Vamshi Jandhyala

A Knight on a Lumpy Board

✠

August 2026

An 8×8 board is tiled by the twelve pentominoes together with one 2×2 tetromino, which is exactly $12 \times 5 + 4 = 64$ squares, so the tiling uses each piece once and covers the board. Think of each of the thirteen regions as a single layer of unit cubes. Now add a *tower* to each region: one extra cube, placed on one square of that region. Every square therefore stands at altitude 1, except the thirteen tower squares, which stand at altitude 2.

Place a knight on the bottom-left square, with a score of 0. It makes knight's moves until it has visited all thirteen towers, never landing on the same square twice. A move here travels 0 units along one axis, 1 along another, and 2 along the third, where the three axes are the two board directions and altitude.

The score is the catch. On the knight's Nth move, if it lands at the same altitude it left, the score increases by N. If it moves up, the score is multiplied by N. If it moves down, the score is divided by N, and that move is legal only when the score is exactly divisible by N.

Every three moves, up to move 18, the knight wrote its score on the square it arrived at. After that it wrote its score only every K moves, for some larger K. Those records are the clues in Figure 1. Reconstruct the path, fill in the score on every visited square, then find the squares the knight never visited. For each unvisited square, add up the scores on its orthogonally adjacent visited squares. The answer is the sum of those neighbour sums.

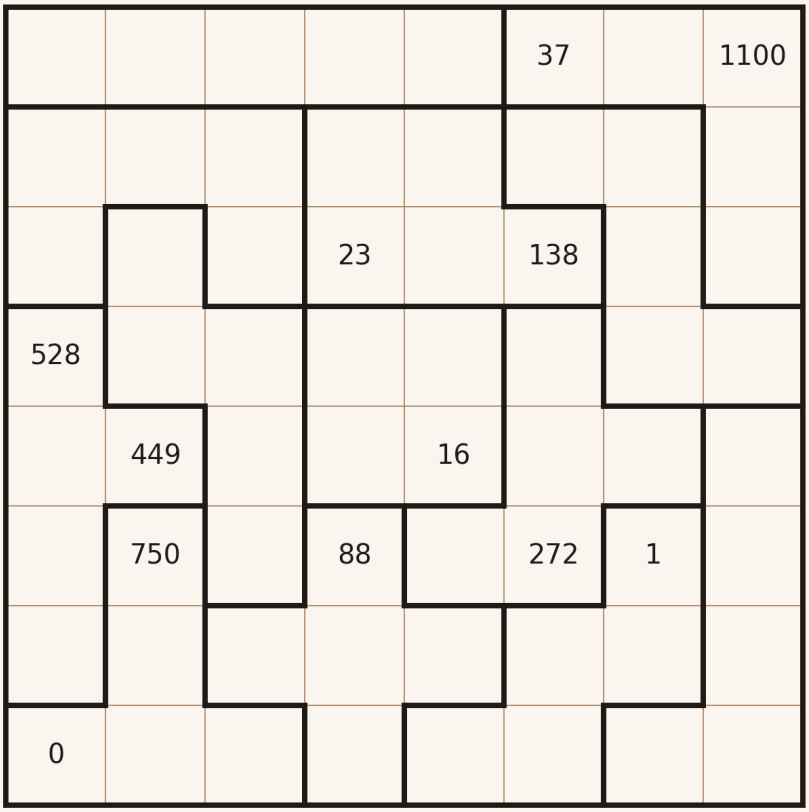

Figure 1. The board. Heavy lines are region boundaries; the twelve numbers are the recorded scores.

Solution

There are only two kinds of move

The whole problem loosens up once you notice how few moves exist.

Write a move as a displacement $(\Delta x, \Delta y, \Delta z)$, where z is altitude. The rule says that $\{|\Delta x|, |\Delta y|, |\Delta z|\} = \{0, 1, 2\}$ as a multiset. Altitudes take only the values 1 and 2, so any two squares differ in altitude by at most 1, which forces $|\Delta z| \in \{0, 1\}$. That leaves precisely two cases.

If $|\Delta z| = 0$, the 0 has been spent on the altitude axis, so the board displacement uses the remaining 1 and 2: the knight makes an ordinary L-shaped knight's move, and its altitude does not change. The score increases by N.

If $|\Delta z| = 1$, the board displacement uses the remaining 0 and 2: the knight moves two squares in a straight line, along a rank or a file, and changes altitude by one. Going from the ground onto a tower multiplies the score by N; stepping off a tower back to the ground divides it by N.

The third case, $|\Delta z| = 2$, would need a board displacement of 0 and 1, a single orthogonal step. It cannot happen, because no two squares differ in altitude by 2. So the knight never takes a single step sideways, an L-move never changes altitude, and a straight two-square move always does. In particular, an L-move between the ground and a tower is illegal, and two ground squares two apart in a line are not connected at

all.

This is worth restating, because it drives everything. Standing on the ground, an L-move keeps you on the ground and adds; a straight two-square move puts you on a tower and multiplies. Standing on a tower, an L-move takes you to another tower and adds; a straight two-square move drops you to the ground and divides.

The knight begins on a tower

The bottom-left square shows 0, so that is where the knight starts, before any move. The first written score is the one after move 3, and it must be one of the twelve numbers on the board.

Start from 0. Multiplication is useless at first, since $0 \times N = 0$, and division of 0 leaves 0. The only way to make the score positive is an addition. So after three moves the score is small. Enumerating every legal sequence of three moves gives the reachable scores exactly:

$$\text{from the ground: } \{0,3,5,6,9\}, \qquad \text{from a tower: } \{0,1,3,5,6\}.$$

Either way the score after move 3 is at most 9. Every clue on the board exceeds 9 except one, the 1. So the move-3 record is the 1, and the knight is standing on that square after three moves.

Now look at the two sets. Starting on the ground, 1 is not reachable. Starting on a tower it is, by exactly one route:

$$0 \xrightarrow{+1} 1 \xrightarrow{+2} 3 \xrightarrow{\div 3} 1.$$

The division at move 3 needs the knight to be on a tower at move 2, and the two additions before it are L-moves that preserve altitude, so the knight was on a tower at moves 1 and 0 as well. The bottom-left square carries its region's tower, and the knight starts on top of it. Nothing else fits.

That single observation is the door into the problem. It also fixes three of the thirteen towers before any search begins, since the two additions were L-moves across the tops of towers.

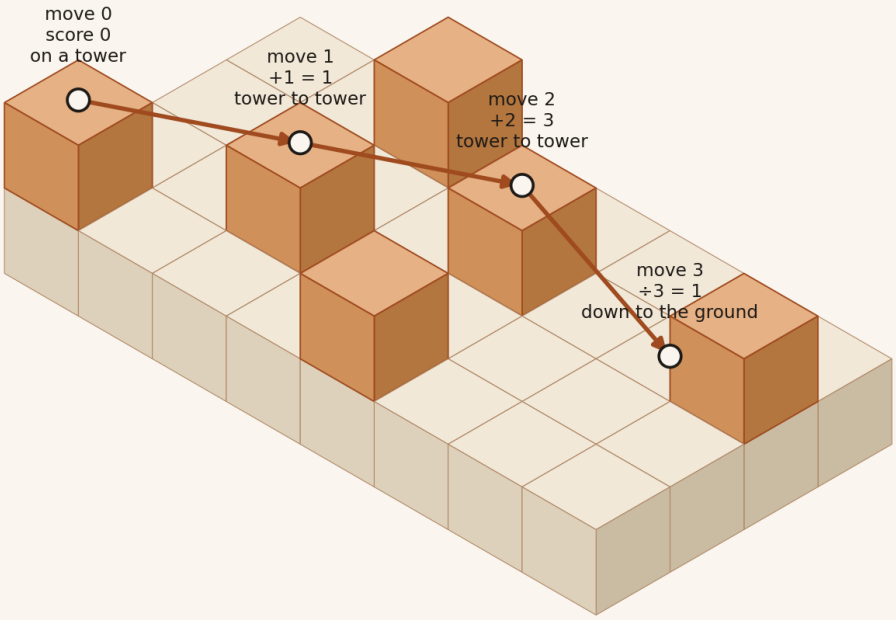

Figure 2. The forced opening, drawn in relief. Every square is one cube high; the thirteen towers are one cube higher. The knight hops across three tower tops, adding 1 *and then* 2*, and drops to the ground on move* 3*, dividing by* 3 *and landing on the clue that reads* 1*.*

Reading the clue list

There are twelve numbers. One of them, the 0, marks the start at move 0. The knight wrote a score every three moves up to move 18, which is six records, at moves $3, 6, 9, 12, 15, 18$. The remaining five numbers are the later records, at moves $18 + K, 18 + 2K, \ldots, 18 + 5K$. So the last record falls at move $18 + 5K$.

Each clue does double duty. It tells us the score, and, because it is written on a particular square, it tells us where the knight was at that moment. Equally useful is the negative information: a square with no number was never a recording square, so at every move that is not a recording move the knight must be standing on a blank square.

K is unknown, so try each value in turn. Only $K = 7$ admits any path at all, putting the records at moves $3, 6, 9, 12, 15, 18, 25, 32, 39, 46, 53$.

Why a solver, and which kind

A score that adds, multiplies and divides is a poor fit for the usual integer-programming machinery, which wants linear relations. Here the score is a running product and quotient of move indices, and it does not stay small: the path below reaches 59400. Modelling that with multiplication constraints is possible and miserable.

Depth-first search with the clues as checkpoints is far better suited, precisely because the clues are so brutal. At move 3 the knight must be on a specific square with a specific score. Three moves later, likewise. A partial path that misses a checkpoint dies immediately, so the tree is pruned to almost nothing.

Running that search against the clues alone yields 17,062 paths. The clues, by themselves, do not determine the answer.

One tower to a region

The constraint that has not been used yet is the tiling, and it is the one that finishes the job.

The towers are not marked on the board. They are decided by the path itself: a square is a tower exactly when the knight is standing on it at altitude 2, which is to say the path's altitude bookkeeping tells us which squares are raised. The tiling then demands that these squares are distributed one to a region, thirteen in all, and that the knight visits every one of them.

Imposing that, at most one raised square per region and all thirteen regions covered, cuts 17,062 down to exactly one path. The clues alone were not enough; the clues plus the tiling are.

One detail falls out of the search rather than into it. Along moves 0 to 53 the knight stands on only twelve towers. The thirteenth region, the one in the top-right corner, is completed at move 54, one move after the final recorded score, when the knight steps up onto $(2, 7)$ and its score becomes $1100 \times 54 = 59400$. The knight stops there, all thirteen towers visited. The path is 55 squares long and leaves 9 squares untouched.

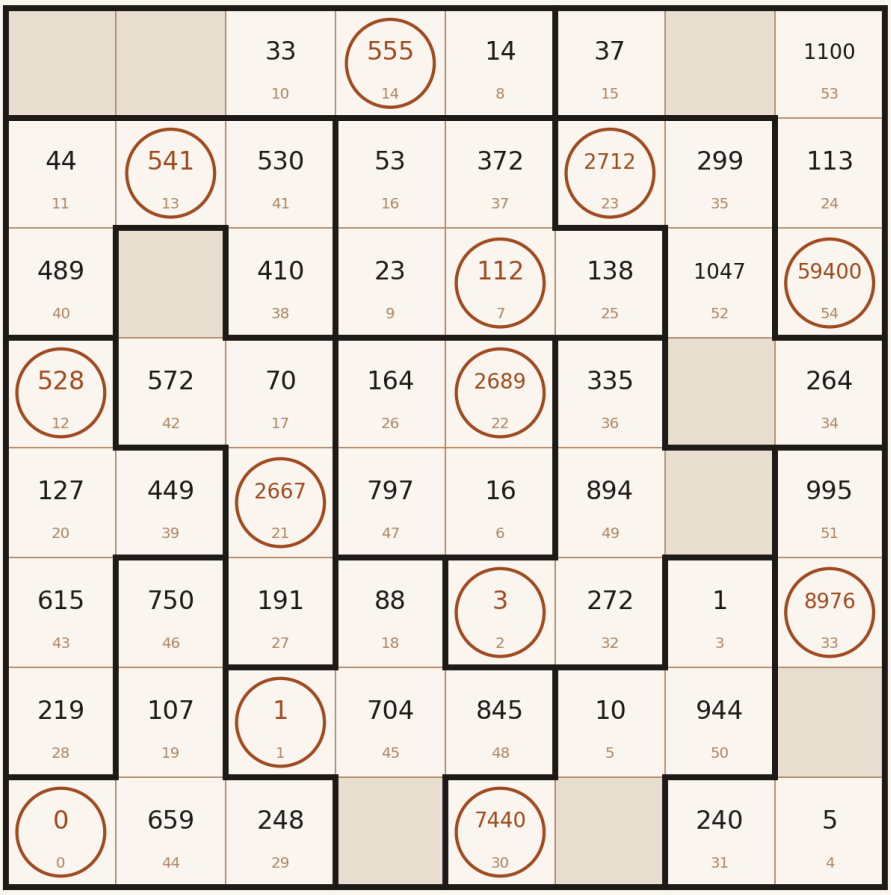

Figure 3. The reconstructed path. Each square carries its score, with the move number beneath in small type. Circled squares are the thirteen towers. Shaded squares were never visited.

The same path in relief makes the arithmetic legible. The knight spends most of its time on the ground, where the score merely accumulates, and every excursion onto a tower is a multiplication that a later step down undoes.

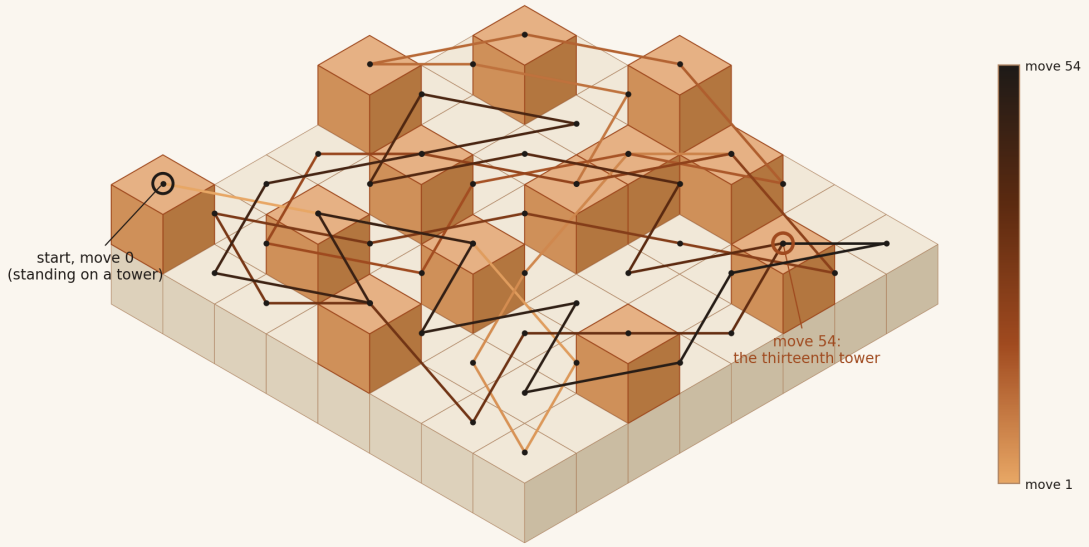

Figure 4. The solution as a landscape. The thirteen towers stand one cube proud of the board. The path runs from pale to dark with the move number.

The answer

Nine squares go unvisited. For each one, add the scores of its orthogonally adjacent visited neighbours:

$$44 + 574 + 1436 + 2012 + 1646 + 1890 + 9925 + 8392 + 7690$$
$$= \mathbf{33609}.$$

Computational Verification

The search below encodes the two move types, the score arithmetic, the clue checkpoints, and the one-tower-to-a-region rule. It returns a single path.

```
N = 8
REGIONS = ["AAAAABBB", "CCCDDEEB", "CFCDDDEB", "GFFHHIEE",
           "GGFHHIIJ", "GKFLIIMJ", "GKLLLMMJ", "KKKLMMJJ"]
CLUES = {(0,5):37, (0,7):1100, (2,3):23, (2,5):138, (3,0):528,
         (4,1):449, (4,4):16, (5,1):750, (5,3):88, (5,5):272, (5,6):1}
START = (7, 0)
L  = [(-1,-2),(-1,2),(1,-2),(1,2),(-2,-1),(-2,1),(2,-1),(2,1)] # same altitude
S2 = [(-2,0),(2,0),(0,-2),(0,2)]                               # altitude +-1

def solve(K):
    checks = {3, 6, 9, 12, 15, 18} | {18 + i*K for i in range(1, 6)}
    last, sols = 18 + 5*K, []
    seen, path = {START}, [(START, 0, 2)] # the knight starts on a tower
    towers = {REGIONS[7][0]: START}

    def step(pos, alt, score, move):
        r, c = pos
        nm = move + 1
        for dr, dc in L + S2:
```

```
            nr, nc = r + dr, c + dc
            if not (0 <= nr < N and 0 <= nc < N) or (nr, nc) in seen:
                continue
            if (dr, dc) in L:              # L-move: altitude unchanged
                nalt, ns = alt, score + nm
            elif alt == 1:                 # step up onto a tower
                nalt, ns = 2, score * nm
            else:                          # step down to the ground
                if score % nm:
                    continue
                nalt, ns = 1, score // nm
            yield (nr, nc), nalt, ns, nm

    def walk(pos, alt, score, move):
        if move == last:
            missing = set("".join(REGIONS)) - set(towers)
            if not missing and alt == 2:
                sols.append(list(path))
            elif len(missing) == 1:
                tail(pos, alt, score, move, missing.pop())
            return
        for cell, nalt, ns, nm in step(pos, alt, score, move):
            if nm in checks:
                if CLUES.get(cell) != ns:  # checkpoint must match exactly
                    continue
            elif cell in CLUES:            # blank squares off-checkpoint
                continue
            rg = REGIONS[cell[0]][cell[1]]
            if nalt == 2 and rg in towers: # one tower to a region
                continue
            if nalt == 2:
                towers[rg] = cell
            seen.add(cell); path.append((cell, ns, nalt))
            walk(cell, nalt, ns, nm)
            path.pop(); seen.discard(cell)
            if nalt == 2:
                del towers[rg]

    def tail(pos, alt, score, move, missing):
        if move - last > 10:
            return
        for cell, nalt, ns, nm in step(pos, alt, score, move):
            if cell in CLUES:
                continue
            if nalt == 2:                  # only the last region may rise
                if REGIONS[cell[0]][cell[1]] != missing:
                    continue
                seen.add(cell); path.append((cell, ns, 2))
                sols.append(list(path))    # all thirteen towers visited
                path.pop(); seen.discard(cell)
                continue
            seen.add(cell); path.append((cell, ns, 1))
            tail(cell, 1, ns, nm, missing)
            path.pop(); seen.discard(cell)

    walk(START, 2, 0, 0)
    return sols

for K in range(4, 13):
```

```
    for p in solve(K):
        score_at = {cell: s for cell, s, a in p}
        total = sum(
            score_at[(r+dr, c+dc)]
            for r in range(N) for c in range(N) if (r, c) not in score_at
            for dr, dc in ((-1,0), (1,0), (0,-1), (0,1))
            if (r+dr, c+dc) in score_at
        )
        print(K, len(p), total)
```

The loop prints one line, `7 55 33609`: a single path, 55 squares long, for $K = 7$.

The puzzle is Jane Street's 'Pent-Up' Frustration 3 / Knight Moves 7, published July 2026 and reproduced here in paraphrase. The figures are redrawn. The answer 33609 *was submitted and confirmed correct by Jane Street's list of correct submissions, as of July 2026.*