In today's lecture, we'll talk about game theory and some of its connections to computer science. The topics we'll cover are:

- 2-player Zero-sum games, and the concept of (minimax) optimal strategies.
- The connection of 2-player zero-sum games to randomized algorithms.
- And time permitting, general sum-games, and the idea of a Nash equilibrium.

1 Introduction to Game Theory

Game theory is the study of how people behave in social and economic interactions, and how they make decisions in these settings. It is an area originally developed by economists, but given its general scope, it has applications to many other disciplines, including computer science.

A clarification at the very beginning: a *game* in game theory is not just what we traditionally think of as a game (chess, checkers, poker, tennis, or football), but is much more inclusive — a game is any interaction between parties, each with their own interests. And game theory studies how these parties make decisions during such interactions.[1]

Since we very often build large systems in computer science, which are used by multiple users, whose actions affect the performance of all the others, it is natural that game theory would play an important role in many CS problems. For example, game theory can be used to model routing in large networks, or the behavior of people on social networks, or auctions on Ebay, and then to make qualitative/quantitative predictions about how people would behave in these settings.

In fact, the two areas (game theory and computer science) have become increasingly closer to each other over the past two decades — the interaction being a two-way street — with game-theorists proving results of algorithmic interest, and computer scientists proving results of interest to game theory itself.

2 Some Definitions and Examples

In a game, we have

- A collection of participants, often called *players*.
- Each player has a set of choices, called *actions*, from which players choose about how to play (i.e., behave) in this game.
- Their combined behavior leads to *payoffs* (think of this as the "happiness" or "satisfaction level") for each of the players.

Let us look at some examples: all of these basic examples have only two players which will be easy to picture and reason about.

2.1 The Shooter-Goalie Game

This game abstracts what happens in a game of soccer, when some team has a penalty shot. There are two players in this game. One called the *shooter*, the other is called the *goalie*. Hence this is a

[1] Robert Aumann, Nobel prize winner, has suggested the term "interactive decision theory" instead of "game theory".

2-player game.

The shooter has two choices: either to shoot to her left, or shoot to her right. The goalie has two choices as well: either to dive to the shooter's left, or to the shooter's right. Hence, in this case, both the players have two actions, denoted by the set $\{\mathsf{L}, \mathsf{R}\}$.[2]

Now for the "payoffs". If both the shooter and the goalie choose the same strategy (say both choose L, or both choose R) then the goalie makes a save. Note this is an abstraction of the game: for now we assume that the goalie always makes the save when diving in the correct direction. This brings great satisfaction for the goalie, not so much for the shooter. On the other hand, if they choose different strategies, then the shooter scores the goal. (Again, we are modeling a perfect shooter.) This brings much happiness for the shooter, but the goalie is disappointed.

Being mathematically-minded, suppose we say that the former two choices lead to a payoff of $+1$ for the goalie, and -1 for the shooter. And the latter two choices lead to a payoff of -1 for the goalie, and $+1$ for the shooter. We can write it in a matrix (called the *payoff matrix*) thus:

payoff matrix M	goalie L	goalie R
shooter L	$(-1, 1)$	$(1, -1)$
shooter R	$(1, -1)$	$(-1, 1)$

The rows of the game matrix are labeled with actions for one of the players (in this case the shooter), and the columns with the actions for the other player (in this case the goalie). The entries are pairs of numbers, indicating who wins how much: e.g., the L, L entry contains $(-1, 1)$, the first entry is the payoff to the row player (shooter), the second entry the payoff to the column player (goalie). In general, the payoff is (r, c) where r is the payoff to the row player, and c the payoff to the column player.

In this case, note that for each entry (r, c) in this matrix, the sum $r + c = 0$. Such a game is called a **zero-sum game**. The zero-sum-ness captures the fact that the game is "purely competitive". Note that being zero-sum does not mean that the game is "fair" in any sense—a game where the payoff matrix has $(1, -1)$ in all entries is also zero-sum, but is clearly unfair to the column player.

One more comment: for 2-player games, textbooks often define the *row-payoff matrix* R which consists of the payoffs to the row player, and the *column-payoff matrix* C consisting of the payoffs to the column player. The tuples in the payoff matrix M contain the same information, i.e.,

$$M_{ij} = (R_{ij}, C_{ij}).$$

The game being zero-sum now means that $R = -C$, or $R + C = 0$. In the example above, the matrix R is

payoff matrix M	goalie L	goalie R
shooter L	-1	1
shooter R	1	-1

[2]Note carefully: we have defined things so that left and right are with respect to the shooter. From now on, when we say the goalie dives left, it should be clear that the goalie is diving to *the shooter's left.*

2.2 Pure and Mixed Strategies

Now given a game with payoff matrix M, the two players have to choose strategies, i.e., decide how to play.

One strategy would be for the row player to decide on a row to play, and the column player to decide on a column to play. Say the strategy for the row player was to play row I and the column player's strategy was to play column J, then the payoffs would be given by the tuple (R_{IJ}, C_{IJ}) in location I, J:

$$\text{payoff } R_{IJ} \text{ to the row player, and } C_{IJ} \text{ to the column player}$$

In this case both players are playing deterministically. (E.g., the goalie decides to always go left, etc.) A strategy that decides to play a single action is called a *pure strategy*.

But very often pure strategies are not what we play. We are trying to compete with the worst adversary, and we may like to "hedge our bets". Hence we may use a randomized algorithm: e.g., the players dive/shoot left or right with some probability, or when playing the classic game of Rock-Paper-Scissors the player choose one of the options with some probability. This means the row player decides on a non-negative real p_i for each row, such that $\sum_i p_i = 1$ (this gives a probability distribution over the rows). Similarly, the column player decides on a $q_i \geq 0$ for each column, such that $\sum_i q_i = 1$. The probability distributions $\mathbf{p}, \mathbf{q}$ are called the *(mixed-strategy) mixed strategies* for the row (mixed-strategy) and column player, respectively. And then we look at the *expected payoff*

$$V_R(\mathbf{p}, \mathbf{q}) := \sum_{ij} \Pr[\text{row player plays } i, \text{column player plays } j] \cdot R_{ij} = \sum_{ij} p_i q_j R_{ij}$$

for the row player (where we used that the row and column player have independent randomness), and

$$V_C(\mathbf{p}, \mathbf{q}) := \sum_{ij} p_i q_j C_{ij}$$

for the column player. This being a two-player zero-sum game, we know that $V_R(\mathbf{p}, \mathbf{q}) = -V_C(\mathbf{p}, \mathbf{q})$, so we will just mention the payoff to one of the players (say the row player).

For instance, if $\mathbf{p} = (0.5, 0.5)$ and $\mathbf{q} = (0.5, 0.5)$ in the shooter-goalie game, then $V_R = 0$, whereas $\mathbf{p} = (0.75, 0.25)$ and $\mathbf{q} = (0.6, 0.4)$ gives $V_R = 0.45 - 0.55 = -0.1$.

2.3 Minimax-Optimal Strategies

What does the row player want to do? She wants to find a vector $\mathbf{p}^\star$ that maximizes the expected payoff to her, over all choices of the opponent's strategy $\mathbf{q}$. The mixed strategy that maximizes the minimum payoff. I.e., the row player wants to find

$$\mathsf{lb} := \max_{\mathbf{p}} \min_{\mathbf{q}} V_R(\mathbf{p}, \mathbf{q})$$

Make sure you parse this correctly:

$$\mathsf{lb} := \overbrace{\max_{\mathbf{p}} \underbrace{\min_{\mathbf{q}} V_R(\mathbf{p}, \mathbf{q})}_{\text{payoff when opponent plays her optimal strategy against our choice } \mathbf{p}}}^{\text{mixed strategy that maximizes the minimum expected payoff}}$$

Loosely, *the row player can guarantee to herself this much payoff no matter what the column player does.* The quantity lb is a **lower bound on the row-player's payoff**.

What about the column player? She wants to find some $\mathbf{q}^\star$ that maximizes her own expected payoff, over all choices of the opponent's strategy $\mathbf{p}$. She wants to optimize

$$\max_{\mathbf{q}} \min_{\mathbf{p}} V_C(\mathbf{p}, \mathbf{q})$$

But this is a zero-sum game, so this is the same as

$$\max_{\mathbf{q}} \min_{\mathbf{p}} (-V_R(\mathbf{p}, \mathbf{q}))$$

And pushing the negative sign through, we get the column player is trying to optimize her own worst-case payoff, which is

$$-\min_{\mathbf{q}} \max_{\mathbf{p}} V_R(\mathbf{p}, \mathbf{q})$$

So the payoff in this case to the row player is

$$\mathsf{ub} := \min_{\mathbf{q}} \max_{\mathbf{p}} V_R(\mathbf{p}, \mathbf{q})$$

The column player can guarantee that the row player does not get more than this much payoff, no matter what the row-player does. This is an **upper bound on the row player's payoff**.

We have two quantities: lb and ub. How do they compare? To figure this out, we first make a simple but important observation: suppose we want to find the row player's minimax-optimal strategy $\mathbf{p}^\star$. Then we can assume that the column player plays a pure strategy (a single column). Why? Once the row player fixes a mixed strategy $\mathbf{p}$, the column player then has no reason to randomize: her payoffs will be some average of the payoffs from playing the individual columns, so she can just pick the best column for her. In other words, we get that the quantity ub can be equivalently defined as

$$\mathsf{lb} = \max_{\mathbf{p}} \min_{j} \sum_{i} p_i R_{ij}.$$

Similarly, the column player wants to optimize

$$\mathsf{ub} = \min_{\mathbf{q}} \max_{i} \sum_{j} q_j R_{ij} = -\max_{\mathbf{q}} \min_{i} \sum_{j} q_j C_{ij}.$$

2.3.1 The Balanced Game Example

For the shooter-goalie game, we claim that the minimax-optimal strategies for both players is $(0.5, 0.5)$. How can we calculate this?

> **Row Player:** For the row player (shooter), suppose $\mathbf{p} = (p_1, p_2)$ is the mixed strategy. Note that $p_1 \geq 0, p_2 \geq 0$ and $p_1 + p_2 = 1$. So it is easier to write the strategy as $\mathbf{p} = (p, 1-p)$ with $p \in [0, 1]$.
>
> OK. If the column player (goalie) plays L, then this strategy gets the shooter a payoff of
>
> $$p \cdot (-1) + (1-p) \cdot (1) = 1 - 2p.$$
>
> If the column player (goalie) plays R, then this strategy gets the shooter a payoff of
>
> $$p \cdot (1) + (1-p) \cdot (-1) = 2p - 1.$$

So we want to choose some value $p \in [0,1]$ to maximize

$$\mathsf{lb} = \min(1-2p, 2p-1)$$

In this case, this maximum is achieved at $p = 1/2$. (One way to see it is by drawing these two lines.) And the minimax-optimal expected payoff to the shooter is 0.

Column Player: The calculation for the column player (goalie) is very similar in this case. The minimax-optimal strategy for the goalie is also $(0.5, 0.5)$ and the guarantees that the shooter cannot make more than 0 payoff.

An observation: the shooter can guarantee a payoff of $\mathsf{lb} = 0$, and the goalie can guarantee that the shooter's payoff is never more than $\mathsf{ub} = 0$. Since $\mathsf{lb} = \mathsf{ub}$, in this case the *"value of the game"* is said to be $\mathsf{lb} = \mathsf{ub} = 0$.

2.3.2 The Asymmetric Goalie Example

Let's change the game slightly. Suppose the goalie is weaker on the left. What happens if the payoff matrix is now:

	L	R
shooter L	$(-\frac{1}{2}, \frac{1}{2})$	$(1, -1)$
R	$(1, -1)$	$(-1, 1)$

Row Player: For the row player (shooter), suppose $\mathbf{p} = (p, 1-p)$ is the mixed strategy, with $p \in [0,1]$. If the column player (goalie) plays L, then this strategy gets the shooter a payoff of

$$p \cdot (-1/2) + (1-p) \cdot (1) = 1 - (3/2)p.$$

If the column player (goalie) plays R, then this strategy gets the shooter a payoff of

$$p \cdot (1) + (1-p) \cdot (-1) = 2p - 1.$$

So we want to choose some value $p \in [0,1]$ to maximize

$$\mathsf{lb} = \min(1 - (3/2)p, 2p - 1)$$

In this case, this maximum is achieved at $p = 4/7$. And the minimax-optimal expected payoff to the shooter is $1/7$. Note that the goalie being weaker means the shooter's payoff increases.

Column Player: What about the calculation for the column player (goalie)? If her strategy is $\mathbf{q} = (q, 1-q)$ with $q \in [0,1]$, then if the shooter plays L then the shooter's payoff is

$$q \cdot (-1/2) + (1-q) \cdot (1) = 1 - (3/2)q.$$

If she plays R, then it is $2q - 1$. So the goalie will try to minimize

$$\mathsf{ub} = \max(1 - (3/2)q, 2q - 1)$$

which will again give $(4/7, 3/7)$ and guarantees that the expected loss is never more than $1/7$.

Again, the shooter guarantees a payoff of $\mathsf{lb} = 1/7$, and the goalie can guarantee that the shooter's payoff is never more than $\mathsf{ub} = 1/7$. In this case the value of the game is said to be $\mathsf{lb} = \mathsf{ub} = 1/7$.

Exercise 1: What if both players have somewhat different weaknesses? What if the payoffs are:

```
(-1/2, 1/2) (3/4, -3/4)
(1, -1)     (-3/2, 3/2)
```

Show that minimax-optimal strategies are $\mathbf{p} = (2/3, 1/3), \mathbf{q} = (3/5, 2/5)$ and value of game is 0.

Exercise 2: For the game with payoffs:

```
(-1/2, 1/2) (3/4, -3/4)
(1, -1)     (-2/3, 2/3)
```

Show that minimax-optimal strategies are $\mathbf{p} = (\frac{4}{7}, \frac{3}{7}), \mathbf{q} = (\frac{17}{35}, \frac{18}{35})$ and value of game is $\frac{1}{7}$.

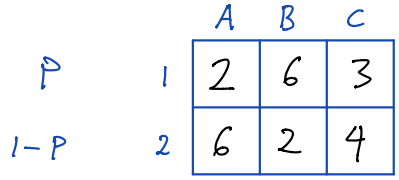
Exercise 3: For the game with payoffs:

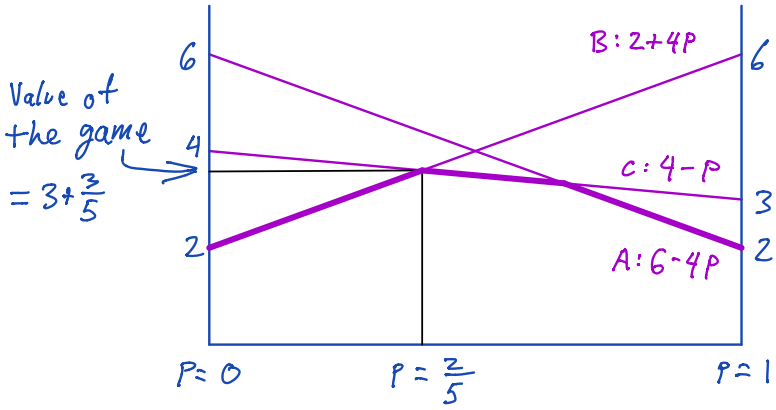
```
(-1/2, 1/2)    (-1, 1)
(1, -1)        (2/3, -2/3)
```

Show that minimax-optimal strategies are $\mathbf{p} = (0, 1), \mathbf{q} = (0, 1)$ and value of game is $\frac{2}{3}$.

3 Solving 2-Row Games

In this section we give a general method for solving games with two rows (or, by symmetry, two columns). We'll develop the method with the following example:





The game we'll analyze is shown in the 2×3 matrix above with values for the row player. We'll call the three options for the column player A, B, and C, and the two options for the row player 1 and 2.

The general mixed strategy for the row player is to choose option 1 with probability p and option 2 with probability $1 - p$.

In the diagram above, the horizontal axis represents p, and the vertical axis represents the payoff to the row player. The three lines in the diagram correspond to the three options for the column player. For example, the line labeled B, which is the graph of the function $2 + 4p$, is the expected value of the game for the row player (as a function of p) if the column player chooses column B.

The lower envelope of the three options for the column player is highlighted. This concave function represents the expected pay-off for the row player if he chooses option 1 with probablility p, assuming that the column player plays optimally knowing the value of p. Thus, it represents a lower bound on the value of the game for the row player for each value of p.

By inspection of this graph, the concave function achieves its maximum value at the point of intersection between lines B and C. This point is $p = \frac{2}{5}$, and with that choice of p the value of the game is $3 + \frac{3}{5}$. That's the lower bound on the game's value.

What's a good strategy for the column player? Consider the convex combination of B and C chosen such that the result is a horizontal line. This is $\frac{4}{5}C + \frac{1}{5}B$. With this mixed strategy, the value of the game for the column player is $3 + \frac{3}{5}$, no matter what the row player does. So this is an upper bound on the value of the game for the row player. Since the lower and upper bound are equal, we know that this is the value of the game.

The same technique can be applied to any game with two rows. The value of the game is obtained by constructing the concave function and finding where it achieves its maximum.

4 Von Neumann's Minimax Theorem

In all the above examples of 2-player zero-sum games, we saw that the row player has a strategy $\mathbf{p}^\star$ that guarantees some payoff lb for her, no matter what strategy $\mathbf{q}$ the column player plays. And the column player has a strategy $\mathbf{q}^\star$ that guarantees that the row player cannot get payoff more than ub, no matter what strategy $\mathbf{p}$ the row player plays. The remarkable fact in the examples was that $\mathsf{lb} = \mathsf{ub}$ in all these cases! Was this just a coincidence? No: a celebrated result of von Neumann[3] shows that we always have $\mathsf{lb} = \mathsf{ub}$ in (finite) 2-player zero-sum games.

Theorem 1 (Minimax Theorem (von Neumann, 1928)) *Given a finite 2-player zero-sum game with payoff matrices $R = -C$,*

$$\mathsf{lb} = \max_{\mathbf{p}} \min_{\mathbf{q}} V_R(\mathbf{p}, \mathbf{q}) = \min_{\mathbf{q}} \max_{\mathbf{p}} V_R(\mathbf{p}, \mathbf{q}) = \mathsf{ub}.$$

This common value is called the value of the game.

The theorem implies that in a zero-sum game, both the row and column players can even "publish" their minimax-optimal mixed strategies (i.e., tell the strategy to the other player), and it does not hurt their expected performance as long as they play optimally.[4]

Von Neumann's Minimax Theorem is an important result in game theory, but it has beautiful implications to computer science as well — as we see in the next section.

5 Lower Bounds for Randomized Algorithms

In order to prove lower bounds, we thought of us coming up with algorithms, and the adversary coming with some inputs on which our algorithm would perform poorly—take a long time, or make many comparisons, etc. We can encode this as a zero-sum game with row-player payoff matrix R. (Keep sorting as an example application in your mind.)

[3] John von Neumann, mathematician, physicist, and polymath.

[4] It's like telling your rock-paper-scissors opponent that you will play each action with equal probability, it does not buy them anything to know your strategy. This is not true in general non-zero-sum games; there if you tell your opponent the mixed-strategy you're playing, she may be able to do better. Note carefully that you are not telling them the actual random choice you will make, just the distribution from which you will choose.

- The columns are various algorithms for the problem (for sorting n elements).
- The rows are all the possible inputs (all $n!$ of them).
- The entry R_{ij} is the cost of the algorithm j on the input i (say the number of comparisons).

This may be a huge matrix, but we'll never write it down. It's just a conceptual guide. But what does the matrix tell us? A lot, as it turns out:

- A deterministic algorithm with good worst-case guarantee is a column that does well against all rows: all entries in this column are small.
- A randomized algorithm with good expected guarantee is a probability distribution $\mathbf{q}$ over columns, such that the expected cost for each row i is small. This is a mixed strategy for the column player. It gives an upper bound.
- Ideally we want to find the minimax-optimal distribution $\mathbf{q}^\star$ achieving the value of this game. This would be the best randomized algorithm.
- What is a lower bound for randomized algorithms? It is a mixed-strategy over rows (a probability distribution $\mathbf{p}$ over the inputs) such that for every column (i.e., deterministic algorithm j), the expected cost of j (under distribution $\mathbf{p}$) is high.

  So to prove a lower bound for randomized algorithms, if suffices to show that lb is high for this game. I.e., give a strategy for the row player (a distribution over inputs) such that every column (deterministic algorithm) incurs a high cost on it.

5.1 A Lower Bound for Sorting Algorithms

Recall from Lecture 2 we showed that any deterministic comparison-based sorting algorithm must perform $\log_2 n! = n \log_2 n - O(n)$ comparisons in the worst case. The next theorem extends this result to randomized algorithms.

Theorem 2 *Let $\mathcal{A}$ be any randomized comparison-based sorting algorithm (that always outputs the correct answer). Then there exist inputs on which $\mathcal{A}$ performs $\lceil \log_2 n! \rceil - 1$ comparisons in expectation.*

Proof: Suppose we construct a matrix R as above, where the columns are possible (deterministic) sorting algorithms for n elements, the rows are the $n!$ possible inputs, and entry R_{ij} is the number of comparisons algorithm j makes on input i. We claim that the value of this game is at least $\lceil \log_2 n! \rceil - 1$: this implies that the best distribution over columns (i.e., the best randomized algorithm) must suffer at least this much cost on some column (i.e., input).

To show the value of the game is large, we show a probability distribution over the rows (i.e., inputs) such that the expected cost of every column (i.e., every deterministic algorithm) is at least $\lceil \log_2 n! \rceil - 1$.

This probability distribution is the uniform distribution: each of the $n!$ inputs is equally likely. Now consider any deterministic algorithm: as in Lecture 2, it can be transformed into a decision tree with $n!$ leaves. No two distinct input permutations go to the same leaf. We will now show that the average depth of a leaf in this tree is at least $\lceil \log_2 n! \rceil - 1$.

Consider such a decision tree T. Each node x in T has has a depth $D(x)$, which is defined as the number of nodes (not counting x) on the path from the root to x. Suppose there were two leaves ℓ_1 and ℓ_2 where $D(\ell_2) \geq D(\ell_1) + 2$. That is, leaf ℓ_2 is at least two deeper than leaf ℓ_1. Let p be the parent of ℓ_2 in T. So $D(p) \geq D(\ell_1) + 1$.

If we were to modify T by swapping ℓ_1 with p in T, the resulting tree T' has average leaf depth less than that of T. This is because the swap moves ℓ_1 deeper but moves p shallower by the same amount. The subtree rooted at p has at least two leaves in it. So the average depth decreases as a result of this swap.

We can continue this process (always decreasing the average leaf depth) until there is no pair of leaves whose depth differs by two or more. So all leaves are at neighboring depths. Call these depths $K-1$ and K. In order for there to be $n!$ leaves, we know that $2^K \geq n!$. Thus $K \geq \lceil \log_2 n! \rceil$. And all leaves are of depth at least $K-1$. This proves the result.

■

Note: The following weaker theorem and proof were presented in lecture. They're included here for reference purposes.

Theorem 3 *Let $\mathcal{A}$ be any randomized comparison-based sorting algorithm (that always outputs the correct answer). Then there exist inputs on which $\mathcal{A}$ performs $\Omega(\lg n!)$ comparisons in expectation.*

Proof: Suppose we construct a matrix R as above, where the columns are possible (deterministic) sorting algorithms for n elements, the rows are the $n!$ possible inputs, and entry R_{ij} is the number of comparisons algorithm j makes on input i. We claim that the value of this game is $\Omega(\lg n!)$: this implies that the best distribution over columns (i.e., the best randomized algorithm) must suffer at least this much cost on some column (i.e., input).

To show the value of the game is large, we show a probability distribution over the rows (i.e., inputs) such that the expected cost of every column (i.e., every deterministic algorithm) is $\Omega(\lg n!)$.

This probability distribution is the uniform distribution: each of the $n!$ inputs is equally likely. Now consider any deterministic algorithm: as in Lecture #2, this is a decision tree with at least $n!$ leaves. No two inputs go to the same leaf.

In this tree, how many leaves can have depth at most $(\lg n!) - 10$? At most the number of nodes at depth at most $(\lg n!) - 10$ in a complete binary tree. Which in turn is

$$1 + 2 + 4 + 8 + \ldots + 2^{(\lg n!)-10} \leq 1 + 2 + 4 + 8 + \ldots + \tfrac{n!}{1024} \leq \tfrac{n!}{512}$$

So $\frac{511}{512} \geq 0.99$ fraction of the leaves in this tree have depth more than $(\lg n!) - 10$. In other words, if we pick a random input, it will lead to a leaf at depth more than $(\lg n!) - 10$ with probability 0.99. Which gives the expected depth of a random input to be $\geq 0.99((\lg n!) - 10) = \Omega(\lg n!)$. ■

6 General-Sum Two-Player Games*

In general-sum games, we don't deal with purely competitive situations, but cases where there are win-win and lose-lose situations as well. For instance, the coordination game of "chicken", a.k.a. *what side of the street to drive on?* It has the payoff matrix:

Note that we are now using the convention that a player choosing L is driving on *their* left. Note that if both players choose the same side, then both win. And if they choose opposite sides, both crash and lose. (Both players can choose to drive on the left—like Britain, India, etc.—or both on

payoff matrix M	Bob L	Bob R
Alice L	$(1,1)$	$(-1,-1)$
Alice R	$(-1,-1)$	$(1,1)$

the right, like the rest of the world, but they must coordinate. Both these are stable solutions and give a payoff of 1 to both parties.)

Consider another coordination game that we call "which movie?" Two friends are deciding what to do in the evening. One wants to see *Citizen Kane*, and the other *Dumb and Dumber*. They'd rather go to a movie together than separately (so the strategy profiles C, D and D, C have payoffs zero to both), but C, C has payoffs $(8, 2)$ and D, D has payoffs $(2, 8)$.

payoff matrix M	Bob C	Bob D
Alice C	$(8,2)$	$(0,0)$
Alice D	$(0,0)$	$(2,8)$

Finally, yet another game is "Prisoner's Dilemma" (or "to pollute or not?") with the payoff matrix:

payoff matrix M	Bob collude	Bob defect
Alice collude	$(2,2)$	$(-1,3)$
Alice defect	$(3,-1)$	$(0,0)$

6.1 Nash Equilibria

In this case, a good notion is to look for a *Nash Equilibrium*[5] which is a stable set of (mixed) strategies for the players. Stable here means that given strategies $(\mathbf{p}, \mathbf{q})$, neither player has any incentive to unilaterally switch to a different strategy. I.e., for any other mixed strategy $\mathbf{p}'$ for the row player

$$\text{row player's new payoff } = \sum_{ij} p'_i q_j R_{ij} \leq \sum_{ij} p_i q_j R_{ij} = \text{ row player's old payoff}$$

and for any other possible mixed strategy $\mathbf{q}'$ for the column player

$$\text{column player's new payoff } = \sum_{ij} p_i q'_j C_{ij} \leq \sum_{ij} p_i q_j C_{ij} = \text{ column player's old payoff.}$$

Here are some examples of Nash equilibria:

- In the chicken game, both $\{\mathbf{p} = (1,0), \mathbf{q} = (1,0)\}$ and $\{\mathbf{p} = (0,1), \mathbf{q} = (0,1)\}$ are Nash equilibria, as is $\{\mathbf{p} = (\frac{1}{2}, \frac{1}{2}), \mathbf{q} = (\frac{1}{2}, \frac{1}{2})\}$.
- In the movie game, the only Nash equilibria are $\{\mathbf{p} = (1,0), \mathbf{q} = (1,0)\}$ and $\{\mathbf{p} = (0,1), \mathbf{q} = (0,1)\}$.

[5] Named after John Nash: CMU graduate, mathematician, and Nobel prize winner.

- In prisoner's dilemma, the only Nash equilibrium is to defect (or pollute). So we need extra incentives for overall good behavior.

It is easy to come up with games where there are no stable *pure* strategies—this is even true for zero-sum games. But what about mixed-strategies? The main result in this area was proved by Nash in 1950 (which led to his name being attached to this concept)

Theorem 4 (Existence of Stable Strategies) *Every finite player game (with each player having a finite number of strategies) has at least one (mixed-strategy) Nash equilibrium.*

This theorem implies the Minimax Theorem (Theorem 1) as a corollary: indeed, take any two-player zero-sum game and consider any Nash equilibrium $(\mathbf{p}, \mathbf{q})$, with value $V = \sum_{ij} p_i q_j R_{ij} = -\sum_{ij} p_i q_j C_{ij}$. Since this is stable, neither player can do better by deviating, even knowing the other player's strategy. So they must be playing minimax-optimal.