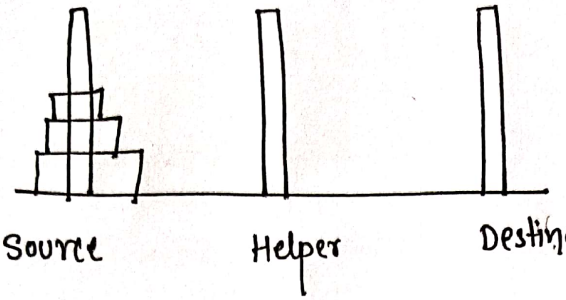
Tower of Hanoi Problem.



Steps 1. The towers of hanoi problem is to find a sequence of disk moves so that all the disks are moved from stand 1 (source) to stand 3 (Destination), adhering to the following rules.

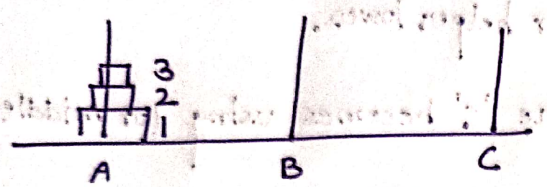
1. Move only one disk at a time.
2. A larger disk cannot be placed on top of a smaller disk.
3. All disks except the one being moved should be on a stand.

Recurrence Relation for solving the Tower of Hanoi problem can be written as :→

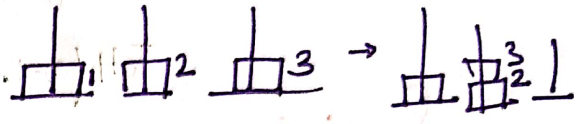
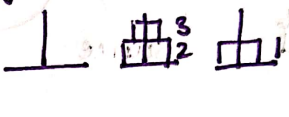
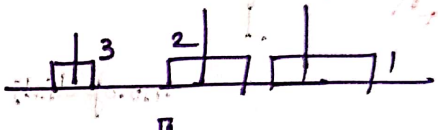
$$\text{TowerofHanoi}(disks) = \begin{cases} \text{move the disk} & \text{if } disks = 1. \\ \text{TowerofHanoi}(disks-1) & \text{if } disks > 1 \end{cases}$$

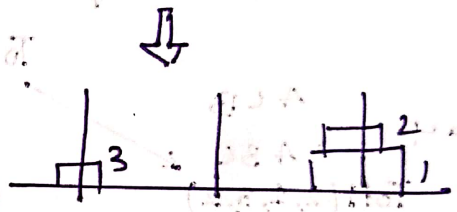
```
#include <      >

void move (int, int, int, int);

void main()
{
  int disks = 3;
  printf("follow these moves:\n");
  move (disks, 1, 3, 2)
}
Void move (int count, int start, int finish, int temp)
{
  if (count > 0)
  {
    move (count-1, start, temp, finish);
    printf ("Move disks %d from %d to %d\n", count, start, finish);
    move (count-1, temp, finish, start);
  }
}
```

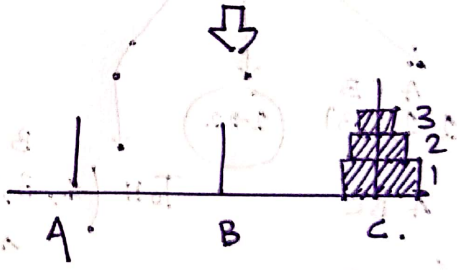
Solution for 3 Disks.



from using to

- Move 2 Disks from A to B using C. 
- Move a Disks from A to C → 
- Move 2 Disks from B to C using A. 





Now if there are n Disks.

- Move n-1 disks from A to B using C
- Move a disks from A to C.
- Move n-1 disks from B to C using A.

Algorithm.

```
void ToH (int n, int A, int B, int C)
{
  if(n>0)
  {
    ToH(n-1, A, C, B);
    printf("Move a disk from %d to %d", A, C);
    ToH(n-1, B, A, C);
  }
}
```

using or helper tower.

source → here 'C' becomes using (or middle tower)

destination

source → here 'A' become using (or middle tower)

destination

no. of disk, no. of towers

ToH (3, 1, 2, 3)

call → A C B, A B C

ToH (2, 1, 3, 2)

1→3

call → B A C, A B C, ~~A B C~~

ToH (2, 2, 1, 3)

ToH (1, 1, 2, 3)

1→2

ToH (1, 3, 1, 2)

A C B

ToH (1, 2, 3, 1)

2→3

B A C

ToH (1, 1, 2, 3)

A B C

1→3

S→D

3→2

S→D

A B C

2→1

S→D

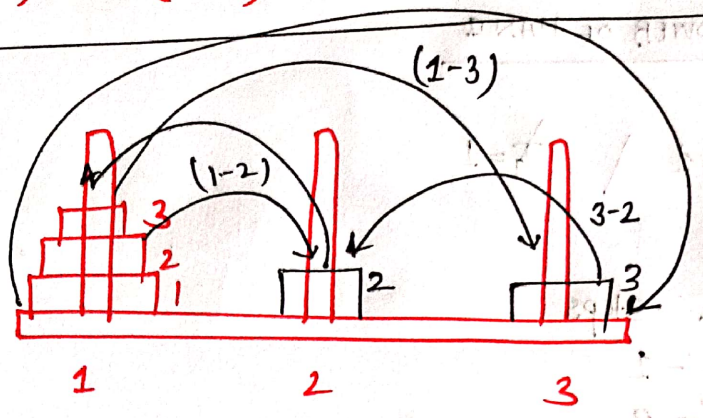
A B C

1→3

1→3 | 1→2 | 3→2 | 1→3 | 2→1 | 2→3 | 1→3

These are moves that we have to take to move disks from Source to destination.

(1-3) → (1-2) → (3-2) → (1-3) → (2-1) → (2-3) → (1-3)



check for this order towers of hanoi.

TIME COMPLEXITY of TOWER OF HANOI

$T(n) = 2T(n-1) + 1$ | $T=1$

$T(n) = 2(2T_{n-2}$

no. of disk.	steps
$n-2$	1 disk – 1
$n-1 =$	2 disk – 3
$n =$	3 disk – 7

So the Recurrence Relation on the basis of Recurrence Relation

$$\boxed{T(n) = 2T_{n-1} + 1}$$

Recurrence Relation

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

$$T(n) = 2[2T(n-2)+1] + 1$$

$$= 2^2 T(n-2) + 3$$

$$= 2^2[2T(n-3)+1] + 3$$

$$= 2^3 T(n-3) + 7$$

or K number of disk

$$\boxed{T(n) = 2^K T(n-K) + 2^K - 1}$$

Calculate :→ total number of steps for N disks.

$$T(n) = 2^K T(n-K) + 2^K - 1$$

if $K \to n$

$$= 2^n T(n-n) + 2^n - 1$$

$$\Rightarrow \boxed{T(n) = 2^n T(0) + 2^n - 1}$$

Another Method to calculate

ANALYSIS OF RECURSION.

$$T(n) = 2T(n-1) + 1 \rightarrow (1)$$

Solving it using Back-Substitution:

$$T(n-1) = 2T(n-2) + 1 \rightarrow (2)$$

$$T(n-2) = 2T(n-3) + 1 \rightarrow (3)$$

Put the value of $T(n-2)$ in the equation 2 with the help of equation 3.

$$T(n-1) = 2\left[2T(n-3) + 1\right] + 1 \rightarrow (4)$$

Now put the value $T(n-1)$ in equation 1 with the help of equation-4.

$$T(n) = 2\left[2\left[2T(n-3) + 1\right] + 1\right] + 1$$

$$= \left[4.\left[2T(n-3) + 1\right] + 2 + 1\right.$$

$$\Rightarrow \left[4.2T(n-3) + 4 + 2 + 1\right]$$

$$= 2^3.T(n-3) + 2^2 + 2^1 + 2^0$$

So if there is K disks

$$= 2^K.T(n-K) + 2^{K-1} + 2^{K-2} + \cdots + 2^2 + 2^1 + 1$$

Base Condition.

$T(1) = 1$

$n - k = 1$

$k = n - 1$

$$T(n) = 2^{n-1}T(1) + \cdots + 2^{(n-2)} + \cdots + 2^2 + 2^1 + 1$$

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

$T(n) = O(2^n - 1)$, or its Time Complexity $= O(2^n)$

which is exponential.

THE SPACE COMPLEXITY OF THE TOWER OF HANOI

$O(N)$ → where n is number of disks.

This is because for each recursion, the disks take up N-1 recursive stack space.