Extensive-Form Games

- An Extensive-Form Game consist of the following elements:
  - A set of players $N$
  - A set of histories $H$ (all possible sequences of moves)
  - A player function $P$, which assigns a player (decision-maker) to every history
  - A payoff function, which assigns payoffs for each player to every **terminal** node

- It differs from a Normal-Form Game
  - It is dynamic (players move in some order)
  - Players may observe histories (what happened so far in the game)
  - Every time a player makes a move, that move can be conditioned on the history
Game Trees

- A game tree is a graph that represents an extensive-form game, like a game matrix for normal-form games.
- In practice, this representation is used only for relatively simple games.
- Game Trees consist of:
  - Nodes (Decision Nodes, Terminal Nodes), that represent histories.
  - Branches (Arcs), that represent the possible decisions (moves, actions) at a decision node.
Game Trees - Examples

Biased matching pennies

![Game Tree Diagram]

Player 1

Player 2

Player 2

H

T

H

T

H

T

(1, -1)

(-1, 1)

(-1, 1)

(1, -1)
Game Trees - Examples

A 3-player game

Player 1

L1

(1, -1, -1)

R1

Player 2

L2

(2, -1, 4)

R2

Player 3

L3

(2, -2, 2)

R3

(0, 0, 3)
Game Trees - Examples

- Ultimatum game
Strategies in ext.-form games

- In extensive-form games, a (pure) strategy is a complete game plan, i.e. it assigns a (pure) decision to every possible decision node.
- In the 3-player game, each player has only two pure strategies.
- In the biased matching pennies, player 1 has 2 strategies, player 2 has 4.
- In the ultimatum game, player 1 has 5, player 2 has 32 strategies.
Reducing to Normal Form

The following game reduces to...

\[
\begin{array}{ccc}
\text{Player 1} & \text{Player 2} \\
U & L & R \\
D & L & R \\
\end{array}
\]

\[
\begin{array}{c}
(2,1) \\
(0,0) \\
(-1,1) \\
(3,2) \\
\end{array}
\]
Reducing to Normal Form

Something like this:

<table>
<thead>
<tr>
<th>Player 1</th>
<th>LL</th>
<th>LR</th>
<th>RL</th>
<th>RR</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>2, 1</td>
<td>2, 1</td>
<td>0, 0</td>
<td>0, 0</td>
</tr>
<tr>
<td>D</td>
<td>-1, 1</td>
<td>3, 2</td>
<td>-1, 1</td>
<td>3, 2</td>
</tr>
</tbody>
</table>
Subgame Perfect Nash Equilibrium (SPNE)

- A **subgame** of an extensive-form game (with perfect information) is a game which begins at any non-terminal history and contains all nodes (histories) and possible moves that can follow after that history.

- A **subgame-perfect** Nash Equilibrium (SPNE) is a pair of strategies (pure or mixed) which forms a NE in every subgame. SPNE is a refinement of NE.

- The optimal algorithm of identifying SPNE is **backward induction**. You start from finding best-responses in the smallest (final) subgames and then consider ever bigger subgames, fixing the best-responses which have been identified in smaller subgames. Problem: cannot be used in infinite-horizon games.
Finding SPNE

- In the last example, only (D, LR) is a SPNE, even though there are 3 NE.
- In the 3-player game (L1, R2, R3) is a SPNE, but (R1, L2, L3) is a NE that is not subgame perfect.
- In the biased matching pennies game, in all SPNEs player 2 plays TH (player 1 is indifferent between T and H).
The drawbacks of SPNE

- Find the SPNE of the Centipede Game

SPNE={SS..S, SS..S}