Introduction

Many problems about finding the expected number of turns required to get some outcome can be solved by imagining a gambling game at the casino. Of course, this is nothing more than the martingale method, with obvious applications to the modelling of asset prices and so on. However, I wanted to find more intuitive explanations than searching for the martingale M_n, or in continuous time, (M_t). Along the way I'll cover the famous ABRACADABRA problem and a few other problems I've encountered in interviews and in discussions with peers.

Contents

1 Basic Idea

Assume we are asked for some expected number of rounds (i.e. we are in the discrete world). There are three steps to transforming the math problem into a gambling scenario:

1. Devise a fair betting game. For every \$1 put in, a player should expect to win \$1 in expectation, so that his expected return is \$0.
2. Make the player invest an additional \$1 every round, so that his expected winnings grow by \$1 every round as well. Note that how he bets with his accumulated winnings, if any, does not affect the expected outcome because this money is already in the system. Because the game is fair, the expected winnings should equal the total investment. Using only the money put in at the start is known in Finance as a self-financing strategy, but I digress.
3. Since how one bets doesn't matter, make him bet in such a way that when the game ends (i.e. when an outcome is reached), we know how much money he has made just by knowing how many rounds the game has been played. Oftentimes, the terminal winnings need not even depend on the amount of the time.

Now for a bit of math. If we call this "stopping time" τ ($\tau \in \mathbb{N}$) and the terminal winnings W_τ, then $M_n := W_n - n$ is a martingale and $\mathbb{E}(W_\tau - \tau) = \mathbb{E}(M_\tau) = M_0 = W_0 - 0 = 0$ by the Optional Sampling Theorem. If W_τ depends on the outcome ω but not on τ, we just need to compute $\mathbb{E}(W_\tau) = \mathbb{E}(\tau)$.

Otherwise, if $W_\tau = g_\omega(\tau)$ where g is linear regardless of the outcome ω, then $\mathbb{E}(W_\tau) = \mathbb{E}_\omega(\mathbb{E}(W_\tau \mid \omega)) = \mathbb{E}_\omega(\mathbb{E}(g_\omega(\tau) \mid \omega)) = \mathbb{E}_\omega(g_\omega(\mathbb{E}(\tau \mid \omega)))$. If we are lucky, we will be able to use $\mathbb{E}(\tau) = \mathbb{E}_\omega(\mathbb{E}(\tau \mid \omega))$ to write $\mathbb{E}(W_\tau) = f(\mathbb{E}(\tau))$ for some f. To finish, solve $\mathbb{E}(W_\tau) = f(\mathbb{E}(\tau)) = \mathbb{E}(\tau)$ for $\mathbb{E}(\tau)$.

There are a few technicalities that need to be established, for instance the fact that $E(\tau) < \infty$ (as mentioned in the Wikipedia article and Problem 3.8 of [2]). Since it is usually not too hard to show that all games end relatively quickly (e.g. by showing that the probability that the $N+1$th round is played decreases faster than N itself), I will skip this step for brevity.

2 Getting N heads in a row

What is the expected number of fair coin tosses required to get N heads in a row? The idea here is the same as the ABRACADABRA problem, which I discuss right after.

2.1 Alternative Solutions

As we toss the coins, we are actually jumping between "states" here (0, 1, 2 consecutive heads etc.), with the end state being N heads and the probability of going to any other state depending only on our current state. Thus, this and all subsequent problems can all be represented as Markov Chains. Before introducing the martingale solution, I present two other solutions that work with the Markov Chain directly without requiring any knowledge of gambling or martingales. I think they both have their respective merits for this relatively tractable problem.

2.1.1 Direct summation

Suppose we start with a hot streak of $k < N$ heads before hitting the first tail. Then those first $k+1$ tosses were wasted and we are back to square 1. We denote by X the expected number of rolls required from this starting state, which is just $\mathbb{E}(\tau)$. However, if $k = N$ (with probability $p = \frac{1}{2^N}$), then we are done. Therefore,

$$\begin{aligned}
X &= \frac{1}{2} \times (X+1) + \frac{1}{2^2} \times (X+2) + \frac{1}{2^3} \times (X+3) + \cdots + \frac{1}{2^N} \times (X+N) + \frac{N}{2^N} \\
&= \left(\sum_{k=1}^{N} \frac{1}{2^k} \right) X + \underbrace{\sum_{k=1}^{N} \frac{k}{2^k}}_{S} + \frac{N}{2^N} \\
&= (1 - \frac{1}{2^N})X + \left(2 - \frac{1}{2^{N-1}} - \frac{N}{2^N} \right) + \frac{N}{2^N} \\
\frac{X}{2^N} &= 2 - \frac{1}{2^{N-1}} \\
\mathbb{E}(\tau) = X &= \boxed{2^{N+1} - 2}
\end{aligned}$$

where S can be evaluated as follows:

$$\begin{aligned}
S &:= \frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \cdots + \frac{N}{2^N} \\
\implies 2S &= 1 + \frac{2}{2} + \frac{3}{4} + \cdots + \frac{N}{2^{N-1}} \\
\implies S &= 2S - S \\
&= 1 + \frac{1}{2} + \frac{1}{4} + \dots \frac{1}{2^{N-1}} - \frac{N}{2^N} \\
&= 1 + (1 - \frac{1}{2^{N-1}}) - \frac{N}{2^N} \\
&= 2 - \frac{1}{2^{N-1}} - \frac{N}{2^N}
\end{aligned}$$

2.1.2 Step-by-step

Instead of unrolling everything, let us define C_k to be the number of coin tosses to get k heads in a row and Y_k ($k \geq 1$) the number of extra tosses to get the last head *when we already have $k-1$ consecutive heads*. Then Y is either 1 or $1 + C_k$, each with probability $\frac{1}{2}$. Thus,

$$\begin{aligned}
C_0 &= 0 \\
C_{k+1} &= C_k + \frac{1}{2}(1) + \frac{1}{2}(1 + C_{k+1}) \\
\implies C_{k+1} &= 2C_k + 2 \\
\implies E(\tau) = C_N &= \underbrace{2(2(2(\dots((0)+2)+\dots)+2)+2)+2}_{N*2's} \\
&= (2^{N+1} - 1) - 1 \\
&= \boxed{2^{N+1} - 2}
\end{aligned}$$

From $C_{k+1} = 2C_k + 2$ an easy proof by induction also suffices.

2.2 Solution by gambling

Follow the three-step framework above. First, the casino pays \$2 for correctly calling the toss (either H or T) and nothing otherwise to make things fair. Since we just want N heads in total, we always bet on H and in each round we put all our money in it - both our accumulated winnings and the extra \$1 we invested in this round. Now, when the game ends, we have the \$2 we made off the last \$1 investment, the \$4 that compounded twice from the second last round, all the way to the $\$2^N$ from $N-1$ rounds ago. Thus

$$\begin{aligned}\mathbb{E}(\tau) = \mathbb{E}(W_\tau) = W_\tau &= 2^N + 2^{N-1} + \cdots + 2 \\ &= 2 \times (2^{N-1} + \ldots 1) \\ &= 2(2^N - 1) \\ &= \boxed{2^{N+1} - 2}\end{aligned}$$

The second equality follows because W_τ here is deterministic.

3 ABRACADABRA

Instead of wanting heads, we now want the expected time it takes to draw the 11 characters

"ABRACADABRA"

randomly and in that order from the English alphabet of length 26. To be clear, we imagine spinning a customized "roulette wheel" with 26 pockets labelled "A" to "Z" as many times as needed.

More ambitiously, we can ask how long it takes for monkeys to type Shakespeare on a typewriter, where each character is picked randomly and independent of all previous characters.

3.1 The classic solution

The numbers at the end are exactly the same as those found all over the internet, but I try to give a different explanation.

For this game, the casino pays \$26 for every correct bet on the next character and \$0 otherwise. As before, invest an extra \$1 each round, but according to the following gambling strategy. Instead of betting on the same thing with all our accumulated winnings, we will separate them into different pools (or "threads" in the terminology of computer science) based on when they were invested. If that money was just invested, put it on "A". If it was invested one round ago (on the "A") and we indeed got an "A", then put the \$26 on "B" now. The third most recent pool, if it's still alive, puts all $\$26^2$ in "R", and so on.

Another way to think about this is imagining a new gambler join the game each round trying to create the 11-character sequence, leaving once he goes bust (in fact this is how I was initially taught). The bottom line is that each pool or thread or substrategy or gambler doesn't care about what the other guys have made thus far. (*Note also that the N heads problem is a special case of this betting strategy in which all threads make the same bet each turn.*)

Each such thread has a maximum length of 11, because that's when the game ends. That lucky thread makes $\$26^{11}$ for getting all the characters right. However, not all threads survive. A thread dies once the wrong character shows up becuase it always goes all in. Of course, any string of characters can show up but we are only interested in the last 11 when the game finally ends, which we know to be "ABRACADABRA". At this point, all threads are between 1 and 11 rounds old and we compare their betting histories with the actual outcome. Some characters below are struck out because they represent the busted threads' last bet and what they *would have* bet on subsequently:

ABRACADABRA	Winnings
ABRACADABRA	26^{11}
~~ABRACADABR~~	0
~~ABRACADAB~~	0
~~ABRACADA~~	0
~~ABRACAD~~	0
~~ABRACA~~	0
~~ABRAC~~	0
ABRA	26^4
~~ABR~~	0
~~AB~~	0
A	26

Again, W_τ is deterministic. Therefore, we sum the column up to get

$$\begin{aligned}\mathbb{E}(\tau) = W_\tau &= 26^{11} + 26^4 + 26 \\ &= \boxed{3670344487444778}\end{aligned}$$

3.2 What if people change strategies midway?

The solution above is repeated nearly verbatim throughout the internet. What if we don't follow that strategy to the tee? Suppose when *ABRACADABR* has come up (i.e. we are missing the final "A"), we put all our money on "A" for what may be the final round. That will mean a different terminal payoff if "A" does show up, right?

The problem arises if the roulette wheel spits out *another relevant letter*, say "R". Then while we are back to \$0 and have to restart, with the original strategy there will be two threads that are still "in the money". And if the following letter is "A", these two threads will profit, whereas in our case, we would have made only \$1. By deviating from the prescribed strategy, the terminal strategy is now non-deterministic.

Therefore, each sub-strategy should be independent and we should "stick to the program". This ensures that the terminal payoff is fixed and can easily be tallied.

3.3 Just for fun...

I found a brainteaser on The Guardian that asks whether the expected time for a monkey to get "ABRACADABRA" or "ABRACADABRX" is longer. The answer is that "ABRACADABRX" takes *shorter time* than "ABRACADABRA" in expectation!

As mentioned above, any failure to get the last "A" resets the game, but failing to get "X" and getting "A" instead leaves us still four letters in! We can also see that "ABRACADBRA" gives higher terminal earnings if we follow the same strategy. This also means that "AAAAAAA" will come up less frequently than "ABCDEFG".

If the last result seems counter-intuitive, it might be because we implicitly allowed the monkey to deviate from truly random gibberish. Indeed, if we ask a real monkey to do something with the keyboard, it's likely to type a lot of "S"'s. However, the problem assumes that the hypothetical monkey has no knowledge of the past and faithfully picks each new character *independently* at random. Randomizing over monkeys rather than observing a monkey's random output leads to what is formally known as Algorithmic probability ([1]), under which simpler strings are likely as one might expect.

4 HHT vs HTT

We toss fair coins until either HHT or HTT is obtained. What is the expected number of rounds before this game ends?

Since there are now two end states, W_τ is not deterministic. Therefore, we need to first find the probability of ending on HHT or HTT.

4.1 Probability of each outcome

This sub-problem can be solved in many ways but they all boil down to analyzing the Markov Chain. Consider when a H appears with no H's preceding it; any history before this point does not matter. Now, to get to HTT, we need two more T's, but if the next toss is H, we are doomed. On a TH we restart at a single H. Thus

$$\begin{aligned} P(HTT \mid H) &= \frac{1}{4} \times 1 + \frac{1}{2} \times 0 + \frac{1}{4} \times P(HTH \mid H) \\ \therefore P(HTT \mid H) &= \frac{1}{3} \end{aligned}$$

Thus the game ends with HHT with $p = \frac{2}{3}$ and with HTT with $p = \frac{1}{3}$.

4.2 Gambling on only one side

While there are two outcomes to gamble towards, we can actually pick one, say HHT, and follow the "pools" approach exactly. With the same odds ($2 for a correct call), bet $1 on H in the first turn, bet $2 on H again if H actually shows up, and so on. If the game does end with HTT, we can still tabulate our winnings if any. This yields the following table:

Bets \Outcomes	HHT	HTT
HHT	8	0
HH	0	0
H	0	0
W_τ	8	0

Thus

$$\begin{aligned} \mathbb{E}(\tau) &= \mathbb{E}(W_\tau) \\ &= p \times \mathbb{E}(\tau \mid HHT) + (1-p) \times \mathbb{E}(\tau \mid HTT) \\ &= \frac{2}{3} \times 8 + \frac{1}{3} \times 0 = \boxed{\frac{16}{3}} \end{aligned}$$

4.3 Verification

We can write a short Python script - see `hht.py`. Alternatively, using the Markov Chain itself and denoting by $\mathbb{E}(\tau \mid X)$ the expected *extra* number of steps given event X,

$$\begin{aligned}
\mathbb{E}(\tau) &= 1 + \frac{1}{2}\left(\mathbb{E}(\tau \mid H) + \mathbb{E}(\tau)\right) \\
\mathbb{E}(\tau \mid H) &= 1 + \frac{1}{2}\left(\mathbb{E}(\tau \mid HH) + \mathbb{E}(\tau \mid HT)\right) \\
\mathbb{E}(\tau \mid HH) &= 2,\ \mathbb{E}(\tau \mid HT) = 1 + \frac{\mathbb{E}(\tau \mid H)}{2} \\
\therefore \mathbb{E}(\tau \mid H) &= 1 + \frac{2 + 1 + 0.5\mathbb{E}(\tau \mid H)}{2} = \frac{5}{2} + \frac{\mathbb{E}(\tau \mid H)}{4} \\
\implies \mathbb{E}(\tau \mid H) &= \frac{10}{3} \\
\implies \mathbb{E}(\tau) &= \mathbb{E}(\tau \mid H) + 2 = \boxed{\frac{16}{3}}
\end{aligned}$$

4.4 Backstory

This problem originated in Summer 2019 (!) and a simulation script (`hht.py`) followed, but it took me until Mid-2023 to work the details out to my heart's content (and another year to put it into LaTeX). The original problem from Jane Street is very old, and it only asked for the probability of the game ending with HHT rather than HTT, and I worked it out with standard methods in my Freshman year to prepare for an interview with Akuna Capital for an internship the following summer.

During the actual interview, I was asked over the phone to simulate an unfair coin with $p = \frac{2}{3}$ of showing up as heads with fair coins. I was like "Bingo!" and gave the stupid answer of flipping them until either HHT or HTT appeared. Of course, I could have said something easier like flipping the coins twice at a time, ignoring HH's and assigning the other three outcomes accordingly (as suggested here).

My interviwer at Akuna was understandably bemused, but he pressed on with the impromptu question of how long I would expect to flip before getting a head or tail using my method. I tried to work out the Markov Chain over the phone but he kept saying there was a "much simpler way". Eventually I ran out of time and said that the answer was between 4 and 5. He asked me to pick one of the two and I said "4". I didn't get the internship, and to this day I don't know which method he had in mind. Hopefully it was the martingale solution!

5 A Biased Random Walk

The gambler now tosses an *unfair* coin each round that shows up as heads with probability $\frac{2}{3}$ and tails with probability $1 - \frac{2}{3} = \frac{1}{3}$. The game ends when the total earnings S_n reaches $S_n = 5$ or $S_n = -3$. We want to find, as usual, the expected stopping time $\mathbb{E}(\tau) := \mathbb{E}(\min\{n \mid S_n = 5 \text{ or } S_n = -3\})$. Since there are again two possible outcomes, we start by finding the probability of each.

5.1 Probability of each outcome

The method here was introduced in Problem 36 "Gambler's Ruin' of [3]. There is a solution using exponential martingales but I find the one here more intuitive (the math is the same).

First, the probability P of the gambler ever hitting $S_n = -1$, ignoring any other end-state, satisfies $P = \frac{1}{3} + \frac{2}{3}P^2$. By taking the positive root, $P = \frac{1}{2}$.

Next, because each cumulative step down or up is independent, the probability of hitting $S_n = -3$, again ignoring other barriers, is $P^3 = \frac{1}{8}$. Now factoring in the upper barrier, either we reach $S_n = -3$ without hitting $S_n = 5$ with some probability $1 - p$, or we reach $S_n = 5$ first with probability p and then take 8 steps down. The probability of reaching $S_{n+} = -3$ from $S_n = 5$ is likewise given by $P^8 = \frac{1}{256}$ due to independence. By the law of total probability, we therefore have

$$\begin{aligned} P(\text{Reach } S_n = -3) &= p \times P(\text{Reach } S_n = -3 \text{ after } S_n = 5) + (1-p) \\ \frac{1}{8} &= p \times \frac{1}{256} + (1-p) \\ \therefore p &= \frac{224}{255} \end{aligned}$$

5.2 A cautious gambler

How much the gambler puts in has been suggested to us by the problem: he will cautiously put in \$1 each turn and just wager that \$1. In fact, we can't bet with our accumulated earnings in the "threads" approach because now the last few tosses before the game ends are not known.

Since the coin is now unfair, the casino will pay $\$\frac{3}{2}$ for calling and getting H, \$3 for calling and getting T, and \$0 for getting it wrong. But our gambler will just bet H all the time to simplify calculations. We know from simple maths that if the game ends at time τ with $S_\tau = 5$, then $\#H = \frac{\tau+5}{2}$, and if $S_\tau = -3$, then $\#H = \frac{\tau-3}{2}$. Therefore, taking into the payoff,

$$\begin{aligned} \mathbb{E}(W_\tau) &= p\left(\frac{\mathbb{E}(\tau \mid S_\tau = 5) + 5}{2} \times \frac{3}{2}\right) + (1-p)\left(\frac{\mathbb{E}(\tau \mid S_\tau = -3) - 3}{2} \times \frac{3}{2}\right) \\ &= \frac{3}{4}\left(p \times \mathbb{E}(\tau \mid S_\tau = 5) + (1-p) \times \mathbb{E}(\tau \mid S_\tau = -3)\right) + \frac{15}{4}p - \frac{9}{4}(1-p) \\ &= \frac{3}{4}\mathbb{E}(\tau) + 6p - \frac{9}{4} \\ &= \mathbb{E}(\tau) \\ \implies \mathbb{E}(\tau) &= 4 \times 6 \times \frac{224}{255} - 9 \\ &= \boxed{\frac{1027}{85}} \approx 12.08 \end{aligned}$$

5.3 Verification

One can easily write a Python script like I did for the previous problem. We could also solve the recurrence relations directly, which amounts to inverting the matrix that defines a linear system of equations. There's a fancier but impractical method of solving the recurrence relations with generating functions [4] which I *do not recommend*.

References

[1] M. Hutter, S. Legg, and P. M.B. Vitanyi. "Algorithmic probability". In: *Scholarpedia* 2.8 (2007). revision #151509, p. 2572. DOI: 10.4249/scholarpedia.2572.

[2] M.S. Joshi, N. Denson, and A. Downes. *Quant Job Interview: Questions and Answers*. CreateSpace Independent Publishing Platform, 2008. ISBN: 9781438217031. URL: https://books.google.com/books?id=y7qEOgAACAAJ.

[3] F. Mosteller. *Fifty Challenging Problems in Probability with Solutions*. Dover Books on Mathematical and Logical Puzzles, Cryptography and Word Recreations. Dover Publications, 1987. ISBN: 9780486653556. URL: https://books.google.com/books?id=sQr5AAAAQBAJ.

[4] H.S. Wilf. *Generatingfunctionology*. Academic Press, 2014. ISBN: 9781483276632. URL: https://books.google.com/books?id=XOPMBQAAQBAJ.