Another Dice Game

Prabhav Kumar

Introduction

Having a crack at math questions via phone at 4AM for a quant position at Wall Street Proprietary Trading Firm isn't what you do on a regular basis, but it was definitely a fun experience. High Frequency Trading Firms (HFTs) are increasingly hiring Math, Physics, and CS majors to make money in financial markets. A typical HFT interview consists of a few math and few coding questions, but this one was a little different because it was one long question. Assuming you know the basics of conditional probability and expectations, I hope to walk you through the question in as much detail as possible.

The Initial Question (Level 1)

Let's play a game G. Imagine we have two fair 6-sided dice. We roll the 1^{st} dice. If we get an outcome that gives us some chance of obtaining a sum greater than 10 after a second roll, we note the outcome of the first dice and end the game. If not, we roll the 2^{nd} dice and note the sum of the two dice. What is the expected value of this game?

The Implicit But Explicit

- **fair 6-sided dice**: The set of outcomes S is $\{1, 2, 3, 4, 5, 6\}$. The probability of each outcome O is $P(O) = \dfrac{1}{6}$.
- **some chance**: The probability $p > 0$.
- The order in which we roll the dice does not matter because the dice are of the same size.

Important Identities

- **expected value**: The expected value of a group of x numbers is its average. If the numbers are consecutive, then we can just take the average of the minimum and maximum number.
- **expected value of sum of random variables**: E(X + Y) = E(X) + E(Y).

Level 1 Solution

Let's draw a table for the sum of two dice rolls.

Table 1: Outcomes of two dice rolls

	1	**2**	**3**	**4**	**5**	**6**
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

According to Table 1:

- If we have a 1, 2, 3, or 4 on the first roll, we can roll again, because the sum of the two dice will be less than 10 regardless of the outcome of the second dice. Let's call this event E_1.
- If we roll a 5 or 6, we have some chance of obtaining a sum greater than 10. Let's call this event E_2.

So we can condition on the first dice roll. Let F and S be the outcomes of the first and second dice roll respectively. Then the expected value of the game G is:

$$\begin{aligned}
E(G) &= \frac{4}{6} \times E(F+G|E_1) + \frac{2}{6} \times E(F|E_2) \\
&= \frac{4}{6} \times (E(F|E_1) + E(G|E_1)) + \frac{2}{6} \times E(F|E_2) \\
&= \frac{4}{6} \times (2.5 + 3.5) + \frac{2}{6} \times 5.5 \\
&= \frac{4}{6} \times 6 + \frac{2}{6} \times \frac{11}{2} \\
&= 4 + \frac{11}{6} \\
&= \frac{35}{6}
\end{aligned}$$

Checking Answer with Python:

```python
import random

# Variables
trials = 1000000
table = {}
dice = 6
target = 10

```

```python
# Level 1 Game
table = {}
for i in range(trials):
    # First Roll
    first_roll = random.randint(1, dice)

    # Following Game Rules
    if first_roll <= target - dice:
        second_roll = random.randint(1, dice)
        result = first_roll + second_roll
    else:
        result = first_roll

    # Storing Outcomes in Dictionary
    if result not in table:
        table[result] = 0
    table[result] += 1

# Calculating Expected Value
ev = sum(map(lambda x: x[0] * x[1], table.items())) / trials

# Calculating Theoretical Answer
answer = 35 / 6

#Outputting Results
print(f'Answer: {round(answer, 2)} | Simulated Expected Value: {round(ev,
↪  2)}')
```

Output

```
Answer: 5.83 | Simulated Expected Value: 5.83
```

The Follow-Up Question (Level 2)

Let's play the Level 1 game G but with two fair n-sided dice. What is the expected value now?

Level 2 Solution

What can we figure out about n? Firstly, let's state the obvious: n is positive integer. Now, if $n > 10$, then there is some chance that we roll a 10 with the 1^{st} dice. Thus, we don't roll at all. The (short) game ends here. Hence, $n \leq 10$.

According Table 1, if $1 \le n \le 5$, then we can always roll twice. Thus:

$$\begin{aligned} E(G) &= E(F+S) \\ &= E(F) + E(S) \\ &= 2E(F) \\ &= 2 \times \frac{n+1}{2} \\ &= n+1 \end{aligned}$$

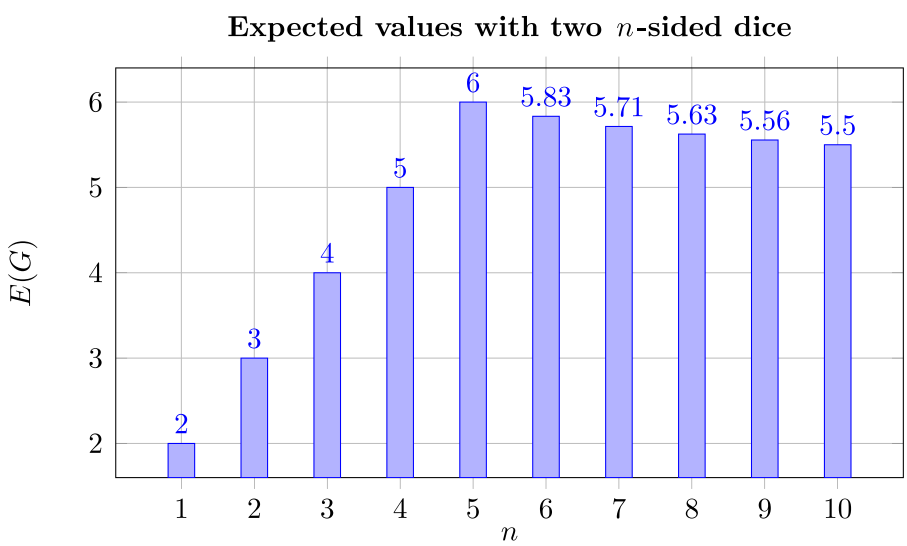
Similar to the Level 1 solution, if $6 \le n \le 10$:

$$\begin{aligned} E(G) &= \frac{10-n}{n} \times \left(\frac{1+10-n}{2} + \frac{1+n}{2}\right) + \frac{2n-10}{n} \times \frac{10-n+1+n}{2} \\ &= \frac{10-n}{n} \times 6 + \frac{11(n-5)}{n} \\ &= \frac{5+5n}{n} \\ &= 5 + \frac{5}{n} \end{aligned}$$

Therefore, the final solution is:

$$E(G) = \begin{cases} n+1 & \text{if } 1 \le n \le 5 \\ 5 + \dfrac{5}{n} & \text{if } 6 \le n \le 10 \end{cases}$$

And if we plot $E(G)$ against n, we obtain:



We clearly see a minimum and maximum expected value for $n = 1$ of 2 and $n = 5$ of 6 respectively.

Checking Answers with Python:

```
# Level 2 Game

# Changing Dice Size
for size in range(1, 11):
    # Resetting Dictionary
    table = {}
    for _ in range(trials):
            # First Roll
            first_roll = random.randint(1, dice)

            # Following Game Rules
            if first_roll <= target - dice:
            second_roll = random.randint(1, dice)
                 result = first_roll + second_roll
            else:
                    result = first_roll

    # Storing Outcomes in Dictionary
    if result not in table:
        table[result] = 0
    table[result] += 1

    # Calculating Expected Value
    ev = sum(map(lambda x: x[0] * x[1], table.items())) / trials

    # Calculating Theoretical Answer
    if 1 <= size <= 5:
        answer = size + 1
    else:
        answer = 5 + 5 / size

    # Outputting Results
    print('{:<9s}{:<15s}{:<10s}'.format(f'n = {size}', f'Answer:
    ↪   {round(answer, 2)}', f'Simulated Expected Value: {round(ev, 2)}'))
```

Output

```
n = 1    Answer: 2      Simulated Expected Value: 2.0
n = 2    Answer: 3      Simulated Expected Value: 3.0
n = 3    Answer: 4      Simulated Expected Value: 4.0
n = 4    Answer: 5      Simulated Expected Value: 5.0
```

```
n = 5     Answer: 6        Simulated Expected Value: 6.0
n = 6     Answer: 5.83     Simulated Expected Value: 5.84
n = 7     Answer: 5.71     Simulated Expected Value: 5.71
n = 8     Answer: 5.62     Simulated Expected Value: 5.62
n = 9     Answer: 5.56     Simulated Expected Value: 5.56
n = 10    Answer: 5.5      Simulated Expected Value: 5.5
```

The Unasked Question (Level 3)

Let's play the Level 2 game G but with a target sum of s instead of 10. What is the expected value now?

Level 3 Solution

Let's start off simple. Let $n = 6$. If $s \leq 6$, then we only roll once, so $E(G) = 3.5$.

If $7 \leq s \leq 11$, then similar to the Level 1 Solution:

$$\begin{aligned}
E(G) &= \frac{s-6}{6} \times \left(\frac{1+s-6}{2} + \frac{1+6}{2}\right) + \frac{12-s}{6} \times \frac{s-6+1+6}{2} \\
&= \frac{s-6}{6} \times \left(\frac{s-5}{2} + 3.5\right) + \frac{12-s}{6} \times \frac{s+1}{2} \\
&= \frac{s-6}{6} \times \frac{s+2}{2} + \frac{12-s}{6} \times \frac{s+1}{2} \\
&= \frac{(s-6)(s+2)}{12} + \frac{(12-s)(s+1)}{12} \\
&= \frac{s^2-4s-12}{12} + \frac{12s+12-s^2-s}{12} \\
&= \frac{7s}{12}
\end{aligned}$$

If $s \geq 12$, then we will always obtain a sum $\leq 12 \leq s$, so

$$E(G) = E(F+S) = E(F) + E(S) = 2E(F) = 2 \times 3.5 = 7$$

Now let n be arbitrary. Following similar reasoning to the case above, if $s \leq n$, we only roll once, so

$$E(G) = \frac{n+1}{2}$$

If $n+1 \leq s \leq 2n-1$, then:

$$
\begin{aligned}
E(G) &= \frac{s-n}{n} \times \left(\frac{1+s-n}{2} + \frac{1+n}{2} \right) + \frac{2n-s}{n} \times \frac{s-n+1+n}{2} \\
&= \frac{s-n}{n} \times \frac{s+2}{2} + \frac{2n-s}{6} \times \frac{s+1}{2} \\
&= \frac{(s-n)(s+2)}{2n} + \frac{(2n-s)(s+1)}{2n} \\
&= \frac{s^2+2s-ns-2n}{2n} + \frac{2ns+2n-s^2-s}{2n} \\
&= \frac{(n+1)s}{2n}
\end{aligned}
$$

Therefore, the final solution is:

$$
E(G) = \begin{cases} \frac{n+1}{2} & \text{if } 1 \leq s \leq n \\ \frac{(n+1)s}{2n} & \text{if } n+1 \leq s \leq 2n-1 \\ n+1 & \text{if } s \geq 2n \end{cases}
$$

Checking Answers with Python:

```python
# Level 3 Game

# Changing Target Sum
for target in range(1, 21):
    print(f'Target Sum: {target}')
    print('---------------------')

    # Changing Dice Size
    for size in range(1, 11):
        # Resetting Dictionary
        table = {}

        for _ in range(trials):
            # First Roll
            first_roll = random.randint(1, size)

            # Following Game Rules
            if first_roll <= target - size:
                second_roll = random.randint(1, size)
                result = first_roll + second_roll
            else:
                result = first_roll

            # Storing Outcomes in Dictionary
```

```python
            if result not in table:
                table[result] = 0
            table[result] += 1

        # Calculating Expected Value
        ev = sum(map(lambda x: x[0] * x[1], table.items())) / trials

        # Calculating Theoretical Answer
        if 1 <= target <= size:
            answer = (1 + size) / 2
        elif size + 1 <= target <= 2*size - 1:
            answer = (size + 1) * target / (2 * size)
        else:
            answer = 1 + size

        # Outputting Results
        print('{:<9s}{:<15s}{:<10s}'.format(f'n = {size}', f'Answer: {round(answer, 2)}', f'Simulated Expected Value: {round(ev, 2)}'))
```

Output

```
Target Sum: 1
---------------------
n = 1    Answer: 1.0    Simulated Expected Value: 1.0
n = 2    Answer: 1.5    Simulated Expected Value: 1.5
n = 3    Answer: 2.0    Simulated Expected Value: 2.0
n = 4    Answer: 2.5    Simulated Expected Value: 2.5
n = 5    Answer: 3.0    Simulated Expected Value: 3.0
n = 6    Answer: 3.5    Simulated Expected Value: 3.5
n = 7    Answer: 4.0    Simulated Expected Value: 4.0
n = 8    Answer: 4.5    Simulated Expected Value: 4.5
n = 9    Answer: 5.0    Simulated Expected Value: 5.0
n = 10   Answer: 5.5    Simulated Expected Value: 5.5
...
Target Sum: 9
---------------------
n = 1    Answer: 2      Simulated Expected Value: 2.0
n = 2    Answer: 3      Simulated Expected Value: 3.0
n = 3    Answer: 4      Simulated Expected Value: 4.0
n = 4    Answer: 5      Simulated Expected Value: 5.0
n = 5    Answer: 5.4    Simulated Expected Value: 5.4
n = 6    Answer: 5.25   Simulated Expected Value: 5.25
```

```
n = 7    Answer: 5.14   Simulated Expected Value: 5.14
n = 8    Answer: 5.06   Simulated Expected Value: 5.06
n = 9    Answer: 5.0    Simulated Expected Value: 5.0
n = 10   Answer: 5.5    Simulated Expected Value: 5.5
...
Target Sum: 20
---------------------
n = 1    Answer: 2      Simulated Expected Value: 2.0
n = 2    Answer: 3      Simulated Expected Value: 3.0
n = 3    Answer: 4      Simulated Expected Value: 4.0
n = 4    Answer: 5      Simulated Expected Value: 5.0
n = 5    Answer: 6      Simulated Expected Value: 6.0
n = 6    Answer: 7      Simulated Expected Value: 7.0
n = 7    Answer: 8      Simulated Expected Value: 7.99
n = 8    Answer: 9      Simulated Expected Value: 9.0
n = 9    Answer: 10     Simulated Expected Value: 10.0
n = 10   Answer: 11     Simulated Expected Value: 11.0
```

The Harder Question (Level 4)

Let's go back to the Level 1 game but with fair dice that can be of different sizes. What is the expected value now?

Level 4 Solution

Let the two dice have sizes n_1 and n_2 respectively. Let's start with $s = 10$.

Firstly, note that if $\min(n_1, n_2) > 10$, then there is a chance we get an outcome greater than 10 on the first roll, so the game ends here.

If $n_1 + n_2 \leq 10$, then we always roll twice. So,

$$E(G) = E(F+S) = E(F) + E(S) = \frac{n_1+1}{2} + \frac{n_2+1}{2} = \frac{n_1+n_2+2}{2}$$

What if $n_1 + n_2 \geq 11$? Suppose $n_1 = 6$ and $n_2 = 5$. If we roll the 6-sided dice first, then:

$$E(G) = \frac{5}{6} \times (3+3) + \frac{1}{6} \times 6 = 5 + 1 = 6$$

However, if we roll the 5-sided first, then:

$$E(G) = \frac{4}{5} \times (2.5+3.5) + \frac{1}{5} \times 5 = \frac{24}{5} + 1 = \frac{29}{5}$$

The order of the rolls affects $E(G)$. More importantly, it seems that rolling the smaller sized dice first has reduced the expected value!

Let's generalise this. Suppose $\min(n_1, n_2) \leq 10$ (for the game to be played) but $n_1 + n_2 \geq 11$ and without the loss of generality that $n_1 \leq n_2$. If we roll the smaller n_1-sided dice first, then:

$$\begin{aligned}
E(G) &= \frac{10-n_2}{n_1} \times \left(\frac{1+10-n_2}{2} + \frac{1+n_2}{2}\right) + \frac{n_1+n_2-10}{n_1} \times \frac{10-n_2+1+n_1}{2} \\
&= \frac{10-n_2}{n_1} \times 6 + \frac{(n_1+n_2-10)(n_1-n_2+11)}{2n_1} \\
&= \frac{20-2n_2}{2n_1} \times 6 + \frac{n_1^2-n_2^2+11(n_1+n_2)-10(n_1-n_2)-110}{2n_1} \\
&= \frac{120-12n_2}{2n_1} + \frac{n_1^2-n_2^2+n_1+21n_2-110}{2n_1} \\
&= \frac{n_1^2-n_2^2+n_1+9n_2+10}{2n_1}
\end{aligned}$$

Now let's roll the larger n_2-sided dice first. For this to be possible, we also need $n_2 \leq 10$. Then we can just swap n_1 and n_2 in the calculation. So,

$$E(G) = \frac{n_2^2 - n_1^2 + n_2 + 9n_1 + 10}{2n_2}$$

Note that if $n_1 = n_2$, then for both formulas

$$E(G) = \frac{10n_1 + 10}{2n_1} = \frac{5n_1 + 5}{n_1} = 5 + \frac{5}{n_1},$$

which is the same as the Level 1 solution (a good sanity check).

Note to the reader: don't just assume the formula is right. Try out your own examples and see if it checks out.

For those who tried out their own examples, they should figure that there is a minor bug in the formula. Let's think about the case when $n_2 \geq 11$. We must roll the the smaller n_1-sided dice first for the game to be played. If we take $n_1 = 1$ and $n_2 = 11$, according to the formula,

$$E(G) = \frac{1^2 - 11^2 + 1 + 9(11) + 10}{2} = -5$$

This is absurd. We are rolling positive-numbered dice and getting a negative expected value! Why does this happen? Intuitively, the formula assumes that, regardless of the values we assign n_1 and n_2, there will always be situations where we roll twice. However, with $n_1 = 1$ and $n_2 = 11$, if we roll the we roll the the smaller 1-sided dice first, then we will never have a second roll (convince yourself why). Hence, let's adjust our formula for the case when $n_2 \geq 11$. Firstly, we must roll the smaller n_1-sided dice first (for the game to be played). We game ends after the first roll, so

$$E(G) = E(F) = \frac{n_1 + 1}{2}$$

Therefore, the solution for $s = 10$ is:

$$E(G) = \begin{cases} \dfrac{n_1 + n_2 + 2}{2} & \text{if } 2 \leq n_1 + n_2 \leq 10 \\ \dfrac{n_1^2 - n_2^2 + n_1 + 9n_2 + 10}{2n_1} & \text{if } n_1 + n_2 \geq 11,\ \ n_1 \leq n_2 \leq 10, \text{ the } n_1\text{-sided dice rolled first} \\ \dfrac{n_2^2 - n_1^2 + n_2 + 9n_1 + 10}{2n_2} & \text{if } n_1 + n_2 \geq 11,\ n_1 \leq n_2 \leq 10, \text{ the } n_2\text{-sided dice rolled first} \\ \dfrac{n_1 + 1}{2} & \text{if } n_1 + n_2 \geq 11,\ n_1 \leq 10,\ n_2 \geq 11 \end{cases}$$

What do these formulae tell us?

- When $2 \leq n_1 + n_2 \leq 10$, $E(G)$ is an increasing function of $n_1 + n_2$, so it is has maximum value of 6 when $n_1 + n_2 = 10$.
- When $n_1 \neq n_2$, $n_1 + n_2 \geq 11$, and $n_1 \leq n_2 \leq 10$, $E(G)$ is maximised when the larger n_2-sided dice is rolled first. Proof (sort of):

$$\begin{aligned} n_2 \geq n_1 \Rightarrow n_2 - n_1 &\geq 0 \\ &\Rightarrow (n_2 - n_1)((n_2 + n_1)^2 - 9(n_2 + n_1) - 10) \geq 0 \qquad \text{(convince yourself)} \\ &\Rightarrow (n_2 - n_1)(n_2^2 + n_1 n_2 + n_1^2 - 9(n_2 + n_1) - 10 + n_1 n_2) \geq 0 \\ &\Rightarrow (n_2 - n_1)(n_2^2 + 2n_1 n_2 + n_1^2 - 9(n_2 + n_1) - 10) \geq 0 \\ &\Rightarrow n_1 n_2 (n_2 - n_1) + (n_2^3 - n_1^3) - 9(n_2^2 - 9n_1^2) - 10(n_2 - n_1) \geq 0 \\ &\Rightarrow n_1(n_2^2 - n_1^2 + n_2 + 9n_1 + 10) \geq n_2(n_1^2 - n_2^2 + n_1 + 9n_2 + 10) \\ &\Rightarrow \frac{n_2^2 - n_1^2 + n_2 + 9n_1 + 10}{n_2} \geq \frac{n_1^2 - n_2^2 + n_1 + 9n_2 + 10}{n_1} \\ &\Rightarrow \frac{n_2^2 - n_1^2 + n_2 + 9n_1 + 10}{n_2} \geq \frac{n_1^2 - n_2^2 + n_1 + 9n_2 + 10}{n_1} \\ &\Rightarrow \frac{n_2^2 - n_1^2 + n_2 + 9n_1 + 10}{2n_2} \geq \frac{n_1^2 - n_2^2 + n_1 + 9n_2 + 10}{2n_1} \end{aligned}$$

- When $n_1 + n_2 \geq 11$, $E(G)$ has a maximum value of 5.5 when $n_1 = 10$.
- $E(G)$ is maximised when $n_1 = 4, 5$ and $n_2 = 10$ and we roll the 10-sided dice first.

Checking Answers with Python:

```
# Level 4 Game

outcomes = []
for n_1 in range(1, 11):
    for n_2 in range(1, 11):
        # Resetting Dictionary
        table = {}
        for _ in range(trials):
            # First Roll
            first_roll = random.randint(1, n_1)
```

```

            # Following Game Rules
            if first_roll <= target - n_2:
                second_roll = random.randint(1, n_2)
                result = first_roll + second_roll
            else:
                result = first_roll

            # Storing Outcomes in Dictionary
            if (n_1, n_2, result) not in table:
                table[(n_1, n_2, result)] = 0
            table[(n_1, n_2, result)] += 1

        # Calculating Expected Value
        ev = sum(map(lambda x: x[0][-1] * x[1], table.items())) / trials

        # Calculating Theoretical Answer
        if n_1 + n_2 <= 10:
            answer = [(n_1, n_2, (n_1 + n_2 + 2) / 2, ev)]
        elif n_1 <= 10 and n_2 <= 10:
            answer = [(n_1, n_2, (n_1**2 - n_2**2 + n_1 + 9*n_2 + 10) / (2*n_1), ev)]
        elif n_2 >= 11:
            answer = [(n_1, n_2,(1 + n_1) / 2, ev)]

        # Adding results to an existing list
        results.extend(answer)

# Sorting list by expected value
results.sort(key=lambda x: x[2])

# Outputting results
for n_1, n_2, answer, ev in results:
    print('{:<15s}{:<15s}{:<15s}{:<15s}'.format(f'n_1 = {n_1}', f'n_2 = {n_2}', f'Answer: {round(answer, 2)}', f'Simulated Expected Value: {round(ev, 2)}'))
```

Output

```
n_1 = 1        n_2 = 10       Answer: 1.0    Simulated Expected Value: 1.0
n_1 = 2        n_2 = 10       Answer: 1.5    Simulated Expected Value: 1.5
n_1 = 1        n_2 = 1        Answer: 2.0    Simulated Expected Value: 2.0
n_1 = 3        n_2 = 10       Answer: 2.0    Simulated Expected Value: 2.0
```

```
n_1 = 1         n_2 = 2         Answer: 2.5     Simulated Expected Value: 2.5
n_1 = 2         n_2 = 1         Answer: 2.5     Simulated Expected Value: 2.5
n_1 = 4         n_2 = 10        Answer: 2.5     Simulated Expected Value: 2.5
n_1 = 1         n_2 = 3         Answer: 3.0     Simulated Expected Value: 3.0
...
n_1 = 9         n_2 = 4         Answer: 6.67    Simulated Expected Value: 6.66
n_1 = 9         n_2 = 5         Answer: 6.67    Simulated Expected Value: 6.67
n_1 = 10        n_2 = 2         Answer: 6.7     Simulated Expected Value: 6.69
n_1 = 10        n_2 = 7         Answer: 6.7     Simulated Expected Value: 6.7
n_1 = 10        n_2 = 3         Answer: 6.9     Simulated Expected Value: 6.9
n_1 = 10        n_2 = 6         Answer: 6.9     Simulated Expected Value: 6.91
n_1 = 10        n_2 = 4         Answer: 7.0     Simulated Expected Value: 7.0
n_1 = 10        n_2 = 5         Answer: 7.0     Simulated Expected Value: 7.0
```

The Boss Question (Level 5)

Now for the final question. Let's play Level 4 game but with a target sum of s instead of 10. What is the expected value now?

Level 5 Solution

Following similar reasoning, if $n_1 + n_2 \leq s$, then

$$E(G) = \frac{n_1 + n_2 + 2}{2}$$

Now, if $n_1 + n_2 \geq s + 1$, $n_1 \leq n_2 \leq s$ and the smaller n_1-sided dice rolled first, then

$$\begin{aligned}
E(G) &= \frac{s - n_2}{n_1} \times \left(\frac{1 + s - n_2}{2} + \frac{1 + n_2}{2} \right) + \frac{n_1 + n_2 - s}{n_1} \times \frac{s - n_2 + 1 + n_1}{2} \\
&= \frac{s - n_2}{n_1} \times \frac{s + 2}{2} + \frac{(n_1 + n_2 - s)(n_1 - n_2 + s + 1)}{2n_1} \\
&= \frac{(s - n_2)(s + 2)}{2n_1} + \frac{n_1^2 - n_2^2 + (s + 1)(n_1 + n_2) - s(n_1 - n_2) - s(s + 1)}{2n_1} \\
&= \frac{s^2 + (2 - n_2)s - 2n_2 + n_1^2 - n_2^2 + s(n_1 + n_2) + (n_1 + n_2) - s(n_1 - n_2) - s(s + 1)}{2n_1} \\
&= \frac{n_1^2 - n_2^2 + n_1 + (s - 1)n_2 + s}{2n_1}
\end{aligned}$$

Now if $n_1 + n_2 \geq s + 1$, $n_1 \leq n_2 \leq s$ and the larger n_2-sided dice rolled first, by swapping n_1 and n_2, we obtain

$$E(G) = \frac{n_2^2 - n_1^2 + n_2 + (s - 1)n_1 + s}{2n_2}$$

Finally, if if $n_1 + n_2 \geq s + 1$, $n_1 \leq s$, $n_2 \geq s + 1$, then

$$E(G) = \frac{n_1 + 1}{2}$$

Therefore (by essentially replacing 10 with s) the final solution is:

$$E(G) = \begin{cases} \dfrac{n_1 + n_2 + 2}{2} & \text{if } 2 \leq n_1 + n_2 \leq s \\ \dfrac{n_1^2 - n_2^2 + n_1 + (s-1)n_2 + s}{2n_1} & \text{if } n_1 + n_2 \geq s + 1,\ \ n_1 \leq n_2 \leq s, \text{ the } n_1\text{-sided dice rolled first} \\ \dfrac{n_1^2 - n_2^2 + n_2 + (s-1)n_1 + s}{2n_2} & \text{if } n_1 + n_2 \geq s + 1,\ n_1 \leq n_2 \leq 10, \text{ the } n_2\text{-sided dice rolled first} \\ \dfrac{n_1 + 1}{2} & \text{if } n_1 + n_2 \geq s + 1,\ n_1 \leq s,\ n_2 \geq s + 1 \end{cases}$$

Checking Answers with Python:

```python
# Level 5 Game

for target in range(1, 11):
    # Resetting Results
    results = []

    print(f'Target Sum: {target}')
    print('---------------------')

    for n_1 in range(1, target+1):
        for n_2 in range(1, target+1):
            # Resetting Dictionary
            table = {}
            for _ in range(trials):
                # First Roll
                first_roll = random.randint(1, n_1)

                # Following Game Rules
                if first_roll <= target - n_2:
                    second_roll = random.randint(1, n_2)
                    result = first_roll + second_roll
                else:
                    result = first_roll

                # Storing Outcomes in Dictionary
                if (n_1, n_2, result) not in table:
                    table[(n_1, n_2, result)] = 0
                table[(n_1, n_2, result)] += 1

            # Calculating Expected Value
            ev = sum(map(lambda x: x[0][-1] * x[1], table.items())) / trials

            # Calculating Theoretical Answer
```

```python
            if n_1 + n_2 <= target:
                answer = [(n_1, n_2, (n_1 + n_2 + 2) / 2, ev)]
            elif n_1 <= target and n_2 <= target:
                answer = [(n_1, n_2, (n_1**2 - n_2**2 + n_1 + (target-1)*n_2 + target) / (2*n_1), ev)]
            elif n_2 >= 11:
                answer = [(n_1, n_2,(1 + n_1) / 2, ev)]

            # Adding results to an existing list
            results.extend(answer)


    # Sorting list by expected value
    results.sort(key=lambda x: x[2])

    # Outputting results
    for n_1, n_2, answer, ev in results:
        print('{:<11s}{:<11s}{:<15s}{:<15s}'.format(f'n_1 = {n_1}', f'n_2 = {n_2}', f'Answer: {round(answer, 2)}', f'Simulated Expected Value: {round(ev, 2)}'))
    print()
```

Output

```
Target Sum: 1
---------------------
n_1 = 1    n_2 = 1    Answer: 1.0    Simulated Expected Value: 1.0

Target Sum: 2
---------------------
n_1 = 1    n_2 = 2    Answer: 1.0    Simulated Expected Value: 1.0
n_1 = 2    n_2 = 2    Answer: 1.5    Simulated Expected Value: 1.5
n_1 = 1    n_2 = 1    Answer: 2.0    Simulated Expected Value: 2.0
n_1 = 2    n_2 = 1    Answer: 2.0    Simulated Expected Value: 2.0

...


Target Sum: 9
---------------------
n_1 = 1    n_2 = 9    Answer: 1.0    Simulated Expected Value: 1.0
n_1 = 2    n_2 = 9    Answer: 1.5    Simulated Expected Value: 1.5
..,
```

```
n_1 = 9     n_2 = 5     Answer: 6.33   Simulated Expected Value: 6.33
n_1 = 9     n_2 = 4     Answer: 6.39   Simulated Expected Value: 6.38

Target Sum: 10
---------------------
n_1 = 1     n_2 = 10    Answer: 1.0    Simulated Expected Value: 1.0
n_1 = 2     n_2 = 10    Answer: 1.5    Simulated Expected Value: 1.5
...
n_1 = 10    n_2 = 4     Answer: 7.0    Simulated Expected Value: 7.02
n_1 = 10    n_2 = 5     Answer: 7.0    Simulated Expected Value: 6.99
```

Food For Thought

We have successfully scrutinised the two-dice game. The question can continue to be extended, but time is money, and it's time for me to enjoy the rest of my summer. The natural next step would be extending the game to D dice, where $D \geq 3$. If you have any ideas, questions, or have further analysed this question, feel free to contact me at kumarprabhav10@gmail.com!

Resources

Python Jupyter Notebook: https://github.com/Prabhav10/DiceQuestion