson method finds a max flow in which \( f(u,v) \) is an integer for all edges \((u,v)\)
Corollaries & Facts

- If Ford-Fulkerson terminates, then it’s found a max flow.
- It will terminate if \( c(e) \) integer or rational (but may not if they’re irrational).
- However, may take exponential time, even with integer capacities:

\[
\begin{align*}
\text{Bipartite matching as a special case of flow} \\
\end{align*}
\]

\[
\begin{align*}
\text{Capacity-scaling algorithm} \\
\end{align*}
\]

- General idea:
  - Choose augmenting paths \( P \) with ‘large’ capacity \( c_P \)
  - Can augment flows along a path \( P \) by any amount \( \Delta \leq c_P \)
    - Ford-Fulkerson still works
  - Get a flow that is maximum for the high-order bits first and then add more bits later

\[
\begin{align*}
\text{Capacity Scaling} \\
\end{align*}
\]
Capacity Scaling

Capacity on each edge is at most 1
(either 0 or 1 times $\Delta=4$)

O(nm) time

Residual capacity across min cut is at most m
(either 0 or 1 times $\Delta=2$)
Residual capacity across min cut is at most $m$

⇒ $\leq m$ augmentations

Residual capacity across min cut is at most $m$
(either 0 or 1 times $\Delta=1$)

After $\leq m$ augmentations
Capacity Scaling Min Cut

- \[ \log_2 U \] rounds where \( U \) is largest capacity
- At most \( m \) augmentations per round
  - Let \( c_i \) be the capacities used in the \( i \)th round and \( f_i \) be the maxflow found in the \( i \)th round
    - For any edge \((u,v)\), \( c_{i+1}(u,v) \leq 2c_i(u,v) + 1 \)
    - \( i+1 \)st round starts with flow \( f = 2f_i \)
  - Let \((A,B)\) be a min cut from the \( i \)th round
    - \( v(f_i) = c_i(A,B) \) so \( v(f) = 2c_i(A,B) \)
    - \( v(f_{i+1}) \leq c_{i+1}(A,B) \leq 2c_i(A,B) + m = v(f) + m \)
- \( O(m) \) time per augmentation
- Total time \( O(m^2 \log U) \)

Edmonds-Karp Algorithm

- Use a shortest augmenting path (via Breadth First Search in residual graph)
- Time: \( O(n m^2) \)

BFS/Shortest Path Lemmas

Distance from \( s \) in \( G_i \) is never reduced by:

- Deleting an edge
  - Proof: no new (hence no shorter) path created
- Adding an edge \((u,v)\), provided \( v \) is nearer than \( u \)
  - Proof: BFS is unchanged, since \( v \) visited before \((u,v)\) examined
**Key Lemma**

Let $f$ be a flow, $G_f$ the residual graph, and $P$ a shortest augmenting path. Then no vertex is closer to $s$ after augmentation along $P$.

**Proof:** Augmentation along $P$ only deletes forward edges, or adds back edges that go to previous vertices along $P$.

---

**Theorem**

The Edmonds-Karp Algorithm performs $O(mn)$ flow augmentations.

**Proof:**

Call $(u,v)$ critical for augmenting path $P$ if it's closest to $s$ having min residual capacity.

- It will disappear from $G_f$ after augmenting along $P$.
- In order for $(u,v)$ to be critical again the $(u,v)$ edge must re-appear in $G_f$ but that will only happen when the distance to $u$ has increased by 2.

It won't be critical again until farther from $s$ so each edge critical at most $n/2$ times.
Corollary

- Edmonds-Karp runs in $O(nm^2)$ time

Project Selection a.k.a. The Strip Mining Problem

- Given
  - a directed acyclic graph $G=(V,E)$ representing precedence constraints on tasks (a task points to its predecessors)
  - a profit value $p(v)$ associated with each task $v \in V$ (may be positive or negative)

- Find
  - a set $A \subseteq V$ of tasks that is closed under predecessors, i.e. if $(u,v) \in E$ and $u \in A$ then $v \in A$, that maximizes $\text{Profit}(A) = \sum_{v \in A} p(v)$

Project Selection Graph

Extended Graph

Each task points to its predecessor tasks
For each vertex $v$:
- If $p(v) \geq 0$, add $(s,v)$ edge with capacity $p(v)$.
- If $p(v) < 0$, add $(v,t)$ edge with capacity $-p(v)$.

Want to arrange capacities on edges of $G$ so that for minimum $s$-$t$-cut $(S,T)$ in $G'$, the set $A = S$-$\{s\}$:
- satisfies precedence constraints
- has maximum possible profit in $G$

Cut capacity with $S = \{s\}$ is just $C = \sum v: p(v) \geq 0 p(v)$
- $\text{Profit}(A) \leq C$ for any set $A$

To satisfy precedence constraints don’t want any original edges of $G$ going forward across the minimum cut:
- That would correspond to a task in $A = S$-$\{s\}$ that had a predecessor not in $A = S$-$\{s\}$
- Set capacity of each of the edges of $G$ to $C + 1$
- The minimum cut has size at most $C$

Cut value:
$= 13 \cdot 3 + 3 + 2 + 3 + 4$
$= 13 + 3$
$+ C$ - $4 - 8 - 10 - 11 - 12 - 14$
Project Selection

- **Claim** Any $s$-$t$-cut $(S, T)$ in $G'$ such that $A = S \setminus \{s\}$ satisfies precedence constraints has capacity
  \[ c(S, T) = C - \sum_{v \in A} p(v) = C - \text{Profit}(A) \]

- **Corollary** A minimum cut $(S, T)$ in $G'$ yields an optimal solution $A = S \setminus \{s\}$ to the profit selection problem

- **Algorithm** Compute maximum flow $f$ in $G'$, find the set $S$ of nodes reachable from $s$ in $G'_f$ and return $S \setminus \{s\}$

---

Proof of Claim

- **A = S \setminus \{s\}** satisfies precedence constraints
  - No edge of $G$ crosses forward out of $A$ since those edges have capacity $C+1$
  - Only forward edges cut are of the form $(v, t)$ for $v \in A$ or $(s, v)$ for $v \in \bar{A}$
  - The $(v, t)$ edges for $v \in A$ contribute \[ \sum_{v \in A : p(v) < 0} p(v) = \sum_{v \in A : p(v) < 0} p(v) \]
  - The $(s, v)$ edges for $v \notin A$ contribute \[ \sum_{v \notin A : p(v) \geq 0} p(v) = C - \sum_{v \in A : p(v) \geq 0} p(v) \]
  - Therefore the total capacity of the cut is
  \[ c(S, T) = C - \sum_{v \in A} p(v) = C - \text{Profit}(A) \]