Expected Rolls in the Coupon Collector's Problem

Krishna Shukla

September 2024

Part A

Let X be the random variable representing the number of rolls until the first appearance of the face $\lfloor k \rfloor$ on an unbiased k-faced die.

1. Definition of the Random Variable

The random variable X follows a geometric distribution with probability of success $p = \frac{1}{k}$. The probability mass function (PMF) of X is given by:

$$P(X = n) = (1-p)^{n-1}p = \left(1 - \frac{1}{k}\right)^{n-1} \cdot \frac{1}{k}$$

for $n = 1, 2, 3, \ldots$.

2. Expectation of X

The expected value $E(X)$ of a geometrically distributed random variable with parameter p is given by:

$$E(X) = \sum_{n=1}^{\infty} n \cdot P(X = n)$$

Substituting the PMF:

$$E(X) = \sum_{n=1}^{\infty} n \cdot \left(1 - \frac{1}{k}\right)^{n-1} \cdot \frac{1}{k}$$

3. Simplifying the Expectation

We can factor out $\frac{1}{k}$:

$$E(X) = \frac{1}{k} \sum_{n=1}^{\infty} n \left(1 - \frac{1}{k}\right)^{n-1}$$

4. Using the Formula for the Sum of a Geometric Series

The sum $\sum_{n=1}^{\infty} nx^{n-1}$ for $|x| < 1$ can be derived from the geometric series:

$$\sum_{n=0}^{\infty} x^n = \frac{1}{1-x} \quad \Rightarrow \quad \sum_{n=1}^{\infty} nx^{n-1} = \frac{1}{(1-x)^2}$$

Substituting $x = 1 - \frac{1}{k}$:

$$\sum_{n=1}^{\infty} n\left(1 - \frac{1}{k}\right)^{n-1} = \frac{1}{\left(\frac{1}{k}\right)^2} = k^2$$

5. Final Calculation

Substituting back into the equation for $E(X)$:

$$E(X) = \frac{1}{k} \cdot k^2 = k$$

Conclusion

Thus, the expected number of rolls until the first occurrence of $\lfloor k \rfloor$ on an unbiased k-faced die is k.

Part B

The expected number of rolls to see every number from 1 to k at least once on a k-faced die is a classic problem known as the Coupon Collector's Problem.

Expected Number of Rolls

The expected number of trials E_k required to collect all k coupons can be expressed mathematically as:

$$E_k = k\sum_{i=1}^{k} \frac{1}{i}$$

1. **Probability of Collecting a New Coupon**: After collecting j distinct coupons, the probability of rolling a new, previously uncollected coupon is given by:

$$p_j = \frac{k-j+1}{k}$$

2. **Expected Rolls for Each New Coupon**: The expected number of rolls required to collect a new coupon after having j distinct coupons is the reciprocal of this probability:

$$E[\text{rolls to collect next coupon}] = \frac{1}{p_j} = \frac{k}{k-j+1}$$

3. **Total Expected Rolls**: Summing these expectations for all j from 1 to k, we find:

$$E_k = \sum_{j=1}^{k} \frac{k}{k-j+1} = k\sum_{i=1}^{k} \frac{1}{i}$$

Thus, the expected number of rolls needed to see every number from 1 to k at least once can be expressed as:

$$E_k = k\left(1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{k}\right) \approx k\log(k)$$

Part C

Now, consider the situation where we have three distinct numbers with the following probabilities:

$$P(1) = \frac{1}{4}$$
$$P(2) = \frac{1}{2}$$
$$P(3) = \frac{1}{4}$$

Step 1: Understanding the Expected Rolls for Each New Number

We can break this down into stages based on how many distinct numbers we've already seen.

First Distinct Number

The very first roll will always yield a new number because there are no prior results. Thus, the expected number of rolls needed to collect the first distinct number is:

$$E_1 = 1$$

Second Distinct Number

After collecting one distinct number (say, 1), the probability of rolling a new distinct number is:

$$\begin{aligned} P(\text{new}) &= 1 - P(\text{same}) = 1 - P(1) \\ &= 1 - \frac{1}{4} = \frac{3}{4} \end{aligned}$$

Thus, the expected number of rolls needed to obtain the second distinct number is:

$$E_2 = \frac{1}{P(\text{new})} = \frac{1}{\frac{3}{4}} = \frac{4}{3}$$

Third Distinct Number

Once we have two distinct numbers (for example, 1 and 2), the probability of rolling the third distinct number is:

$$P(\text{new}) = P(3) = \frac{1}{4}$$

So, the expected number of rolls to get this third distinct number is:

$$E_3 = \frac{1}{P(\text{new})} = \frac{1}{\frac{1}{4}} = 4$$

Step 2: Total Expected Number of Rolls

Now that we have the expected number of rolls for each distinct number, we can calculate the total expected number of rolls needed to collect all three distinct numbers:

$$E = E_1 + E_2 + E_3$$

Substituting the values we found:

$$E = 1 + \frac{4}{3} + 4$$

To simplify this, we need a common denominator (which is 3 in this case):

$$E = 1 + \frac{4}{3} + \frac{12}{3} = \frac{3}{3} + \frac{4}{3} + \frac{12}{3} = \frac{19}{3}$$

Step 2: Total Expected Number of Rolls

1 Part D

The value at face $k = 3$ gives an expected value of 6.3, as obtained in the previous part. The code for this plot generation is written in a separate file.

Consider a general k-faced geometric die, where the probability that the upward face is i from a random roll is defined as:

$$P(i) = \begin{cases} \frac{1}{2(k-1)} & \text{if } i = 1 \text{ or } i = k \\ \frac{1}{2(i-1)} & \text{if } 2 \leq i < k \end{cases}$$

Thus, we have: - $P(1) = P(k) = \frac{1}{2(k-1)}$ - $P(2) = \frac{1}{2}$ - $P(3) = \frac{1}{4}$, and so on.

We will write a program to show how the expected number of rolls changes as k increases. If we use a closed-form solution for the expected number of rolls, we will check if it matches the plot.

The following Python code implements the simulation and the closed-form solution:

```
import numpy as np
import matplotlib.pyplot as plt

def simulate_die_rolls(k, num_trials=1000):
    probabilities = [1 / (2 * (k - 1)) if i == 1 or i == k else 1 / (2 * (i - 1)) for i in r
    total_probability = sum(probabilities)
    probabilities = [p / total_probability for p in probabilities]

    def roll_die(probabilities):
        return np.random.choice(range(1, k + 1), p=probabilities)

    def rolls_to_see_all_faces():
        seen_faces = set()
        roll_count = 0
        while len(seen_faces) < k:
            face = roll_die(probabilities)
            seen_faces.add(face)
            roll_count += 1
        return roll_count

    total_rolls = [rolls_to_see_all_faces() for _ in range(num_trials)]
    return np.mean(total_rolls)

def closed_form_solution(k):
    expected_rolls = 0
    for i in range(1, k + 1):
        expected_rolls += 1 / (i * (1 - 1 / (2 * (k - 1))))  # Adjust for probabilities
```

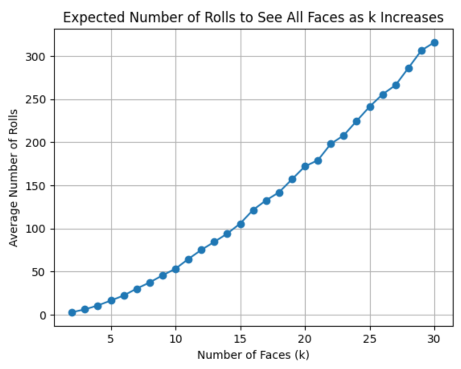

Figure 1: Plot of number of faces vs expected number of rolls

```
    return expected_rolls

k_values = list(range(2, 11))
average_rolls = [simulate_die_rolls(k) for k in k_values]
closed_form_values = [closed_form_solution(k) for k in k_values]

plt.plot(k_values, average_rolls, marker='o', label='Simulated Average Rolls')
plt.plot(k_values, closed_form_values, marker='x', linestyle='--', label='Closed-Form Soluti
plt.xlabel('Number of Faces (k)')
plt.ylabel('Expected Number of Rolls')
plt.title('Expected Number of Rolls to See All Faces as k Increases')
plt.legend()
plt.grid(True)
plt.show()
```

This code is also written in a file This code simulates rolling a k-faced die until all faces are seen and compares the results with the closed-form solution. The plot generated by the code illustrates how the expected number of rolls changes as k increases.

The value according to code for k=3 is 6.307 that is same a svlue we got this verifies the answer