print zig zag. →

Input1 -> 1
Output1 -> 1 1 1

Input2 -> 2
Output2 -> 2 1 1 1 2 1 1 1 2

Input2 -> 3
Output3 -> 3 2 1 1 1 2 1 1 1 2 3 2 1 1 1 2 1 1 1 2 3

Expectation pzz(n) = n (n-1) n (n-1) n

faith → pzz(n-1) = n-1 (n-2) n-1 (n-2) n

Merging :D

pzz(n) = n $\underbrace{(n-1)}_{\text{faith}}$ n $\underbrace{(n-1)}_{\text{faith,}}$ n

pzz(n) = Syso(n) pzz(n-1) Syso(n) pzz(n-1) Syso(n)

pzz(n) {

Done by myself

Syso(n)
pzz(n-1)
Syso(n)
pzz(n-1)
Syso(n)

}

faith

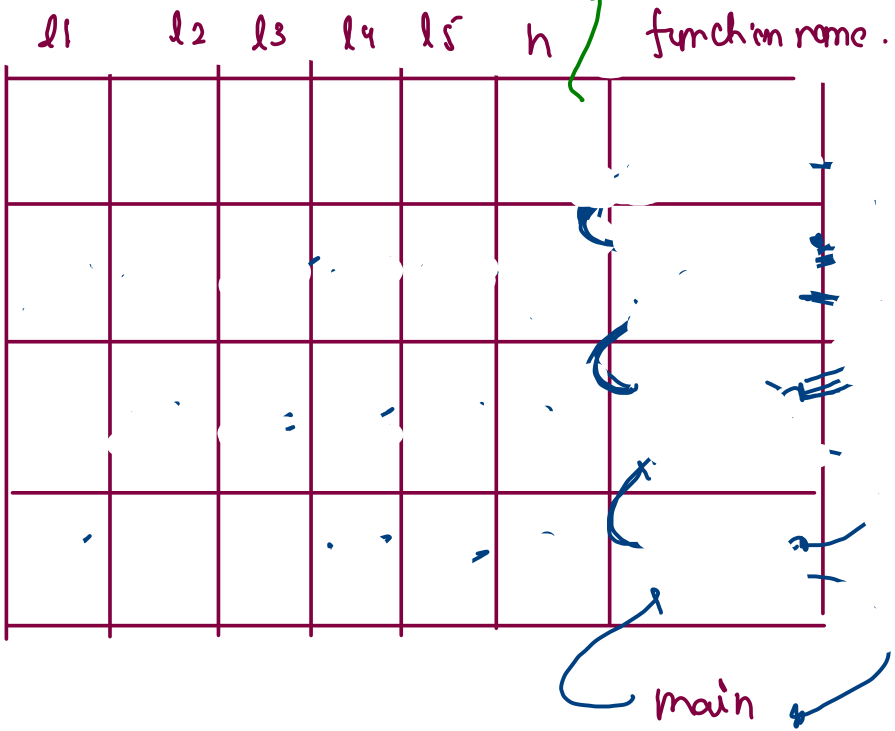
low level Analysis (print zig-zag)

n = 3

if(n = = 0) return;

```java
public static void pzz(int n) {
    // pre area

    System.out.println("Pre : " + n);
    pzz(n - 1);

    // In area
    System.out.println("In : " + n);

    pzz(n - 1);

    // post area
    System.out.println("Post : " + n);
}
```

Base case



STACK

console

pre : 3
pre : 2
pre : 1
In : 1
post : 1
In : 2
pre : 1
In : 1
post : 1
post 2

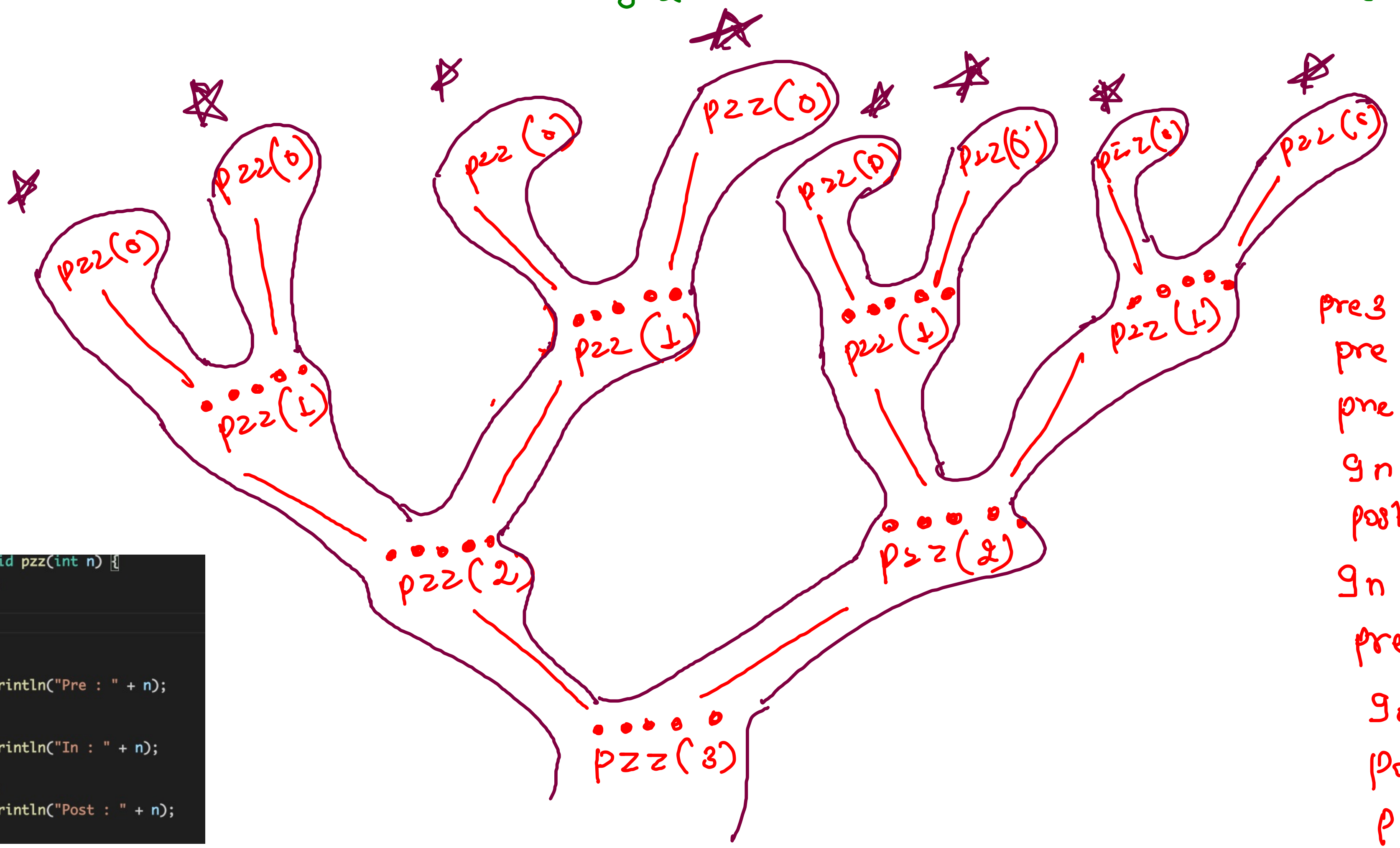
In : 3
pre : 2
pre : 1
In : 1
post : 1
In : 2
pre : 1
In : 1
post : 1
post 2
post 3

> 3 2 1 1 1 2 1 1 1 2 3 2 1 1 1 2 1 1 1 2 3

✓ 3 2 1 1 1 2 1 1 1 2 3 2 1 1 1 2 1 1 1 2 3

Recursion Tree / Euler Diagram / Euler traversal

3 2 1 1 1 2 1 1 1 2 3 2 1 1 1 2 1 1 1 2 3

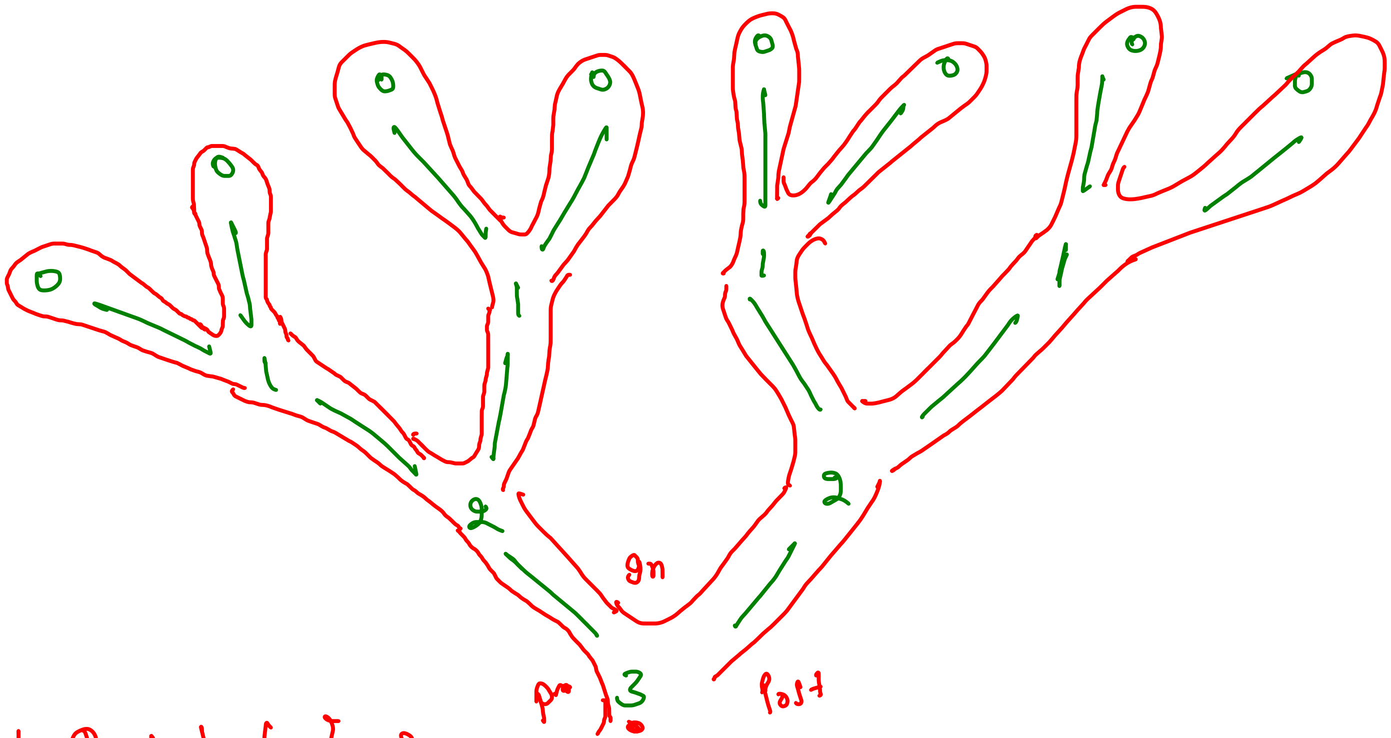



```
public static void pzz(int n) {
    // base case
    if(n == 0)
        return;

    // pre area
1.  System.out.println("Pre : " + n);
2.  pzz(n - 1);
    // In area
3.  System.out.println("In : " + n);
4.  pzz(n - 1);
    // post area
5.  System.out.println("Post : " + n);
}
```

pre 3	In 3
pre 2	pre 2
pre 1	pre 1
In 1	In 1
post 1	post 1
In 2	In 2
pre 1	pre 1
In 1	In 1
Post 1	post 1
post 2	post 2
	post 3

sys o(n)
p z z(n-1)
syso(n)

p z z (n-1)
syso(n)



3 2 1 1 1 2 1 1 1 2 3

2 1 1 1 2 1 1 1 2 3

toh(n, source, destination, Helper)

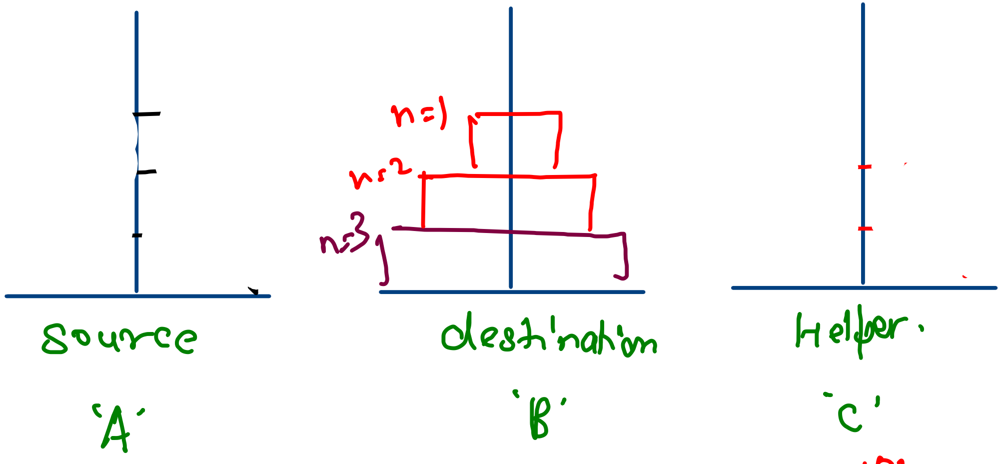
Expectation →

toh(3, A, B, C) → print steps to move 3 discs from A to C using B.

src dst hlp

(n-1) disc - Recursion

toh(3, A, B, C)



// faith → src dst hlp

toh(n-1, src, helper, dst)

A C B

// my work

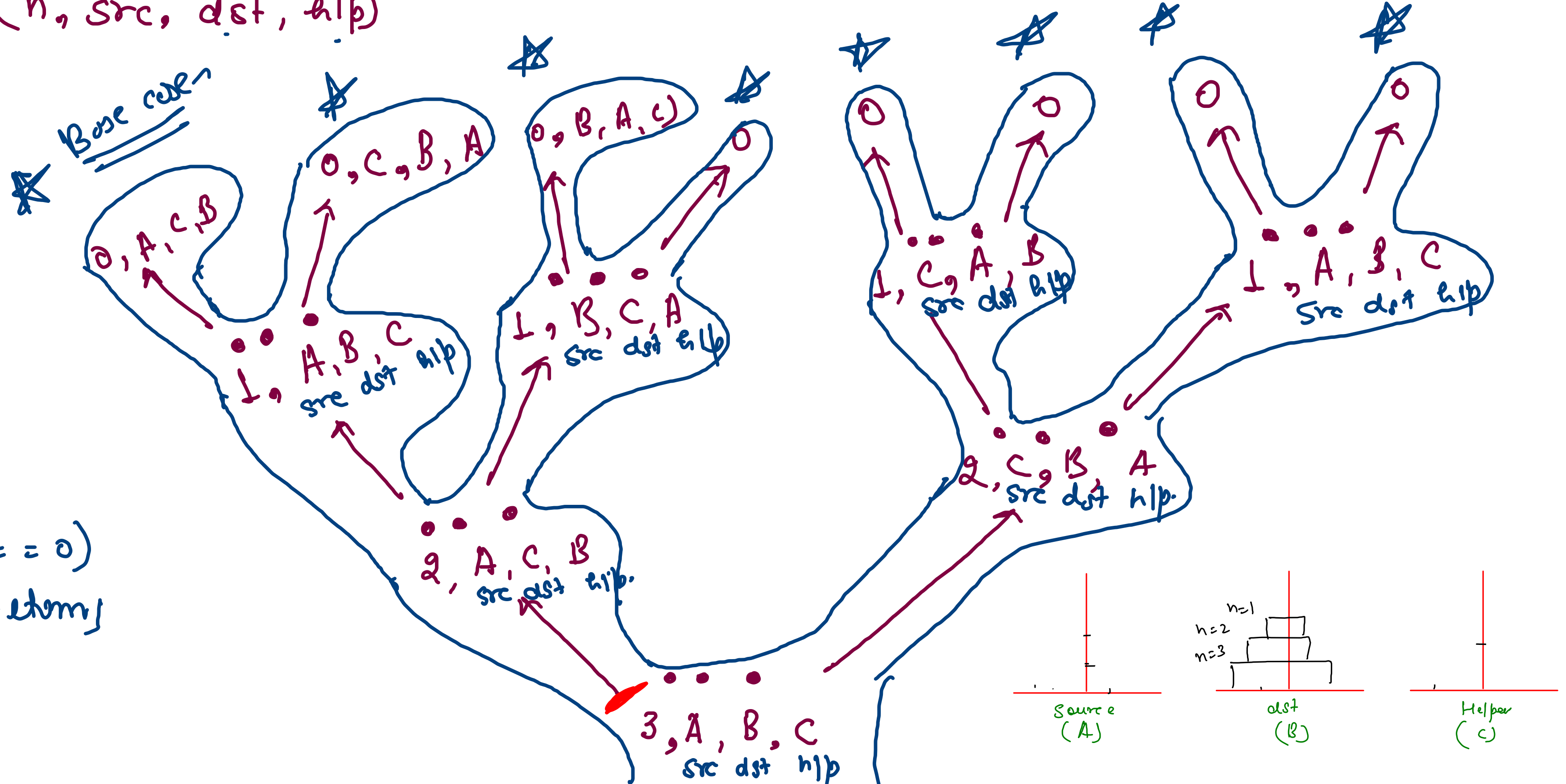
Syso('move' n^th disc from src to dst);

// faith → B

toh(n-1, hlp, dst, src)

src dst hlp

toh(n, src, dst, hlp)



if(n == 0)
return;

```
public static void toh(int n, char src, char dst, char hlp) {
    // faith
    toh(n - 1, src, hlp, dst);
    // my work
    System.out.println("move " + n + "th disc from " + src + " to " + dst);
    // faith
    toh(n - 1, hlp, dst, src);
}
```

(i) move 1 A → B
(ii) move 2 A → C
(iii) move 1 B → C
(iv) move 3 A → B
(v) move 1 C → A
(vi) move 2 C → B
(vii) move 1 A → B

toh(n) {

toh(n-1) $\longrightarrow$ C1

sysо()

toh(n-1) $\longrightarrow$ C2

}

$\boxed{C1 + C2 + 1}$

$T(n) = T(n-1) + T(n-1) + 1$ } Reccurence Relation

$T(n) = \cancel{2T(n-1)} + 1 \longrightarrow (1)$

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

$\cancel{2^2 T(n-2)} = \cancel{2^3 T(n-3)} + 2^2 \longrightarrow (3)$

$\cancel{2^3 T(n-3)} = \cancel{2^4 T(n-4)} + 2^3 \longrightarrow (4)$

$\vdots$

} x.

$n - x = 1$

$\boxed{x = n-1}$

$\cancel{2^{x-1} T(n-(x-1))} = 2^x T(n-x) + 2^{x-1} \longrightarrow (x)$

$\boxed{T(1) = 1}$

$T(n) = 2^x T(n-x) + [2^0 + 2 + 2^2 + 2^3 + \cdots + 2^{n-1}$

$n = 3$

$T(n-x) = T(0)$

$n - x = 0, \boxed{n = x}$

$T(n) = 2^n + 1\left(\frac{2^n - 1}{2-1}\right)$

$\frac{a(r^n - 1)}{r-1} \longrightarrow$ sum of G.P.

$\longrightarrow a, ar, ar^2, ar^3, \ldots\ldots ar^{n-1}$

$= 2^n + 2^n - 1$

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

Reduce $n \to n-1$

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

$O(2^n) =$