n = 4

4

3

2

1

1

2

3

4

n = 2

2

1

1

2

expectation

n=4

4

3

2

1

1

2

3

4

Gain

n=3

3

2

1

1

2

3

ML

4

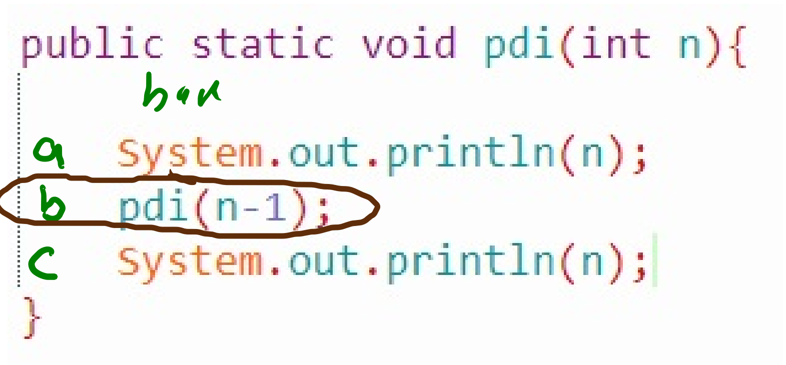
PDI (3)

4

MLW → E , F FE

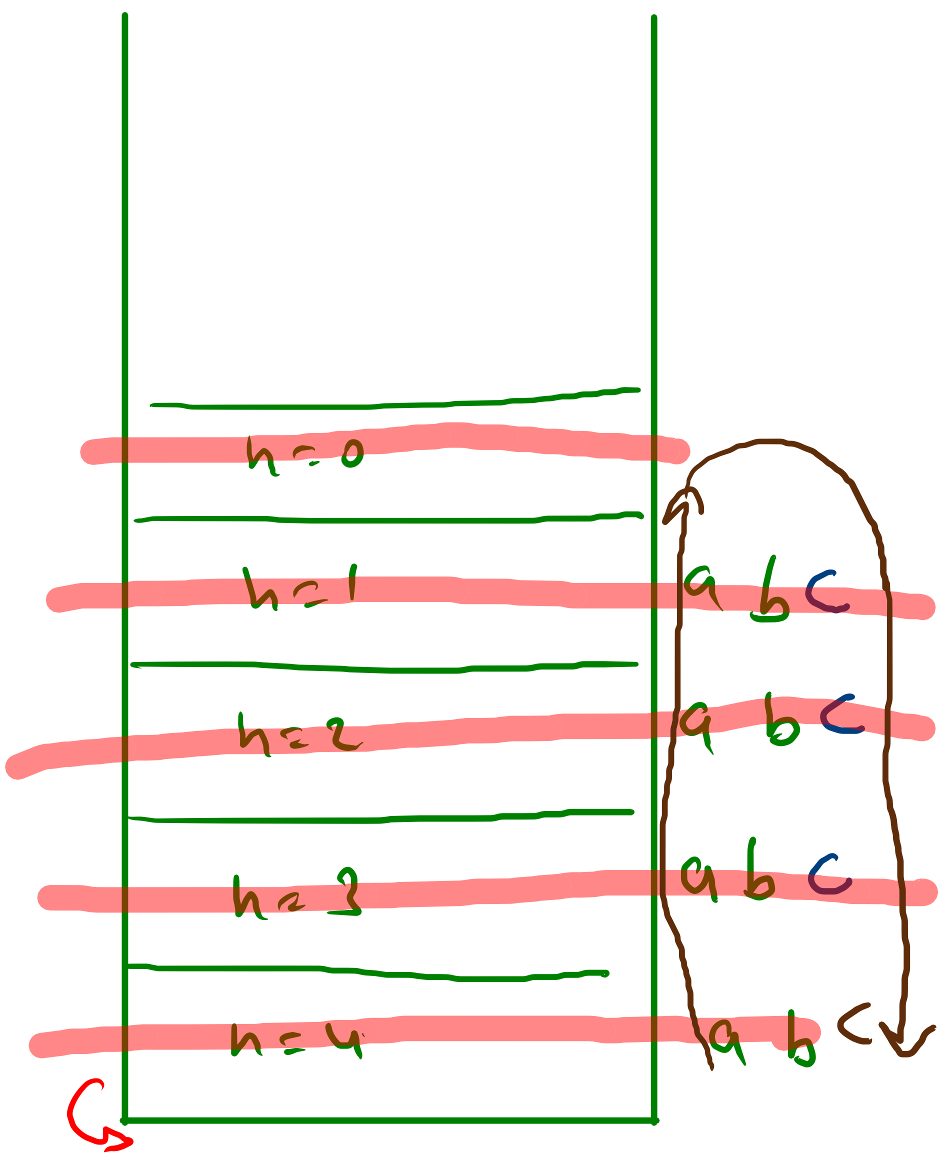
LL → Dry

n = 4



```
public static void pdi(int n){
      bar
a   System.out.println(n);
b   pdi(n-1);
c   System.out.println(n);
}
```

a	4	
a	3	
a	2	
a	1	
	1	b
	2	b
	3	b
	4	b



n = 5

5 × 4 × 3 × 2 × 1

$\boxed{120}$ ←

n = 1

1

n = 0

$\textcircled{1}$

expectation

$n=5$

$5\times\boxed{4\times3\times2\times1}$

reality

$n=4$

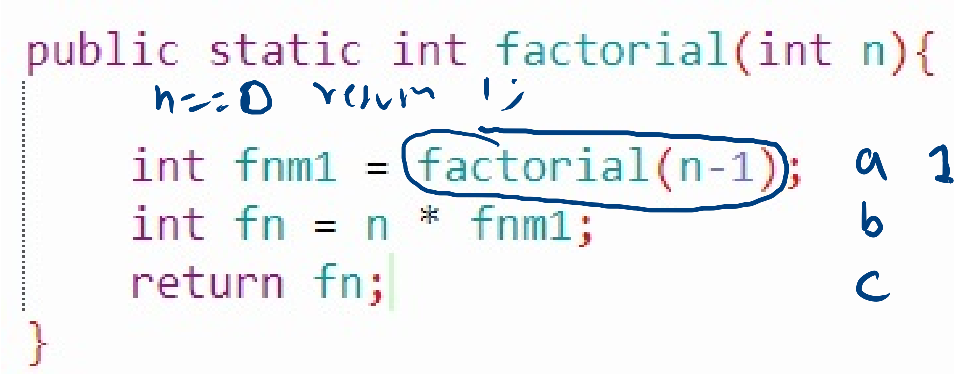
$4\times3\times2\times1$

$n=5$

$5\times$ factorial(4)

$5\times\ 4\times3\times2\times1$

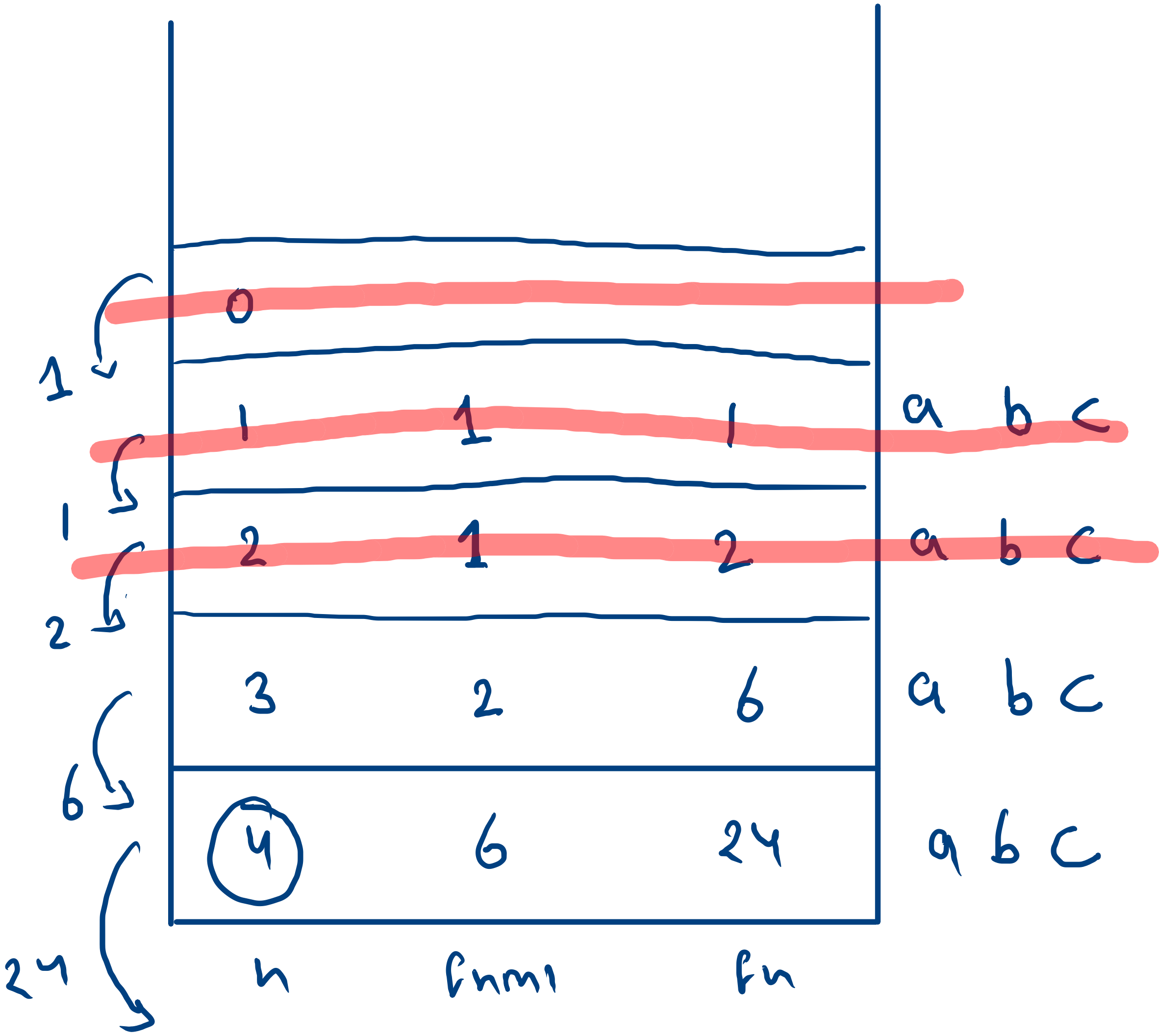
n = 4



```
public static int factorial(int n){
     n==0 return 1;
     int fnm1 = factorial(n-1);      a  1
     int fn = n * fnm1;              b
     return fn;                      c
}
```

n

3×2 = 6



1	~~0~~			
1	~~1~~	~~1~~	~~1~~	~~a b c~~
2	~~2~~	~~1~~	~~2~~	~~a b c~~
6	3	2	6	a b c
24	4	6	24	a b c
	n	fnm1	fn	

$x \to 2$

$n \to 6$

$\boxed{x^n}$ $\quad 2^6$

$2 \times 2 \times 2 \times 2 \times 2 \times 2$

$x \quad ?$

$n = 0$

$\boxed{x^0 = 1}$

E	F	
$x \to 2$	$x = 2$	
$n = 6$	$n = 5$	power$(x, n-1) \times x$
2^6	2^5	

x

n

$$\boxed{x^n = x^{n-1} \cdot x}$$

2^{16} $2^{15} \times 2$ $2^8 \times 2^8$

n

$$2^8 = 2^4 \times 2^4$$

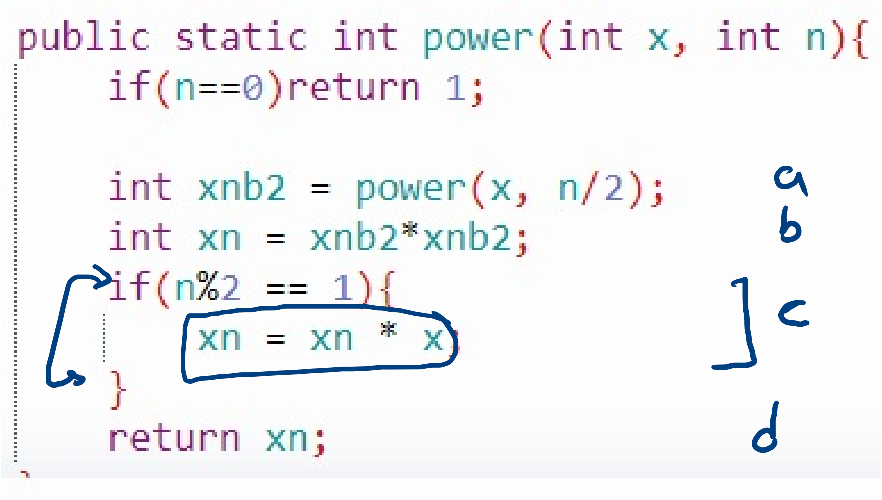
$$2^9 = 2^4 \times 2^4 \boxed{\times 2}$$

$$2^{10} = 2^5 \times 2^5$$

$$2^{11} = 2^5 \times 2^5 \boxed{\times 2}$$

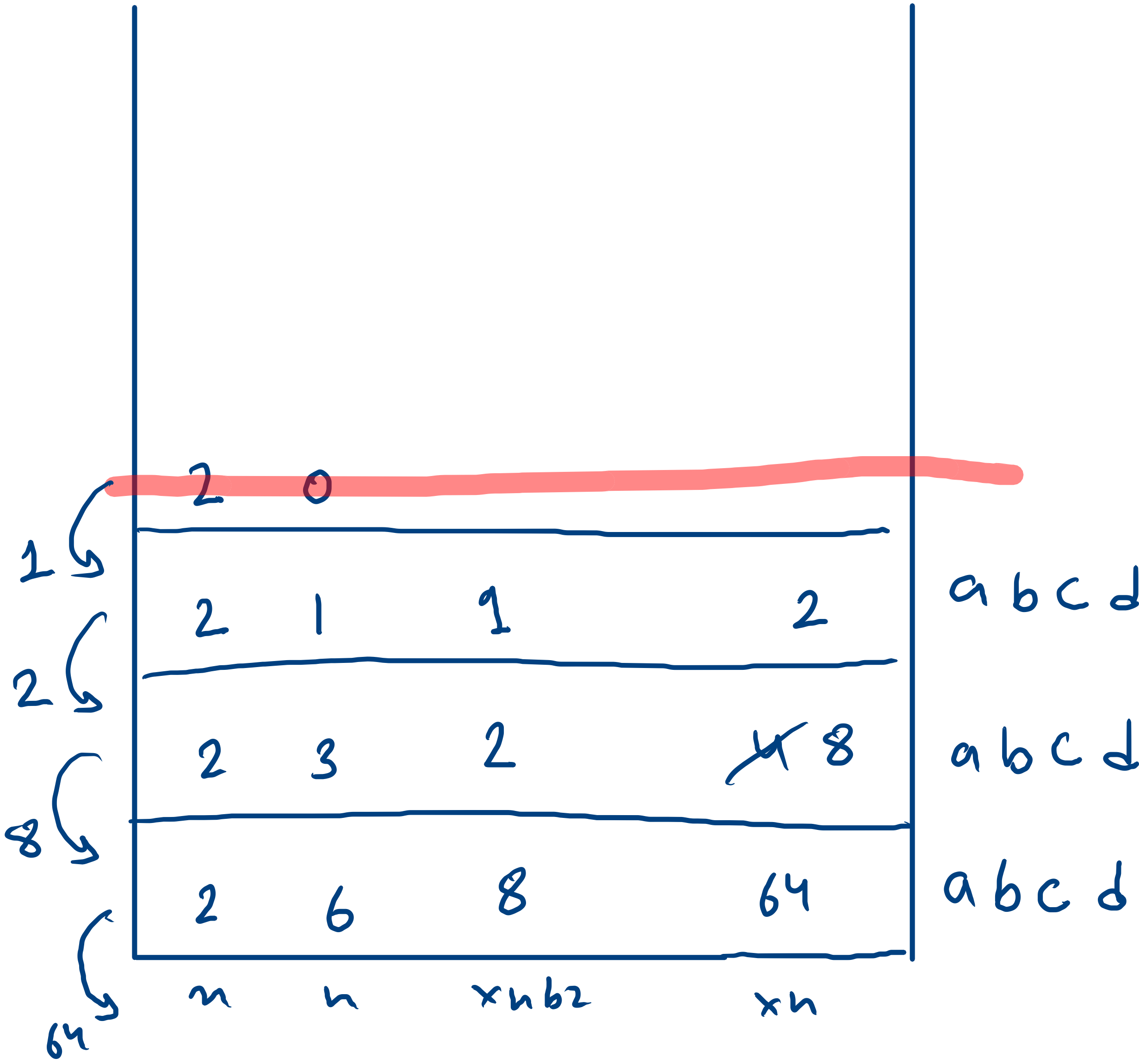
$$x^n = \begin{cases} x^{n/2} \cdot x^{n/2} & n \text{ is even} \\ x^{n/2} \cdot x^{n/2} \cdot x & \end{cases}$$

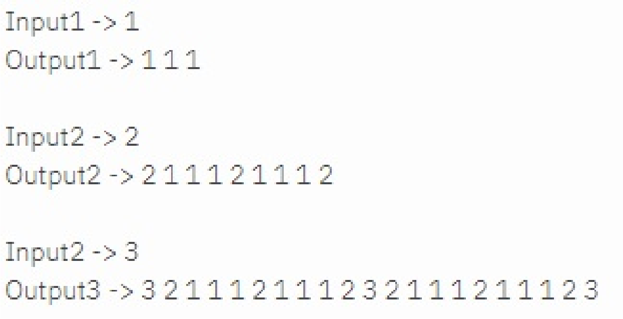
2^6



```
public static int power(int x, int n){
    if(n==0)return 1;

    int xnb2 = power(x, n/2);
    int xn = xnb2*xnb2;
    if(n%2 == 1){
        xn = xn * x;
    }
    return xn;
```

xn = 4x2 = 8





```
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
```

n = 1

1 1 1

$1 \; f(0) \; 1 \; f(0) \; 1$

n = 2

$2 \; f(1) \; 2 \; f(1) \; 2$

$2 \; \underbrace{1\,1\,1}_{f(1)} \; 2 \; \underbrace{1\,1\,1}_{f(1)} \; 2$

↕

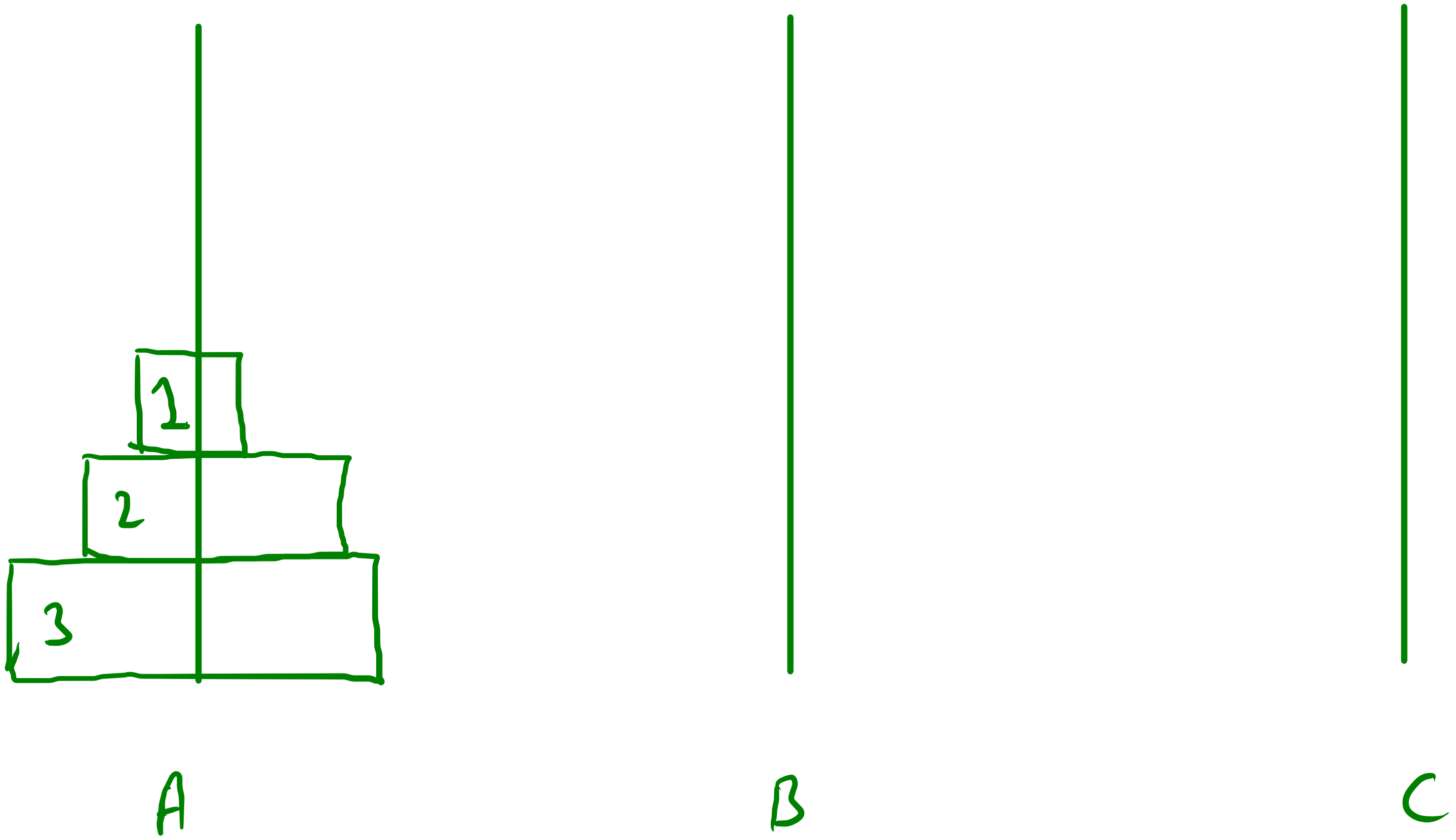
n = 3

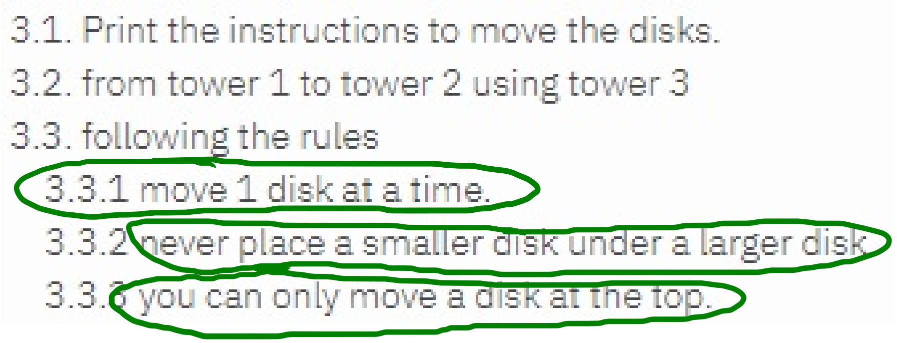
$3 \; f(2) \; 3 \; f(2) \; 3$

✓

$3 \; \underbrace{2 \; 1\,1\,1\,2 \; 1\,1\,1\,2}_{f(2)} \; 3 \; \underbrace{2 \; 1\,1\,1\,2 \; 1\,1\,1\,2}_{f(2)} \; 3$

n = 3




3.1. Print the instructions to move the disks.

3.2. from tower 1 to tower 2 using tower 3

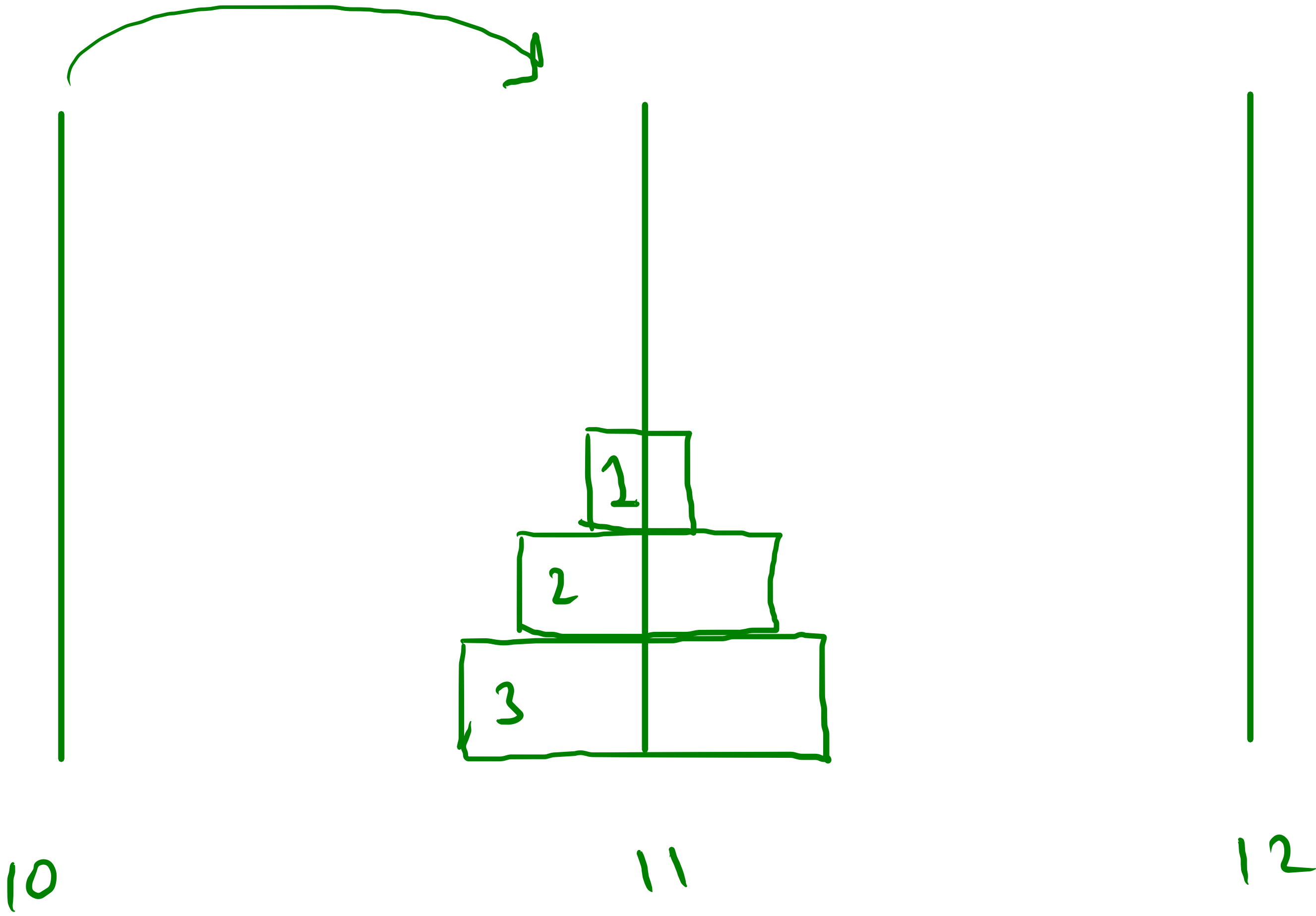
3.3. following the rules

3.3.1 move 1 disk at a time.

3.3.2 never place a smaller disk under a larger disk

3.3.3 you can only move a disk at the top.

n=3



1[10 -> 11] ✓
2[10 -> 12] ✓
1[11 -> 12] ✓
3[10 -> 11] ✓
1[12 -> 10] ✓
2[12 -> 11] ✓
1[10 -> 11] ✓

n = 3

src des

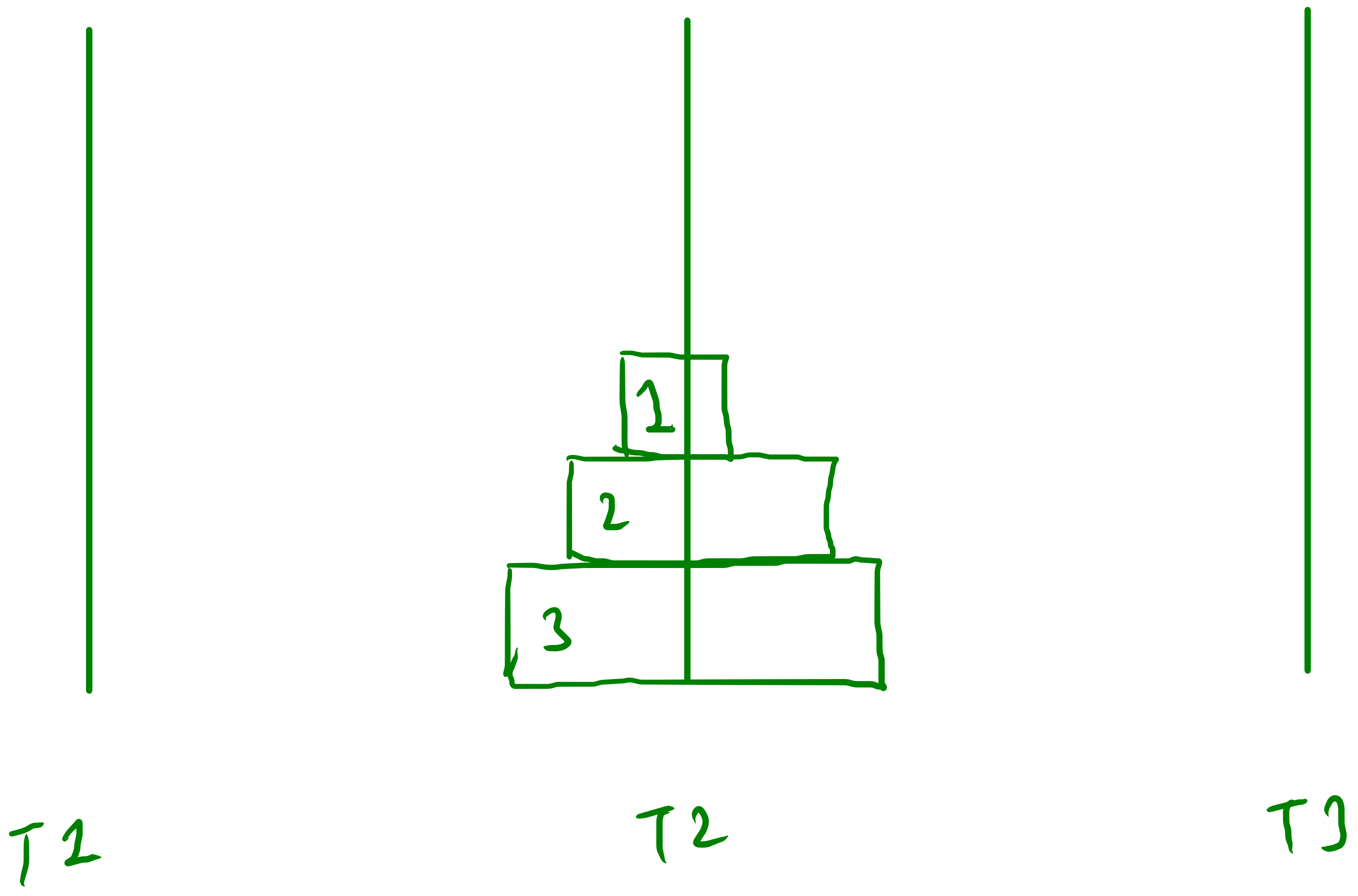
toh(3, T1, T2, T3)

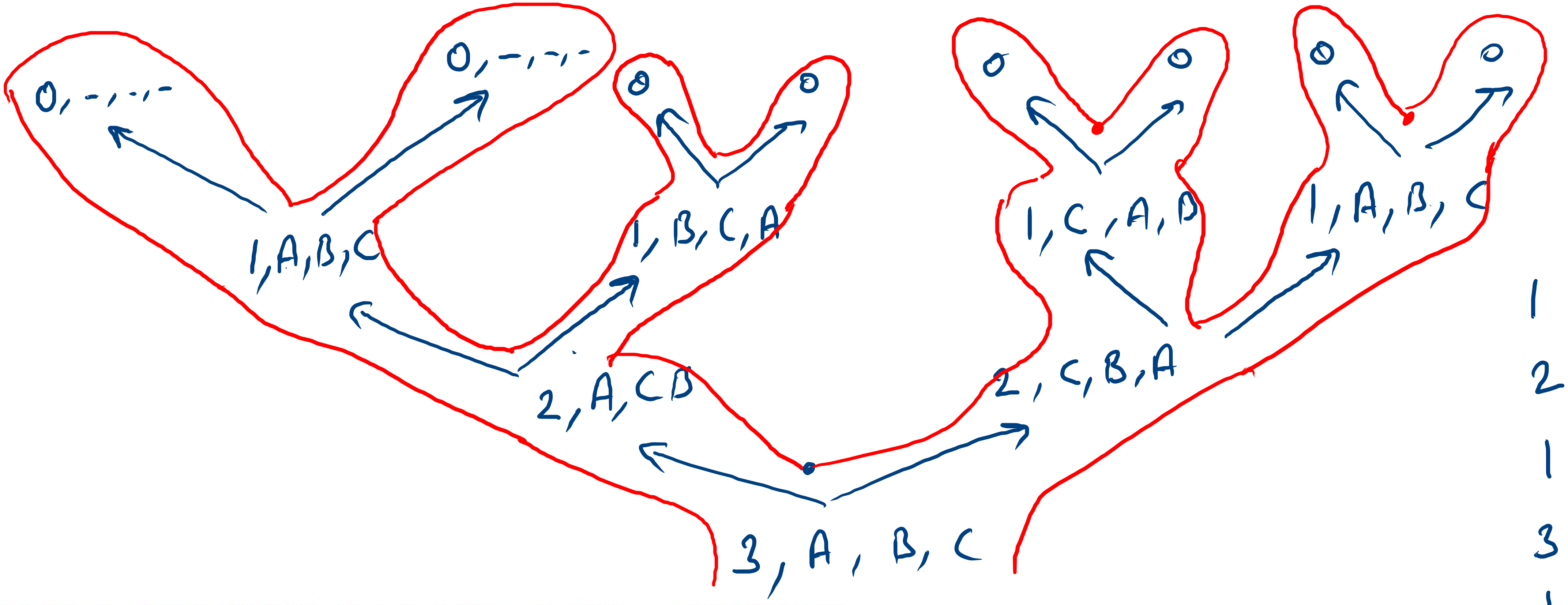
toh(2, T1, T3, T2)

3 T1 → T2

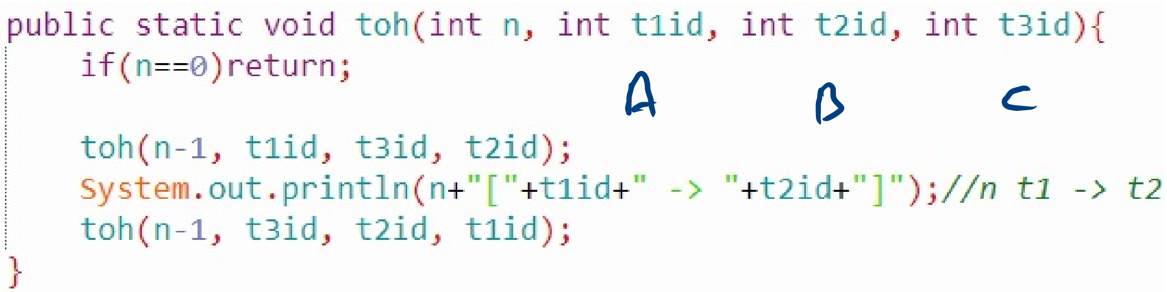
toh(2, T3, T2, T1)

NL



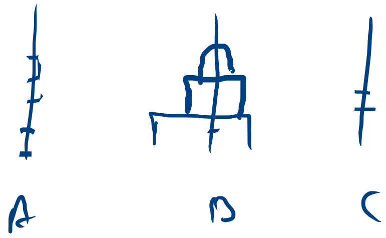


1	A → B
2	A → C
1	B → C
3	A → B
1	C → A
2	C → B
1	A → B



```
public static void toh(int n, int t1id, int t2id, int t3id){
    if(n==0)return;

    toh(n-1, t1id, t3id, t2id);
    System.out.println(n+"["+t1id+" -> "+t2id+"]");//n t1 -> t2
    toh(n-1, t3id, t2id, t1id);
}
```

A B C



$$\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ \hline 7 & 5 & 3 & 4 & 2 & 1 \end{array}$$

7
5
3
4
2
1

C.W.

E

arr idx
0

7
5
3
4
2
1

faith

arr idx
1

5
3
4
2
1

H.W

E

arr idx
5

7
5
3
4
2
1

faith

arr idx
4

7
5
3
4
2

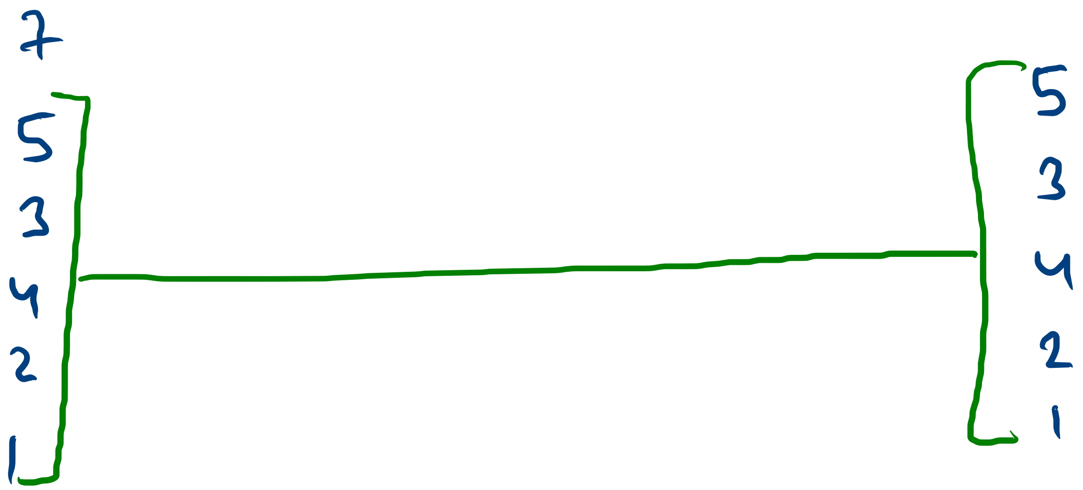
	0	1	2	3	4	5
arr →	7	5	3	4	2	1

Expectation

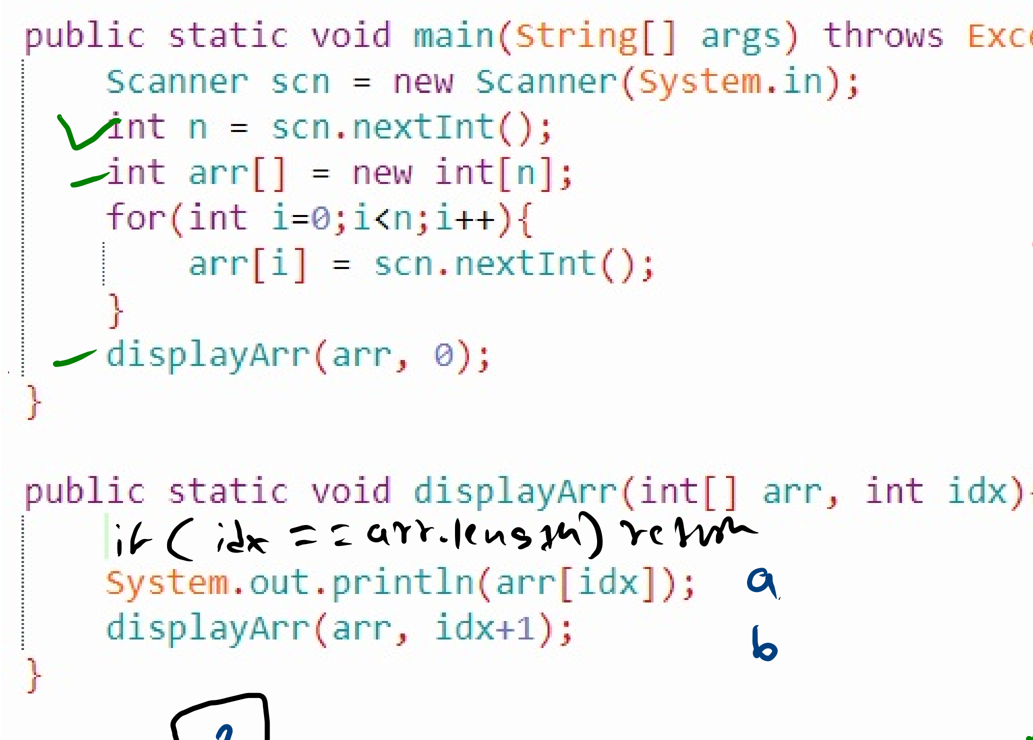
arr	idx
	0

Faith

arr	idx
	1

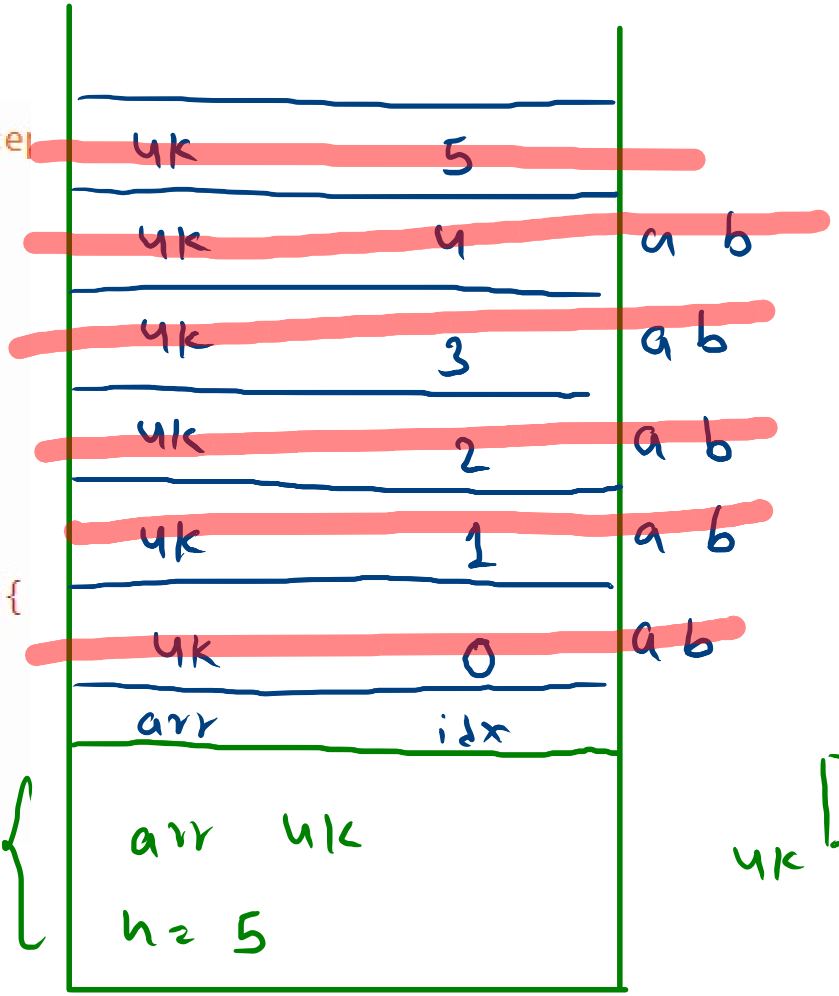


```
print(arr[idx])
Faith(arr, idx+1)
```

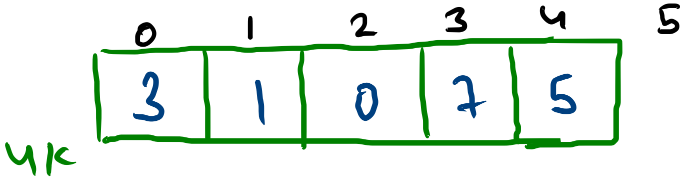


```
public static void main(String[] args) throws Excep
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int arr[] = new int[n];
    for(int i=0;i<n;i++){
        arr[i] = scn.nextInt();
    }
    displayArr(arr, 0);
}

public static void displayArr(int[] arr, int idx){
    if ( idx = = arr.length) return
    System.out.println(arr[idx]);   a
    displayArr(arr, idx+1);         b
}
```

3
1
0
7
5



main



length-1

H.W. Display Array In Reverse

$$\begin{array}{lcccccc} & 0 & 1 & 2 & 3 & 4 & 5 \\ arr \rightarrow & [\ 7 & 5 & 3 & 4 & 2 & 1\ ] \end{array}$$

1

2

4

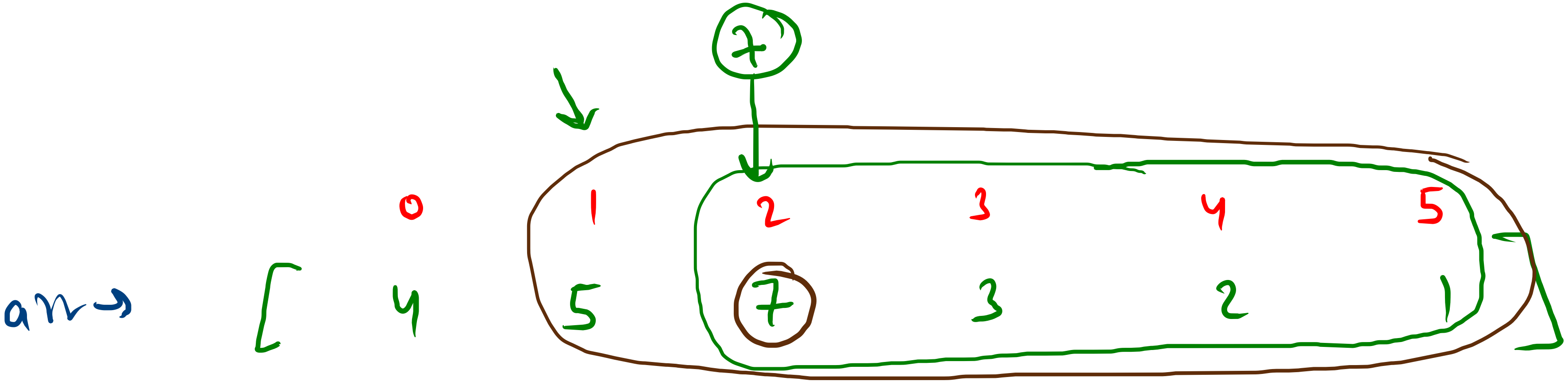
3

5

7

$$\begin{array}{cc} & \begin{matrix}0 & 1 & 2 & 3 & 4 & 5\end{matrix} \\ \text{arr} \rightarrow & \begin{bmatrix}4 & 5 & \textcircled{7} & 3 & 2 & 1\end{bmatrix} \end{array}$$

$\textcircled{7}$



arr →

0	1	2	3	4	5
4	5	7	3	2	1

Expectation

idx = 1

1 ----

7

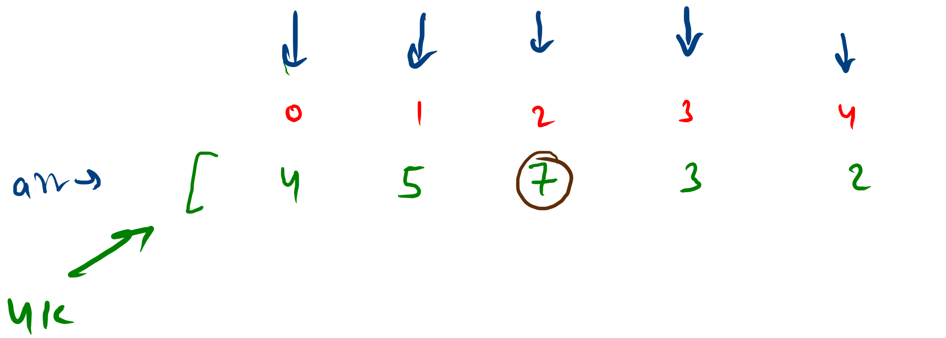
Faith

idx 2

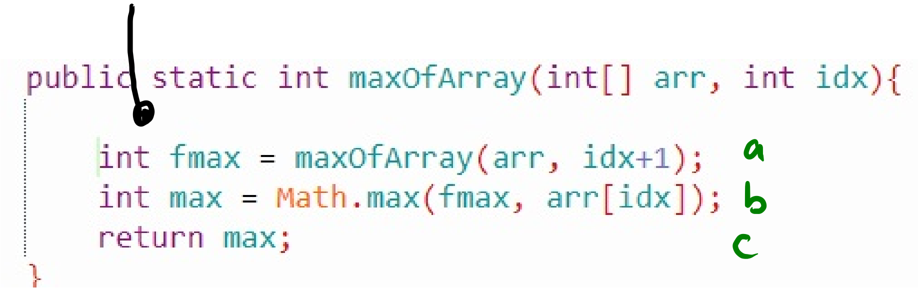
2 ----

7

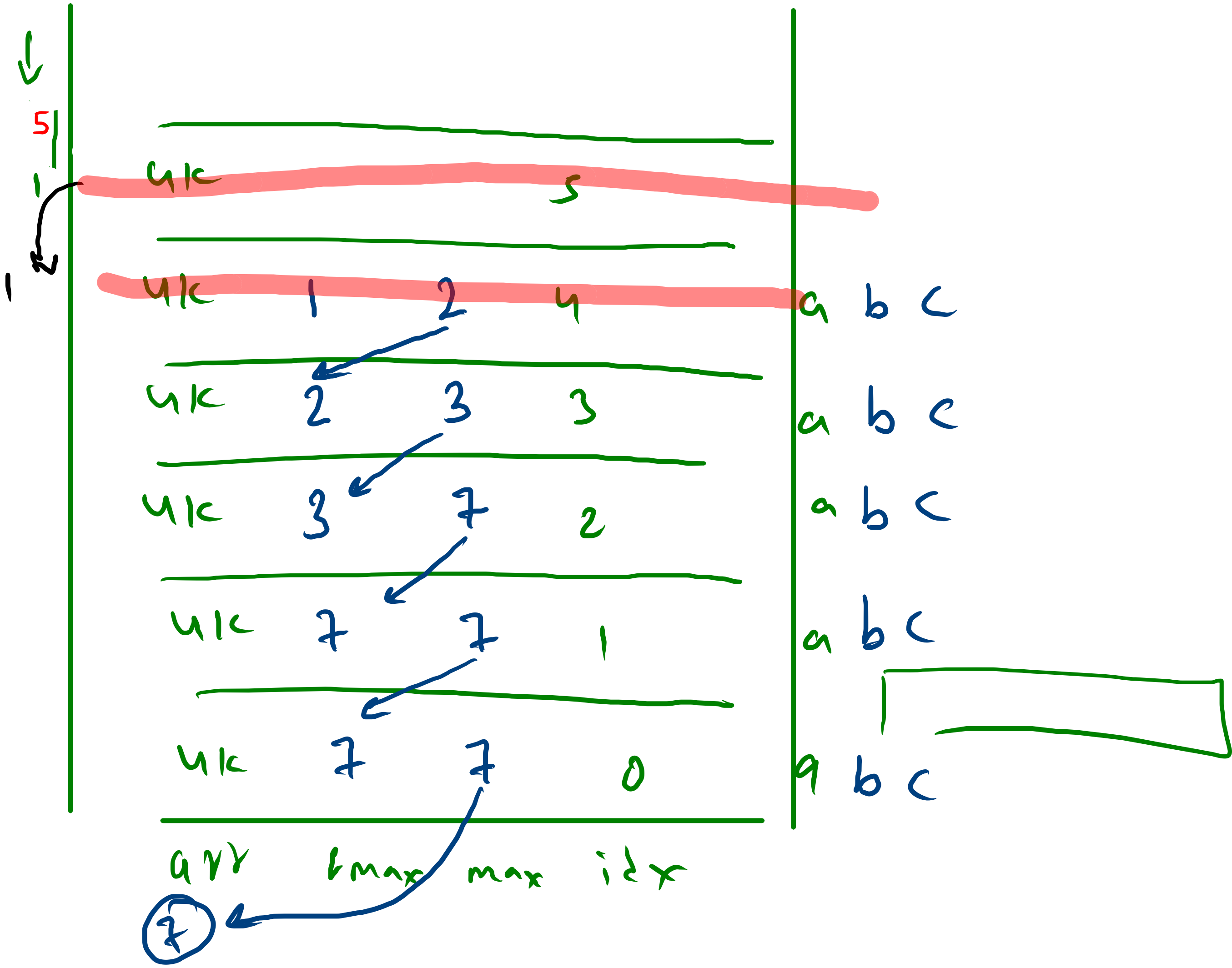
```
int lmax = max(idx+1)

return max(lmax,
           arr[i])
```



```
idx == arr.length-1
        return arr[idx]}
```



```
public static int maxOfArray(int[] arr, int idx){

    int fmax = maxOfArray(arr, idx+1);   a
    int max = Math.max(fmax, arr[idx]);  b
    return max;                          c
}
```



$$\begin{array}{lcccccc} & & \checkmark & & \checkmark & & \checkmark \\ & 0 & \textcircled{1} & 2 & 3 & 4 & 5 \\ \text{arr} \rightarrow & [\;7 & 4 & 3 & 4 & 2 & 4\;] \end{array}$$

x → 4

1

x = 9

-1

arr→

0	1	2	3	4	5
7	4	3	4	2	4

arr[idx] != x

E

idx 0 x=4

Call

idx = 1 x=4

arr[idx] == x

E

idx= 1

F

X

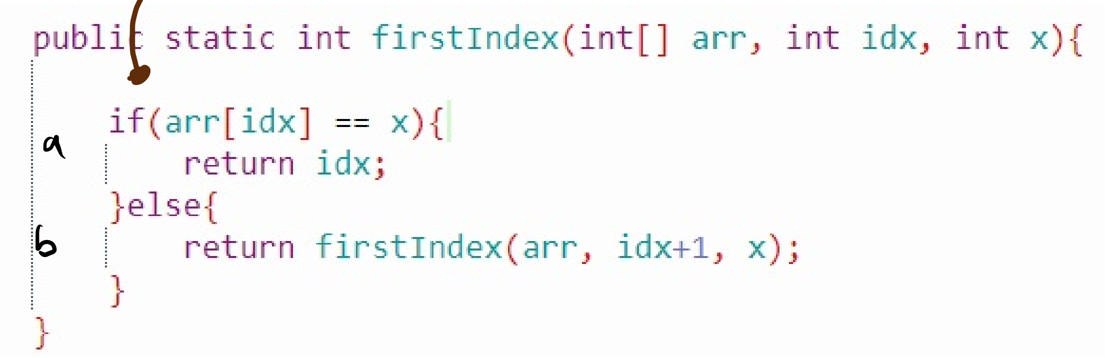
1

1

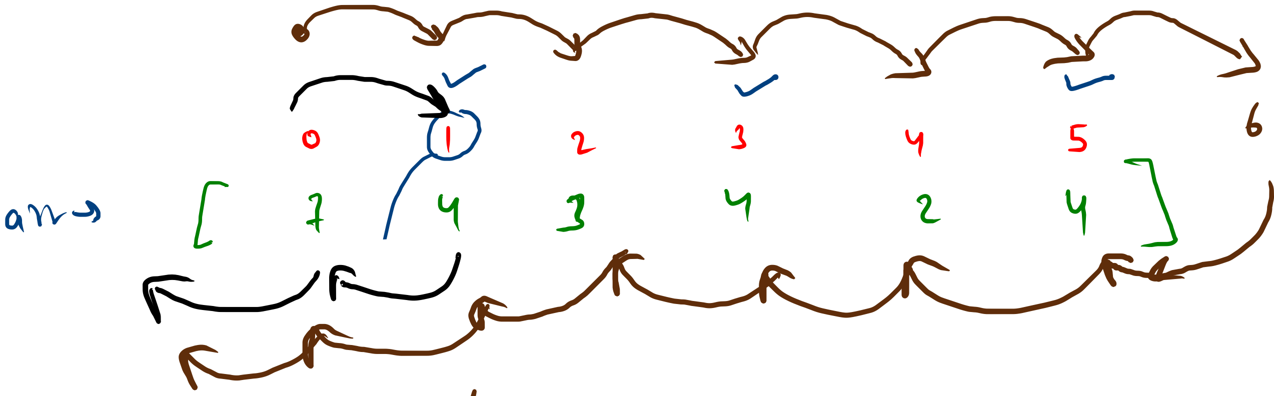
1

ans Call(idx+1)

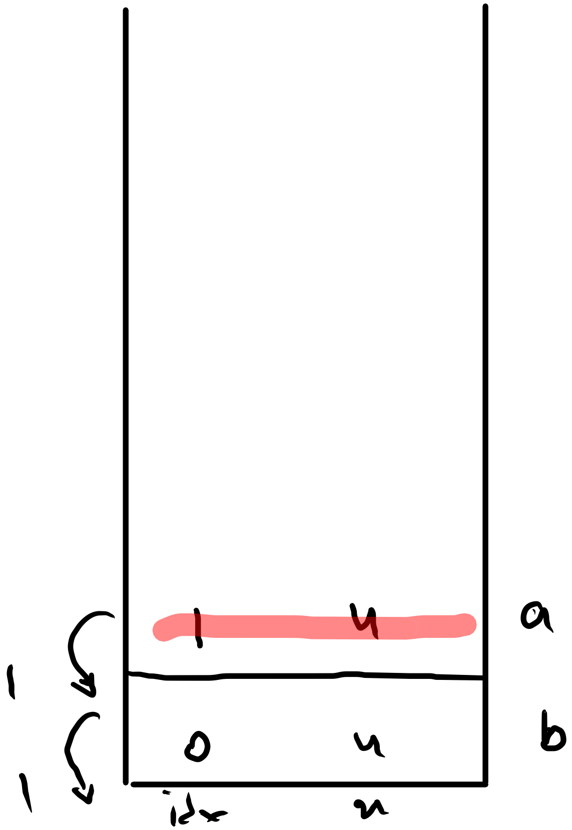
idx == length return -1



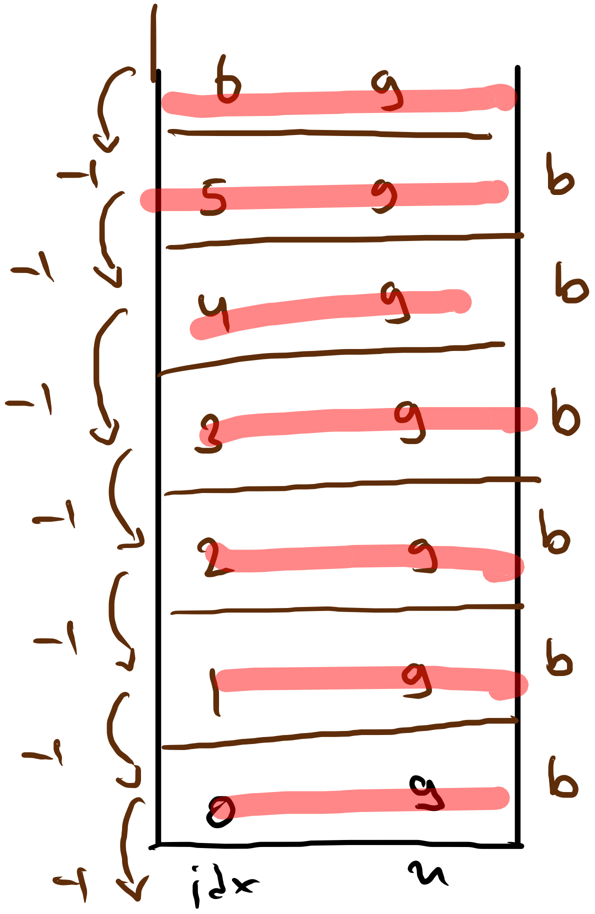
```
public static int firstIndex(int[] arr, int idx, int x){

    if(arr[idx] == x){
        return idx;
    }else{
        return firstIndex(arr, idx+1, x);
    }
}
```



arr→
0
1
2
3
4
5
6
[7
4
3
4
2
4]



1 4 a
0 4 b
1
1
idx n



6 9
5 9 b
4 9 b
3 9 b
2 9 b
1 9 b
0 9 b
-1
-1
-1
-1
-1
-1
-1
idx n