A typical C Program:

```
1. #include <stdio.h>
2. main ( ){
3.
4.
5.      A(3);   function with actual argument
6.
7.
8.
9.      }
```

Function with formal argument

```
5a. A(n) {
5b.     if (n>0){
5c.         Printf("%d", n-1);
5d.         A(n-1);
5e.     }
5f. }
```

recursion is just when a function calls itself i.e. caller function and called function is same.

The most common question from recursion is they tell us to trace the recursive function and tell us the order of activation record of function.

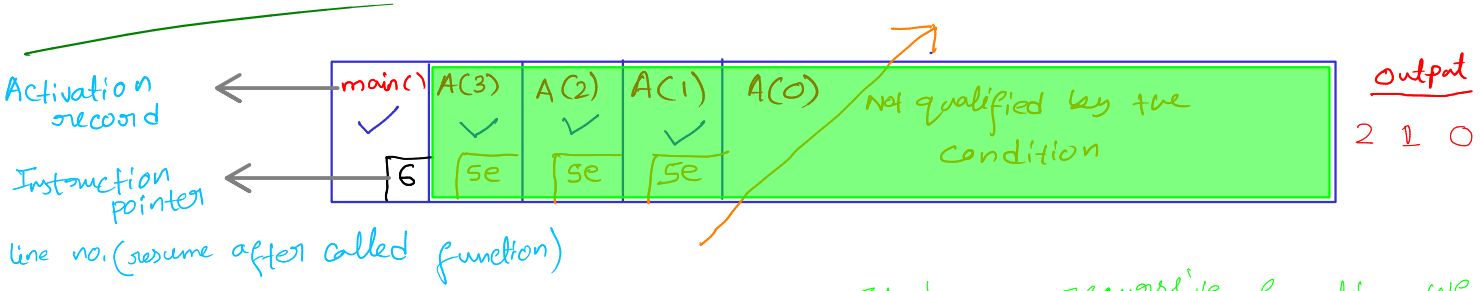
Activation record of function:

when a function is called, it creates a activation record. Normally these records are used to see the order of the function → which function executed after which function → Determining order of function through these records are called tracing. Tracing is shown in two ways → 1. Stack order 2. Tree order

In recursions these records are used as resume option → when a caller function calls another function This activation records saves the line no. which will resume this function after function calling is over. They saves line no. in activation records with instruction pointer.

Tracing

1. with Stack:

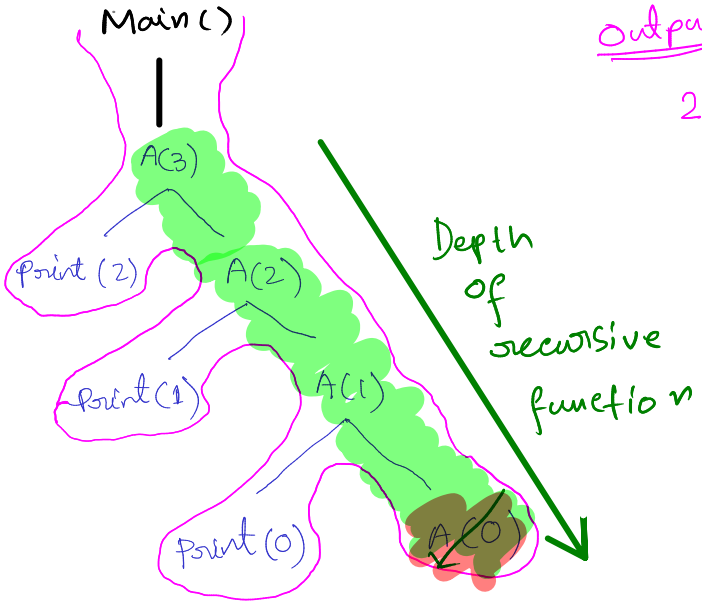


It is a recursive function, we can know that by seeing the same activation record with different local variable and they also have same instruction pointer.

It is also called depth of recursive function (It is normally asked in GATE)

Here Depth =

2) with Tree



output:

2 1 0

Depth = 4

function call = 4

If there is a big recursive program go with STACK otherwise you can use TREE method.

Example - 1

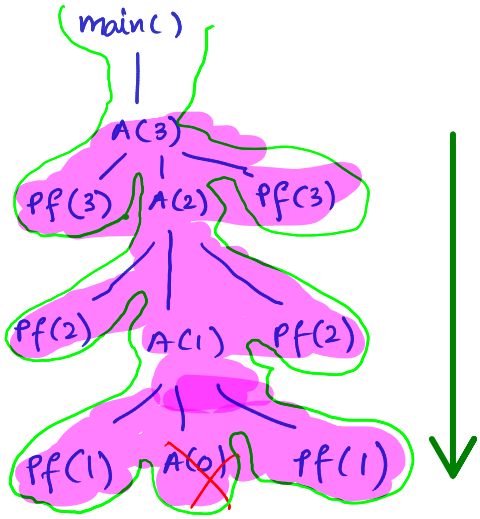
```
1. A(n) {
2.     if(n>0){
3.         Pf(n);
4.         A(n-1);
5.         Pf(n)
6.     }
7. }
```

STACK

output: 3211123

Depth = 4

Tree:



output:
3211123

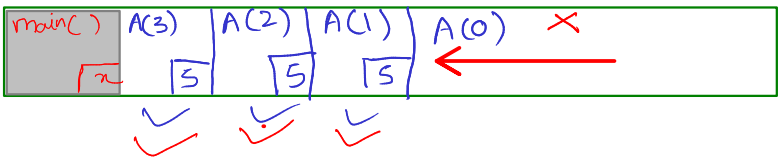
Depth = 4

4 depth
4 function call

Example - 2

```
1. A(n){
2.     if(n>0){
3.         pf(n-1);
4.         A(n-1);
5.         pf(n-1);
6.     }
7. }
```

output: 2 1 0 0 1 2

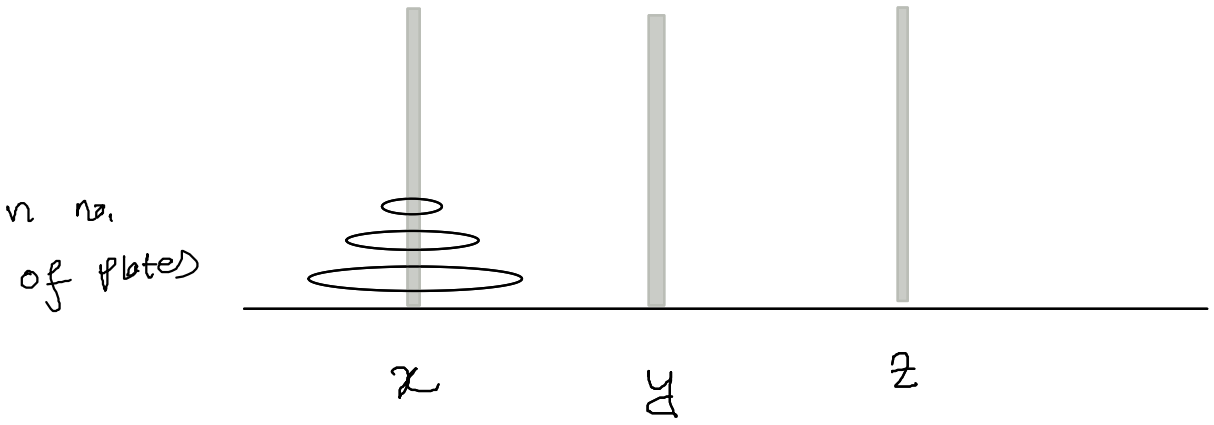


TOWERS OF HANOI

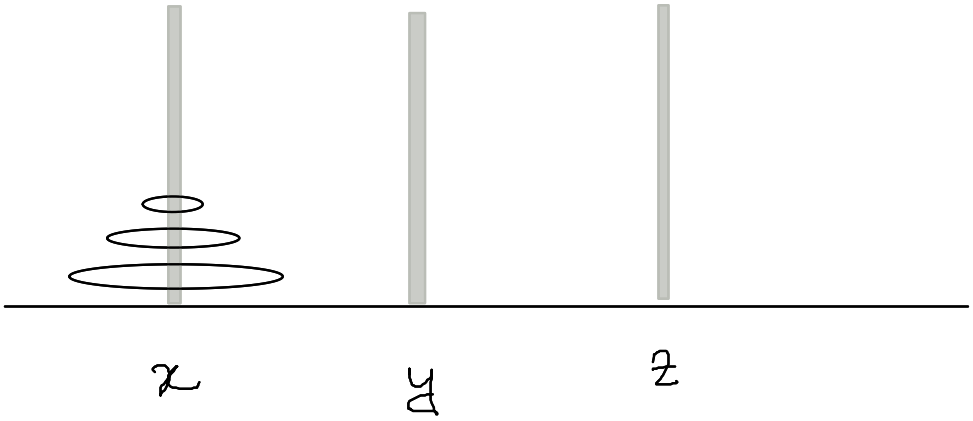
A story from hindu mythological book is that some brahmins is toying to destroy the world. They have 3 dimond towers → x, y & z.

In x tower there are few disks made of gold Now if the brahmins can put all golden disks from x to y then the world will end.

But we have to put the plates one by one and also we can't put heavy plate over small plate otherwise the plate will break and these work will be done with the help of z tower.



But the brahmins are dumb so they couldn't able to do that so we are still alive.



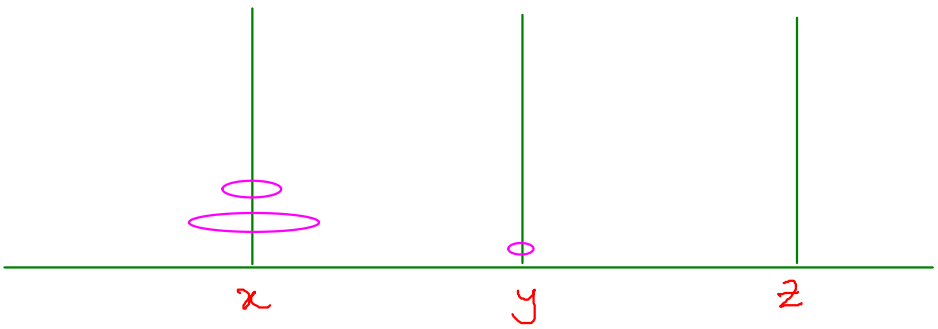
Let's assume, $n = 3$.

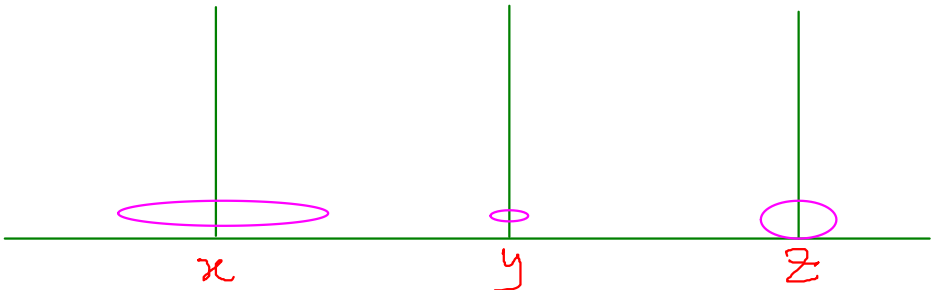
1. First we have to somehow put $n-1 = 3-1 = 2$ plates from x to z with help of y

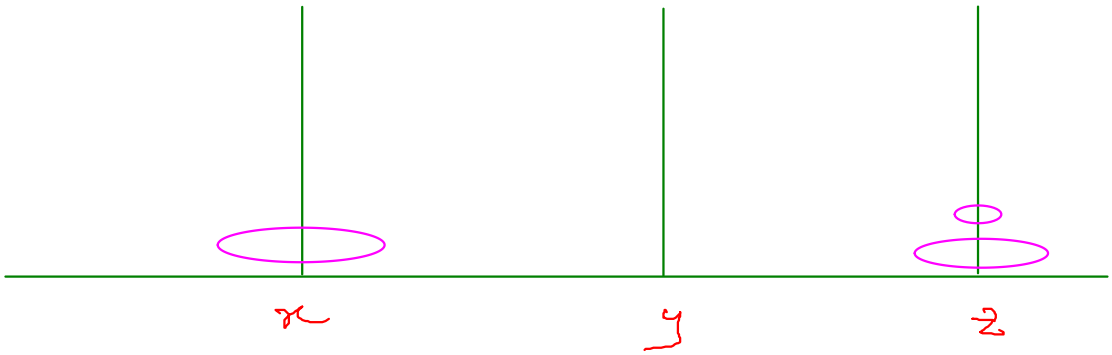
2) Now we have only the heviest plate left in x, so we put that in y

3) Now rest of $(n-1)$ plates that we put in z we will put in y with help of x.

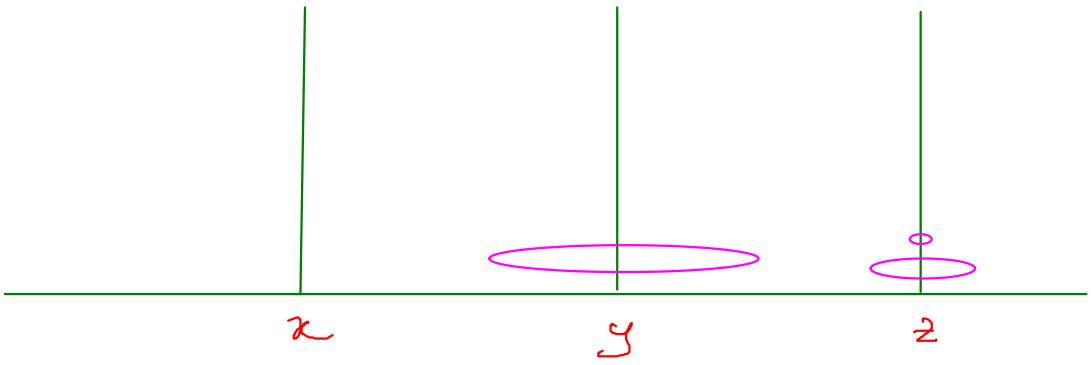
1. TOH $(n-1, x, z, y)$



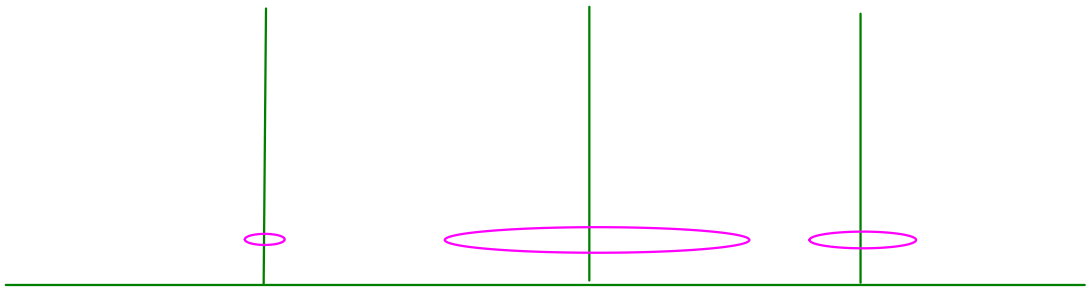


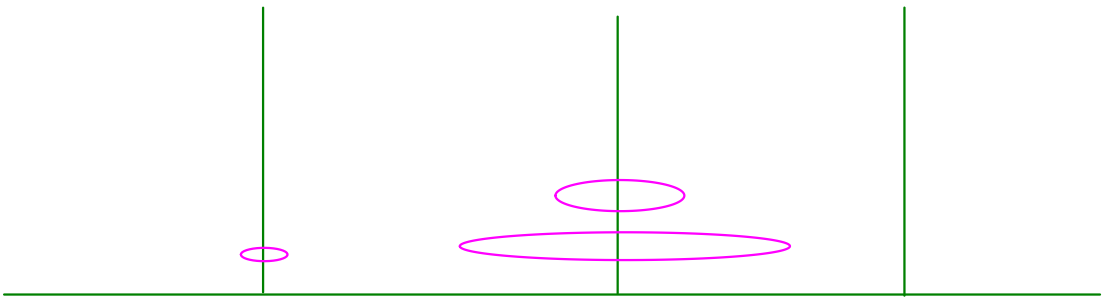


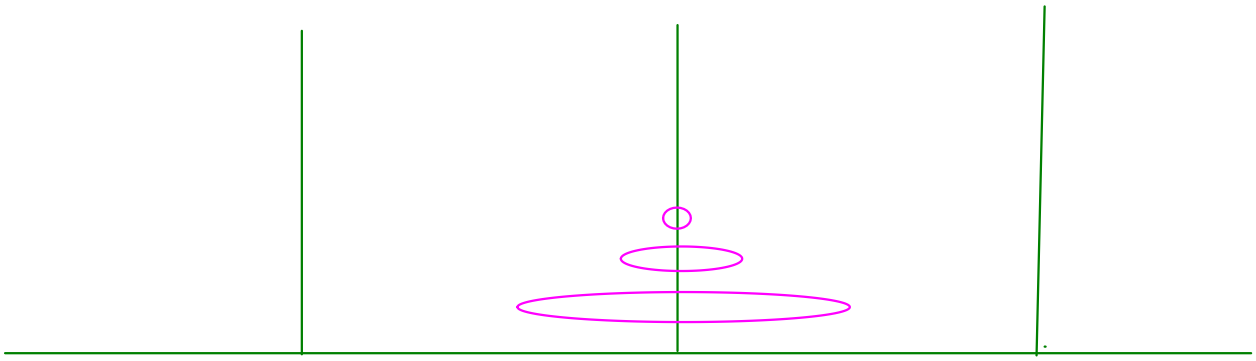
2. move TOP 'x' to 'y'



3. TOH (n-1, z, y, x)



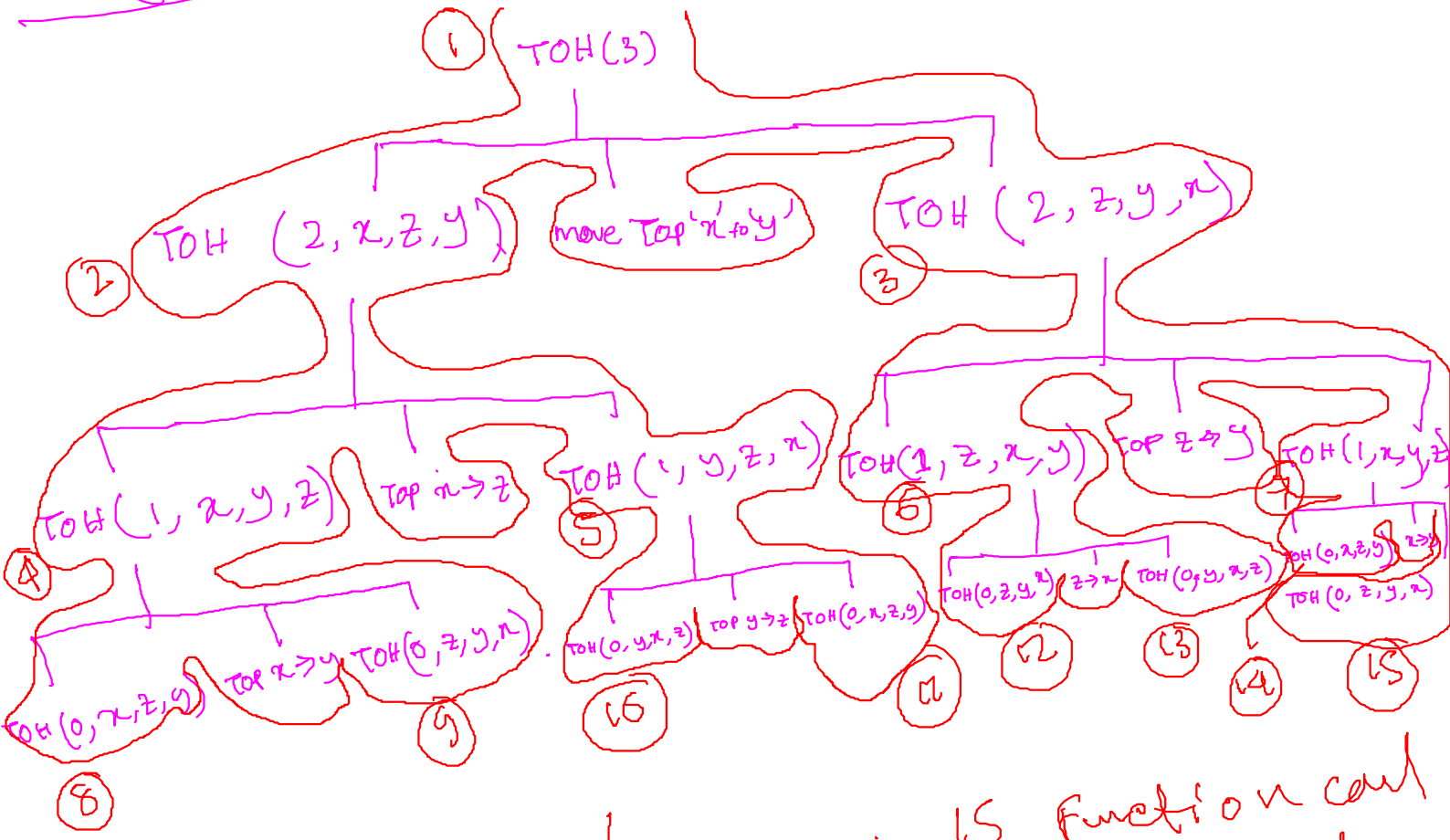




Algo of TOH:

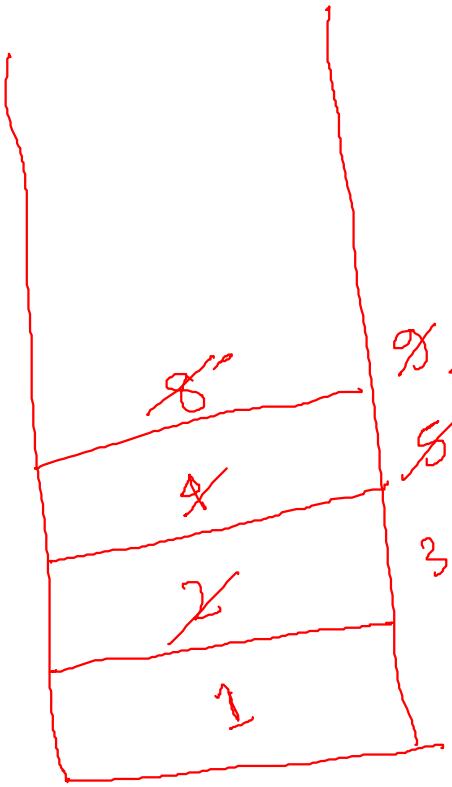
```
TOH (n) {
    if (n >= 1) {
        TOH(n-1, x, z, y);
        move TOP 'x' to 'y';
        TOH(n-1, z, y, x);
    }
}
```

Tracing with Tree : Let's say n = 3



15 function call

7 movements



4 depth