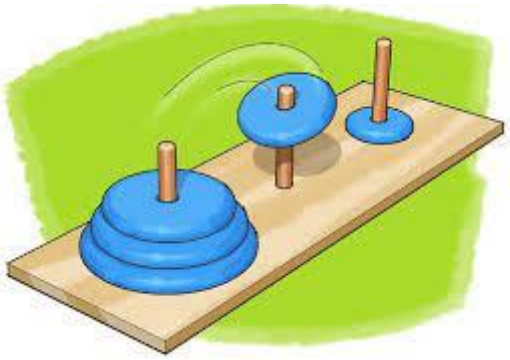
The Tower of Hanoi

By Abhishek Rawat

The tower of Hanoi (also called the tower of Brahma or the Lucas tower) was invented by a French mathematician Édouard Lucas in the 19th century. It is associated with a legend of a Hindu temple where the puzzle was supposedly used to increase the mental discipline of young priests.

If you've gone through the recursion, then you're ready to see another problem where recursing multiple times really helps. It's called the Towers of Hanoi. You are given a set of three pegs and *n* disks, with each disk a different size. Let's name the pegs A, B, and C, and let's number the disks from 1, the smallest disk, to *n*, the largest disk. At the outset, all *n* disks are on peg A, in order of decreasing size from bottom to top, so that disk *n* is on the bottom and disk 1 is on the top.



The objective of the puzzle is to move the entire stack to the last rod, 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 rod.**

3. **No disk may be placed on top of a disk that is smaller than it.**

With 3 disks, the puzzle can be solved in 7 moves. The minimal number of moves required to solve a Tower of Hanoi puzzle is $2^n - 1$, where *n* is the number of disks.

Solution:

Assuming all *n* disks are distributed in valid arrangements among the pegs; assuming there are *m* top disks on a *source* peg, and all the rest of the disks are larger than *m*, so they can be safely ignored; to move *m* disks from a source peg to a *target* peg using a *spare* peg, without violating the rules:

1. **Move *m* − 1 disks from the source to the spare peg, by *the same general solving procedure*. Rules are not violated, by assumption. This leaves the disk *m* as a top disk on the source peg.**
2. **Move the disk *m* from the source to the target peg, which is guaranteed to be a valid move, by the assumptions — *a simple step*.**
3. **Move the *m* − 1 disks that we have just placed on the spare, from the spare to the target peg by *the same general solving procedure*, so they are placed on top of the disk *m* without violating the rules.**
4. **The base case is to move *0* disks (in steps 1 and 3), that is, do nothing – which obviously doesn't violate the rules.**

The full Tower of Hanoi solution then consists of moving *n* disks from the source peg 1 to the 3.

This approach can be given a rigorous mathematical proof with mathematical induction and is often used as an example of recursion when teaching programming.

Algorithm

To write an algorithm for Tower of Hanoi, first we need to learn how to solve this problem with lesser amount of disks, say → 1 or 2. We mark three towers with name, **source**, **destination** and **aux** (only to help moving the disks). If we have only one disk, then it can easily be moved from source to destination peg

The steps to follow are –

Step 1 – Move n-1 disks from **source** to **aux**
Step 2 – Move n^{th} disk from **source** to **dest**
Step 3 – Move n-1 disks from **aux** to **dest**

A recursive algorithm for Tower of Hanoi can be driven as follows –

```
START
Procedure Hanoi(disk, source, dest, aux)

   IF disk == 1, THEN
      move disk from source to dest
   ELSE
      Hanoi(disk - 1, source, aux, dest)     // Step 1
      move disk from source to dest          // Step 2
      Hanoi(disk - 1, aux, dest, source)     // Step 3
   END IF

END Procedure
STOP
```

CODE FOR TOWER OF HANOI IN C++

```cpp
#include <bits/stdc++.h>
using namespace std;

void towerOfHanoi(int n, char from_rod,
                  char to_rod, char aux_rod)
{
    if (n == 0)
    {
        return;
    }
    towerOfHanoi(n - 1, from_rod, aux_rod, to_rod);
    cout << "Move disk " << n << " from rod " << from_rod <<
                                " to rod " << to_rod << endl;
    towerOfHanoi(n - 1, aux_rod, to_rod, from_rod);
}

int main()
{
    int n = 4; // Number of disks
    towerOfHanoi(n, 'A', 'C', 'B'); // A, B and C are names of rods
    return 0;
}
```

Time Complexity Analysis | Tower Of Hanoi (Recursion)

```
Analysis of Recursion

Recursive Equation : T(n) = 2T(n-1) + 1   ——-equation-1

Solving it by Backsubstitution :
T(n-1) = 2T(n-2) + 1   ————equation-2
T(n-2) = 2T(n-3) + 1   ————equation-3

Put the value of T(n-2) in the equation—2 with help of equation-3
T(n-1)= 2( 2T(n-3) + 1 ) + 1   ——equation-4

Put the value of T(n-1) in equation-1 with help of equation-4
T(n)= 2( 2( 2T(n-3) + 1 ) + 1 ) + 1
T(n) = 2^3 T(n-3) + 2^2 + 2^1 + 1

After Generalization :
T(n)= 2^k T(n-k) + 2^{(k-1)} + 2^{(k-2)} + ..... +2^2 + 2^1 + 1

Base condition T(1) =1
n – k = 1
k = n-1
put, k = n-1
T(n) =2^{(n-1)}T(1) + + 2^{(n-2)} + ..... +2^2 +2^1 + 1

It is a GP series, and the sum is 2^n - 1

T(n)= O( 2^n - 1)   , or you can say O(2^n)   which is exponential

for 5 disks i.e. n=5 It will take 2^5-1=31 moves.
```