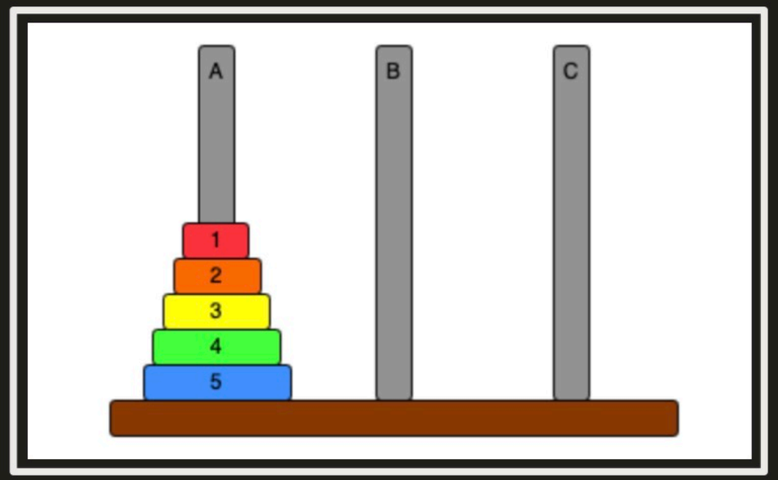
Tower of Hanoi

Tuesday, 16 January 2024 11:07 AM

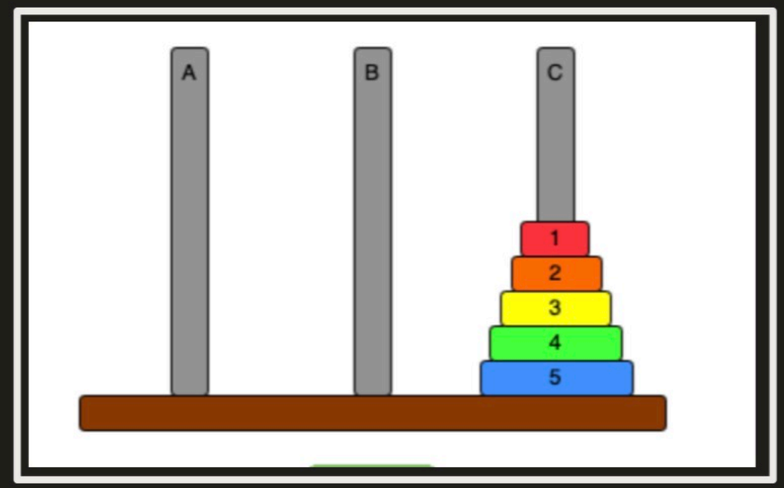
Input



toh(N, A, C, B)

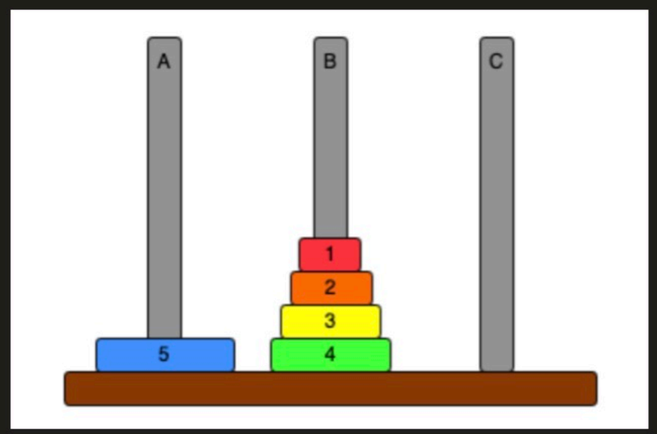
expectation from the func. implementation

Output.



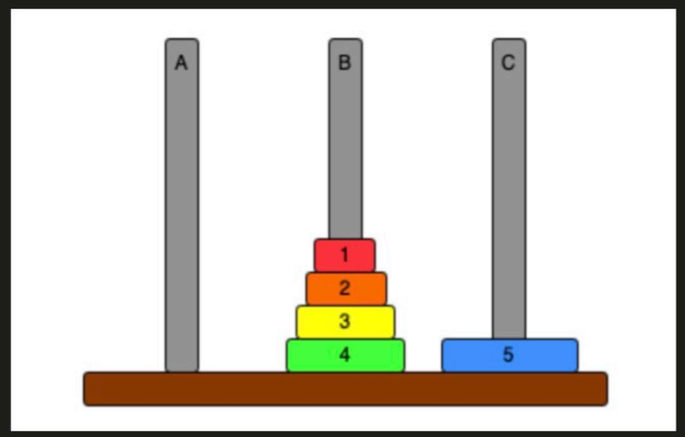
toh (N-1, A, B, C)

faith that the recursive call for a smaller input will do its job.



N : A → C

move the last block from A to C



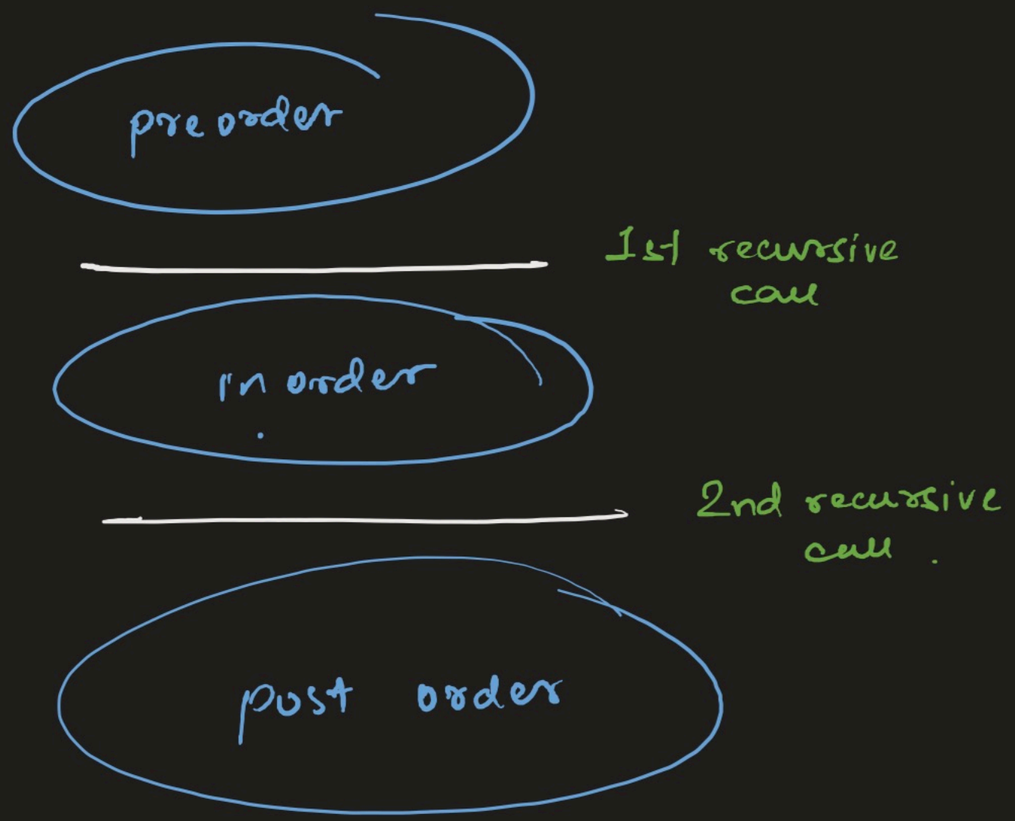
faith step.

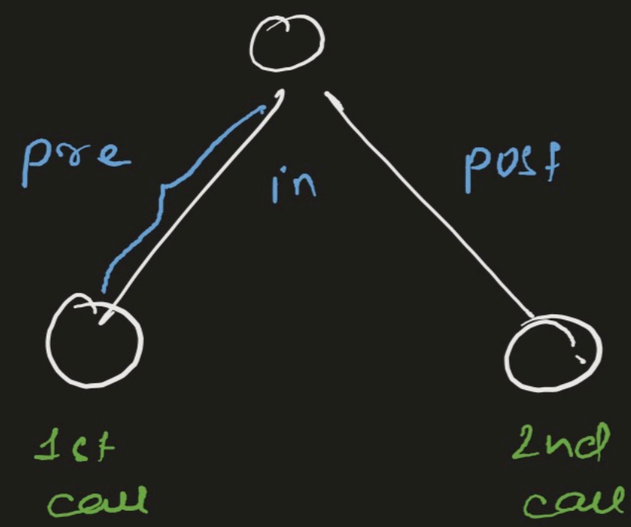
toh (N-1, B, C, A).

from, to, aux.

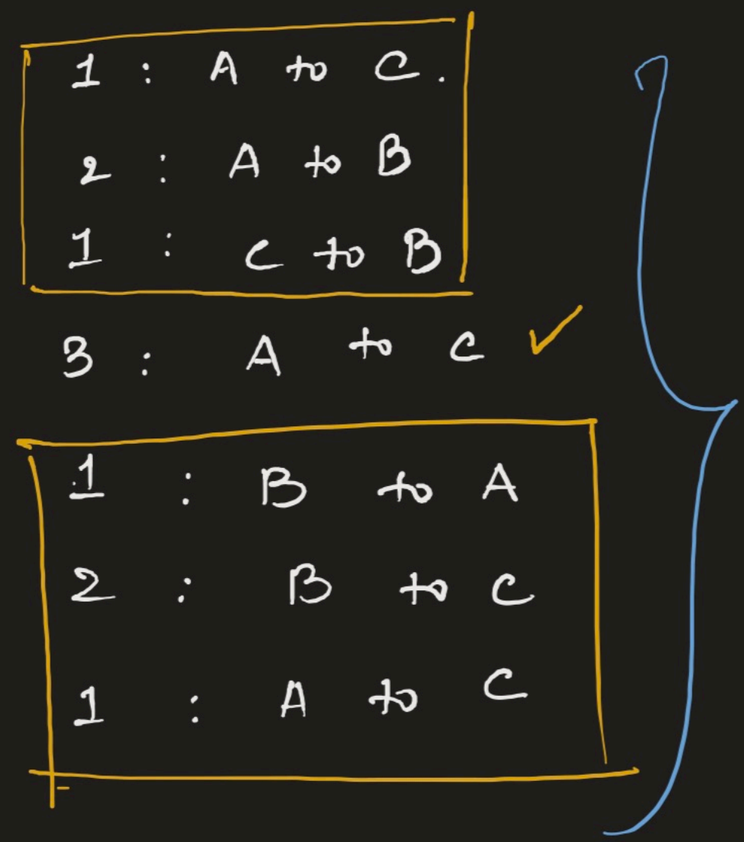
toh(n, A, C, B) :

{
- toh(n-1, A, B, C) ✓
- move last block from A → C
- toh(n-1, B, C, 1)
}



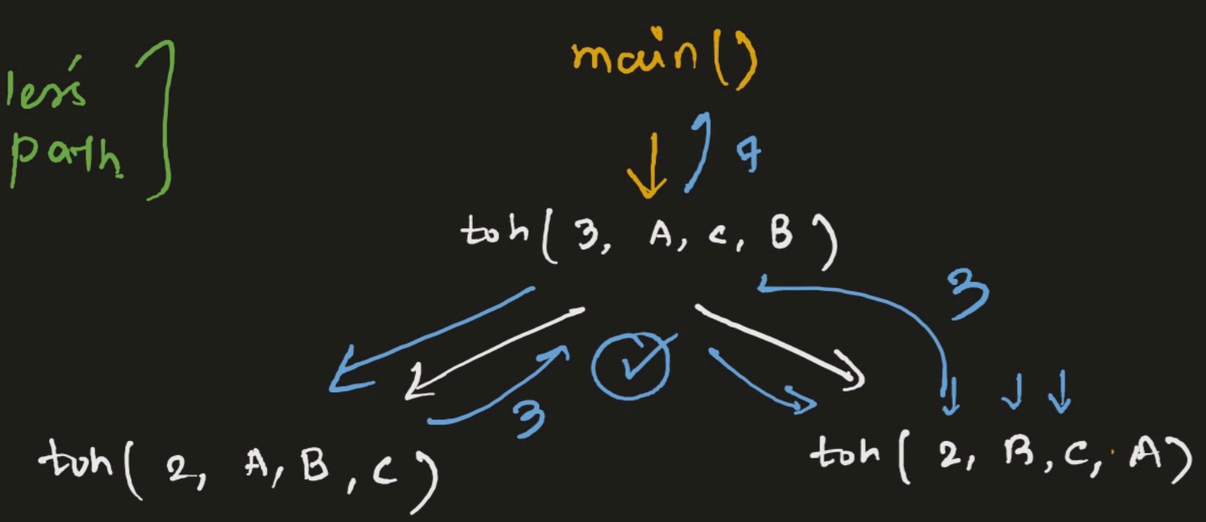


```cpp
long long toh(int n, int A, int C, int B) {

    // preorder work
    if ( n == 0 )   return 0; // base case

    long long ans1 = toh( n-1, A, B, C ); //faith

    //inorder work
    cout << "move disk " << n << " from rod " << A << " to rod " << C << endl; // do yourself

    long long ans2 = toh( n-1, B, C, A ); //faith

    // post order work
    long long ans = ans1 + 1 + ans2;
    return ans;

}
```



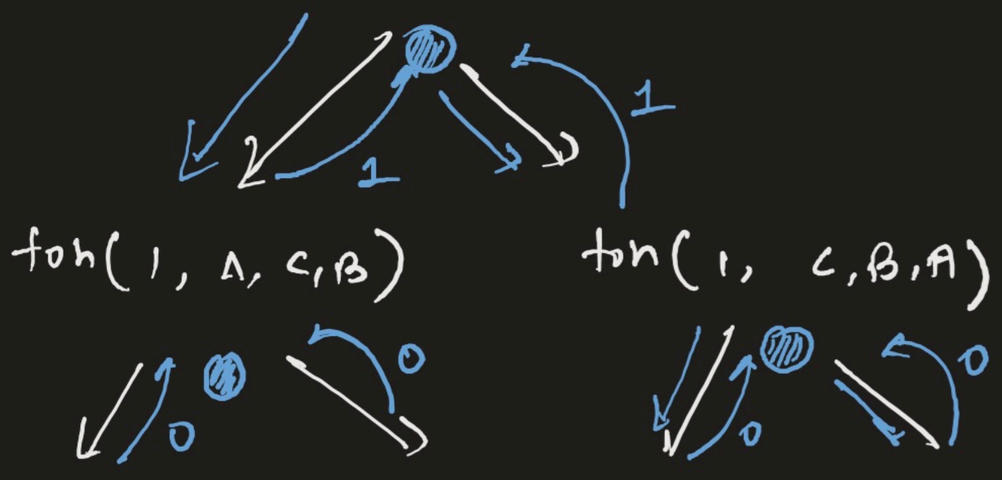
Dry Run

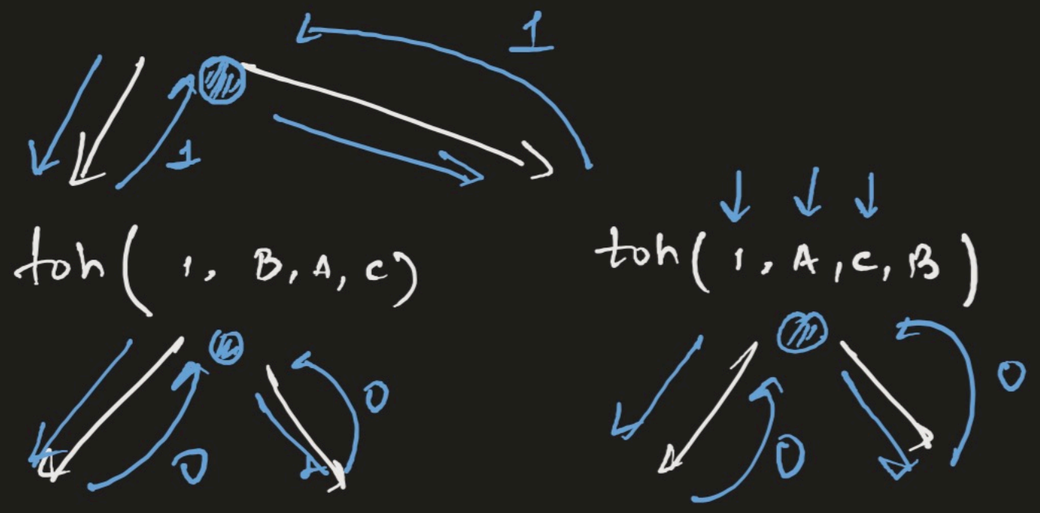
[Euler's path]



Logic

- Swap last 2
- reverse all 3.





Time complexity : $\longrightarrow O(2^n)$.

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

$$\Rightarrow T(n) = 2T(n-1) + 1.$$

$$\Rightarrow T(n) \simeq 2T(n-1) \Rightarrow 2^n \cdot c$$

$$\Rightarrow T(n-1) = 2T(n-2)$$

$$\Rightarrow T(n-2) = 2T(n-3)$$

$$\Rightarrow T(3) = 2T(2) = 2(2^2 c) = 2^3 c.$$

$\therefore T(3) = 2T(2) = 2(2^2 c) = 2^3 c.$

$\therefore T(2) = 2T(1) = 2 \cdot 2c = 4c = 2^2 \cdot c$

$\therefore T(1) = 2T(0) = 2^1 c$

$\therefore T(0) = c.$