Towers of Hanoi (Induction & Recursion, Supplementary)

Table of contents

Supplementary material reinforcing 2025-10-20-recursion[1] and python-recursion[2] — a formal induction proof that the Towers of Hanoi puzzle is always solvable, plus a fully validated worked trace. (Sourced from a guest set of slides that used its own "Lecture 4A" numbering — treated here as extra practice material rather than a separate CSSE1001 lecture.)

The puzzle

Given a tower of n discs (each a different size) stacked on one of several poles, move the entire stack to another pole, one disc at a time, never placing a bigger disc on top of a smaller one.

Proving it's always solvable

Let $P(n)$ be the proposition "a stack of n discs can be moved to an arbitrary pole under the rules above."

- **Base case** $P(0)$: vacuously true — there's nothing to move.
- **Inductive step**: assume $P(n-1)$ (a stack of $n-1$ discs can always be moved to an arbitrary pole). Then, for a stack of n discs:

    1. Move the top $n-1$ discs onto the spare pole (possible by the inductive hypothesis).

[1] courses/csse1001/2025-10-20-recursion.pdf
[2] courses/csse1001/python-recursion.pdf

2. Move the remaining (largest) disc onto the target pole directly.
3. Move the $n-1$ discs from the spare pole onto the target pole, on top of the largest disc (again possible by the inductive hypothesis).

  This gives a valid sequence of moves for n discs, so $P(n)$ holds.

- **Conclusion**: by the Principle of Mathematical Induction, $P(n)$ holds for all n.

This is exactly the recursive structure of the `hanoi` function in 2025-10-20-recursion[3]: the inductive hypothesis *is* the recursive call.

Validated example: 4 discs

Moving 4 discs from peg 0 to peg 2 (moves written as `(disc, destination_peg)`):

```
(1,1) (2,2) (1,2) (3,1) (1,0) (2,1) (1,1)
(4,2)
(1,2) (2,0) (1,0) (3,2) (1,1) (2,2) (1,2)
```

That's $2^4 - 1 = 15$ moves, split as 7 moves to clear discs 1–3 onto the spare peg, 1 move for disc 4, then 7 moves to bring discs 1–3 back on top — matching the proof's structure exactly. This trace is exactly what `hanoi(4, 0, 2)` from 2025-10-20-recursion[4] produces:

```
>>> hanoi(4, 0, 2)
[(1, 1), (2, 2), (1, 2), (3, 1), (1, 0), (2, 1), (1, 1),
 (4, 2),
 (1, 2), (2, 0), (1, 0), (3, 2), (1, 1), (2, 2), (1, 2)]
```

Exercise stub

The source slides left the implementation as an exercise:

```
def hanoi(n: int, from_peg: int, target_peg: int) -> list[tuple[int, int]]:
    """Return the sequence of moves to shift n discs from from_peg to
    target_peg, obeying the Towers of Hanoi rules."""
    ...
```

See 2025-10-20-recursion[5] for a complete implementation (using the third, "off", peg as scratch space).

[3] courses/csse1001/2025-10-20-recursion.pdf
[4] courses/csse1001/2025-10-20-recursion.pdf
[5] courses/csse1001/2025-10-20-recursion.pdf

Related: [[2026-07-27-welcome-and-intro-problems|MATH1061's welcome & intro problems]] previews the same puzzle (and the $2^n - 1$ minimum move count) as motivation for MATH1061's own later induction/recursion content — a nice example of the same idea showing up in two different courses.