Vamshi Jandhyala

Two Subsets, One Sum

✠

March 2026

A pigeonhole puzzle from the Monthly Mindbenders.[1]

Problem

You challenge your bestie to the following game. She chooses 10 distinct integers between 1 and 100; you try to find two disjoint, nonempty subsets of her 10 numbers that have the same sum. With best play, who will win?

Solution

You win, and you win whatever she picks. There is no clever set of ten numbers for her to find, because the count that follows leaves her no room at all.

Step one: more subsets than sums

Write S for her ten numbers. A set of ten elements has $2^{10} = 1024$ subsets, counting the empty set and S itself. Each subset has a sum, and every one of those sums is an integer lying between two bounds. The lower bound is 0, from the empty set. The upper bound comes from the whole set, and here the constraints on her choice bite: ten *distinct* integers, none larger than 100, total at most

$$91 + 92 + \cdots + 100 = 955.$$

So every subset sum is one of the 956 integers $0, 1, \ldots, 955$. We have 1024 subsets to place into 956 boxes, a surplus of 68, so by the pigeonhole principle some box holds at least two of them. That gives distinct subsets $A \neq B$ with

$$\sigma(A) = \sigma(B),$$

writing σ for the sum of a set.

[1] Peter Winkler, *Monthly Mindbenders*, National Museum of Mathematics, March 2026, momath.org/mindbenders, based on a problem from the 1972 International Mathematical Olympiad.

Step two: making them disjoint

Nothing so far stops A and B from overlapping, and in general they do. Throw the overlap away. Put

$$A' = A \setminus (A \cap B), \qquad B' = B \setminus (A \cap B).$$

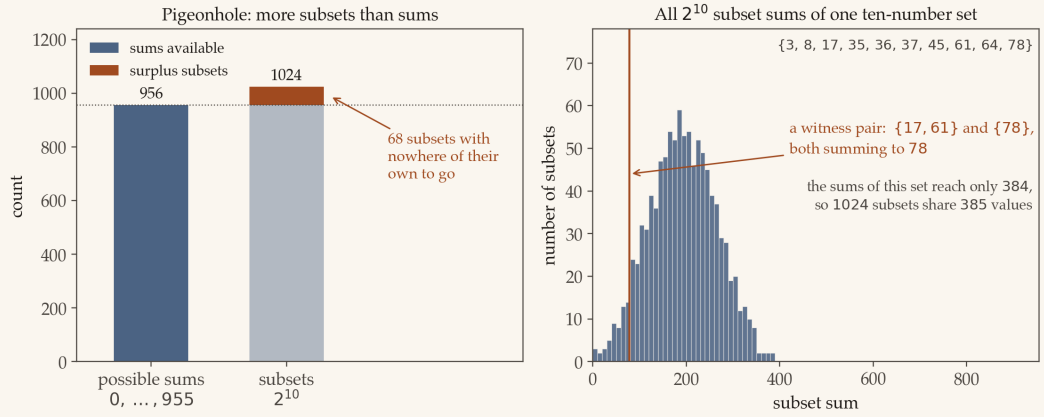
Both sets lose exactly the elements of $A \cap B$, hence both lose the same total $\sigma(A \cap B)$, so

$$\sigma(A') = \sigma(A) - \sigma(A \cap B) = \sigma(B) - \sigma(A \cap B) = \sigma(B').$$

They are disjoint by construction: anything in both A' and B' would lie in $A \cap B$, which was removed from each.

It remains to check that neither is empty, and this is where positivity does its work. Suppose A' were empty. Then $\sigma(B') = \sigma(A') = 0$, and since all ten numbers are at least 1, a subset sums to zero only if it is empty, so B' is empty too. But $A' = B' = \emptyset$ means $A \subseteq B$ and $B \subseteq A$, that is $A = B$, contradicting the fact that the pigeonhole gave us two distinct subsets. So A' and B' are nonempty. They are disjoint, nonempty, and have equal sums, which is exactly what you were asked to produce.

You win, against every choice she can make.



Left: the whole argument in two bars. There are 1024 *subsets and only* 956 *conceivable sums, so at least* 68 *subsets must share a value with another. Right: all* 1024 *subset sums of one randomly drawn ten-number set. They pile up in the middle and, for this set, never exceed* 384*, so the crowding is far worse than the worst case allows for.*

How much slack is there

A great deal, and it is worth seeing where it hides. The bound 955 is attained only by the set $\{91, 92, \ldots, 100\}$, and that set is about the least useful thing she could pick: its ten numbers sit within 9 of each other, so its subset sums cluster savagely by size. Any set that spreads out to keep sums apart pays for it by pushing the total down, which shrinks the range of sums.

The count survives a much harsher version of itself. Restrict attention to subsets of size at most 5. There are

$$\binom{10}{0} + \binom{10}{1} + \cdots + \binom{10}{5} = 1 + 10 + 45 + 120 + 210 + 252 = 638$$

of them, and each has sum at most $96 + 97 + \cdots + 100 = 490$, so at most 491 values are available. Since $638 > 491$ the argument goes through unchanged, and it now delivers a witness in which each of A' and B' has at most five elements. The search in the last section never finds it needing more than five between them.

The version set at the 1972 olympiad is tighter still: ten distinct *two-digit* numbers, so between 10 and 99, where the largest possible total is $90 + 91 + \cdots + 99 = 945$. The MoMath phrasing opens the range down to 1, which costs the argument only ten of its 1024 pigeons' worth of comfort and none of its force.

Why the range matters

The proof rests on nothing about the numbers except that there are ten of them, they are positive, and they are squeezed under 100. Drop the last condition and the conclusion fails at once. Take the powers of two,

$$\{1, 2, 4, 8, 16, 32, 64\},$$

and every one of its 128 subsets has a different sum, since a subset sum here is just a binary numeral and binary numerals are unique. Your bestie would be delighted with such a set. The trouble is that it has seven elements, not ten, and the next three powers of two are 128, 256 and 512, all of them out of bounds.

Counting says the shortfall is unavoidable. Suppose ten distinct positive integers had all 2^{10} subset sums different. Those sums are distinct integers in $\{0, 1, \ldots, \sigma(S)\}$, so $\sigma(S) \geq 1023$. If the largest of the ten is M then, the numbers being distinct, they are at most $M, M-1, \ldots, M-9$, giving

$$\sigma(S) \leq 10M - 45.$$

Hence $10M - 45 \geq 1023$, so $M \geq 106.8$, and $M \geq 107$. Ten numbers with all subset sums distinct cannot live below 107, and the true threshold is higher than this crude bound suggests, since no construction attains it. Capped at 100, she is short by a wide margin, and the surplus of 68 in Step one is precisely the shortfall $1023 - 955$ seen from the other side.

That is the shape of the whole puzzle. Distinct subset sums demand exponential room, ten numbers demand 1024 values of it, and the interval $[1, 100]$ has 956 to give.

Python code

The proof is short enough to be suspicious of, so the code tests it against the game rather than against the argument: draw ten distinct numbers, then search all 3^{10} ways of assigning each number to A', to B', or to neither, in order of size so the first hit is the smallest witness. The last part hunts for a set that resists, first at random and then by hill-climbing.

```
import itertools
import numpy as np

TOP = sum(range(91, 101)) # largest possible total: 91 + ... + 100
print(f"2^10 = {2**10} subsets vs sums 0..{TOP} = {TOP + 1} values: forced")

# every split of the ten numbers into A (+1), B (-1), unused (0), both nonempty,
```

```
# grouped by size so the search stops at the smallest witness
C = np.array(list(itertools.product((0, 1, -1), repeat=10)), dtype=np.int64)
C = C[(C > 0).any(1) & (C < 0).any(1)]
BLOCK = [(s, C[np.abs(C).sum(1) == s]) for s in range(2, 11)]

def witness(x): # smallest disjoint equal-sum pair, or None
    for s, B in BLOCK:
        h = np.flatnonzero(B @ np.asarray(x) == 0)
        if h.size:
            return s, B[h[0]]
    return None, None

rng = np.random.default_rng(2026)
draw = lambda: np.sort(rng.choice(np.arange(1, 101), 10, replace=False))
draws = [draw() for _ in range(20_000)]
sizes = [witness(x)[0] for x in draws]
u, c = np.unique([k for k in sizes if k is not None], return_counts=True)
print(f"witness found in {sum(k is not None for k in sizes) / 200:.1f}%"
      f" of 20000; smallest-witness sizes {dict(zip(u.tolist(), c.tolist()))}")

# adversarial: hill-climb, one swap at a time, to push the witness up
hard = draws[int(np.argmax(sizes))]
best, bestset = 0, None
for x in [hard] + [draw() for _ in range(30)]:
    k, moved = witness(x)[0], True
    while moved:
        moved = False
        for i in range(10):
            for v in np.setdiff1d(np.arange(1, 101), x):
                y = np.sort(np.r_[np.delete(x, i), v])
                if witness(y)[0] > k:
                    x, k, moved = y, witness(y)[0], True
    if k > best:
        best, bestset = k, x
print(f"best adversarial set {bestset.tolist()}: witness uses {best}")

# powers of two keep every subset sum distinct, but they run out below 100
p2 = [1, 2, 4, 8, 16, 32, 64]
sums = np.array(list(itertools.product((0, 1), repeat=7))) @ np.array(p2)
print(f"{p2}: {len(set(sums.tolist()))} distinct sums from {2**7} subsets")
for extra in ([65, 66, 67], [96, 98, 99]):
    x = np.array(sorted(p2 + extra))
    k, w = witness(x)
    A, B = x[w > 0].tolist(), x[w < 0].tolist()
    print(f"{x.tolist()}: {A} and {B} both sum to {sum(A)}, size {k}")
# 2^10 = 1024 subsets vs sums 0..955 = 956 values: forced
# witness found in 100.0% of 20000; smallest-witness sizes {3: 17185, 4: 2814, 5: 1}
# best adversarial set [4, 40, 57, 58, 64, 73, 76, 78, 88, 99]: witness uses 5
# [1, 2, 4, 8, 16, 32, 64]: 128 distinct sums from 128 subsets
# [1, 2, 4, 8, 16, 32, 64, 65, 66, 67]: [2, 65] and [67] both sum to 67, size 3
# [1, 2, 4, 8, 16, 32, 64, 96, 98, 99]: [32, 64] and [96] both sum to 96, size 3
```

A witness was found every time, as it must be. What the numbers add is a sense of how easy the win is in practice: in 17,185 of the 20,000 random sets the smallest witness used three numbers in total, meaning two of her numbers summing to a third. A witness of size two is impossible, since it would need two equal numbers. Only one set in twenty thousand pushed the minimum to five, and hill-climbing from that set could not improve on it.