Complexity of Towers of Hanoi

Ibrahim Nasser

November 2025

1 Problem Definition

The Towers of Hanoi is a puzzle that consists of three pegs (A, B, and C) and a number of disks of different sizes, which can slide onto any peg. The puzzle starts with the disks arranged in a stack in ascending order of size on one peg, with the smallest disk at the top, forming a conical shape.

The objective of the puzzle is to move all disks from peg A to peg B, while obeying the following rules:

1. Only one disk may be moved at a time.
2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty peg.
3. No larger disk may be placed on top of a smaller disk.

The idea of the algorithm (for $N > 1$) is to move the top $N-1$ disks onto the auxiliary peg, then move the bottom disk to the destination peg, and finally move the remaining $N-1$ disks from the auxiliary peg to the destination peg.

2 PROLOG Solution

We can write a PROLOG predicate that prints out a solution for the Towers of Hanoi puzzle. We will use the `write(X)` predicate that prints the value of X (X can be simple text or any type of argument) to the screen and `nl` that prints a new line to write a rule `move(N, A, B, C)` that prints out the solution for moving N disks from peg A to peg B, using C as the auxiliary peg. Each step of the solution is of the form "Move top disk from X to Y".

```
move(1, A, B,_):-
    write("Move top disk from "),
    write(A), write(" to "), write(B), nl.

move(N,A,B,C):-
    N>1,
    N1 is N-1,
    move(N1, A,C,B),
    move(1, A,B,_),
    move(N1, C, B, A).
```

Example $N = 3$:

```
?- move(3, left, center,right).
Move top disk from left to center
Move top disk from left to right
Move top disk from center to right
Move top disk from left to center
Move top disk from right to left
Move top disk from right to center
Move top disk from left to center
true
```

3 Time Complexity

Let $T(N)$ be the number of moves needed to move N disks.

The Base Case:

If there is only one disk, you move it once:

$$T(1) = 1$$

Recursion:

To move N disks from peg A to B using C:

1. Move $N-1$ disks from A to C (*that takes* $T(N-1)$ *moves*)
2. Move the largest disk from A to B (*one move*)
3. Move the $N-1$ disks from C to B (*another* $T(N-1)$ *moves*)

Hence,

$$T(N) = T(N-1) + 1 + T(N-1) = 2T(N-1) + 1$$

Computing $T(N-1)$ the same way gives us

$$T(N-1) = T(N-2) + 1 + T(N-2) = 2T(N-2) + 1$$

Substitute:

$$2T(N-1) + 1 = 2(2T(N-2) + 1) + 1$$

Expand:

$$2(2T(N-2) + 1) + 1 = (4T(N-2) + 2) + 1 = 4T(N-2) + 3$$

Computing $T(N-2)$ the same way gives us

$$T(N-2) = T(N-3) + 1 + T(N-3) = 2T(N-3) + 1$$

substitute:

$$4T(N-2)+3 = 4(2T(N-3)+1)+3$$

Expand:

$$(8T(N-3)+4)+3 = 8T(N-3)+7$$

And so on $\cdots$

Notice that for a recursive step k, we get the following:

- $k=1 \rightarrow T(N) = 2T(N-1)+1$
- $k=2 \rightarrow T(N) = 4T(N-2)+3$
- $k=3 \rightarrow T(N) = 8T(N-3)+7$

This can be generalized to:

$$T(N) = 2^k T(N-k) + (2^k - 1)$$

We stop the recursion when we reach the base case $T(1)$, i.e. when $N-k=1$, i.e. when $k=N-1$.

For $k=N-1$, we have

$$\begin{aligned} T(N) &= 2^{N-1}T(N-(N-1)) + (2^{N-1}-1) \\ &= 2^{N-1}T(1) + (2^{N-1}-1) \\ &= 2^{N-1} + (2^{N-1}-1) \\ &= 2 \cdot 2^{N-1} - 1 \end{aligned}$$

The complexity is $\mathcal{O}(2 \cdot 2^{N-1} - 1) = \mathcal{O}(2^{N-1})$, i.e. **exponential**.