4. RECURSION

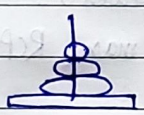
→ Disadvantages: • Need more overhead space (extra space)
• Recursion is function call overhead [many function calls are there in stack]

→ Tail Recursive: A function is tail recursive when last thing to happen is function call → This is optimised by modern compiler (Take less time & Auxilary space)

n^{th} number in a fibonacci Series

> fibonacci Series: 1, 1, 2, 3, 5, 8, 13 . . . .
> ↳ any number is actually sum of last 2 number in Series

```
int fib(int n){
    if(n<=1)
        return 1;
    return fib(n-1) + fib(n-2);
}
```

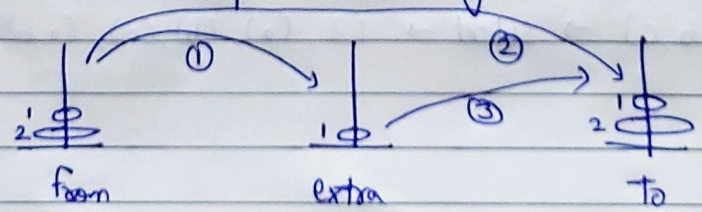
Tower of hanoi  → this game

to solve this problem,
we can observe pattern as follow:

```
toh(n, from, to, extra){
    toh(n-1, from, extra, to);
    cout << n << from << to;
    toh(n-1, extra, to, from);
}
```

if(n==0) return 0;

to understand this problem, always take case n=2

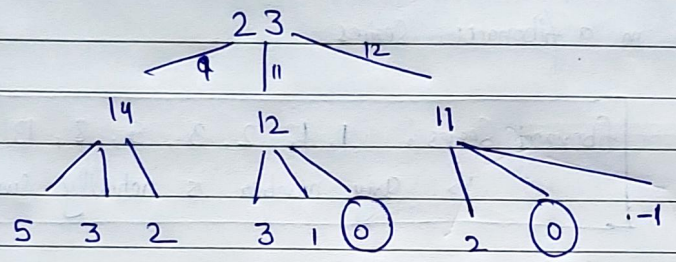


★★★ Total no. of movement = $2^n - 1$

Priyansh

Rope Cutting problem: given a rope of length (n), and possible cutting length (a), (b) and (c). find possible no of ways to cut the given rope completely

eg $n = 23$ $a = 9$ $b = 11$ $c = 12$



$\Rightarrow$ ans = 2

```
int RCP ( int n, int a, int b, int c ) {
    if ( n<0)
        return -1;
    if ( n==0)
        return 0;
    int ans = max( RCP(n-a,a,b,c), max(RCP(n-b,a,b,c),
                                       RCP(n-c,a,b,c));
    if ( ans == -1)
        return -1;
    return ans+1;
}
```

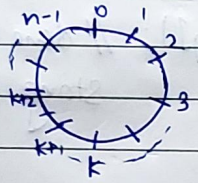
Subset / Subsequence generation

Total no of Subset = 2^n

(a,b,c) → subset → (), (a), (b), (c), (a,b), (b,c), (a,c), (a,b,c)

```
int subset(string str, string cur = "", int i = 0) {
    if (i == str.length()) {
        cout << cur << endl;
        return 0;
    }
    subset(str, cur, i+1);
    subset(str, cur + str[i], i+1);
}
```

Josephus problem → given n people in circle, and a given k, eleminate every k^{th} each round. (Starting index = 0)




```
int Jos(int n, int k) {
    if(n == 1)
        return 0;
    return [ jos(n-1, k) + k ] % n;
}
```

TC : O(n)

→ using Bit magic pattern is like for every diff greater than power of 2 (ans = 2*diff + 1)
like $n = 2^x + l$ then ans = 2*l + 1

OR

TRICK : n → binary take MSB and put it in LSB then ans = new no. like
n = 41 : 101001 then MSB at 6th position so new no.
after shift 010011 = 19 (in decimal) (ans)

To do this, we find MSB pos then make it like (100000) and do XOR then add an extra '0' in end (<<1 use) and then add '1' (11)

Priyansh

Permutation of a String [total possible permutation = n!]

eg abc → { abc, acb, bac, bca, cab, cba }

```
int perm.( string Str, int i=0) {
    if( i == str.length() - 1 ){
        cout << str << endl;
        return 0;
    }
    for (int j = i ; j < str.length() ; j++)
    {
        swap( str[i] , str[j] ) ;
        perm ( str , i+1 ) ;
        swap( str[i], str[j]); // done to get original
                               string back
    }
}
```

Priyansh