The River Crossing Game

By Tyler Bramhall

Table of Contents

How To Play

The River Crossing Game simply put is a game of chance between two players. Each player begins the game with a set number of tokens (anything from money to goldfish crackers). Both players distribute their tokens among the numbers 2 through 12, typically using some sort of number line as shown here for a 12 token game.

						x					
P1	x	x	x	x	x	x	x	x	x	x	x
	2	3	4	5	6	7	8	9	10	11	12
P2					x	x	x				
					x	x	x				
					x	x	x				
						x					
						x					
						x					

After both players have established their arrangement of tokens, the two players take turns rolling a pair of dice. For each roll, if either player has a token on the space that corresponds with the sum of the dice, they remove one token from that space. A token can be removed if either player rolls that specific number.

The first player to remove all of his/her chips from the board wins the game.

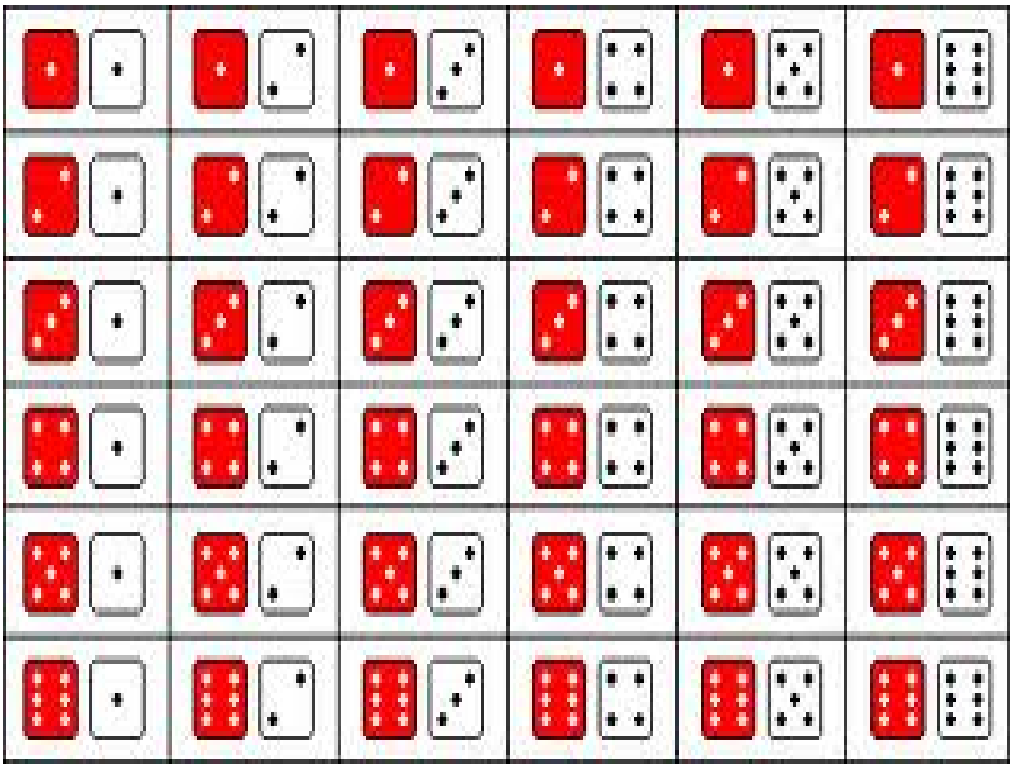
We denote these two strategies as follows:

P1: $[2^1\ 3^1\ 4^1\ 5^1\ 6^1\ 7^2\ 8^1\ 9^1\ 10^1\ 11^1\ 12^1]$ P2: $[6^3\ 7^6\ 8^3]$

The exponent on each number represents the number of tokens placed on that number.

Basic Dice Probabilities

Combinations For Rolling a Pair of Dice



Given these 36 combinations we can see that the possibilities for the sum of two dice are {2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}. The probabilities for each number are as follows:

Pr(Rolling a 2) = 1/36

Pr(Rolling a 3) = 2/36

Pr(Rolling a 4) = 3/36

Pr(Rolling a 5) = 4/36

Pr(Rolling a 6) = 5/36

Pr(Rolling a 7) = 6/36

Pr(Rolling a 8) = 5/36

Pr(Rolling a 9) = 4/36

Pr(Rolling a 10) = 3/36

Pr(Rolling a 11) = 2/36

Pr(Rolling a 12) = 1/36

NOTE: There is **symmetry** in the probabilities, with 7 being the middle probability at 6/36. Due to this fact, strategies that are mirror images of one another with 7 being the axis of symmetry will naturally have identical probability of winning the game.

Notation/Calculations

When discussing certain aspects of the River Crossing Game as well as explaining how to calculate certain probabilities, some notation may be used that is unfamiliar. Here is a description of what each notation means.

Pr(#) = The probability of a certain number being rolled (e.g. Pr(6) = 5/36)

Pr(# before #) = The probability of a certain number being rolled before another number is rolled (e.g. Pr(6 before 7)= 5/11)

Pr(A beats B)= The probability that strategy A beats strategy B in a game

ED(A)= The expected duration, i.e. the expected number of rolls for a given strategy A to eliminate all of their tokens

Exact calculations for certain values such as the probability of a strategy winning can be difficult as well as time consuming. Fortunately with the assistance of Matlab software we are able to approximate these values by running 10,000 or more simulations. With the large amount of iterations, the approximation error is small.

Although simulation is handy for working with large token games, exact calculations are fairly easy to compute for small token games as shown in the following demonstration.

How To Calculate

Lemma:

$$\text{Pr(n before m)} = \frac{\Pr(n)}{\Pr(n)+\Pr(m)}$$

Using our prior example:

$$\text{Pr(6 before 7)} = \frac{\Pr(6)}{\Pr(6)+\Pr(7)} = \frac{5/36}{5/36 + 6/36} = 5/11$$

In general, the probability of rolling x before any of the other rolls such as a, b, c… is

$$\text{Pr(x before a, b, c…)} = \frac{\Pr(x)}{\Pr(x)+\Pr(a)+\Pr(b)+\Pr(c)\ldots}$$

Calculating Exact Game Probability In General

The probability of one strategy beating another can be calculated with two methods. We can either do an exhaustive search of all possible outcomes and sum the probabilities that correspond to each strategy winning, or we can do a recursive process by breaking the game down into multiple pieces. Although both methods produce the same result, when using software such as Matlab or R, the exhaustive search method is significantly quicker. We will begin by demonstrating the exhaustive search method.

Start by identifying what numbers need to be rolled to eliminate both strategies. Let Strategy A = $[6^1\ 7^2]$ and let B = $[6^1\ 7^1\ 8^1]$.

Therefore $[6^1\ 7^2\ 8^1]$ are the numbers needed to eliminate both strategies. There are a total of $\frac{4!}{2!}$ = 12 possible roll combinations that would eliminate both these strategies. Of these possible combinations, there are three possible outcomes:

- If the last number rolled is an 8, then strategy A beats strategy B.
- If the last number rolled is a 7, then strategy B beats strategy A.
- If the last number rolled is a 6, then it is a tie.

Thus, the Pr(A beats B) = $\sum$ Pr (specific order ending in 8)

The following 12 order combinations represent every possible outcome for a game between the given strategies. The combinations highlighted represent the ones in which strategy A beats strategy B.

[6 7 7 8] [7 6 7 8] [7 7 6 8] [6 8 7 7] [6 7 8 7] [8 7 6 7] [8 6 7 7] [7 6 8 7] [7 8 6 7] [7 7 8 6] [7 8 7 6] [8 7 7 6]

To calculate these individual values, we use a sequential process.

Pr([6 7 7 8]) = Pr(6 before 7 or 8) × Pr(7 before 8)^2 = $\left(\frac{5}{16}\right) \times \left(\frac{6}{11}\right)^2$ = .09297

Pr([7 6 7 8]) = Pr(7 before 6 or 8) × Pr (6 before 7 or 8) × Pr(7 before 8) = $\left(\frac{6}{16}\right) \times \left(\frac{5}{16}\right) \times \left(\frac{6}{11}\right)$ = .06392

Pr([7 7 6 8]) = Pr(7 before 6 or 8)^2 × Pr(6 before 8) = $\left(\frac{6}{16}\right)^2 \times \left(\frac{5}{10}\right)$ = .07031

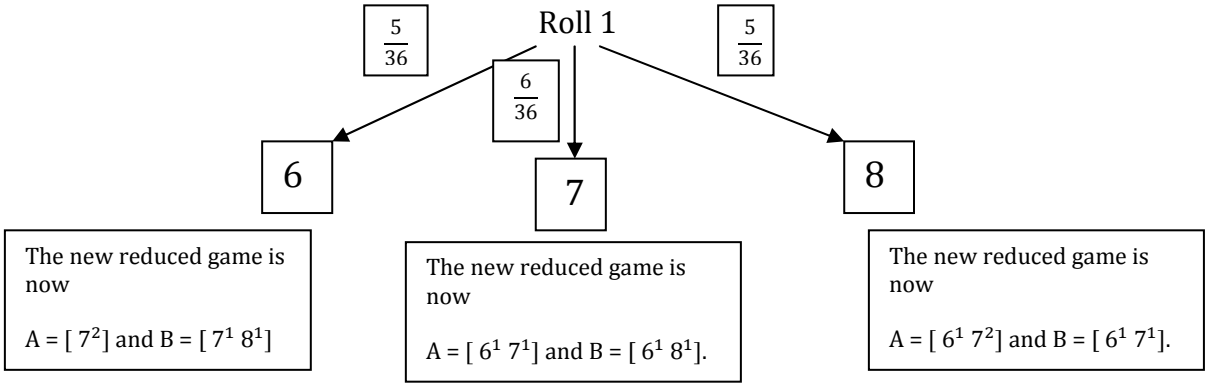
Pr(A beats B) = .09297 + .06392 + .07031 = **.2272**

This same logic can be applied to calculating Pr(B beats A); then, the Pr(Tie) = 1 – Pr(A beats B) –Pr(B beats A).

Recursive Process

Start by identifying what numbers need to be rolled to eliminate both strategies. Let Strategy A = $[6^1\ 7^2]$ and let B = $[6^1\ 7^1\ 8^1]$.

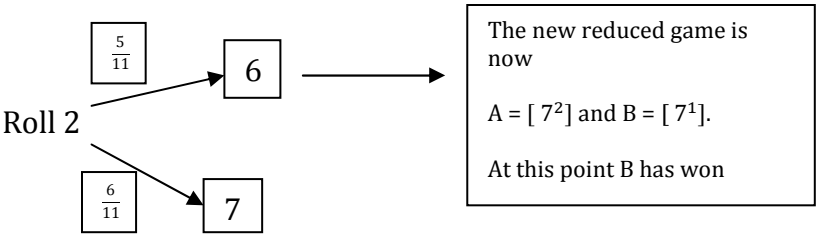
Therefore $[6^1\ 7^2\ 8^1]$ are the numbers needed to eliminate both strategies. We start with the first roll…



This process continues until either strategy A or B is eliminated or if A and B are reduced and have identical strategies resulting in a tie. Each "branch" of the recursive breakdown corresponds to either A wins, B wins, or a tie. To calculate Pr(A beats B) we sum the probability of each branch corresponding with A winning, but to calculate the probability of a "branch" we multiply the probabilities down the branch.

Example: Calculating a single branch of Pr(B beats A)

We start with the initial branch of rolling an 8 with probability = $\frac{5}{36}$ and now start with strategies: A = $[6^1\ 7^2]$ and B = $[6^1\ 7^1]$.



Single Branch Pr(B beats A) = $\left(\frac{5}{36}\right) \times \left(\frac{5}{11}\right)$ = .06313

Expected Duration Calculation

We will start with the basics; the expected duration of rolling a single number is merely the reciprocal of that number. Since the probability of rolling a 2 is 1/36, we would expect to roll a 2 once every 36 rolls, therefore the expected duration of rolling a 2 would be 36 rolls.

The expected duration of rolling one of two numbers is similar to the expected duration of rolling a single number. Since the probability of rolling a 2 or 3 is 3/36, we would expect to roll a 2 or 3 three times out of every 36 rolls, therefore the expected duration of rolling a 2 or 3 would be 12 rolls.

Expected Duration of a Game

The expected duration of a game is calculated by breaking the game down into multiple pieces. We start by identifying what numbers need to be rolled to eliminate both strategies. Let Strategy A = $[6^1\ 7^2]$ and let B = $[6^1\ 7^1\ 8^1]$

Therefore $[6^1\ 7^2\ 8^1]$ are the numbers needed to eliminate both strategies. To advance the game, a 6, 7, or 8 must be rolled, so the expected duration just to start the game is…

$$\text{ED(6, 7, or 8)} = \frac{1}{\Pr(6,7,or\ 8)} = \frac{36}{16} = 2.25 \text{ rolls}$$

But now we must account for the three possible situations, keeping in mind that each reduced game occurs with a probability:

- If a 6 is rolled first ($\Pr(\text{6 before 7 or 8}) = \frac{5}{16}$) the new game becomes $[7^2]$ vs. [7 8]

We must now account for two possible situations within this situation:

  - If a 7 is rolled first ($\Pr(\text{7 before 8}) = \frac{6}{11}$), the new game becomes [7] vs[8] and $\text{ED}([7] \text{ vs. } [8]) = \frac{1}{\Pr(7,or\ 8)} = \frac{36}{11} = 3.27$ rolls
  - If an 8 is rolled first ($\Pr(\text{8 before 7}) = \frac{5}{11}$), the new game becomes $[7^2]$ vs. [7] and $\text{ED}([7^2] \text{ vs. } [7]) = \frac{1}{\Pr(7)} = \frac{36}{6} = 6$ rolls

- If a 7 is rolled first ($\Pr(\text{7 before 6 or 8}) = \frac{6}{16}$), the new game becomes [6 7] vs. [6 8]

We must now account for three possible situations within this situation:

- If a 6 is rolled first (Pr(6 before 7 or 8) = $\frac{5}{16}$), the new game becomes [7] vs. [8] and ED([7] vs. [8]) = $\frac{1}{\Pr(7\ or\ 8)}$ = $\frac{36}{11}$ = 3.27 rolls
- If a 7 is rolled first (Pr(7 before 6 or 8) = $\frac{6}{16}$), the new game becomes [6] vs. [6 8] and ED([6] vs. [6 8]) = $\frac{1}{\Pr(6)}$ = $\frac{36}{5}$ = 7.2 rolls
- If an 8 is rolled first (Pr(8 before 6 or 7) = $\frac{5}{16}$), the new game becomes [6 7] vs. [6] and ED([6 7] vs. [6]) = $\frac{1}{\Pr(6)}$ = $\frac{36}{5}$ = 7.2 rolls

❖ If an 8 is rolled first (Pr(8 before 6 or 7) = $\frac{5}{16}$), the new game becomes [6 7^2] vs. [6 7]

We must now account for two possible situations within this situation:

- If a 6 is rolled first (Pr(6 before 7) = $\frac{5}{11}$), the new game becomes [7^2] vs. [7] and ED([7^2] vs. [7]) = $\frac{1}{\Pr(7)}$ = $\frac{36}{6}$ = 6 rolls
- If a 7 is rolled first (Pr(7 before 6) = $\frac{6}{11}$), the new game becomes [6 7] vs. [6] and ED([6 7] vs. [6]) = $\frac{1}{\Pr(6)}$ = $\frac{36}{5}$ = 7.2 rolls

We now sum the expected duration of each smaller game after multiplying them by their proper probability weights.

$$\frac{36}{16} + \left(\frac{5}{16}\right)\left(\frac{6}{11} \times \frac{36}{11} + \frac{5}{11} \times \frac{36}{6}\right) + \left(\frac{6}{16}\right)\left(\frac{5}{16} \times \frac{36}{11} + \frac{6}{16} \times \frac{36}{5} + \frac{5}{16} \times \frac{36}{5}\right) + \left(\frac{5}{16}\right)\left(\frac{5}{11} \times \frac{36}{6} + \frac{6}{11} \times \frac{36}{5}\right)$$

$$\approx 7.98 \text{ rolls}$$

Method Comparison

We have two possible methods for calculating the probability one strategy beats another, the Exhaustive method and the Recursive method. Is one method superior to the other? For small token games the exhaustive method is very easy to implement by hand as demonstrated earlier. The recursive method involves calculating all "branch'' probabilities and only summing the desired ones; calculating all the probabilities will be slower than only focusing on certain situations when using the exhaustive method.

For games involving more than 4 tokens, by-hand calculations become extremely tedious. Using the two methods discussed, we created a program to run these methods for larger token games in both R and Matlab.

Recursive Method

This method was at first implemented in Matlab, but because each branch is unique we must signify as such and R is better suited for labeling unique objects. We are running a process over and over breaking an *n* token game into multiple *n-1* token games and so on. After breaking the game down until a winner is identified, we must sum up the probabilities of certain branches. This method becomes painfully slow for any 5+ token game.

Exhaustive Method

The exhaustive method was our primary choice that we ran in Matlab. The program would produce every combination of possible roll outcomes to eliminate both strategies. It would then sum up the corresponding probabilities for each correct combination. This method is quicker than the recursive method, but as the tokens increase, Matlab begins to struggle enumerating every combination.

What Makes A Good Strategy?

In games of chance there exists no strategy that guarantees a win every time due to the randomization (dice are random). So what makes one strategy better than another? In a two-person game, a strategy is determined superior to another if that strategy is said to **dominate** its opponent.

Dominate: Strategy A dominates strategy B iff Pr(A beats B) > Pr(B beats A).

It may seem obvious that one would want the highest probability of winning possible. What allows a strategy to win more than another? Because the River Crossing Game is normally played with both players unaware of their opponent's strategy, it is not always possible to recognize the better strategy. Occasionally one knows their opponent's strategy; there are methods for devising a strategy to beat your opponents known strategy that will be addressed later but for now we will focus on the situation in which your opponent's strategy is not revealed.

Determining the better strategy seems like an easy thing for a computer to calculate but because of the complex nature behind calculating the probability of a given strategy winning, our software Matlab takes upwards of 45 minutes to calculate the results of a six token game. Instead of staring at a computer screen for hours on end, we investigated whether certain properties indicate if a strategy will dominate another. Some common properties are lowest expected duration or the highest multinomial probability.

Using Matlab software to run thousands of simulations we tested whether any of these conditions are correlated with a strategy dominating. For example, if ED(A) < ED(B), does strategy A dominate strategy B? The results of these simulations as well as the code for these simulations are included in the properties section.

Transitive Strategy Property

Is Dominance Transitive?

You might think that If Strategy A **dominates** Strategy B and Strategy B **dominates** Strategy C, then Strategy A **dominates** Strategy C. This is what is called a *transitive property* . But, in fact, this property does not hold for all games therefore one counter example will show that this property is false.

Example:

A = [2, 5, 6] B = [2, 4, 7] C = $[2, 7^2]$

Using the methods show earlier, the calculated values are provided

Pr(A beats B) = .21 Pr(B beats A) = .17

Pr(B beats C) = .18 Pr(C beats B) = .17

Pr(A beats C) = .1747 Pr(C beats A) = .1777

In this situation strategy A *dominates* B, strategy B *dominates* C, but strategy A ***does not dominate*** C.

Expected Duration Property

You might think that if Strategy A is expected to be eliminated in fewer rolls than Strategy B, then Strategy A should have a better probability of winning than Strategy B.

Let **ED(A)** be the **expected duration** of strategy A and **ED(B)** be the **expected duration** of strategy B, then if E(A) < E(B) does…

Strategy A **dominates** Strategy B?

Parallel to my example on the previous page, this property does not hold for all games therefore one counter example will show that this property is in fact false using a 3 token game for simplicity.

Example:

A = [3, 5, 6] B = $[6^3]$

Using the methods show earlier, the calculated values are provided

ED(A) = 22.33 rolls ED(B) = 21.6 rolls

Pr(A beats B) = .5459 Pr(B beats A) = .4541

In this situation ED(B) < ED(A) but Strategy B does **not** dominate Strategy A. Therefore this property does not hold for all strategies.

Not Knowing Opponent Strategy

In the situation in which one does not know their opponents strategy, it is impossible to tell if your strategy dominates or not. So we must determine what other property is most efficient for determining a good strategy. According to our simulations on the next page as well as our intuition that one would choose a strategy that is most likely to be eliminated in the shortest amount of rolls. This is in fact correct logic and a strategy that is most likely to be eliminated in the shortest amount of rolls is one with the highest **multinomial probability.**

There are three components to calculating multinomial probability; let x_i represent the number of tokens place on the ith number for i = 2…12. Let p_i represent the probability of rolling the ith number for i = 2…12. Let n represent the total number of tokens. The general formula for *multinomial probability* is as follows.

$$\frac{n!}{x_1!\ldots x_{12}!}p_2^{x_2}\ldots p_{12}^{x_{12}}$$

Example: Consider the strategy A= [2 4 5^2 6^3 7^2 9 10^2].

The Multinomial probability of strategy A is

$$\left(\frac{12!}{1!0!1!2!3!2!0!1!2!0!0!}\right)\left(1/36\right)^1\left(2/36\right)^1\left(3/36\right)^3\left(4/36\right)^3\left(5/36\right)^3\left(6/36\right)^2 = .0000009098$$

Because of the large number of possibilities and the nature of chance games, every strategy will have a low multinomial probability because eliminating any strategy in exactly *n* rolls is very specific. We are in search of the strategy with the largest probability even though it may only be .001.

Using this formula for multinomial probability, we created a program that will find the strategy with the highest multinomial probability for any specified number of *n* tokens. The program starts with a base strategy of 0 tokens in every slot, then a single token is added to each slot one at a time while recording which new strategy has the highest multinomial probability. (i.e. the best 1 token strategy is just 1 token on 7). This recursive process continues until there are a total of *n* tokens on the board, revealing the *n* token strategy with the highest multinomial probability.

The program is included in the appendix but we will use it to show another property.

Notice that for 1 to 12 token games as shown below, the best multinomial strategies for any *n* token game can be produced by adding a single chip to the best multinomial strategy for a *n*-1 token game.

We will start with the trivial 1 token game where it is obvious that putting a single token on 7 is the best multinomial strategy. Because of symmetry there may be more than one optimal strategy for certain token games (e.g. [6, 7] is the mirror image of [7, 8] and therefore [6, 7] and [7, 8] have equivalent multinomial probabilities)

Tokens	**STRATEGY**	**MULTINOMIAL PROBABILITY**
1	[7]	.1667
2	[6,7]	.0463
3	[6,7,8]	.0193
4	[5,6,7,8]	.0086
5	[5,6,7,8,9]	.0048
6	[5,6,7^2,8,9]	.0024
6	[4,5,6,7,8,9]	
7	[4,5,6,7^2,8,9]	.0014
8	[4,5,6,7^2,8,9,10]	.00092614
9	[4,5,6^2,7^2,8,9,10]	.00057884
10	[4,5,6^2,7^2,8^2 ,9,10]	.00040917
11	[3,4,5,6^2,7^2,8,9,10]	.00024565
11	[3,4,5,6^2,7^2,8^2,9,10]	
11	[4,5^2,6^2,7^2,8^2,9,10]	
11	[4,5,6^2,7^3,8^2,9,10]	
12	[3,4,5,6^2,7^2,8,9,10,11]	.00016377
12	[3,4,5^2,6^2,7^2,8^2,9,10]	
12	[3,4,5,6^2,7^3,8^2,9,10]	
12	[3,4,5 ,6^2,7^2,8^2,9^2,10]	

Notice that for some amount of tokens, there are multiple strategies that have the highest multinomial probability. This fact shows that although the best strategy for a given game has the highest multinomial probability, the strategy with the highest multinomial probability is not guaranteed to be the best.

Knowing Your Opponents Strategy

The more uncommon situation for the River Crossing Game is one in which your opponents strategy is known and you must devise a strategy that will dominate it. One may assume the highest multinomial probability point s to the optimal strategy as discussed in the situation where one does not know their opponent's strategy. Although high multinomial probability is a general property that "good" strategies possess, we showed that having a higher multinomial probability then your opponent does not guarantee that your strategy will dominate your opponent's.

In our previous situation, it was not possible to calculate the probability of a strategy winning without knowing the other players strategy. Now that we know both strategies, we are able to test whether a strategy dominates or not by using our original definition.

Now that the opponent's strategy is revealed, our goal is to create a strategy that will win majority of the time. For a three or four token game, the "better" strategy may be obvious but visualizing the best strategy when dealing with 12 or even 36 token games becomes very difficult and tedious. Fortunately there exists a method to reduce these larger token games down to a size that is easier to comprehend.

Strategy Reduction Property

Definitions:

Reduced: Given any two strategies A and B each strategy can be reduced by examining the number of tokens on each number. If strategy A has more tokens on a certain number than strategy B does, then strategy B reduces the number of tokens on that number to zero and strategy A does not reduce its tokens on that number or vice versa. After using this process for each entry slot, the remaining altered strategies are considered **reduced.**

Property:

For strategies (A) and (B), let (a) and (b) denote the **reduced** strategies then….

$$Pr(A \text{ beats } B) = Pr(a \text{ beats } b)$$

Example: We will use the 3 token game to show this for simplicity

$A=[6^2\ 7\ 8]$ $B=[6\ 7^2\ 8]$

For both strategies to be eliminated, a combination of (6,6,7,7,8) needs to be rolled, this has 5!/(2!2!)= 30 total combinations. Strategy A combos will be the total number of combinations of (6,6,7,7,8) that would allow strategy A to be rolled before strategy B therefore allowing A to win.

A Wins: All the combinations that end in 7 result in an A win, because strategy B has two 7's so ending with a 7 means a had already been eliminated before the fifth toll:

Strategy A winning combinations(66787), (66877), (67687), (67867),(68677), (68767),(76867),(76687),(78667),(86767),(87667),(86677)

= 12 total combinations

Example revisited:

$a=[6^2\ 8]$ $b=[7^2\ 8]$

For both strategies to be eliminated, a combination of (6,6,7,7,8) needs to be rolled, this has 5!/(2!2!)= 30 total combinations. This will be our denominator. Our numerator will be the total number of combinations of (6,6,7,7,8) that would allow strategy A to be rolled before strategy B therefore allowing A to win.

A Wins: All the combinations that end in 7 result in an A win, because strategy B has two 7's so ending with a 7 means a had already been eliminated before the fifth toll:

Strategy A winning combinations(66787), (66877), (67687), (67867),(68677), (68767),(76867),(76687),(78667),(86767),(87667),(86677)

= 12 total combinations

Because our denominator is the total number of combinations needed to wipeout all the tokens off the board, which is the combinations of the max number of tokens on each number. Therefore after we reduce our strategies we maintain the same maximum tokens for each number because when reducing we reduce the minimum to 0.

We have identical total number of game combinations and so as long as we maintain identical combinations for strategy A and B to win, our probabilities will match.

***NOTE:** The 12 combinations that end in strategy A winning are the exact same as before in the full strategy example.

Recall that in our example strategy A won when the combination of rolls ended in a 7 because strategy B had one more 7 than strategy A

This philosophy works to find any total number of combinations for a given strategy to be rolled before the opposing strategy. Our numerator is any of the combinations in the denominator that end in a number that strategy **B** has more of then strategy **A**. Therefore because our reduced strategies **a** and **b** still contain the same total combinations of die rolls to end the game, it consequently has identical combinations for strategy **A** and **a** to win.

One Player Expected Duration (RCG.m)

Explanation

RCG.m calculates how many rolls are expected to completely eliminate any given strategy A. The input should be some strategy A written in the following format.

A= [X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12]

Code

```
function[expected]= RCG(a)
duration=zeros(1,10000);
for i= 1:10000
x=a;
duration(i)=0;
while sum(x)> 0
    duration(i)= duration(i) +1;
    die1= randi([1,6]);
    die2= randi([1,6]);
    toss= die1 + die2;
if x(toss-1)> 0
        x(toss-1)= x(toss-1)-1;
end
end
end
expected= sum(duration)/10000;
```

Two Player Game Simulation(RCG2.m)

Explanation

RCG2.m simulates a game between two input strategies A and B and returns Pr(A beats B), Pr(B beats A), and Pr(Tie). The input should be some strategy A and B written in the following format.

A= [X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12]

Code

```
function[probability]= RCG2(a,b)
prob=[0,0,0];
for i= 1:10000
x=a;
y=b;
    duration=0;
while sum(x)> 0 && sum(y) >0
    duration= duration +1;
    die1= randi([1,6]);
    die2= randi([1,6]);
    toss= die1 + die2;
if x(toss-1)> 0
        x(toss-1)= x(toss-1)-1;
end
if y(toss-1)> 0
        y(toss-1)= y(toss-1)-1;
end
rolls=duration;
end
if sum(x)==0 & sum(y)>0
    prob(1)= prob(1)+1;
elseif sum(y)==0 & sum(x)>0
    prob(2)= prob(2)+1;
else
    prob(3)= prob(3)+1;
end
end
probability= prob/10000;
a;
RCG(a);
b;
RCG(b);
```

Smart Strategy Generator(strategy.m)

Explanation

strategy.m generates a "good" strategy based on the input of how many tokens are in the game.

Code

```
function[strategy] = strategy(n)
x=[0,0,0,0,0,0,0,0,0,0,0];
for i= 1:n
die1= randi([1,6]);
die2= randi([1,6]);
toss= die1 + die2;
x(toss-1)= x(toss-1)+1;
end
strategy=x;
```

Strategy Converter(converter.m)

Explanation

converter.m takes a given strategy inputed in the form,

A= [0, 0, 1, 2, 0, 1, 1, 0, 2, 0, 0]

And converts the strategy into the following form,

A= [4, 5, 5, 7, 8, 10, 10]

Code

```
function[new]= convert(A)
m= zeros(1,sum(A));
list = [2,3,4,5,6,7,8,9,10,11,12];
k=1;
for i=1:11
    for j=k:A(i)
        if A(i)>0
        m(j)= list(i);
        else
        end
    end
    k=k+A(i);
end
new=m;
end
```

Multinomial Probability Calculator (perfectprob.m)

Explanation

perfectprob.m calculates the multinomial probability for any given input strategy

Code

```
function[prob]= perfectprob(a)
choose= factorial(sum(a))/
(factorial(a(1))*factorial(a(2))*factorial(a(3))*factorial(a(4))*factor
ial(a(5))*factorial(a(6))*factorial(a(7))*factorial(a(8))*factorial(a(9
))*factorial(a(10))*factorial(a(11)));
pr= ((1/36)^(a(1) + a(11)))*((2/36)^(a(2) + a(10)))*((3/36)^(a(3) +
a(9)))*((4/36)^(a(4) + a(8)))*((5/36)^(a(5) + a(7)))*((6/36)^(a(6)));
prob= choose*pr;
```

Best Multinomial Strategy Builder(bestonestrategy.m)

Explanation

bestonestrategy.m determines the strategy with the highest multinomial probability for a given amount of tokens (n) in the input.

Code

```
function[best] = bestonestrategy(A)
 m= zeros(11,11);
while sum(A)<12

   prob= zeros(1,11);
for i=1:11

        A(i)= A(i)+1;
        m(i,:)=A;
        A(i)= A(i)-1;
       prob(i)= perfectprob(m(i,:));
end
    [~, maxInd]= max(prob);
    entry= maxInd;
    A=m(entry,:);
end
best= A;
end
```

Exact Two Player Game (ExactProbability.m)

Explanation

ExactProbability.m calculates the probability of outcomes for a game between two input strategies A and B and returns Pr(A beats B), Pr(B beats A), and Pr(Tie). The input should be some strategy A and B written in the following format.

A= [X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12]

Code

```
function[game] = ExactProbability(A,B)
u =(max(A,B));
w= convert(u);
v = perms(w);
combos = unique(v, 'rows');
probability = vectorprob(combos);
[row,col] = size(combos);
outcome = zeros(row,1);
probA=0;
probB=0;
probT=0;
for i=1:row
    rollorder = combos(i,:);
    for j=1:(col-1)
        A(rollorder(j)-1)=  A(rollorder(j)-1)-1;
        B(rollorder(j)-1)=  B(rollorder(j)-1)-1;
        for k=1:11
           if A(k)<0
               A(k)=0;
           else
           end
           if B(k)<0
               B(k)=0;
           end
        end
    end
    if sum(A)==0 && sum(B)>0
        outcome(i) = 'A';
    elseif sum(B)==0 && sum(A)>0
        outcome(i) = 'B';
    elseif sum(A)>0 && sum(B)>0
        outcome(i) = 'T';
    end

    if outcome(i)=='A'
        probA= probA+ probability(i);
    elseif outcome(i)=='B'
        probB= probB+ probability(i);
    elseif outcome(i)=='T'
        probT= probT+ probability(i);
    end
```

```
end
game=[probA,probB,probT];
end
```

Permutation Probability Calculator (vectorprob.m)

Explanation

vectorprob.m calculates the probability of a given combination vector occurring exactly in that order. (i.e if x = [6,7,8] vectorprob.m calculates the probability of rolling a 6 before a 7 and a 7 before an 8)

Input in the following form: A= [3, 3, 5, 6, 7, 7, 8, 9, 11, 11]

```
function[probability]= vectorprob(A)
[row,col] = size(A);
probability = zeros(row,1);
Pr= [0,1/36,2/36,3/36,4/36,5/36,6/36,5/36,4/36,3/36,2/36,1/36];
for i=1:row
    total=1;
    order = A(i,:);
    for j=1:col
       U=unique(order);
       [~,y] = size(U);
       denom=0;
       for k=1:y
       denom = denom + Pr(U(k));
       end
        total= total*(Pr(order(j))/denom);
        order(j)=1;
    end
    probability(i)= total;
end
end
```