1 Problem Statement

You are offered to play a game where you win if you flip heads the number of times I previously flipped tails. That is, to win, you need to flip heads N times where $N = T + 1$, with T the cumulative total of previous coin flips being tails. (Assume a fair coin.) What is the probability of winning this game? (This comes from the fivethirtyeight Riddler from August 3, 2018. The solution method is also from there, as I originally just did the simulation in python.)

2 Answer

It is easier to think of this from the other perspective calling a win above a loss below. Then a "win" only occurs if you flip tails often enough (avoid a flipping heads $T + 1$ times with T the number of times you previously flipped tails).

The probability of winning on zero previous flips is 2^{-1} times the probability of winning on 1 tail flip, because you have to get a tail initially or you lose. The probability of winning on one previous tail flip is $(1 - 2^{-2})$ times the probability of winning on 3 tail flips. Here, the reasoning is that the only way you lose, is if you flip two heads after your first tail. In any other case you are then on your second tail. Clearly this will continue, for the Tth tail flip. We have $\Pr(T)$ for us "winning" as

$$\Pr(T) = \left(1 - 2^{-T-1}\right) \Pr(T + 1) \tag{1}$$

Thus, our probability of winning after T flips is recursive. The probability of the losing the game (that is from the beginning) is $\Pr(0)$ which we can write as

$$\Pr(0) = \left(1 - 2^{0-1}\right) \Pr(1) = \left(1 - 2^{-(0+1)}\right) \left(1 - 2^{-(1+1)}\right) \Pr(2) \tag{2}$$

$$= \prod_{s=1}^{\infty} \left(1 - 2^{-(s+1)}\right) \tag{3}$$

where we assume $\lim_{N\to\infty} \Pr(N) \to 1$ because then the factor that prevents us from always getting to the Nth coin flip becomes $2^{-(N+1)} \to 0$. That is eventually it is "certain" we will win because the likelihood of a string of heads of that size becomes basically impossible.

Okay, let's return back to the terminology in the question. What we have found is the probability of losing after T flips in the actual game is given by (1). We would like to know the probability of winning the game overall. This is simply $1 - \Pr(0)$ because $\Pr(0)$ is now the probability of losing the game. Thus the answer is

$$1 - \prod_{s=0}^{\infty} \left(1 - \frac{1}{2^{s+1}}\right) = 1 - \prod_{s=1}^{\infty} \left(1 - \frac{1}{2^{s}}\right) \tag{4}$$

Let's see what we get for "partial" products. We have

$$\Pi(T) = \prod_{s=1}^{T} \left(1 - \frac{1}{2^{s}}\right) \tag{5}$$

$$\Pi(T) = \prod_{s=1}^{T} \left(\frac{2^s - 1}{2^{s}}\right) \tag{6}$$

$$\Pi(1) = \frac{1}{2} \approx 0.5 \tag{7}$$

$$\Pi(2) = \frac{1}{2}\frac{3}{4} = \frac{3}{8} \approx 0.375 \tag{8}$$

$$\Pi(3) = \frac{1}{2}\frac{3}{4}\frac{7}{8} = \frac{21}{64} \approx 0.328\,125 \tag{9}$$

$$\Pi(4) = \frac{1}{2}\frac{3}{4}\frac{7}{8}\frac{15}{16} = \frac{315}{512} \approx 0.307\,617 \tag{10}$$

$$\Pi(N) = \frac{\prod_{s=1}^{N}(2^N - 1)}{2^{N(N+1)/2}} \tag{11}$$

$$\Phi \equiv \lim_{N\to} \Pi(N) \tag{12}$$

We find

$$\Phi \approx 0.288\,788\,095\,086\,602\,9 \tag{13}$$

$$1 - \Phi \approx 0.711\,211\,904\,913\,397\,4 \tag{14}$$

and so the probability of winning (in the original formulation of the game) is about 71%.

We can find these numbers and do a simulation trial with the file below.

headTailsgame.py

```
#!/usr/bin/env python3

import numpy as np
import matplotlib.pyplot as plt

# From The Riddler Aug. 3 2018 puzzle

# find the probability to win in a game where you need T+1 heads in a row where T is the number of previous tails found (with a fair coin)


# calculate series prod_j=1^infty (2^j-1)/2^j
def Phi(digits):
  phi=0.5
  eps=10**(-digits)
  err=1
  j=2
  while err>eps:
    phiold=phi
    phi=(2**j-1)/2**j*phi
    err=np.abs(phi-phiold)/phi
    j=j+1
  return phi

# set random seed
np.random.seed(1)

# looks through series to see if the game is a win.
def searchseries(series,comment):
  if comment:
    print(series)
  tails=0
  heads=0
  for j in range(len(series)):
    if (heads>tails):
      return 1
    if (series[j]==0):
      tails=tails+1
      heads=0
```

```
39      if (series[j]==1):
40        heads=heads+1
41    if (heads>tails):
42      return 1
43    return 0
44
45  # constructs a single game, inefficient as continues
46  # after game is won to cutoff
47  # However, this should work well since victories
48  # should mostly come from short runs, so a small
49  # cutoff should get us a good approximation
50  def single_game(cutoff,comment):
51    trials=np.random.random(cutoff)
52    trials=(trials>0.5)*1
53    x=searchseries(trials,comment)
54    return x
55
56
57  # set cutoff and number of trials of game to play
58  def trials(num,cutoff,comment=False):
59    trial_list=np.zeros(num)
60    for i in range(num):
61      trial_list[i]=single_game(cutoff,comment)
62      if comment:
63        print(trial_list[i])
64    win=trial_list.sum()/np.shape(trial_list)[0]
65    return win
66
67  # actual answer to 10 digits
68  tru=Phi(15)
69  print("actual",1-tru)
70  # two trials to see the accuracy we are approximately at
71  a=trials(10000,9)
72  b=trials(10000,9)
73  print("a",a)
74  print("b",b)
```