The Tower of Hanoi : The Eternal Stack

Nysha Massand
March 2025

1.) Introduction

Deep in a temple in Hanoi, Vietnam, 64 golden disks shine on three diamond rods, stacked in a perfect pyramid in ascending order from top to bottom. Legend whispers that monks strive endlessly to transfer this tower to another rod, following ancient rules, when they complete it the world shall end. This tale, invented by French mathematician **Édouard Lucas** in 1883, frames the Tower of Hanoi puzzle, a deceptively simple challenge that portrays mathematical insights concerning recursion, and problem solving. Far from entertainment, it forms the basis of computer science algorithms, inspires games and mirrors real world complexities from virus spreads to simulations. In this essay, I dissect the puzzle's rules , explore looping and alternate solutions, prove its mathematical depth, dive into variants and trace its legacy and reveal why it remains a mathematical wonder for over 140 years.

2.) What is the Tower of Hanoi?

At its heart, the Tower of Hanoi consists of three vertical rods labeled A(source), B (auxiliary), and C (target) with a stack of n disks of decreasing size placed on rod A , the smallest disk at the top. The objective is to relocate the entire stack to rod C, preserving the original order. Three unbreakable rules control every move: 1) only one disk may be shifted at a time, 2) no larger disk can ever lie upon a smaller one, 3) all three rods are available for use. These constraints transform a simple, straightforward task into an intellectual enigma.

Consider a tower of three disks. Visual:

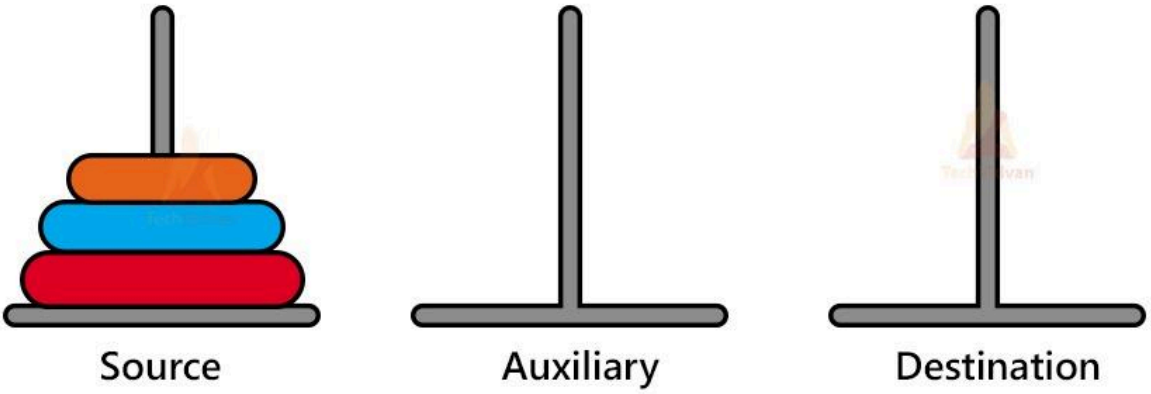

Figure 1: Initial diagram (3 discs) of Tower of Hanoi

Solving it manually reveals escalating difficulty. For one disk: simply move it from A to C (1 move). Two disks demand three precise steps: small from A to B, large from A to C, small from B to C. Three disks leap to seven moves, hinting at an astonishing pattern where each additional disk doubles the prior moves plus one. This isn't a coincidence, it's the recursion.

n (number of disks)	Recursive pattern moves
3	Disk 1 move from A to C Disk 2 move from A to B Disk 1 move from C to B Disk 3 move from A to C Disk 1 move from B to A Disk 2 move from B to C Disk 1 move from A to C Disks Pattern: 1,2,1,3,1,2,1
4	Disk 1 move from A to B Disk 2 move from A to C Disk 1 move from B to C Disk 3 move from A to B Disk 1 move from C to A Disk 2 move from C to B

	Disk 1 move from A to B Disk 4 move from A to C Disk 1 move from B to C Disk 2 move from B to A Disk 1 move from C to A Disk 3 move from B to C Disk 1 move from A to B Disk 2 move from A to C Disk 1 move from B to C Disks Pattern: 1,2,1,3,1,2,1,4,1,2,1,3,1,2,1

3.) Solving the Puzzle - The Recursive Method

The solution consists of recursion, a strategy where the problem solves smaller versions of itself, like a mirror reflecting mirrors infinitely. To shift *n* disks from A to C using B:

3.1 The Recursive Algorithm

1.) Recursively move the top *n-1* disks from A to B (treating C as the temporary help/auxiliary).
2. Shift the largest (nth) disk directly from A to C
3. Recursively move the n-1 disks from B to C (now using A as helper).

This process lies beautifully. For n=3- step 1 handles 2 disks to B using 3 moves, Step 2 takes 1 move, Step 3 mirrors Step 1 (3 moves), coming to a sum of 7 moves. Scaling up, the recursion develops dramatically, each level halving the stack but doubling sub steps plus one.

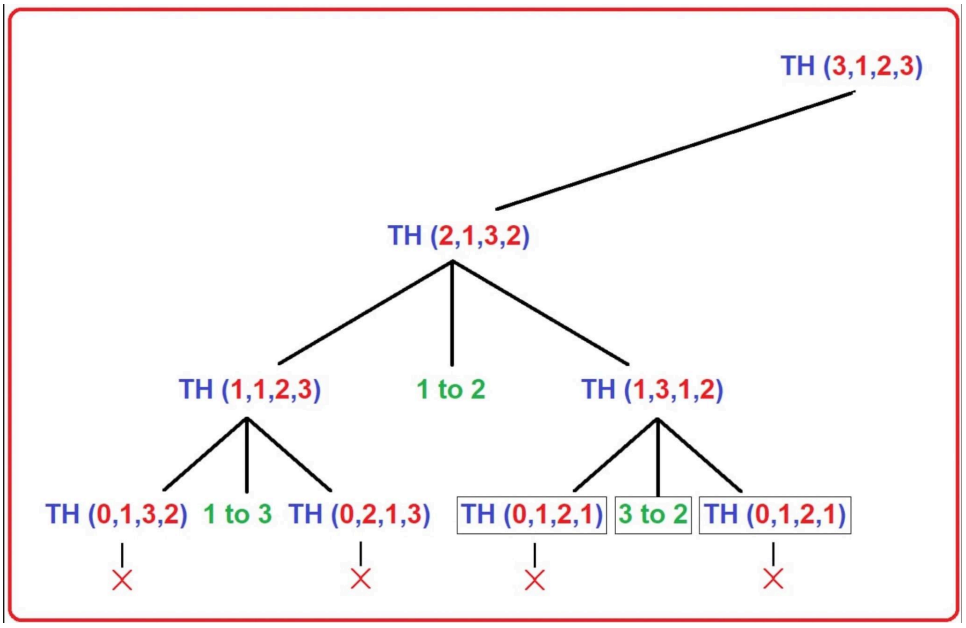

Figure 2: Recursion tree for 3 discs

3.2 Other Methods

3.2.1 Binary Cycle Method:

Number rods A= 0, B= 1, C= 2. For move *k* (1 to 2^n - 1): Disk 1 moves in a circle: A - B - C - A - B - C… (After every 3 moves, back to start)

Then, Move whichever other disk you can rightfully move according to the rules (there's always one exact choice)

Move 1: Disk 1 A - B (1st in A- B - C - A circle)
Move 2: Disk 2 A - C (only legal non disk 1 move)
Move 3: Disk 1 B - C (2nd move in the circle)
Move 4: Disk 3 A - B (only legal move)
…. And so on

3.2.2 Direct Recurrence Unrolling

Recursion can crash computers with call stacks such as n>1000. Simulate these recursion calls with direct stack.
Push main task (n, A, C, B). While stack: grab top task, if n=1, execute move, otherwise push 3 sub tasks in reverse order. Time O (2^n) space O(n) this beats recursion call stack overflow when n=1000+

3.2.3 Frame Stewart Algorithm (4+ pegs)

For k ≥ 4 rods, T(n,k) = min1 ≤ i ≤ n{2T(i,k) + T(n - i,3)}. Thought to give the fewest moves approx $2^{\sqrt{2n}}$ vs classic 2^n . Unproven minimum sparks research simulations show approx 20% faster for n=20.

T (n,p)= minimum moves to solve tower of Hanoi with n disks and p rods
i = the number of the disks from the top you move first

4. Mathematics, Proving T(n) = 2^n - 1

Recurrence: T (1) = 1, T(n) = 2T (n - 1) + 1.

Method 1 - Back Substitution
T (2) = 2T(1) + 1 = 3
T (3) = 2T(2) + 1 = 7
T (4) = 2T(3) + 1 = 15

Pattern: T(n) = 2^{n-1} T(1) + $\sum_{i=0}^{n-2} 2^i = 2^{n-1} - 1 = 2^n - 1.$

Method 2 - Generating Functions: Let G(x) = $\sum_{n=1}^{\infty} T(n)x^n$.

Then $G(x) = x + 2xG(x) + x$, so $G(x) = \frac{x}{1-3x+2x^2}$. Coefficient extraction confirms T(n) = 2^n - 1.

Method 3- Master Theorem:
Rewrite T(n) = 2T(n - 1) + O(1). Dominant 2^n term wins: $\Theta(2^n)$.

Disk Movement: Disk k moves exactly $2^{n-k+1} - 1$ times? NO disk 1: 2^{n-1} moves, disk 2: 2^{n-2} , …… , disk n: 1 move. Total $\sum_{k=1}^{n} 2^{n-k} = 2^n - 1$

.

64 disks: $2^{64} - 1 \approx 1.84 \times 10^{19}$ moves. Disk 1 alone: $2^{63} \approx 9 \times 10^{18}$ moves > atoms in earth! At 1/sec, 585 billion years.

Add figure 3
Figure 3: Log scale T(n) = 2n - 1 vs n = 1-10. Vertical asymptote at n=64

5. Variations

5.1 Reve's Puzzle (4 pegs): FRame-Stewart recurrence above.

5.2 Cyclic Hanoi: Clockwise moves only: $T(n) = 3T(n-1) + 2$
5.3 Largest Disk Once: Fibonacci:
$T(n) = T(n-1) + T(n-2) + 1$
5.4 Sierpiński Connection: Odd numbered moves form Sierpiński triangle (fractal, Hausdorff dimension log(3)/log(2) ≈ 1.585).

6. Key of each variable

O(n)= works same speed as n
O (2^n) = Work explodes very fast
n = number of disks
p = number of pegs
T(n) = total moves required
k = disk index in sigma summation
G (x) = generating function for disk movements
i = move number track

7. Applications

The Tower of Hanoi reveals its true power beyond the game board. While the Sierpiński fractal emerges from disk moves, the same recursive logic solves real problems.

Clinical Psychology: Neurologists use Hanoi to diagnose frontal lobe damage. Patients with planning disorders fail at larger disk counts, revealing cognitive deficits.

Data Storage: Companies rotate backup tapes using Hanoi patter (3 tapes: A to C using B). Ensures full recovery even if 2 tapes fail simultaneously.

Recursive Programming: Teaches the “divide and conquer” strategy. Every major sorting algorithm uses a similar recursive structure.

Robotics Path Planning: Robots move objects between positions without collision exactly like disks between pegs. Used in automated warehouses.

The connection is beautiful: The same rule creating Sierpiński fractals also powers hospital diagnostics, protects corporate data, and guides self-driving robots. The Tower of Hanoi proves simple recursions defeat complexity everywhere.

8 . Conclusion

The Tower of Hanoi isn’t just a puzzle, its maths ultimate con artist.

Picture This: you start with simple disks on pegs, thinking “easy stacking game”. Wrong. Five minutes in, those disks are mocking you, the smallest one sprinting back and forth like an over-caffeinated squirrel while the giant bottom disk sits there like “I’ll move when dinosaurs invent Wi-Fi”. Then the real betrayal hits, tracks the moves and a Sierpiński triangle appears. Your kids toy just drew a fractal that laughs at Euclid. Add 64 disks? Even God needs 584 billion years. Your calculator quits in protest.

But here’s the knockout, the same recursion? It reads brains in hospitals. It shuffles corporate backup tapes so you never lose company secrets. It powers sorting algorithms that run your Netflix recommendations. Robot arms in Amazon warehouses dodge collisions using Hanoi logic.

Three pegs. One rule. Infinite genius.

Math didn't just trick you, it handed you the universe’s source code. Next time someone calls this a “game” laugh. Those disks just outsmarted Einstein while teaching robots table manners. The tower falls slow, but its shadow runs the world. Game Over? NOPE. World domination achieved! Yes.

Refrences

Alsina, Aida. “Towers of Hanoi: A Math and Programming Challenge - Wiris.” *Wiris*, 21 May 2025, www.wiris.com/en/blog/towers-of-hanoi-a-math-and-programming-challenge/.

GeeksforGeeks. “Time Complexity Analysis | Tower of Hanoi (Recursion).” *GeeksforGeeks*, 27 Feb. 2018, www.geeksforgeeks.org/dsa/time-complexity-analysis-tower-hanoi-recursion/.

Kidwell, Eugene. “Tower of Hanoi.” *Maths Careers*, 2 July 2015, www.mathscareers.org.uk/tower-of-hanoi/.

Team, TechVidvan. “Tower of Hanoi in Data Structure - TechVidvan.” *TechVidvan*, 21 Aug. 2021, techvidvan.com/tutorials/tower-of-hanoi/.

Tutorials, Dot Net. “Tower of Hanoi Using Recursion in C.” *Dot Net Tutorials*, 22 Nov. 2021, dotnettutorials.net/lesson/tower-of-hanoi-using-recursion-in-c/.