Introduction to Game Theory

Game Representations and Nash Equilibrium

February 10, 2009

Representing Games

Before talking about games we should get some idea of how to represent them mathematically. We will focus on a certain class of games, called ***perfect information*** games, in which each player knows all moves that have taken place and the possible moves each player has at all times.

We'll start off with some definitions:

Normal Form Game

We define a ***normal form (stragetic form) game*** as follows:

- A finite set of players $N = \{1, 2, ...n\}$
- Each player $i \in N$ chooses actions from a strategy set S_i
- Outcomes are defined by ***strategy profiles*** which consist of all the strategies chosen by individual players. Mathematically, the set of strategy profiles are defined as $S = S_1 \times S_2 \times \ldots \times S_n$
- Players have preferences over the outcomes (note, NOT preferences over their actions), denoted as $\preceq_i$. Usually these preferences are defined by a ***utility function***, $u_i : S \rightarrow \mathcal{R}$ that maps each outcome to a real number. (This is sometimes called a ***payoff function***).

So an instance of a game might be represented by the tuple $(N, (S_i), (\preceq_i))$, containing the players, strategy sets, and preferences.

Let's look at an example and get some practice working with the notation.

Example

First we'll consider a simple problem we can model as a normal form game. Consider the following situation: you want to meet three of your friends somewhere on MIT's campus to go hacking. Each of you might go to one of four places: State Center (C), the Green Building (G), the dome (D), or Kresge (K). Now that we have a "game," we'll analyze it using the definition above:

Players

To make things easier, we'll name you and your friends, and call the set of players $N = \{\text{You, Stavros, Helen, Paris}\}$

Stategy Sets

In this situation, each player's strategy set, $S_{You}, S_{Stavros}, S_{Helen}, S_{Paris}$, is the same: (C, G, D, K). It consists of the four possible meeting places.

Outcomes

Above we definited an "outcome" to be some $s \in S$, where $S = S_{You} \times S_{Stavros} \times S_{Helen} \times S_{Paris}$. First we'll quickly review some mathematical notation. Also see the cheat sheet at the end of these notes that might be useful.

The symbol $\times$ when used with sets is called a ***Cartesian Product***. It means to take all ordered tuples consisting of one element from each set in the product. So for example if $B = \{1, 2\}$ and $C = \{3, 4\}$, then $A = B \times C$ consists of the tuples $(1, 3), (1, 4), (2, 3)$, and $(2, 4)$. So the first item in each element of A comes from the set B, and the second comes from the set C.

Now we are ready to definte the outcomes, $S = S_{You} \times S_{Stavros} \times S_{Helen} \times S_{Paris}$, in our game. The elements in S are:
(D, D, D, D)
(K, K, K, K)
(C, C, C, C)
(G, G, G, G)
(G, D, D, D)
etc., including all 4^4 combintations and possible orderings of K, C, D, and G.

So, for example (K, G, C, D) is the outcome where You go to Kresge, Stavros goes to the Green Building, Helen goes to the Stata Center, and Paris goes to the dome.

Preferences

First we will define some notation for preferences: we will say $a \preceq b$ if in all cases b is preferred to at least as much as a.

As noted above, it is easier to make a preference relationship if outcomes are mapped to real numbers using a utility function u_i for each player i. So for any outcomes s_i and s_j, $s_i \preceq_k s_j$ iff $u_k(s_i) \leq u_k(s_j)$. That is, a player k prefers one outcome, s_j, over another, s_i, if and only if that player's utility function is higher for that outcome.

In this case it makes sense that you should prefer outcomes where as many of you meet at the same place as possible, since hacking by yourself isn't nearly as fun. (In the case where there might be more than 4 people we are talking about, we might define a utility function that ranked huge groups of people lower, since you don't want a mob of a hundred people hacking, but for now we'll set our goal at getting all four people in the same place). So we'll set the utility function to be the same for each player and define it as follows:

u_i : (max # people meeting at the same place)+0.5∗(# times this max occurs - 1)

So if all four of you meet at the same place, the utility (as we will call the value of the utility function) will be 4, if only 3 of you meet it will be 3, if two of you meet at the dome and two of you at State, the utility will be 2.5, etc. Note that outcomes with the same utility have the same preference level.

If we wanted to make things even more complicated, we could make each player have preferences over meeting spots, some players hoping that annoyingly loud person won't come along, etc.

Now that we know about the preference relation, we'll define some more conventional notation related to strategy:

- We define s_{-i} to be the set of strategy choices of all players except i. So if Stavros has chosen G, Helen C, and Paris D, then $s_{-You} = (G, C, D)$.
- The ***best response*** function for a player $i \in N$ is defined as

$$BR_i(s_i) = \{s_i \in S_i : u_i(s_i', s_{-i}) \leq u_i(s_i, s_{-i}) \forall s_i' \in S_i\}$$

  Or, in English, the best response function gives the strategy s_i for player i from his possibly strategy set S_i, given that he knows all the other players' strategies in s_{-i}, that maximizes player i's utility function.

Now we will look at perhaps one of the most famous game theory model problems, the ***prisoner's dilemma***. Two suspects are arrested as suspects in a crime investigation. For our sake, let's say someone has stolen Pandora's box, and both Helen and Stavros are accused and thrown into jail. If both of them don't confess, they will both spend three years in prison. If only one of them confesses, he or she will be freed and used as a witness against the other, who will then have to spend four years in jail. If both confess, the judge will go easy on them and give them both sentences of only one year. This situation can be modeled with the table below:

	Don't Confess	Confess
Don't Confess	3,3	0,4
Confess	4,0	1,1

If the players cooperate and both confess, then they can both get off easy with only one year in prison. But after investigating things further it is clear that no rational player would confess. No matter whether one prisoner chooses to confess or not confess, the other is always better off *not* confessing. If Stavros confesses, then Helen gets 0 years in prison if she doesn't confess versus 1 year in prison if she confesses. If Stavros doesn't confess, then Helen gets 3 years in prison if she doesn't confess versus 4 if she does. Therefore, if both players are rational, and for the moment we are assuming Stavros and Helen are, even though they were involved in that whole silly Trojan horse business back in the day and then had the foolishness to play with Pandora's box, both will choose not to confess and therefore will end up with three year sentences.

Game theoriests talk about this relationship among strategies in terms of ***dominance***. We say that a strategy s_i' is ***strongly dominated*** by another s_i if:

$$\forall s_{-i} \in S_i, u_i(s_i, s_{-i}) > u_i(s_i', s_{-i})$$

That is, for any combinations of actions for all of the other players, the move s_i has a higher payoff than the action s_i'. A strategy is ***weakly dominated*** if :

$$\forall s_{-i} \in S_i, u_i(s_i, s_{-i}) \geq u_i(s_i', s_{-i})$$

and for at least one choice of s_{-i} the inequality is strict, that is, the $\geq$ turns into a $>$. In games where there are more than two actions, it is also possible to have one action dominated not by a single other action, but by a combination of others.

For Stavros and Helen, confessing is strongly dominated, and so neither player has any reason to confess. Keep these games and definitions in mind as they will be useful to us in the next section.

Extensive Form

Another way to represent a game is in ***extensive form***. This type of game notation represents games using ***game trees***, which is how we will represent most of the two-player strategy games we look at. A ***game tree*** is a directed graph with nodes being game board positions and edges being moves. Each "level" of the tree, representing a move by one player, is called a ***ply***. The leaves of the tree are possible game outcomes. Each vertex, called a ***decision node***, is labeled with the player that makes a decision at that point. A decision node represents every possible state of a game as it is played, starting from the ***initial node*** and ending with the ***terminal node*** which assigns final payoffs to the players.

So an extensive form game is represented with the following items:

- players $N = \{1, 2, 3, \ldots, n\}$
- a set H of sequences (the ***histories***) of the game, where no sequence is a subhistory of another sequence, defined by:
  - $\emptyset$ is a member
  - if $(s^k)_{k=1\ldots K} \in H$ and $L < K$, then $(s^k)_{k=1\ldots L} \in H$, where s^k is an action after step k of the game.
  - if an infinite sequence $(s^k)_{k=1}^{\infty}$ has $(s^k)_{k=1\ldots L} \in H$ for all integers L, then $(s^k) \in H$. The set of histories in H that are infinite or terminal (there is no $s^{k+1} : (s^k_{k=1\ldots k}) \in H$) is denoted as Z.

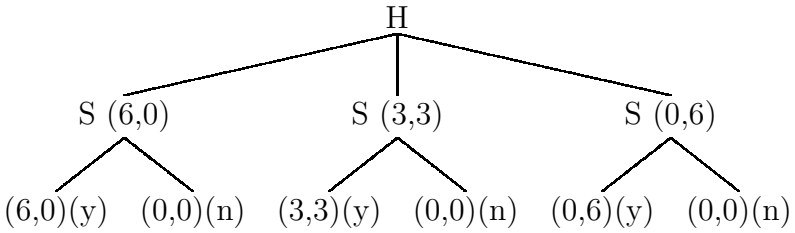
- a function P that assigns a player to every proper subsequence of every terminal history. We define $P(h)$ to be the player who moves after the subhistory h.
- for each player $i \in N$, a preference relation $\preceq_i$ over the set of terminal histories, Z.

So the tuple $\langle N, H, P, (\preceq_i)\rangle$ defines an extended form game.

Games in which there are a finite number of branches are called ***finite games***, and those for which the longest branch is finite are said to have a ***finite horizon***.

We define a player j's ***strategy*** as a function that assigns an action for player j to each h for which $P(h) = j$. That is, a strategy defines what action j should take at every point it gets to make a decision during the game.

As an example, consider the following situation: Helen and Stavros happen upon six olives that have fallen on the ground near their home in Athens. Helen suggests a scheme for dividing up the olives between them, and Stavros can say either "yes," in which case they divide the olives according to what Helen suggested, or "no," in which case both of them get zero olives.



So if Helen proposes (6,0), with her getting six and Stavros getting 0, if Stavros says yes the distribution will be (6,0), and if he says no both of them will get no olives.

In terms of extensive form game notation, we have the following:

- $N = \{Helen, Stavros\}$
- $H = \{\emptyset, (6,0), (3,3), (0,6), ((6,0), y), ((6,0), n), ((3,3), y), ((3,3), n), ((0,6), y), ((0,6), n)\}$
- $P(\emptyset) = Helen, P(h) = Stavros \forall h \neq \emptyset$.

Nash Equilibrium

Now that we have learned how to define games, we will begin looking at how to "solve" games. The most commonly used concept in looking at game solutions is the ***Nash Equilibrium*** (NE). The NE is conceptually the state of the game where if each player was given the chance to change strategies given all of the other players' strategies, the player would not be able to profitably deviate from its current strategy. The first type of NE we examine assumes ***pure strategy*** games, where every player chooses one strategy and sticks with it.

Normal Form NE

The NE for a normal form game $\langle N, (S_i), (\preceq_i)\rangle$ is a strategy profile $s^* \in S$ of actions such that for each player $i \in N$ we have:

$$(s^*_{-i}, s_i) \preceq_i (s^*_{-i}, s^*_i) \forall s_i \in S_i$$

So every player prefers the strategy profile s^* to any other strategy, given all the other players' strategies s^*_{-i}.

We can also define the Nash equilibrium in terms of best response functions. If $BR_i(s_{-i})$ is player i's best response to a strategy profile s_{-i}, then the Nash Equilibrium is a strategy profile s^* for which:

$$s^*_i \in BR_i(s^*_{-i}) \forall i \in N$$

So a game could be solved by finding the best response function of each player, and then finding a profile s^* that is in each BR_i. This can be modeled as a set of N equations with N unknowns and solved as a linear program.

Let's find the NE for the games we have examined so far. In the original problem meeting your friends to go hacking, can you find the equilibrium strategy profile? Well, there are several:
$(D, D, D, D); (C, C, C, C); (G, G, G, G); (K, K, K, K)$.
As long as you all meet at the same place, no player can profitably deviate (i.e., increase the value of their utility function) by choosing a different strategy.

Consider the following game, called ***matching pennies***: Stavros and Helen flip one coin each. If the coins match, Helen has to give Stavros a dollar, but if the coins differ, Helen gets to take Stavros's dollar. We can represent this game in a table:

	T	H
T	1,-1	-1,1
H	-1,1	1,-1

Helen is the "column player" and Stavros is the "row player." So when Helen flips tails and Stavros flips heads, this corresponds to the entry in the second row of the first column, or (-1,1). Meaning the row player, Stavros, loses a dollar while the column player, Helen, gains a dollar.

This type of game, in which what one player loses is exactly what the other player gains, is called a ***zero sum game***. Another name is ***constant sum***. You can think of it as the entire sum of money between Stavros and Helen staying the same the whole time, but the amount of money each individual player has changes with game play.

What is the NE for matching pennies? (Let's assume for a second that each player chooses whether or not the pennies are heads or tails.) In this game, there is no NE. If Helen picks heads, Stavros will pick heads so the coins are the same and he wins. But given this strategy profile, Helen would pick tails so that the coins differ. There is no strategy profile in which each player would not want to deviate. This is an example of a pure strategy game in which the players have ***diametrically opposed*** goals, and there is no NE. The same situation happens in rock paper scissors. Can you see why?

Now look at prisoner's dilemma. We said earlier that no matter what another player chooses, a player always does better by choosing not to confess. So this game has a unique NE, which is (Don't Confess, Don't Confess).

We have seen examples of games using pure strategy that have no NE, more than one NE, and one unique NE. It can be shown that if a player has a finite number of strategies to choose from and can choose different strategies given by a probability distribution, that an NE does exist. As a quick example,

consider matching pennies and rock paper scissors. In matching pennies, the NE occurs when each player chooses heads and tails with 50-50 probability. This is called ***mixed strategy*** and we will deal with this more when we talk about maximinima. Nash showed that every game has a mixed strategy NE.

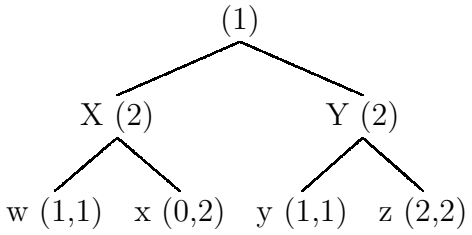
Extensive Form NE

If we define $O(s_i)$ to be the outcome associated with strategy profile s_i, then the NE for an extensive form game has:

$$O(s^*_{-i}, s_i) \preceq_i O(s^*_{-i}, s_i*) \forall S_i \text{ of player } i$$

So in our olive game with Helen and Stavros, the strategy profile ((3,3),y) is an NE, since neither Helen nor Stavros would get more olives (their utility in this case) by changing strategies.

Let's look at another extensive form game:



where the number in parentheses is the player (1 or 2) that makes the decision at that point, and the tuples at the terminal nodes are the payoffs for player 1, then player 2. The NE of this "game" as defined above would be the strategy profile (Y, z), yielding the payoff (2,2). But suppose for a second that player one chose X, (which he shouldn't, since it is weakly dominated by Y). Now player 2 has to make a decision based on the history $h = (X)$. Now player 2 is looking at the subgame of h, or the part of the game that follows h, defined as $\Gamma(h) = \langle N, H|_h, P|_h, (\preceq_i)|_h \rangle$. We can define another type of NE, the ***subgame equilbibrium*** for an extended game, as follows:

$$O(s_{-i}|^*_h, s_i|_h) \preceq_i |_h O(s^*_{-i}|_h, s_i * |_h) \forall S_i|_h \text{ of player } i$$

So the subgame equilibrium given the h defined above is (X, x) (a bad deal for player 1...).

Solving Games

Solving a Game

Many board games with seemingly simple rules have long stumped both humans and computers. Solving games depends on the ***decision complexity***, or the difficulty in making decisions, and the ***space complexity***, or the size of the search space over the possible moves and their consequences.

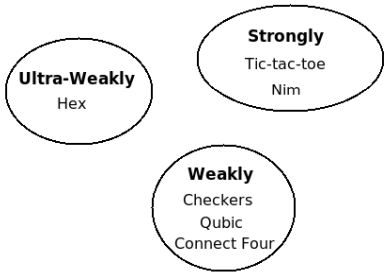
Good players look ahead to the possible game situations a move might lead to, which is no simple task when the search space reaches billions of billions of possibilities! There is much research in the field of artificial intelligence in creating computers that are able to look many branches ahead in a game tree to determine the best move.

What it means to solve a game

There are three basic types of "solutions" to games:

1. ***Ultra-weak*** The result of perfect play by each side is known, but the strategy is not known specifically.
2. ***Weak*** The result of perfect play and strategy from the start of the game are both known.
3. ***Strong*** The result and strategy are computed for all possible positions.

By perfect play, we mean each player cannot do any better, given the other's strategy. So to "solve" a game means to find it's Nash equilibrium.



Iterated Dominance

The method of ***iterated dominance*** is one way to find the pure strategy NE (if it exists) for a normal form game.

Consider a game with the following payoffs (player 1 is the row player, player 2 is the column player):

	A	B	C	D
A	(5,2)	(2,6)	(1,4)	(0,4)
B	(0,0)	(3,2)	(2,1)	(1,1)
C	(7,0)	(2,2)	(1,5)	(5,1)
D	(9,5)	(1,3)	(0,2)	(4,8)

Now when player 1 chooses a move, it will not choose A, since it is weakly dominated by C. But player 2 knows player 1 is rational, and knows he will not choose A. So the choices left are:

	A	B	C	D
A				
B	(0,0)	(3,2)	(2,1)	(1,1)
C	(7,0)	(2,2)	(1,5)	(5,1)
D	(9,5)	(1,3)	(0,2)	(4,8)

Now for player 2, D is dominates A, so player 2 won't choose A.

	A	B	C	D
A				
B		(3,2)	(2,1)	(1,1)
C		(2,2)	(1,5)	(5,1)
D		(1,3)	(0,2)	(4,8)

Now for player 1, B dominates both C and D, so he won't play either of those:

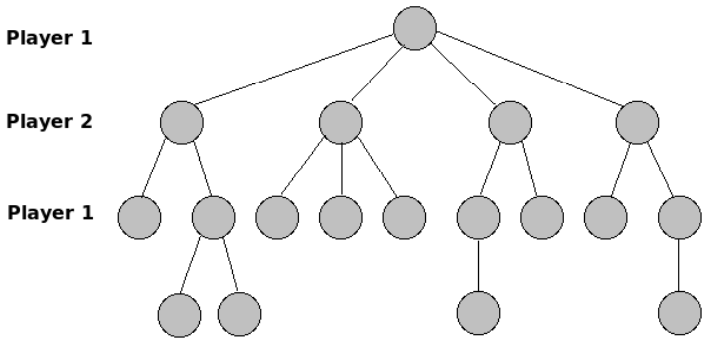
	A	B	C	D
A				
B		(3,2)	(2,1)	(1,1)
C				
D				

Player 2 has B dominating both C and D:

	A	B	C	D
A				
B		(3,2)		
C				
D				

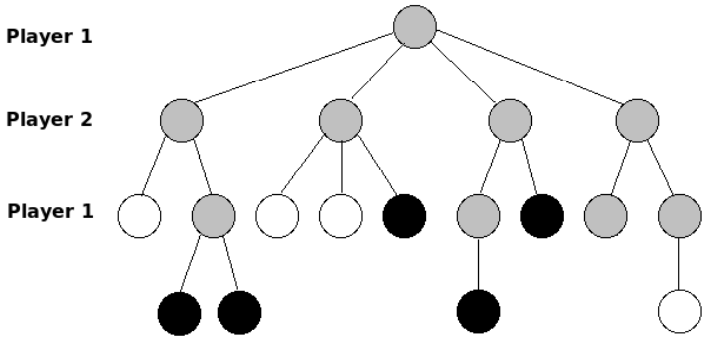
So the only choice left, the NE, is B for each player.

Finding Solutions from Game Trees



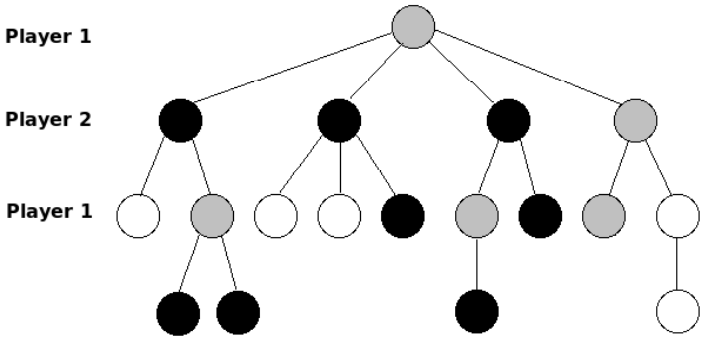
Once a complete game tree is known, we can use the following steps to solve the entire tree:

1. Color the final play according to which player won. We will use black for player 1, and white for player 2:

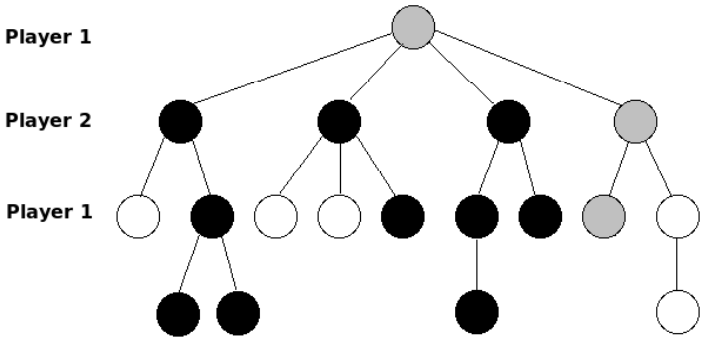


2. Look at the next ply up.

- If there is a node imeediately below the current node colored opposite than the current player at that node, then color the current node the color of the opposite player.



- If all immediately lower nodes are colored the same as the current player, also color the node that same color.



- Else color the node as a tie (we will use gray for a tie).

3. Move up the tree, repeating this for each ply. (In this case, we already completed the tree with step 2).

Exercises

- Begin drawing a tic-tac-toe game tree. See how many branches you can color as wins for each player.
- Consider the "centipede" game, with the following rules: two players alternate taking turns. On each turn, the player can decide to either

continue the game, or stop. A player gets a higher payoff with stopping at turn t compared to the other player stopping the game at time $t+1$, but gets an even higher payoff if neither of these events happen (i.e., the game continues).

- Define the player function for this game.
- Draw a possible tree for this game (there are many), remembering to label nodes with players and terminal nodes with the payoffs for each player.
- Write out the possible histories for this game (going only up to period $T = 6$).
- What are the possible subgame equilibria?

Appendix

Additional Topics

Complexity

We will often form questions about solving games in terms of some n, usually the size of a game board or a similar measure, and want to know how the number of steps needed to find the solution grows with n. It is worth mentioning a couple things to note about complexity here that we will refer to later on.

Problems are often classified using the following terms:

- **P** (polynomial) The problem is solvable in polynomial time. That is, the number of steps needed to find a solution is a polynomial in n.
- **NP** (non-deterministic polynomial) These problems are in general very difficult to solve, but once a solution is found it can be verified in polynomial time. All P problems are also NP problems, so $P \subseteq NP$, but you can get a million dollars if you can prove or disprove whether P=NP or not. One argument that there is reason to believe P=NP is that the existence of a good algorithm for verifying a solution to an NP problem might somehow imply that there is a good algorithm for solving it.

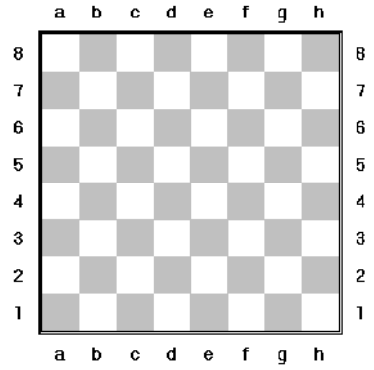
- **NP-complete** All NP problems can be reduced to a group of problems called NP-complete by a polynomial transformation. If there is a solution to one NP-complete problem then there is a solution to all.
- **NP-hard** This term means a problem is at least as hard as any NP-complete problem.
- **PSPACE** This describes problems that can be solved with a limited amount of storage space but unlimited time. Both P and NP belong to PSPACE, so $P \subseteq NP \subseteq PSPACE$.
- **EXPTIME** The set of all decision problems solvable in $O(2^{p(n)})$ time. EXPTIME is a superset of PSPACE.

Representing Games on Computers - Game Boards

We have looked at mathematical representations of games. Now we will see how to create computer representations.

Arrays

Probably the most intuitive way to represent a game board is by an array, here one array entry corresponds to one cell on the game board. For example, for a chessboard the cells might be labeled as follows:

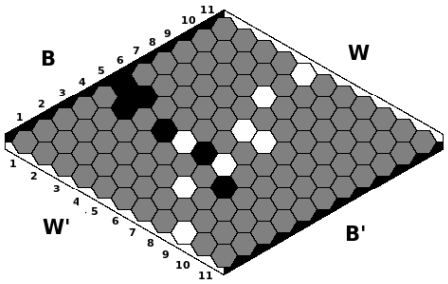


and stored in an 8×8 array. The entries in the array could be strings, such as "empty(e)", "wN", "bK" corresponding to an empty cell, white Knight, and black king, for example. The array below represents the starting postion of a chess board:

```
[[wC wN wB wQ wK wB wN wC]
[wP wP wP wP wP wP wP wp]
[e  e  e  e  e  e  e  e]
[e  e  e  e  e  e  e  e]
[e  e  e  e  e  e  e  e]
[e  e  e  e  e  e  e  e]
[bP bP bP bP bP bP bP bP]
[bC bN bB bW bK bB bN bC]
```

It is also common to use numbers to represent pieces: 0 for empty, 1 for white pawn, -1 for black pawn, 2 for white castle, -2 for black castle, etc.

A Hex Board (we will meet the game of Hex next week) might be represented as an 11 by 11 array:




```
[[0 0 0 0 0 1 0 0 0 0 0]
[0 0 0 0 1 1 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 1 0 0 0 0 0 0]
[0 0 0 0 -1 0 0 0 -1 0 -1]
[0 0 0 0 1 0 -1 0 0 0 0]
[0 0 -1 0 -1 0 -1 0 0 0 0]
[0 0 0 1 0 0 0 0 0 0 0]
[-1 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0]]
```

with entries 0 for empty, 1 for black, and -1 for white.

Bit Boards

One interesting scheme used widely in computer chess is the ***bit board*** . This representation uses a 64 bit string for each piece, with a 1 if that piece is in the square and a 0 if not. So the bit board for white pawns in the initial chess game state is:

```
[[0 0 0 0 0 0 0 0]
[1 1 1 1 1 1 1 1]
[0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0]]
```

As one example of how this notation can be useful, white can compute all the squares occupied by black by bitwise ORing the bitboards for each of black's pieces, etc. More resources on this method and a great tutorial applying it to checkers can be found online at http://www.3dkingdoms.com/checkers/bitboards.htm.

Graphs

While not used that widely, it might make sense to represent some game boards on a graph. A good graphing library for python is networkx, but there are plenty out there. You can create a graph by connect adjacent cells with edges and name them by their position on the board. We'll talk more about graph theory next week.

There will be a full tutorial online about making game boards.

Cheat Sheet

Symbols

- $\forall$ - "Forall"
- $\in$ - "In."
- $\emptyset$ - "Empty set"

- $x|h$ - x given that h has already occurred
- $\preceq_i$ - preference relation for player i
- $u_i(s_{-i}, s_i)$ player i's utility function when he chooses strategy s_i and everyone else's strategies are defined by s_{-i}