Vamshi Jandhyala

Fifty Strings in a Box

✠

May 2026

A counting puzzle from the Monthly Mindbenders.[1]

Problem

To keep your visiting five-year-old niece busy, you mix 50 strings of orange yarn in a box and ask her to successively grab two random ends and tie them together until there are no loose ends remaining. The result will of course be some number of loops of string. How many, on average?

Solution

Write L_n for the expected number of loops made from n strings. A box of n strings holds $2n$ loose ends, and every tie uses two of them, so there are exactly n ties and the process always ends with the box empty of loose ends.

Look at the first tie. Pick up any end at all; by symmetry it does not matter which, since the ends are interchangeable before anything has been tied. That end is joined to one of the other $2n-1$ ends, each equally likely. Exactly one of those candidates is the other end of the same piece of string. Two cases follow.

- With probability $\dfrac{1}{2n-1}$ the end meets its own partner. The string closes into a loop, which leaves the box and is never touched again. One loop has been completed and $n-1$ strings remain.
- With probability $\dfrac{2n-2}{2n-1}$ the end meets an end of a different string. The two strings become one longer string with two free ends. No loop is completed and $n-1$ strings remain.

Both branches leave $n-1$ strings in the box, each with two free ends, and nothing about the future depends on how long any of those strings is or on which ties produced it. So the whole process restarts on $n-1$ strings, and the expected number of loops still to come is L_{n-1} in either case. Conditioning on the first tie,

$$L_n = \frac{1}{2n-1}(1 + L_{n-1}) + \frac{2n-2}{2n-1} L_{n-1} = \frac{1}{2n-1} + L_{n-1}.$$

[1] Peter Winkler, *Monthly Mindbenders*, National Museum of Mathematics, May 2026, momath.org/mindbenders, a variation of a classic.

That self-similarity is the whole of the argument. It is also why the answer is a plain sum of reciprocals and not something more intricate: the state of the box is described by a single number, the count of strings, and each tie reduces that count by one at a cost that depends on nothing else.

With $L_0 = 0$ the recursion telescopes:

$$L_n = \sum_{k=1}^{n} \frac{1}{2k-1}.$$

For the niece's box of fifty,

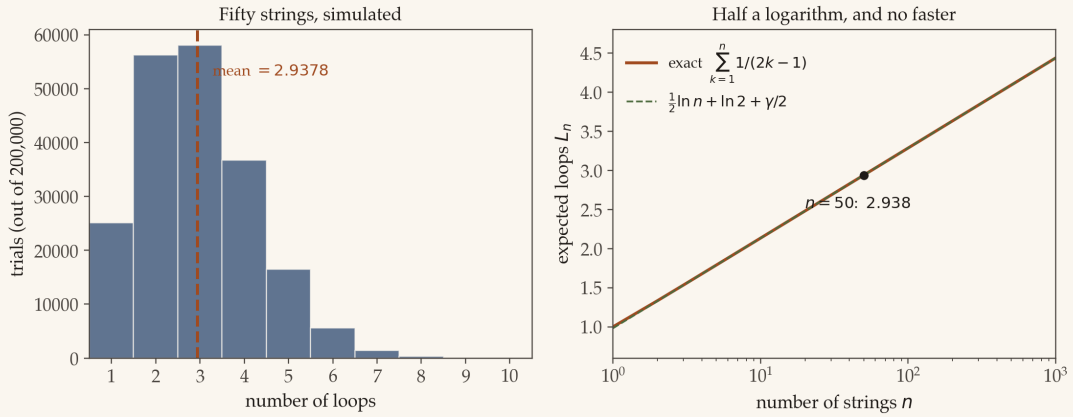
$$L_{50} = 1 + \frac{1}{3} + \frac{1}{5} + \cdots + \frac{1}{99} = \boxed{\sum_{k=1}^{50} \frac{1}{2k-1} \approx 2.93777.}$$

A shade under three loops, from fifty strings and fifty ties.

The same sum without induction

Linearity of expectation gets there in one line, and the counting is worth seeing on its own. Number the ties by the state they act on: some tie takes the box from k strings to $k-1$ strings, for each k from n down to 1. At that tie the box holds $2k$ ends, the chosen end has $2k-1$ possible partners, and exactly one of them closes a loop, so the tie completes a loop with probability $1/(2k-1)$. Let X_k be the indicator of that event. The total number of loops is $\sum_k X_k$, since every loop is closed by exactly one tie, and expectation adds without any need for independence:

$$\mathbb{E}\Big[\sum_{k=1}^{n} X_k\Big] = \sum_{k=1}^{n} \frac{1}{2k-1}.$$



Left: the simulated number of loops from fifty strings over 200,000 trials, with the exact mean 2.9378 dashed in copper; almost every box yields between one and six loops. Right: L_n against n on a logarithmic axis, with the asymptote $\frac{1}{2}\ln n + \ln 2 + \gamma/2$; the two are indistinguishable beyond the first few values.

The logarithmic crawl

Sums of odd reciprocals reduce to harmonic numbers. Since the even terms $1/2 + 1/4 + \cdots + 1/(2n)$ are $\frac{1}{2}H_n$, removing them from H_{2n} leaves the odd ones:

$$L_n = \sum_{k=1}^{n} \frac{1}{2k-1} = H_{2n} - \tfrac{1}{2}H_n.$$

Substituting $H_m = \ln m + \gamma + O(1/m)$ gives

$$L_n = \ln(2n) + \gamma - \tfrac{1}{2}(\ln n + \gamma) + O(1/n) = \tfrac{1}{2}\ln n + \ln 2 + \tfrac{\gamma}{2} + O(1/n),$$

with constant $\ln 2 + \gamma/2 = 0.693147 + 0.288608 = 0.981755$. The approximation is already good at fifty: it returns 2.937767 against the exact 2.937775, an error in the sixth decimal place.

Half a logarithm is very slow growth, and that is the part worth sitting with. Doubling the number of strings adds $\frac{1}{2}\ln 2 \approx 0.347$ to the expected count, so each extra loop costs roughly a sevenfold increase in the yarn. A thousand strings average 4.44 loops, and a box would need about a hundred thousand strings before the expected count reached seven.

n	1	2	5	10	50	1000
L_n	1.0000	1.3333	1.7873	2.1333	2.9378	4.4356

One further remark makes the whole distribution available, not just its mean. The indicators X_k of the previous section are independent, because the conditional probability $1/(2k-1)$ never depended on what the earlier ties did. So the number of loops is a sum of n independent coin flips with success probabilities $1, \frac{1}{3}, \frac{1}{5}, \ldots$, and convolving them gives the exact distribution for fifty strings: 0.1256 for one loop, 0.2814 for two, 0.2896 for three, 0.1844 for four, and 0.0823 for five, with a standard deviation of 1.307. Six loops or fewer covers 99.7 per cent of boxes. The niece is very likely to hand back a small handful, and about one time in eight a single loop of yarn some fifty strings long.

Python code

The recursion is short enough to be suspicious of, so the check ignores it and plays the game. Shuffle the $2n$ ends into a random pairing, walk through the pairs, and use a disjoint-set structure over the strings: if the two ends already belong to the same string a loop has closed, otherwise the two strings merge. Nothing in the simulation knows about $1/(2k-1)$.

```
import numpy as np

L = lambda n: sum(1.0/(2*k-1) for k in range(1, n+1)) # 1 + 1/3 + ... + 1/(2n-1)
G = 0.5772156649015329                          # Euler-Mascheroni
asym = lambda n: 0.5*np.log(n) + np.log(2) + G/2

def simulate(n, trials, rng):  # shuffle 2n ends, tie in pairs, count loops
    total = 0
```

```
    for _ in range(trials):
        parent = list(range(n)) # disjoint set over the strings in the box
        def find(x):
            while parent[x] != x:
                parent[x] = parent[parent[x]]; x = parent[x]
            return x
        # ends 2i and 2i+1 are the two ends of string i
        for a, b in rng.permutation(2*n).reshape(-1, 2):
            ra, rb = find(a//2), find(b//2)
            if ra == rb: total += 1 # both ends of one string: a loop closes
            else: parent[ra] = rb # two strings merge into one longer string
    return total/trials

rng = np.random.default_rng(20260516)
print(f"exact L_50 = {L(50):.6f}")
print(f"sim  L_50 = {simulate(50, 40_000, rng):.6f} (40,000 trials)")
print(f"asymp L_50 = {asym(50):.6f} ln2 + gamma/2 = {np.log(2)+G/2:.6f}")
for n in (1, 2, 5, 10, 50, 100, 1000):
    print(f" n = {n:4d} L_n = {L(n):.6f} asymptotic = {asym(n):.6f}")
# exact L_50 = 2.937775
# sim  L_50 = 2.934200 (40,000 trials)
# asymp L_50 = 2.937767 ln2 + gamma/2 = 0.981755
#  n =   1  L_n = 1.000000 asymptotic = 0.981755
#  n =   2  L_n = 1.333333 asymptotic = 1.328329
#  n =   5  L_n = 1.787302 asymptotic = 1.786474
#  n =  10  L_n = 2.133256 asymptotic = 2.133048
#  n =  50  L_n = 2.937775 asymptotic = 2.937767
#  n = 100  L_n = 3.284342 asymptotic = 3.284340
#  n = 1000 L_n = 4.435633 asymptotic = 4.435633
```

Forty thousand simulated boxes give 2.9342 against the exact 2.9378, a discrepancy of 0.0036 against a standard error of $1.307/\sqrt{40000} = 0.0065$.