MODELLING INCENTIVES IN COMPUTER SCIENCE

1.The Draw and College Admissions

The Draw

1. Each student submits a ranked list. Each item in the list is a type of room in a specific dorm or house, and they are ordered from most preferred (at the top) to least preferred. (No ties are allowed.) This is how you express your preferences—that is, what you want—to the algorithm implementing the Draw.
2. Each student is assigned a number in {1, 2, . . . , 3500}.
3. For i = 1, 2, . . . , 3500: (a) Student i is assigned to her favorite choice among the options still available. In effect, the algorithm goes down student i's ranked list (from top to bottom), giving her the first option that is available.

Pareto optimal (adj.): the property of an outcome that you can't make anyone better off without making someone else worse off.

Proposition 1.1 The outcome of the Draw is Pareto optimal

strategyproof (synonym: truthful) (adj.): the property of a mechanism that honesty is always the best policy, meaning that lying about your preferences cannot make you better off.

Proposition 1.2 The Draw is strategyproof.

mechanism (n.): a procedure for making a decision or taking an action, as a function of what people want (i.e., of participants' preferences).

serial dictatorship (n.): a mechanism that orders the participants, and in this order allows each player to dictate their favorite feasible option (given the choices made by previous players).

two-sided market (n.): a market with two distinct groups of participants, each with their own preferences

2. Stable Matching

The Draw Mechanism for one-sided markets motivates us to look for a similar analogue for two-sided markets, to regulate processes like college admissions.

In other words, can there be a sensible mechanism to regulate the two sided market of college admissions, given:

- The set S of Students and the set C of Colleges
- Each student has a ranked list of preferences over colleges
- Each college has a ranked list of preferences over students
- Each college c has a capacity to accommodate p_c students
- $|S| = \sum p_c$

What exactly are we looking for?

Matching: An assignment of each student to exactly one college and each college c to exactly p_c students

We are looking for a mechanism which produces a matching, and that matching should have certain nice properties, so that both sides of the markets are satisfied by it.

Blocking pair: a pair (s, c_1) where $s \in S$ and $c \in C$ such that:

1. *s is assigned to $c_2 \neq c_1$*
2. *s prefers c_1 to c_2*
3. *c_2 prefers s to one of the students assigned to c_2*

The existence of a blocking pair in a matching is something we do not want, because that would mean that s and c_1 would both not be satisfied with the mechanism and are likely to negotiate outside of the system.

Stable Matching*: A matching with no blocking pair.*

Thus, we would like to find a mechanism that produces a Stable matching.

The Deferred Acceptance Algorithm:

***while** there is at least one unassigned student*

do

every unassigned student proposes to his favourite college that hasn't rejected him yet

every college keeps track of its favourite p_c proposals that it ever receives, and rejects all other proposals

all unrejected proposals are made final

Some properties of the DAA:

1. *The DAA pro*duces a matching.
2. The matching DAA produces is stable (and hence Pareto-Optimal).

Sometimes, more than one stable matchings are possible. How should we pick one of them?

Student-Optimality: A stable matching is said to be student-optimal if it matches every student $s \in S$ to c(s) where c(s) denotes the highest ranked (in s's preference list) college, to which s is matched in any stable matching.

3. *The DAA produces a Student Optimal Matching.*

Strategyproof-ness (for students): Students cannot get better colleges by lying in their preference lists.

4. The DAA is Strategyproof for students.

3. Markets, Everywhere

Markets from (Almost) the 21st Century:
Perhaps the most obvious example of a market that wouldn't exist without advances in Computer science is Amazon.
Example #1: Amazon.
There would be no Amazon without the Internet. There would be no Amazon without the World Wide Web. Amazon would not have quite the same reach without modern wireless Amazon's main contribution was to make an existing
market better. For example:
• More convenience. Open 24 hours a day, no travel required, all genres of products in the same place.
• More efficiency. Ideally, because of its scale, Amazon should be able to supply a given set of products at a relatively low cost, hopefully translating to lower prices for consumers.
Switching from brick-and-mortar shops to Amazon wasn't all upside, however. On Amazon, you generally don't know much about the seller you're buying from, while you may have a relationship with the proprietors at your local shop. Shopping on Amazon was potentially less safe than traditional shopping.

How about other examples of companies whose primary contribution was improving a market that already existed? Advertising companies like Google and Facebook are good examples.
Example #2: Google; Facebook.
Maybe you think of Google and Facebook as a search engine and a social network, respectively. But if you I but the rise of platforms like Google and Facebook allowed ads to be targeted at a far finer level of granularity than had ever been possible before.

What about ride-sharing companies?
Example #3: Uber; Lyft.
To the extent that there was a ride-sharing market before Uber and Lyft, it was small, consisting of taxis plus various ad hoc arrangements.Thus Uber and Lyft really created a new market, or at least grew an existing market by an order of magnitude or more. Uber started with limos, which made sense, as limo drivers could traditionally only be booked by appointment, and as a result often had long idle periods between rides. Many of these drivers were happy to earn some extra money through Uber during these idle times. Of course, it's now clear that the ride-sharing market is much bigger than was initially thought, with lots of amateur drivers happy to lend their time and car part-time to pick up some extra cash.

There are many more examples of the same type.
Example #4: Airbnb. Airbnb opened up the market for spare rooms.8
Example #5: eBay. These days, you might regard eBay primarily as a smaller and less organized version of Amazon. But when eBay debuted back in 1995, it looked rather different, like the union of all the world's garage sales happening at the same time and place. Thus eBay first rose to prominence by opening up the market for second-hand bric-a-brac.
Example #6: StubHub. StubHub opened up the market for second-hand tickets to events

like concerts and sporting contests. This was not a new market—there's always been ticket scalpers and ticket-resellers—but StubHub grew the market by making it more convenient and in many cases safer.
Example #7: Upwork, TaskRabbit, Postmates, Fiverr, etc. These companies brought parts of the labor market online (especially temp jobs and freelancing). Again, there's always been freelancing, but not at today's scale.
Example #8: Tinder, etc. There's always been a market for dates, but it's never been as explicit as it is now

Centralized vs. Decentralized Markets
A market is centralized if the transactions that get executed are dictated by a third party. The Draw and the Deferred Acceptance algorithm, correspond to centralized markets (all matches are centrally computed by the mechanism, and all participants can do is report their preferences). Conversely, in a decentralized market, participants are free to transact directly. In the examples #1-#8 mentioned above, mostly, they are decentralized, but with one exception: Uber and Lyft. Both manage a two-sided market of drivers and riders, with every match centrally dictated by the platform. Drivers can to some extent pick which ride assignments to accept, but as a rider, you have to accept the driver assigned to you or face a cancellation penalty.
Markets are often somewhere in between centralized and decentralized. Even a seemingly decentralized market like Amazon influences the transactions you engage in via its (centralized) search functionality and user interface. More generally, an oft-used middle ground between the two extremes is for a third party (like the platform) to recommend several transactions, perhaps based on the data it has about a user and its possible transaction partners, while users remain free to pick any or none of them. (Uber/Lyft is the extreme case where only one transaction is recommended, and users cannot engage in a non-recommended transaction.)

Batched transactions
A second way to interpolate between the extreme points of decentralization and centralization is to *batch* subsets of transactions for centralized processing. Uber is one such market which realised the importance of this algorithm in 2017 and changed its manner of functioning. Before 2017, Uber was more on a greedy side. It collected information about the rider and assigned him the nearest available driver. In this mechanism, riders might have to wait for a longer duration. But if we take an interval of 1 second to save the data of riders and drivers, we may find a ride which reaches sooner than the previous one. This grouping of information is called batching.

New York stock exchange (NYSE) again seeks some need of batching. Here, the high frequency trading appears to be unjustified for some. The people who are in proximity of the main transaction executing server find themselves on an advantageous side. To give one example of how this comes up, imagine that you submit an order to buy one share of Facebook stock at a price of $200. Suppose the order remains outstanding, say because all standing sell orders are at prices $210 or higher. Now imagine that the Cambridge Analytica scandal hits, and you suspect that Facebook's stock price is about to drop dramatically. The race is on: you want to cancel your standing buy order, while opportunistic sellers want to beat you to it and clear your order (at the now-lucrative price of $200). Whether or not the transaction gets executed depends on who gets to the market first.

Could we redesign the NYSE to mitigate the incentive to engage in high-frequency trading? Budish, Cramton, and Shim propose an elegant solution: to collect transactions into batches and compute all the trades in a batch at the same time. For example, imagine that we break time into one-second intervals. At the end of each second, the exchange considers all the standing buy and sell orders (possibly submitted during the preceding second, or possibly submitted earlier and not yet executed or canceled), and removes any orders that were canceled in the preceding second. With the remaining buy and sell orders, the exchange executes as many trades as possible, all at a common price, subject to the buyers paying no more than their bid and sellers receiving at least their ask.

Types of market failure

There are plenty of things which can go wrong in the market. We have discussed four of them:

Timing issue :- NYSE is a good example of this problem as we have seen earlier. On the NYSE even a fraction of a second can influence the market. Imagine if Stanford and other schools start admitting students on a first come first basis then what will happen? What will the reputation of schools remain in the next five years??

Safety issues :- This issue arises when customers do not feel safe to do transactions with the market. It becomes a serious issue for the markets which engage in face-to-face meetings like dating platforms.

Not thick enough :- If a customer finds it's desired trading partner easily then that market is considered as thick and otherwise it is thin. Amazon and ebay are good examples of a thick market.

The main reason behind the thin market is that not enough people show up or one can find it's desirable trading partner easily. In this case technology , your design, keeping transaction cost minimum , offering transactions to multiple parties at once can help you to solve this problem.

Despite all this, thickness is not always good. Let's say you are selling a car and the market is thick now from the buyer's perspective he is happy because he is getting more options and definitely he would end up getting a good deal but from the seller perspective he is unhappy because the chance of losing a sale to his competitor increased.

Congestion :- This type of problem arises when a customer has too many options for his trade.For a centralized market, as the market grows bigger and thicker, Formulating and communicating preferences definitely gets harder for the participants,the more options you have to rank, the more onerous the task.For decentralization, let's say you are buying shoes from amazon then you have plenty of choices and one can not simply go through all of them and compare them to find his best deal but a good search functionality or a good recommendation system can solve this problem. The problem gets harder for some markets or platforms like dating. In dating platforms famous profiles get all the attention.A natural idea is to push back against thickness and low transaction costs, so that an offer to transact becomes meaningful again-hunting for a sweet spot between the thinness and congestion problems. One possible solution for this problem is signaling.

Digression

Network Effects :- The main reason behind congestion or thinness of market is network effect. It is the effect that one user of a good or service has on the

value of that good or service to its other users. Most of the time the network effect is assumed to be good. For example we never get worried about the less no. of social media platforms because we want a platform that everyone is using and this leads to a single platform getting much more attention that is suffering from congestion and others suffering from thin-ness.

Antitrust Regulation :- It is Digression within digression. Sometimes the government imposes regulations on tech companies because when there are strong network effects, a platform with many users can exploit its users' high switching costs by failing to innovate, raising prices, or engaging in anticompetitive behavior. In 2001 U.S imposed charges on Microsoft they claimed that Microsoft was exploiting its dominance in the operating systems market to also dominate the Internet browser market by bundling Internet Explorer with Microsoft's operating system.

Signaling

Signaling is a solution to the congestion problem where participants are overwhelmed by the number of options. One approach to exporting more information and guidance to participants is to allow a limited amount of signaling. For example, you can signal your qualifications in a college application (grades, test scores, letters, etc.).

For example in dating platforms the most popular profiles receive far too many messages to reply to. So one can play a smart game by sending messages to not so famous profiles in order to get a response from them but this generates a new problem, some profiles can slip through the cracks with no response from anybody who incorrectly guessed that the applicant was out of their league. In order to solve this problem Korean dating site did an experiment, users were given two "roses" per week. A user could send a large number of messages, but could attach a rose to at most two of them.The roses were especially effective when sent to someone with a similar or non-famous profile (who might otherwise think the sender was out of their league). This experiment increases the probability of message acceptance (from 15% to 18%, averaged over all cases), and also increases the overall number of messages.

4. Strategic Voting

1. Voting in Computer Science:
    A. Rank Aggregation- Here, voters correspond to different ranking algorithms. Consider a prediction problem where the goal is to come up with a ranked list. Imagine you have several different heuristics for producing such ranked lists. Maybe one heuristic is based on anchor text, another on link structure, and another on the page content. It's natural to try to combine these different and likely conflicting rankings into a "consensus ranking." The hope is that the consensus ranking is somehow "better" or "more accurate" than the individual rankings
    B. Crowdsourcing- Here, voters correspond to "workers" (in crowdsourcing terminology). Suppose each worker produces a ranking — maybe you showed them three different candidate user interfaces and asked about their most and least favorite options. A natural goal is to summarize the workers' results with a single ranking.
    C. Participatory democracy- The goal in participatory democracy is to get more people involved in government decisions, especially at the local level

2. Examples of Voting Rules
    a. Formalism- Let A denote a set of alternatives. Each voter i has her own ranked list over the alternatives in A, and declares some ranked list L_i to a voting mechanism. (L_i may or may not be a voter i's true preferences—the mechanism has no way of knowing.) A voting rule can then have two different forms, depending on whether its output is a full ranked list or just a single alternative (the "winner"). So it's either a map of the form;
    $$L_1, L_2, \ldots, L_n \rightarrow L^*,$$
    where L* is a ranking of A,or a map;
    $$L_1, L_2, \ldots, L_n \rightarrow a^*,$$
    where $a^* \in A$.

    b. Majority Rule- Suppose there are only two alternatives ($|A| = 2$). One obvious voting rule is majority vote: elect that alternative that appears first in the largest number of voters' lists. (If there is an even number of voters and a tie, break the tie in some canonical way, perhaps randomly.) The majority vote rule is strategyproof in this case.

    c. Plurality Rule- Suppose now that $|A| \geq 3$, as is the case in many applications. plurality rule: which elects the candidate with the most first-place votes. (With this result, a voter does not need to submit her full ranked list; just her top choice suffices.) The plurality rule is not strategyproof. More generally, the plurality rule tends to be biased toward "extreme" candidates.

    d. Ranked-Choice Voting- For ranked-choice voting, voters submit a full ranked list. First, if there is some alternative a* that receives more than 50% of the first-choice votes, then a* is the winner. Otherwise, the alternative with the

fewest first-choice votes is deleted and the winner is computed recursively on the remaining alternatives. Eventually, there will be only two alternatives remaining at which point the rule is the same as the majority rule.
Ranked-choice voting is not strategyproof. The intuition is that there can be an incentive to influence who gets eliminated early on, so that your preferred candidate gets more favored matchups in later rounds.

e. The Borda Count- Voters submit their full ranked lists. An alternative gets |A| points for each first-choice vote, |A| − 1 points for each second-choice vote, and so on, with 1 point for each last-choice vote. The Borda count rule also fails to be strategyproof. For example, you might want to rank the closest competitor to your preferred alternative last.

3. The Gibbard-Satterthwaite and Arrow Impossibility Theorems
   - Dictator: a dictator rule has a dictator voter i, and always elects i's first choice.
   - Duple: a duple rule chooses a pair a, b ∈ A of alternatives (independent of voters' preferences), and runs a majority vote between a and b.

   Theorem 3.1 (Gibbard-Satterthwaite Theorem) Every strategyproof voting rule that can produce at least three different outcomes is a dictator.

   Theorem 3.2 (Arrow's Impossibility Theorem) With three or more alternatives, no voting rule satisfies the following three properties:
   1. Non-dictatorship- It just means that there is no dictator (a voter i such that the output of the voting rule is always the same as i's ranked list).
   2. Unanimity - It means that if every voter ranks a over b, then the voting rule should also rank a over b.
   3. Independent of irrelevant alternatives (IIA) - It means that, for every pair a, b of alternatives, the relative order of a, b in the produced ranking should be a function only of the relative order of a, b in each voter's list, and not depend on the position of any "irrelevant" alternative c in anyone's preferences.

4. A Tractable Special Case: Single-Peaked Preferences
   Suppose the set of alternatives is the unit interval [0, 1]. A voter i has single-peaked preferences if there is a "peak" pi ∈ [0, 1] such that whenever z is farther from pi than y (meaning z < y < pi or z > y > pi), the voter prefers y to z.
   Suppose each voter votes by providing a reported peak xi ∈ [0, 1].
   One idea is to choose the average, 1 n Pn i=1 xi . But note that this is not strategyproof: a voter might be able to pull the chosen outcome closer to her peak by reporting an overly extreme peak.
   A second and better idea is to choose the median of the reported peaks. (For simplicity, assume there is an odd number of voters.) The median voting rule is strategyproof. The only way a voter can manipulate the median is to report a peak on the opposite side of the median from her true peak. But this can only pull the median farther away from her true peak, a worse outcome for her.

5. Voting, Machine Learning, and Participatory Democracy

1. Voting Rules as Maximum Likelihood Estimators

A. Two Interpretations-

1. *As a consensus between different subjective opinions.* Here, voters have genuinely different preferences, and a voting rule is a method for aggregating them. With this interpretation, voters are motivated to influence the output of the voting rule, and strategy proofness is a relevant property.
2. *To recover the "ground truth" from noisy, imperfect estimates.* Here, voters are effectively cooperating in a quest for the truth. There is an objectively correct answer and the voters do their best to figure it out but inevitably make errors. Under this interpretation, strategyproofness is not a relevant property.

B. Digression on Regression-

To see how the analogy might work, we review a canonical machine learning problem, linear regression. The setup is: we are given n data points $(x_1, y_1), \ldots ,(x_n, y_y)$, where the x_i 's are points in R^d and the y_i 's are real values. The goal is to compute a "best-fit line"(Or more generally, for d > 1, the best-fit affine subspace.). This is sort of like a voting problem, where each data point is effectively voting for a line that passes through it. Ordinary least squares is a canonical choice of how to define the "best-fit line," as the one minimizing the total squared vertical distance:

$$\sum_{i=1}^{n}\left(\sum_{j=1}^{d} ajxij \;-\; yi\right)^2$$

where $(a_1, \ldots , a_d, b)$ are the coefficients specifying the subspace (a line, if d = 1).

Notion of "best fit" over others

Technical convenience- the objective function is convex and can be minimized by gradient descent (for example), and there is even a closed-form solution.

Probabilistic interpretation- For clarity, we assume that d = 1. Suppose there is a "ground truth" line a ∗x + b—the relationship that we're trying to learn. For data points $x_1, \ldots , x_n$, assume that each label y_i has the form $y_i = a*x_i + b + \epsilon_i$, where theϵ_i's are drawn i.i.d. from a Gaussian distribution with mean 0. Now suppose that, given $(x_1, y_1), \ldots ,(x_n, y_n)$, we reverse engineer "best guess" of the unknown ground truth. This is known as *the maximum likelihood solution.*That is, the maximum likelihood solution is the solution to $\max_{a,b}$**Pr**[data | *a, b*].

The punchline is: Ordinary least squares computes the maximum likelihood solution—the most likely linear relationship between the xi 's and the yi 's, given our assumption about how the data is generated.

2. Majority Vote as a Maximum Likelihood Estimator

Condorcet was the first to give a probabilistic justification of a voting rule, specifically for the majority rule when there are only two alternatives (A = {a, b}).

Formal model: Assume that one of the two alternatives is the “correct” one (i.e., the ground truth), and that each voter independently votes for the correct alternative with probability p and the incorrect alternative with probability 1 − p, where p ∈ (1/2 , 1] is a parameter. Thus, a voter is more likely to be correct than incorrect.

Suppose the data consists of k votes with a on top, and n − k votes with b on top. (So n voters total, and for simplicity assume that n is odd) If the ground truth is that a is the better alternative, then the probability of seeing this data:

$$\binom{n}{k} p^k (1-p)^{n-k}$$

since there are (n k) choices for the subset of k correct votes, and given such a choice, all of the allegedly correct votes really should be correct (this has probability p^k) and similarly for the incorrect votes (probability $(1 - p)^{n-k}$).

Similarly, if b is the better alternative, then the probability of seeing the given data is

$$\binom{n}{k} p^{n-k} (1-p)^k$$

Since p >1/2 , the first expression is bigger than the latter if and only if k > n − k, or equivalently k > n/2 . That is, alternative a is the maximum likelihood estimator if and only if it is the winner by majority vote. This argument provides a justification of the majority rule.

3. The Kemeny Voting Rule

A. The Maximum Likelihood Estimator for Independent Pairwise Comparisons

4. Participatory Budgeting

1. k-Approval Voting

The systems currently in place typically use “k-approval voting” — each voter is told the overall budget and a list of project descriptions with costs, and the voter picks their k favorite projects, with no ordering between them.

k-Approval Voting
1. Each voter votes for at most k projects. 2. Sort the projects in decreasing order of number of votes. 3. Fund the maximal prefix such that the total cost is at most the budget B.

A drawback with k-approval voting is that voters need not take into account projects’ costs, which results in more expensive projects being overrepresented.

Given that a voter can only vote for one project, it is possible (even likely) that she will vote for the single project for which she has the most value. This

would result in everybody voting for the first project, a suboptimal result. Thus the outcome of straightforward voting in the k-approval scheme need not be Pareto optimal. Approval voting is also not strategyproof.

2. Knapsack Voting

We next take a glimpse into the current state-of-the-art, and describe one current prototype for a replacement: knapsack voting. The idea is to allow a voter to approve any number of projects, as long as their total cost is at most the budget.

Knapsack Voting
1. Each voter votes for a subset Si of projects for which the total cost is at most the budget B. 2. Sort the projects in decreasing order of number of votes. 3. Fund the maximal prefix such that the total cost is at most the budget B.

For the proof, we also allow the final project considered—the most popular one that can't be fully funded—to be partially funded, so that the entire budget of B is spent. It may or may not be realistic to partially fund projects, but partial funding and insistence on spending all of one's budget are both common in practice. The intuition behind knapsack voting is that it forces voters to account for project costs— voting for more expensive projects decreases the number of projects that you can vote for— and hence should result in a better choice of projects. In the three-project example in the preceding section, straightforward knapsack voting results in the optimal choice (with the second and third projects getting funded).

3. Properties of Knapsack Voting

We'll strive for both strategyproofness and Pareto optimality guarantees, and this requires making fairly specific assumptions (otherwise impossibility results kick in). The two assumptions are:

1. Voter i has some set of projects $S * i$ that she wants to fund, with the total cost of S^*_i at most the budget B.
2. Voter i wants as much money as possible to be spent on the projects in S^*_i.

Thus the utility of voter i is

$$\sum_{j \in s^* i} [\text{money spent on project j}]$$

The definition implies that if a project (of S^*_i) is partially funded, then the utility earned is prorated accordingly. The most unrealistic aspect of this utility model is the extreme assumption that a voter i has absolutely no value for any project outside its preferred set $S * i$. Under the assumptions, knapsack voting has several good properties.

Proposition A. *With voter utilities as in (ii), knapsack voting is strategyproof, meaning that a player always maximizes her utility by voting for her true set* S^*_i .
Intuitively, misreporting your preferred set can only transfer funds from projects you do want to projects that you don't want.

Proposition B. *With voter utilities as in (ii), and assuming that voters report their true sets, knapsack voting results in a Pareto optimal choice of projects.*
Intuitively, this follows from the greedy nature of the way projects get funded, from most popular to least popular.

Proposition C. *Knapsack voting is the maximum likelihood estimator for a seminatural generative model of votes*.
The setup is reminiscent of the Mallows model for ranked lists. The model here is: there is a ground truth set S* of "correct" projects, and the probability of seeing a given vote S from a voter decays exponentially with the cost of the projects in S*\S, that is, with the amount of funds not spent on S*. Proposition C then states that knapsack voting is the maximum likelihood solution for this Mallows-type model..

5. Fair Division

1. Cake Cutting
1.1 Properties of the Cut and Choose Protocol
A reasonable protocol for two-person cake-cutting(the good, herein the cake, may be heterogeneous in nature):

The Cut and Choose Protocol
1. Player 1 splits the good into two pieces A and B, such that the player's value for each is exactly half that of the entire good. 2. Player 2 picks whichever of A, B she likes better.

Let us assume the cake is the interval [0,1] (in general we can assume the interval to be [a,b] but to make our work simpler we take a=0 and b=1). Now each player 'i' has its valuation function v_i, which specifies the value $v_i(S)$ of the subset S of the cake. For example, let player 1 divide the cake in 4 equal parts, so the subset S can be: [0,0.25], [0.25,0.5], [0.5,0.75], [0.75,1]. Note that the subsets are disjoint (no element common) and their union is equal to the interval [0,1]. Now we make two assumptions about each valuation v_i:

***Strategyproof (synonym: truthful) (adj.)**: the property of a mechanism that honesty is always the best policy, meaning that lying about your preferences cannot make you better off.*

Now let's check whether the cut and choose protocol is strategyproof and Pareto optimal.

1. v_i is normalized. That is, $v_i[0,1] = 1$. In the above example let the 4 subsets be S_1, S_2, S_3 and S_4. Then $v_i[S_1] + v_i[S_2] + v_i[S_3] + v_i[S_4] = 1$.

2. v_i is additive on disjoint sets. That is, in the above example since all the subsets are disjoint, then $v_i[S_1] + v_i[S_2] = v_i[S_1 \cup S_2]$.

- Since player 1 splits the cake into two halves A and B so player 2 cannot affect splitting. She has to choose the piece she likes better so she has no incentive to deviate.
- Now let's assume the cake has a strawberry, the first player likes all parts of the cake equally, while the second player really cares about the strawberry. If player 1 has some information about the second's player valuation function, the first player could split the cake into strawberry and the rest, knowing that the second player would take the strawberry, leaving a very valuable piece for the first player.
- If the first player doesn't know anything about what the second player wants, and assumes that the second player will always leave the piece that is worse for the first

player, then the first player is incentivized to follow the protocol (to guarantee herself a piece with value ½).

- In any case, the cut and choose protocol is not strategyproof.

> ***Pareto optimal (adj.)**: the property of an outcome that you can't make anyone better off without making someone else worse off.*

- Now let's assume the cake is on the x-y plane with centre at origin and the first player likes the cake in first and second quadrant while the second players like the third and fourth quadrant of the cake.
- Splitting the cake about the y-axis will result in both players getting a piece valued at ½ whereas splitting it around the x-axis will result in both players getting a piece valued at one.
  So, we conclude that the cut and choose protocol is neither strategyproof nor Pareto optimal.

Can we say something about "fairness"? First of all, let us understand what fairness is.

One possible definition of fairness would be that both players wind up equally happy. But this property is also not satisfied by the cut and choose protocol: the first player is guaranteed to get a piece that she values at ½, while the second player might well end up with a piece that she values at greater than ½.

> ***Definition 1.1** An allocation A1, A2, . . ., An of cake to n players is proportional if*
>
> $$v_i(Ai) \geq 1/n$$
>
> *for every player 'i'.*

The cut and choose protocol satisfies this property as the first player gets the piece that she values at ½ and the second player will always choose the better piece from the two and so $v_2(A_2) \geq ½$.

> ***Defintion 1.2** An allocation A1, A2, . . ., An of cake to n players is envy-free if*
>
> $$v_i(Ai) \geq v_i(Aj)$$
>
> *for every pair 'i', 'j' of players.*

This means every player likes her piece more than others piece, that is, no player wants to trade pieces with any other player.

The second definition itself implies the first one. To see this, recall our first assumption i.e., for any player i, the summation over j=1 to j=n i.e., $\sum_j v_i(A_j) = v_i([0,1]) = 1$. Now if player 'i' likes A_i better than any other piece it must imply $v_i(A_i) \geq 1/n$.

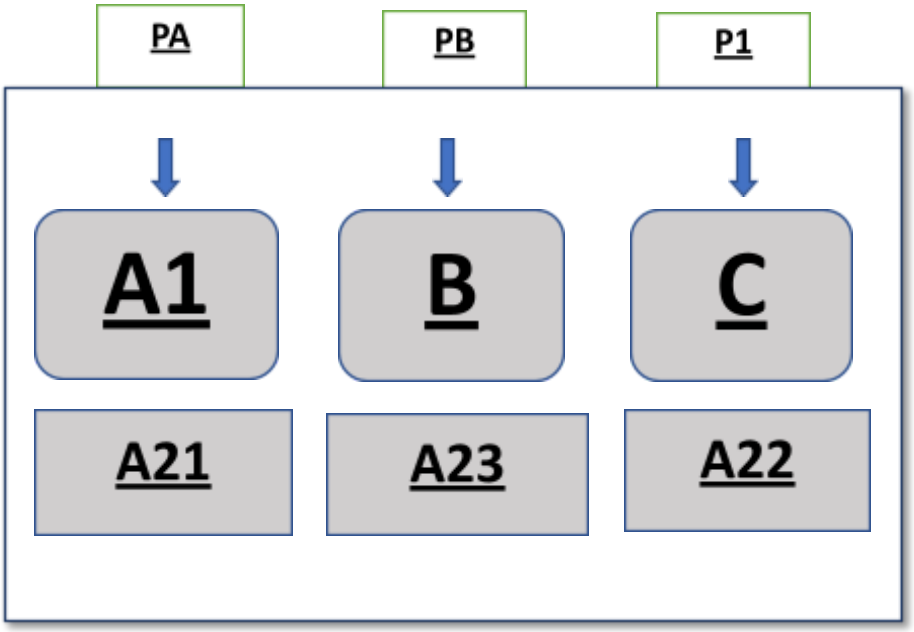
Cake cutting beyond two players

- Selfridge-Conway protocol (for three players):

1. Suppose we have three players P1, P2 and P3. P1 divides the cake into three pieces he considers of equal piece.
2. Let's call A the largest piece according to P2.
3. P2 cuts off a bit of A to make it the same size as the second largest. Now A is divided into: the trimmed piece A1 and the trimmings A2. Leave the trimmings A2 to the side for now.
    - If P2 thinks that the two largest parts are equal (such that no trimming is needed), then each player chooses a part in this order: P3, P2 and finally P1 and we are done here.
4. P3 chooses a piece among A1 and the two other pieces.
5. P2 chooses a piece with the limitation that if P3 didn't choose A1, P2 must choose it.
6. P1 chooses the last piece leaving just the trimmings A2 to be divided.

It remains to divide the trimmings A2. The trimmed piece A1 has been chosen by either P2 or P3; let's call the player who chose it PA and the other player PB.

1. PB cuts A2 into three equal pieces.
2. PA chooses a piece of A2 - we name it A21.
3. P1 chooses a piece of A2 - we name it A22.
4. PB chooses the last remaining piece of A2 - we name it A23.



A. Rent Division: Fair division in practice

- One place where fair division protocols are used in practice is the rent division problem, where there are n people, n rooms, and a rent of R. The goal is to assign people and rents to rooms, with one person per room and with the sum of rents equal to R, in the “best” way possible.
- We assume that each person i has a value v_{ij} for each room j, and normalize these values so that $\sum_j v_{ij} = R$. (In effect, we force each player to acknowledge the constraint that the entire rent must get paid). We assume that each player ‘i’

wants to maximize v_{ij} minus the rent paid for her room j. It enables envy-free solutions and is reasonable in this context.

- A solution to a rent division problem is envy-free if

$$v_{i\,f(i)} - r_{f(i)} \geq v_{i\,f(j)} - r_{f(j)}$$

for every pair i, j of players, where f(i) denotes the room to which i is assigned and r_j denotes the rent assigned to the room j. That is, no one wants to trade places with anyone else (where trading places means swapping both rooms and rents).

- The good news is that an envy-free solution is guaranteed to exist, and that one can be computed efficiently.
- The bad news is that there can be many envy-free solutions, and not all of them are reasonable.
- For example, let there be 3 rooms and 3 persons and that the total rent is 1500. Now if 1st player only wants 1st room, 2nd person only wants room 2 and 3rd person only wants room 3, i.e.
    1. $V_{1\,1} = 1500$, $V_{1\,2} = 0$, $V_{1\,3} = 0$.
    2. $V_{21} = 0$, $V_{2\,2} = 1500$, $V_{2\,3} = 0$.
    3. $V_{3\,1} = 0$, $V_{3\,2} = 0$, $V_{3\,3} = 1500$.
- The only reasonable room assignment is to give each person the room that they want. Intuitively, by symmetry, each person should pay 500 in rent. But every division of the rent is **envy-free**! Because $v_{i\,f(i)} - r_{f(i)} \geq 0$ whereas $v_{i\,f(j)} - r_{f(j)} < 0$.
- So even if you make the first person pay almost 1500 for her room, she still doesn't want to swap with the other person.
- So, we need a method for selecting one out of the many envy-free solutions. One of the methods is discussed below:
    1. Choose the room assignment f to maximize $\sum_i v_{i\,f(i)}$.
    2. Set the room rents so that envy-freeness holds, and subject to this, maximize the minimum utility:

$$\max\left(\min\left(v_{i\,f(i)} - r_{f(i)}\right)\right)$$

6. BITCOIN

Digital currencies have been around for decades, but only over the past 7 or so years has one really taken off: Bitcoin. Bitcoin enables digital payments between untrusted parties in a fully decentralized way, meaning with no central authority involved (no banks, credit card companies, governments, etc.).

Why use Bitcoin? One prosaic but representative example is transferring money internationally. Through Bitcoin, you can accomplish the same goal, saving an order of magnitude in both time and cost.

The basic element of Bitcoin is a Transaction.

A Bitcoin Transaction

1. One or more senders.
2. One or more receivers.
3. The amount of BTC (Bitcoins) transferred from each sender to each receiver.
4. A proof of ownership of the coins being transferred, in the form of a pointer back to the most recent transactions involving the transferred coins
5. A transaction fee, paid by the sender to the authorizer of the transaction.

A transaction is valid if:

1. It has been cryptographically signed by all of the senders. (This can be verified using the senders' public keys.)
2. The senders really do own the coins being transferred.

Blocks

Transactions are added to the ledger in groups (rather than one-by-one), known as blocks. Specifically, a block has the following ingredients:

1. One or more transactions.
2. A hash of the previous block.
3. A nonce. (I.e., a bunch of bits that can be set arbitrarily.)

Blocks typically have on the order of 1000–2000 transactions (there is a 1 MB cap on the block size). The second ingredient imposes a natural linked list-type structure on the ledger, with the predecessor of block b_2 being block b_1 whose hash matches the hash stored in b_2.

How do new blocks get added to the blockchain? Who can do it? Why should they bother? How can we make sure that everybody agrees on the contents of the blockchain? The two-prong solution to these questions is brilliant in its elegance.

1. Any user can authorize a block. Bitcoin incentivizes users to do authorizations through explicit monetary rewards (in BTC, naturally).

2. To avoid anarchy, authorizing a new block of transactions involves proof of work, meaning that the authorizer must solve a computationally difficult puzzle.

The Computationally Difficult Puzzle

A block b (consisting of transactions, the hash of the previous block, and the nonce) is valid if h(b) is sufficiently close to 0, where h is a pre-agreed upon hash function (currently, SHA-256, so h(b) is 256 bits). Thus, finding a valid block involves inverting a one-way/cryptographic hash function. By "sufficiently close to 0," we mean that the leading £ bits of h(b) should all be 0, where £ is a parameter. Note that £ provides a knob to control the difficulty of the problem: the bigger the choice of £, the harder the problem.[5] How is £ chosen? To keep the rate of valid block creation roughly constant, averaging around one block every ten minutes. The parameter £ is re-calibrated roughly every two weeks, based on the average time between blocks during this window. (So if the average time between blocks has dropped to 9 minutes, it's time to increase £.) Why 10 minutes? Because to maintain the property that all miners are on the same page about what the current blockchain is, it seems essential to have block creation happening on a time scale slower than the latencies in the peer-to-peer network.

Block Rewards and Bitcoin Mining

Bitcoin mining refers to the process of finding new valid blocks. The intended behavior for a bitcoin miner is: choose a subset of the outstanding transactions that it knows about (e.g., the ones with the highest transaction fees), insert the hash of the current last block of the blockchain, arbitrarily set the bits in the nonce, and hope that the resulting block b is valid. Since h is a cryptographic hash function, the accepted belief is that there is no algorithm for finding a valid block that is smarter or faster than random guessing or exhaustive search. Assuming that SHA-256 acts like a random function, for any given block b, the probability that b is valid is $2^{-£}$. This means that the expected number of tries necessary to find a valid block is $2^{£}$. In practice, Bitcoin miners typically search for a valid block by fixing the set of transactions and varying the nonce until the block becomes valid. When a new valid block has been found, the miner is supposed to immediately broadcast it across the entire peer-to-peer network, so that it gets appended to the blockchain (and the miner can collect its reward, see below). If someone else announces a new valid block first, then the miner restarts this procedure, now using only transactions not already authorized by the new block, and using the hash of the new block.

1. A flat reward that does not depend on the contents of the block (other than it is valid). When Bitcoin debuted this reward was 50 BTC, but the protocol dictates that this amount gets cut in half every four years.

2. The sum of the transaction fees of the transactions in the block. Currently, transaction fees are non-zero but typically constitute only a few percent of the overall reward.

Forks

Once in a while, two different bitcoin miners will discover valid blocks at roughly the same time. This results in a fork in the blockchain (Figure 2), where two valid blocks, each with its own set of transactions, point to the same previous block. There needs to be a mechanism for deciding which branch of the fork is the "right" one, for two reasons: (i) so that everybody knows which transactions have been authorized; and (ii) so that bitcoin miners know which block they should be trying to extend.

The Bitcoin protocol specifies the intended behavior when there's a fork: a user should regard the longest branch as the valid one, breaking ties according to the block that it heard about first. When there is a fork as in Figure 2, it is completely possible that different users will have different opinions about which branch is the valid one (user 1 may have heard about b_2 before b_3, while user 2 heard about b_3 before b_2).

Eventually, some bitcoin miner will authorize a new block, which extends only one of the branches (depending on which hash the miner put in the new block). If there were previously branches with equal length, then this new block will break the tie and create a chain longer than any other. At this point, all users will again have a consistent view of the blockchain (as the longest chain). When this happens, blocks not on this longest chain are "orphaned," and the transactions in them are not regarded as authorized. (Some of them may be authorized instead by a block on the longest chain.) Similarly, the creators of these orphaned blocks do not get any reward for them.

Blocks will occasionally get orphaned even when all miners are obediently following the protocol. Thus, one should not regard a transaction as authorized until it has been included in a block on the blockchain, and also been extended by another block

Sybil Attacks

Bitcoin users are identified with public keys. It's not hard to generate many public keys, so many Bitcoin "users" might actually correspond to the same person. Deliberately creating multiple identities in a system is often called a Sybil attack. Sybil attacks plague many systems, but remarkably, they cause no issues in Bitcoin. Your influence in Bitcoin is determined entirely by the amount of computational power that you wield—the number of identities is irrelevant.

6.BITCOIN INCENTIVE ISSUES

There are many ways that a bitcoin miner could conceivably deviate from the intended behavior, and we explore some of these next. At the outset, we should say that there have been little to no sightings of any of these attacks “in the wild.” The usual explanation for this is that any major attack on Bitcoin would immediately devalue the currency, a losing situation for everyone (including the attacker). Moreover, recent attacks have been hard to detect. However it is necessary to look for future possible attacks.

The Double-spend Attack

In this deviation, miners deliberately create forks. This is simple to do: when searching for a valid block, just insert the hash of a block on the blockchain that is not the last block. To see why a miner might want to create a fork, suppose in the transaction T, Alice transfers some bitcoins to Bob. Suppose this transaction gets added to the blockchain as part of block b1. Per the discussion above, Bob only ships the purchased goods to Alice on another block b2 has been appended to b1. When Alice has the goods, she could try the following attack: try to find a valid block b3 extending b0, another block b4 extending b3, and a third block b5 extending b4. Alice does not put transaction T in any of these blocks. If Alice successfully creates these three blocks before any other miner extends b2, then she rips off Bob: b1 and b2 are orphaned and Alice’s payment to Bob gets canceled, while the goods have already been sent. This attack is sometimes called the double-spend attack, especially in the case where Alice puts a payment T 0 to Carol in the block b3, promising the same coins to Carol that she already promised to Bob. Since bits are easily copied, every digital currency must address double-spending attacks. The probability that Alice succeeds in her double-spend attack depends on how much computational power she has. Suppose that of all the computational cycles being devoted to bitcoin mining, Alice controls an α fraction. The fraction α is sometimes called Alice’s mining power. Since finding valid blocks just boils down to random guessing or exhaustive search, the probability that Alice is the one who finds the next valid block is well-approximated by α. Finding three blocks in a row before anyone else, as needed in the double-spend attack above, happens with probability only α 3 . More generally, if Bob waits for k ≥ 1 blocks to be appended to b1 before shipping the goods, then the probability that a double-spend attack succeeds is only α k+2 . How big is α? If just a solo miner, then not very big. However many miners participate in mining pools, where the miners join forces, all working on the same puzzle, and sharing the rewards of finding a valid block among the pool members. The reward is shared proportionally, according to the amount of computation contributed by each member of the pool. And big mining pools can control a significant fraction of the total computational power (e.g. α = .3).

The 51% Attack

On the surface, our discussion of a double-spend attack above would seek to apply even if α = .51—even here the success probability is only ≈ 1 8 . But when α > 1 2 , a more patient strategy is guaranteed to succeed: Alice just continues to extend her own chain (b3, b4, b5, . . .) until it is the longest chain. Since on average Alice creates more than every other block, her chain will eventually overtake any other chain. More generally, Bitcoin is not intended to function when a single entity controls more than 50% of the computational power. Such an entity can effectively act as a centralized authority, defeating the whole point of Bitcoin. For

example, while such an entity cannot outright steal bitcoins from another user's account (because of the cryptographic protections), it can freeze the assets of any user that it wants, by forcing any blocks involving that user to be orphaned. In general, it is only interesting to study Bitcoin when no one controls more than 50% of the overall computational power.

Selfish Mining

Above attacks involved deliberate forking. Now we discuss a different type of deviation from the intended behavior: block withholding. Recall that a miner is supposed to announce a valid block to everyone as soon as she finds one; but if she desires, she can opt to keep it a secret. What is the incentive for Alice to withhold a block and forego the corresponding reward? The intuition is that Alice can trick all of the other miners into working on the wrong computational problem (extending the last publicly announced block, not Alice's secret block). Meanwhile, Alice can work privately on extending her own block. This trick boosts Alice's fraction of the (useful) computation being done, and hence has the potential to boost her expected reward. In detail, here's the selfish mining strategy. Suppose the last block of the current blockchain is b0, and Alice just discovered a new valid block s1 (which she will withhold from the others).

Selfish mining

1. Work privately to find a valid block s2 that extends s1. If some other miner announces a new valid block b1 (extending b0), then give up and start over.

2. If Alice finds s2 (extending s1) before any other miner finds b1 (extending b0), then Alice continues to mine her secret chain s1, s2, . . . , sk until her "lead" over the public blockchain b1, b2, . . . , b` drops to 1 (i.e., ` = k−1).

3. Announce her entire secret chain s1, . . . , sk.

For example, suppose Alice gets lucky and finds s2 and s3, and only then does some other miner find b1. Alice continues to try to extend the end s3 of her secret chain (with a lead of 2, this is still safe). If some other miner finds b2 first, however, then Alice cashes in her chips and announces s1, s2, and s3. Assume that all miners other than Alice behave as intended. Would selfish mining give Alice a higher expected reward than honest mining? This problem can be calculated and reaches an interesting solution.

Theorem :If Alice's mining power α is bigger than 1 3 , and all other miners are honest, then selfish mining yields greater expected reward than honest mining.

Interestingly, the original white paper on Bitcoin, by a mysterious individual or team known only as Satoshi Nakamoto, seems to suggest that there should be no incentive issues provided no miner has more than 50% of the overall computational power. Presumably Nakamoto had in mind attacks based on forking (like the double-spend attack), rather than attacks based on block withholding. Theorem effectively says that all miners being honest is not an "equilibrium," when at least one miner controls more than a third of the overall computational power.

7.Game Theory

Introduction

Definition: The study of strategically interdependent behaviour.

-Strategic Interdependence: actions of two individuals affect their respective outcomes.

Reasons to study game theory-

The logic of strategically interdependent situations gets extremely complicated extremely fast.

Game theory allows us to quickly draw parallels from one situation to another.

The Prisoner's Dilemma and Strict Dominance

Two criminals are detained. The police suspect them of having conspired on a major crime but only have evidence of a minor crime. To charge them for the greater crime, they need to elicit a confession.

To do this, they lock the criminals in separate interrogation chambers and offer each of them the following deal: If both keep quiet, then they will each spend a minimal time in jail. If one confesses while the other keeps quiet, the police will let the rat go free while throwing the book at the silent one. If both confess, the testimony is no longer necessary, and so both will be charged for the larger crime—though the time in jail will not be as bad as getting the book thrown at them.

Takeaway Points

1. We solve the prisoner's dilemma using the strict dominance solution concept.
2. Strategy x strictly dominates strategy y for a player if x generates a greater payoff than y regardless of what the other players do.
3. Strict dominance does not allow for equal payoffs. For example, suppose playing x and y both generated a payoff of 2 for an opposing strategy. Then x does not strictly dominate y.
4. Rational players never play strictly dominated strategies. After all, why would you want to play a strategy that always does worse than something else?
5. In a prisoner's dilemma, confessing strictly dominates keeping quiet for both players. Thus, we expect both players to confess.
6. The outcome of the game is inefficient. Both players would be better off if they both kept quiet instead. However, each player's individual interest is to confess if they know that the other player will keep quiet. As such, silence is unsustainable.
7. The conclusions a game theory model produces are only as the assumptions built into it. Here, we assumed that players only wanted to minimize their own jail time. In practice, this might not be the case. For example, a mafia boss may kill a lieutenant who acts as an informant to the police. If we wanted to know what might happen in that sort of scenario, we would need to change the payoffs to accommodate that additional strategic concern.
8.

Iterated Elimination of Strictly Dominated Strategies

If it is known the opponent has a strictly dominated strategy, one should reason that the opponent will never play that strategy. Internalizing that might change what one wants to do in the game.
This concept formalizes that idea, showing how to use strict dominance to simplify games. In fact, the logic can grow more complicated. It may be that after player-1 factors in player-2's strictly dominated strategy, one of player-1's strategies becomes strictly dominated. Then player-2 can reason that player-1 will not play something because he knows that player-1 can reason that he will not play something. Iterated elimination of strictly dominated strategies is the process that guides that thinking.
Takeaway Points

1. We may remove strictly dominated strategies from a game matrix entirely.
2. A reduced matrix will still give us all the necessary information we need to solve a game.
3. We may continue eliminating strictly dominated strategies from the reduced form, even if they were not strictly dominated in the original matrix. We call this process iterated elimination of strictly dominated strategies.
4. If a single set of strategies remains after eliminating all strictly dominated strategies, then we have a prediction for the game's outcome.
5. Iterated elimination of strictly dominated strategies cannot solve all games. We will have to broaden our solution concept if we want to make progress elsewhere.

The Stag Hunt and Pure Strategy Nash Equilibrium

Sometimes strict dominance takes us nowhere. Other times, we may make one or two inferences based on it but then get stuck. The next option is to look for Nash equilibrium. We use the stag hunt to introduce the concept of pure strategy Nash equilibrium (PSNE). In a pure strategy Nash equilibrium, all players take deterministic actions with no element of randomness.
Takeaway Points

1. Holding all other players' actions constant, a best response is the most profitable move a particular player can make.
2. A game is in Nash equilibrium when all players are playing best responses to what the other players are doing.
3. Put differently, a Nash equilibrium is a set of strategies, one for each player, such that no player has incentive to change his or her strategy given what the other players are doing.
4. Nash equilibria can be inefficient.
5. At least one Nash equilibrium exists for all finite games. This is known as Nash's Theorem. John Nash, the person that A Beautiful Mind is based on, first proved this, hence why his name is attached to both the theorem and the solution concept.
6. A game is finite if the number of players in the game is finite and the number of pure strategies each player has is finite. The stag hunt has two players, each of whom has two pure strategies. Therefore, it is a finite game.
7. There may or may not be a Nash equilibrium in infinite games.

What Is a Nash Equilibrium?

Nash equilibrium is the most important solution concept in game theory.It is a set of strategies, one for each player, such that no player has incentive to change his or her strategy given what the other players are doing. Stated like this, Nash equilibrium does not have a clear conceptual application.
This is deceiving. In fact, Nash equilibrium has a basic underlying interpretation.
Takeaway Points

1. Another way to think of a Nash equilibrium is as a law that no one would want to break even in the absence of an effective police force.
2. Suppose that the police do not exist.
3. Imagine that the government passes a law.
4. The required behaviors of people that the law outlines is a Nash equilibrium if everyone still wants to abide by it.

Example: Suppose two cars are sitting perpendicular from each other at a stoplight. The light is green for one of them and red for the other. If the red light car goes, it will cause a crash. If the green light does not go, it is just wasting time. Thus, everyone is happy to follow the law as stated, even if there is no police to ticket the drivers for breaking it. Thus, following the rules of the stoplight is a Nash equilibrium.

Best Responses and Safety in Numbers

We know that a Nash equilibrium is a set of strategies, one for each player, such that no player has incentive to change his or her strategy given what the other players are doing. In order to find Nash equilibria in complicated games with many strategies, We are introduced with a simple algorithm to find all of a game's pure strategy Nash equilibria. All it requires is some time to go through and mark all of a player's best responses.
Takeaway Points

1. Review: A player's best response is the strategy (or strategies) that generate the greatest payoff for him or her given what the other players are doing.
2. In larger games, it may prove helpful to mark best responses with asterisks (*) in the payoff matrix.
3. Best responses allow for indifference. For example, if the best payoff a player can earn in response to a particular opposing strategy is 0, then all instances of 0 receive the asterisk.
4. After doing this for all strategies, if all the payoffs in a particular cell have an asterisk next to them, then that strategy profile is a pure strategy Nash equilibrium.
5. Any outcome that does not have asterisks for all of its payoffs are not equilibria.

Matching Pennies and Mixed Strategy Nash Equilibrium

We have till now seen how to find pure strategy Nash equilibria. Sometimes the methods will not find any. However, Nash's Theorem says that all finite games have at least one Nash equilibrium. What are we missing?
The answer is mixed strategy Nash equilibrium. When a player selects a mix strategy, he or she randomizes among two or more pure strategies. We use matching pennies to introduce the concept of mixed strategy Nash equilibrium.
Takeaway Points

1. If the game is finite and there is no pure strategy Nash equilibrium, then there must be a mixed strategy Nash equilibrium.
2. In a mixed strategy Nash equilibrium, at least one of the players plays multiple strategies with positive probability.
3. This mixed strategy leaves the opponent indifferent to playing his pure strategies. (When there are more than two strategies, this gets a little more complicated—it may be that the mixed strategy leaves the other player indifferent between playing two of his strategies and strictly worse off playing a third.)
4. The mixed strategy that makes the opponent indifferent is not always obvious. We need to develop an algorithm to do that, which is the subject of the next lecture.

The Mixed Strategy Algorithm

We now look at the algorithm we use to solve for mixed strategy Nash equilibrium in simple 2×2 games.
Takeaway Points

1. If there is a mixed strategy Nash equilibrium, it usually is not immediately obvious.
2. However, there is a straightforward algorithm that lets you calculate mixed strategy Nash equilibria. We will employ it frequently.
3. The algorithm involves setting the payoffs for a player's two pure strategies equal to each other and solving for the mixed strategy of the other player that makes this equation true.
4. The mixed strategy algorithm does not involve any fancy mathematics. All one needs is basic knowledge of algebra. However, the process requires that the analyst be careful in putting the right values in the right places.
5. The trickiest part of the logic of mixed strategy Nash equilibrium is that we must use one player's payoffs to solve for the player's strategy. This is because the point of my mixed strategy is to make you indifferent, and vice versa.

How NOT to Write a Mixed Strategy Nash Equilibrium

Here we see why we should not use decimals to express mixed strategy Nash equilibria. Use fractions instead!
Takeaway Points

1. Fractions are more precise than decimals. For example, 1/3 is not equal to .33.

2. Even slight differences like 1/3 versus .33 mean that a player has a profitable deviation, and thus the “equilibrium” you have is not an equilibrium at all.
3. Be safe and express everything as a fraction.

Battle of the Sexes

Here we see how to solve the battle of the sexes.
A man and a woman want to get together for a date this evening. There are only two forms of entertainment in town: a ballet and a fight. The woman wants to see the ballet. The man wants to see the fight. But if they wind up in different locations, they will both have to head back home unhappy. What are the rational ways to resolve this dilemma?
Takeaway Points

1. Battle of the Sexes has three equilibria: two in pure strategies and one in mixed strategies.
2. The mixed one is worse than either of the pure strategy equilibria for both players. We will see this when we learn how to calculate payoffs.
3. safe and express everything as a fraction.