More Applications of Max Flow
Edge Disjoint Paths

**Disjoint path problem.** Given a directed graph \( G = (V, E) \) and two nodes \( s \) and \( t \), find the max number of edge-disjoint \( s \)-\( t \) paths.

**Def.** Two paths are **edge-disjoint** if they have no edge in common.

**Application:** Communication networks.
Edge Disjoint Paths

Max flow formulation: assign unit capacity to every edge.

Theorem. Max number edge-disjoint s-t paths equals max flow value.

Proof. ≤

- Suppose there are $k$ edge-disjoint paths $P_1, \ldots, P_k$.
- Set $f(e) = 1$ if $e$ participates in some path $P_i$; else set $f(e) = 0$.
- Since paths are edge-disjoint, $f$ is a flow of value $k$. 
Edge Disjoint Paths

Max flow formulation: assign unit capacity to every edge.

Proof. $\geq$

- Let $f$ be a max flow in $G'$ of value $k$ computed by Ford-Fulkerson
- $f(e) = 1$ or 0 for every edge $e$ (integrality property).
- Consider any edge $(s, u)$ with $f(s, u) = 1$.
  - By conservation, there exists an edge $(u, v)$ with $f(u, v) = 1$
  - Continue to find the next unused edge out of $v$ until reaching $t$.
- After finding one path, flow value decreases by 1.
- Repeat the process $k$ times to find $k$ edge-disjoint paths.
- The proof above also provides an algorithm.
Circulation with Demands

**Input:** A directed connected graph $G = (V, E)$, where

- every edge $e \in E$ has a capacity $c(e)$;
- a number of source vertices $s_1, s_2, \ldots$, each with a supply of $\text{sup}(s_i)$ and a number of target vertices $t_1, t_2, \ldots$, each with a demand of $\text{dem}(t_i)$;
- $\sum_i \text{sup}(s_i) \geq \sum_i \text{dem}(t_i)$

**Output:** A flow $f$ that meets capacity and conservation conditions, and

- At each source vertex $s_i$, $\sum e \text{ out of } s_i \text{ f}(e) - \sum e \text{ into } s_i \text{ f}(e) \leq \text{sup}(s_i)$;
- At each target vertex $t_i$, $\sum e \text{ into } t_i \text{ f}(e) - \sum e \text{ out of } t_i \text{ f}(e) = \text{dem}(s_i)$. 

![Circulation Diagram](image_url)
Solving Circulation with Demands using Max Flow

Algorithm:
- Add a “super source” \( s \) and a “super target” \( t \).
- Add an edge from \( s \) to each \( s_i \) with capacity \( \text{sup}(s_i) \).
- Add an edge from each \( t_i \) to \( t \) with capacity \( \text{dem}(t_i) \).
- Compute the max flow \( f \).
- If \( |f| = \sum_i \text{dem}(t_i) \), then return \( f \); else return “no solution”.

\[ G' : \]

\[ \begin{align*}
\text{supply} & : \\
\text{demand} & :
\end{align*} \]
### Baseball Elimination

<table>
<thead>
<tr>
<th>Team $i$</th>
<th>Wins $w_i$</th>
<th>To play $r_i$</th>
<th>Remaining Against = $r_{ij}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

**Rule:** Order teams by the number of wins.

**Q:** Does Team 4 still have a chance to finish in the first place (tie is OK)?

**A:** No, obviously.
Q: Does Team 4 still have a chance to finish in the first place (tie is OK)?

A: No, because

- Team 4 has to win both remaining games against team 2 and 3.
- Team 1 has to lose both remaining games against team 2 and 3.
- Then 2 and 3 will both have 3 wins.
- The game between team 2 and 3 will give one of them one more win.

Suppose you need to do this for MLB / Premier League...
Baseball Elimination: Formal Definition

Input:
- \( n \) teams: 1, 2, \ldots, \( n \)
- One particular team, say \( n \) (without loss of generality)
- Team \( i \) has won \( w_i \) games already
- Team \( i \) and \( j \) still need to play \( r_{ij} \) games, \( r_{ij} = 0 \) or 1.
- Team \( i \) has a total of \( r_i = \sum_j r_{ij} \) games to play

Output:
- “Yes”, if there is an outcome for each remaining game such that team \( n \) finishes with the most wins (tie is OK).
- “No”, if no such possibilities.

Brute-force algorithm:
- For each remaining game, consider two possible outcomes.
- Try all \( 2^r \) possible combinations, where \( r = \sum_{i,j} r_{ij} \)
Can team \( n \) finish with most wins?

- Assume team \( n \) wins all remaining games \( \Rightarrow w_n + r_n \) wins.
- All other teams must have \( \leq w_n + r_n \) wins.

Flow network construction:

- A source \( s \) and a target \( t \)
- A node for each remaining game \((i, j)\); and an edge from \( s \) to it with capacity 1
- A node for each team \( i = 1, 2, \ldots, n - 1 \); and an edge from it to \( t \) with capacity \( w_n + r_n - w_i \)
- Game node \((i, j)\) has edges to team node \( i \) and \( j \), with capacity 1

Baseball Elimination: Max Flow Formulation
Baseball Elimination: Max Flow Formulation

**Claim:** There is a way for team \( n \) to finish in the first place iff the max flow has value \( r = \sum_{i,j} r_{ij} \).

**Proof:** “⇒”: Suppose there is an outcome for each remaining game such that team \( n \) finishes the first. First set \( f(s, (i, j)) = 1 \) for all \( (i, j) \).

For each remaining game \((i, j)\):
- if \( i \) wins, set \( f((i, j), i) = 1 \) and \( f((i, j), j) = 0 \);
- if \( j \) wins, set \( f((i, j), j) = 1 \) and \( f((i, j), i) = 0 \).

Team \( i \) wins \( \leq w_n + r_n - w_i \) games, so it can send all incoming flow to \( t \).
Proof: “⇐”: Suppose the max flow $f$ has $|f| = r$. It must saturate all edges out of $s$.

Look at each game node $(i, j)$. Exactly one of its outgoing edges must have 1 unit of flow (integrality property):

- If $f((i, j), i) = 1$, let $i$ win the game;
- If $f((i, j), j) = 1$, let $j$ win the game.

Team node $i$ receives $\leq w_n + r_n - w_i$ units of flow, each corresponding to one win, so it cannot beat team $n$.

Baseball Elimination: Max Flow Formulation

game nodes

team nodes
Baseball Elimination: Extensions

Q: What if $r_{ij}$ can be more than 1?

Q: Can this be used for football (soccer) leagues?
  - Using the old rule: Winner takes 2 points, loser 0 point; each team gets 1 point in case of a tie.