$\underline{pzz(3)} = 3\ \underbrace{2\ 1\ 1\ 1\ 2\ 1\ 1\ 1\ 2}\ 3\ \underbrace{2\ 1\ 1\ 1\ 2\ 1\ 1\ 1\ 2}\ 3$

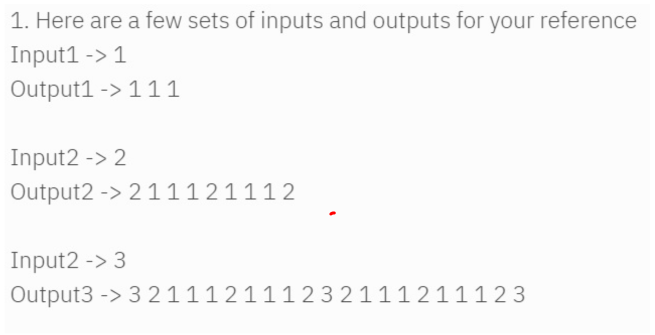
expectation

$\underline{pzz(2)} = 2\ 1\ 1\ 1\ 2\ 1\ 1\ 1\ 2$

faith

$pzz(3) = syso(n) + pzz(2) + syso(n$

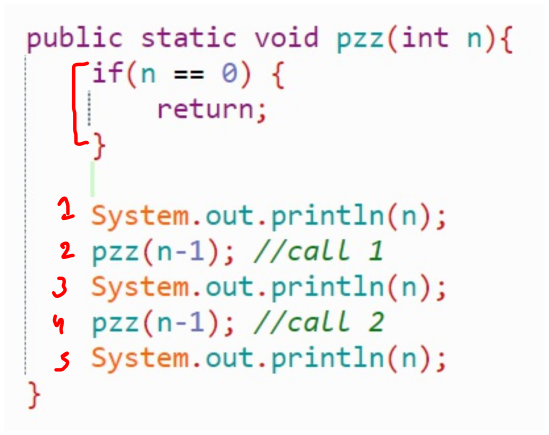
$+ pzz(2) + syso(n)$



1. Here are a few sets of inputs and outputs for your reference
Input1 -> 1
Output1 -> 1 1 1

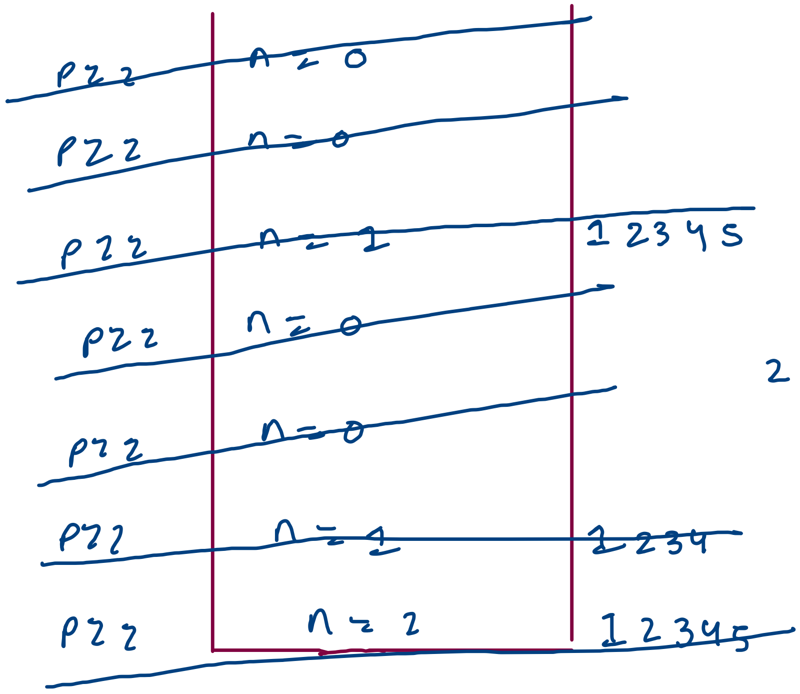
Input2 -> 2
Output2 -> 2 1 1 1 2 1 1 1 2

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



```
public static void pzz(int n){
    if(n == 0) {
        return;
    }

1   System.out.println(n);
2   pzz(n-1); //call 1
3   System.out.println(n);
4   pzz(n-1); //call 2
5   System.out.println(n);
}
```

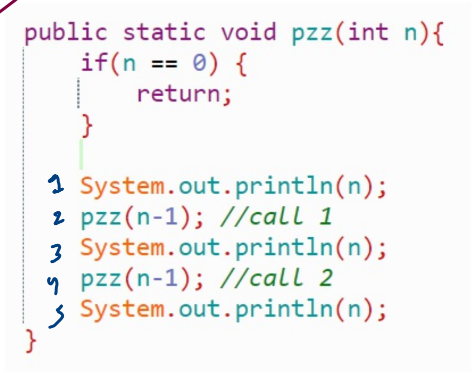
2,4 $\rightarrow$ pzz(n-1)



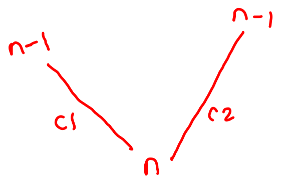
n = 2

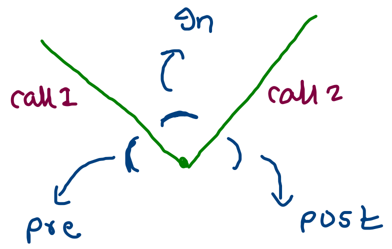
2 1 1 1 2 1 1 1 2

Euler diagram



```
public static void pzz(int n){
    if(n == 0) {
        return;
    }

1   System.out.println(n);
2   pzz(n-1); //call 1
3   System.out.println(n);
4   pzz(n-1); //call 2
5   System.out.println(n);
}
```





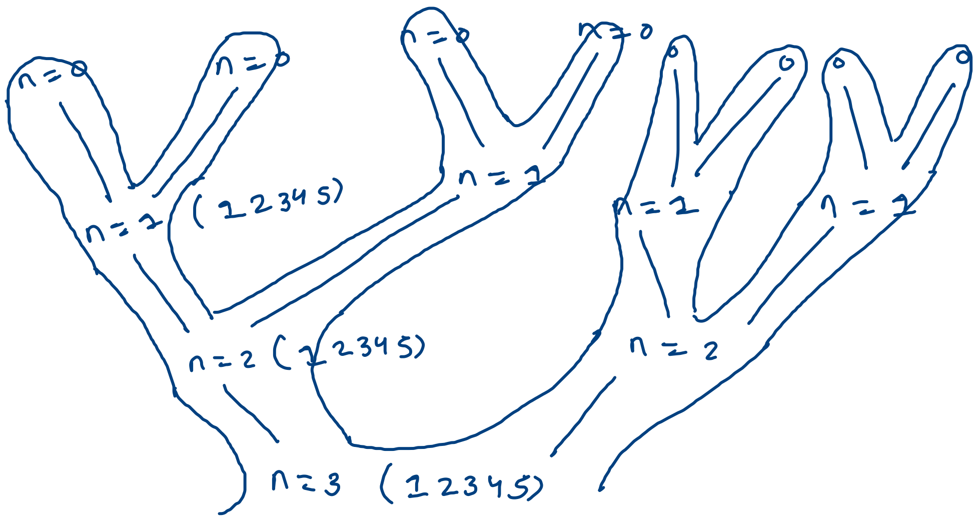
pre → 1

call 1 → 2

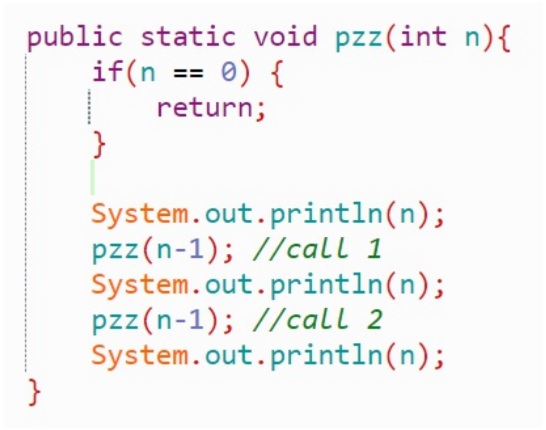
in → 3

call 2 → 4

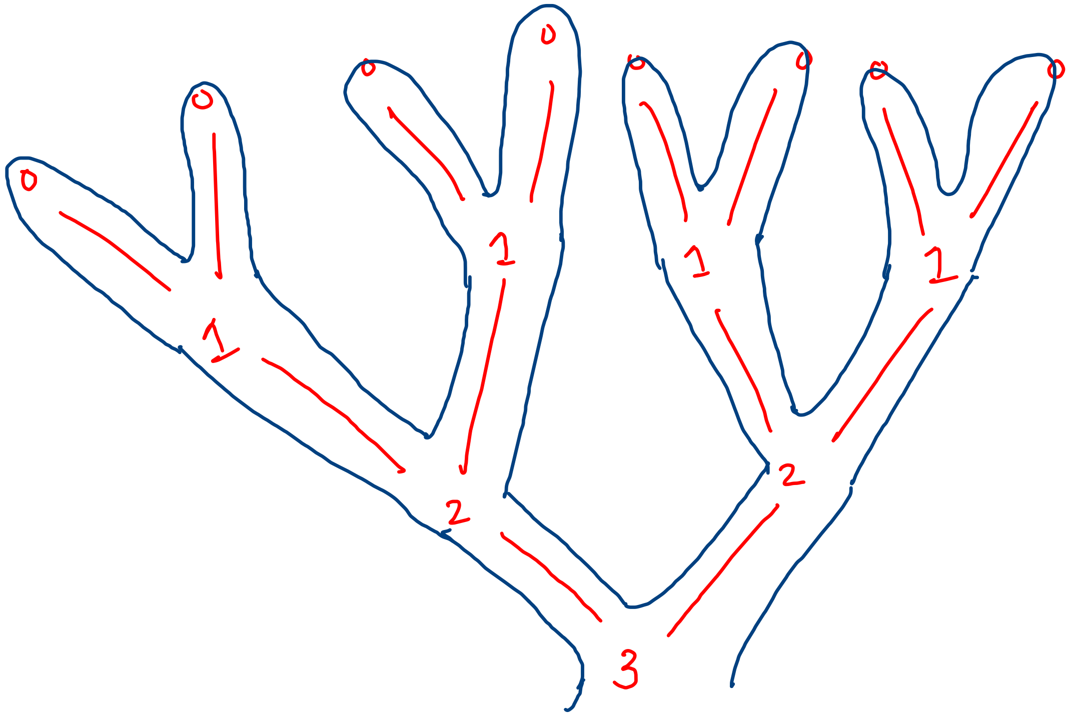
post → 5



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



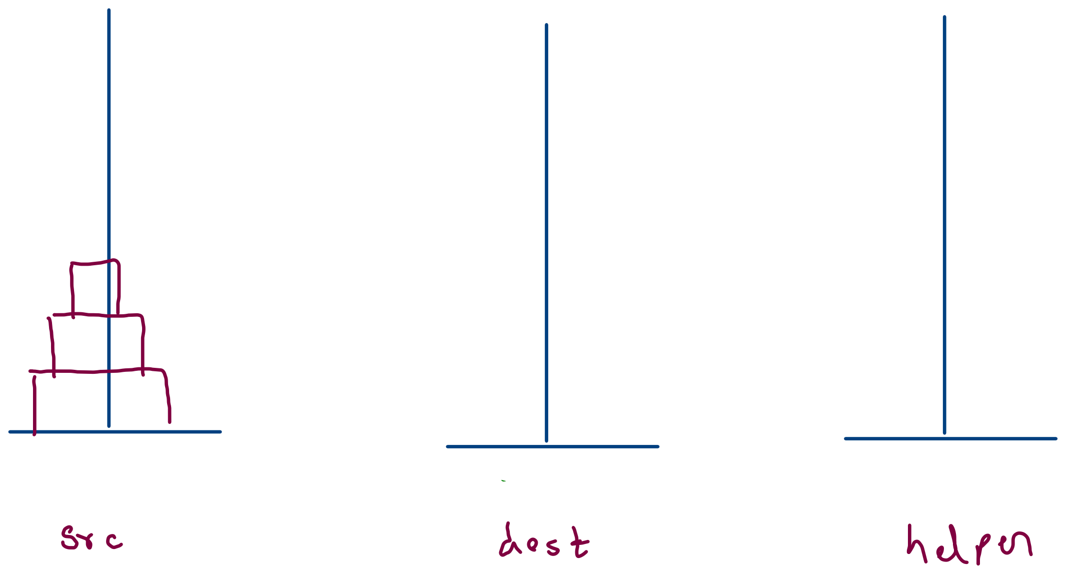
```java
public static void pzz(int n){
    if(n == 0) {
        return;
    }

    System.out.println(n);
    pzz(n-1); //call 1
    System.out.println(n);
    pzz(n-1); //call 2
    System.out.println(n);
}
```



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

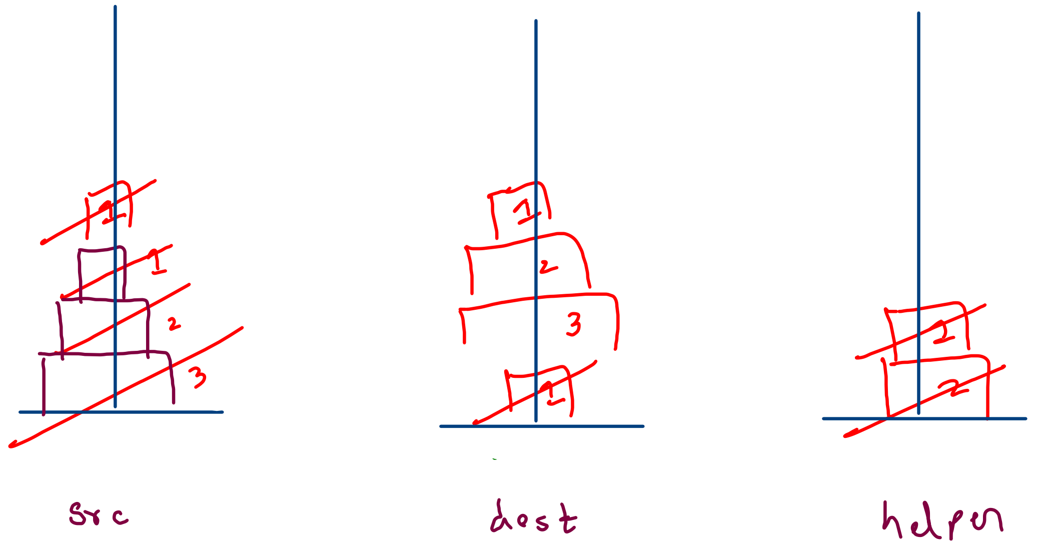
tower of hanoe

n = 3



(i) you can't place a heavier disk on a lighter disk.

(ii) you can only pick one disk at a time



move 1 from src to dest

move 2 from src to helper

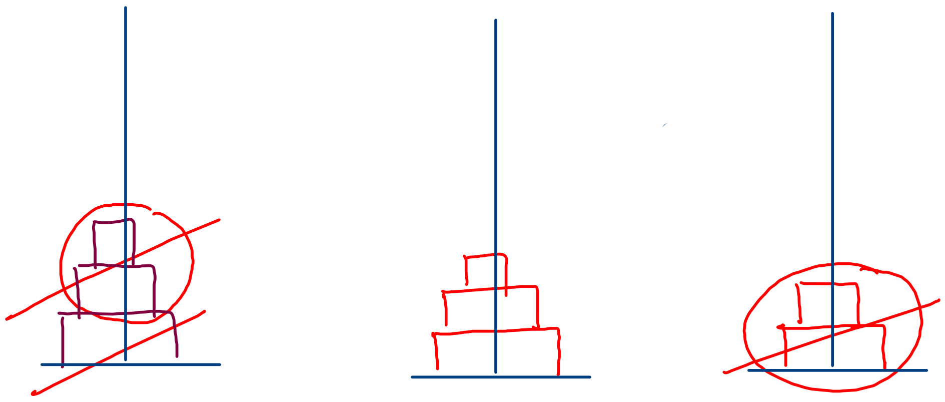
move 1 from dest to helper

move 3 from src to dest

move 1 from helper to src

move 2 from helper to dest

move 1 from src to dest



src	dest	helper
10	11	12
t1	t2	t3

1 [10 → 11]

disk no.

(i) faith → transfer (n−1) dishs from src to helper.

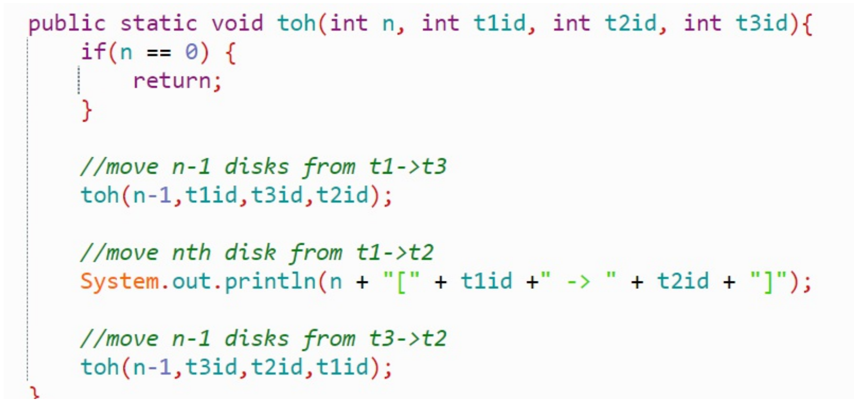
(ii) self work → move nth dish from src to dest

(iii) faith → transfer (n−1) dishs from helper to dest.

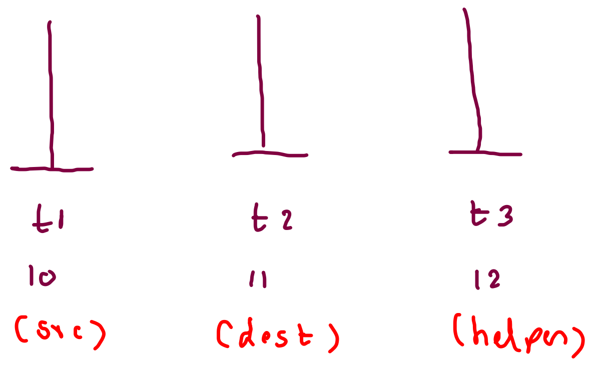
problem → transfer n dishs from src to dest

Recursion

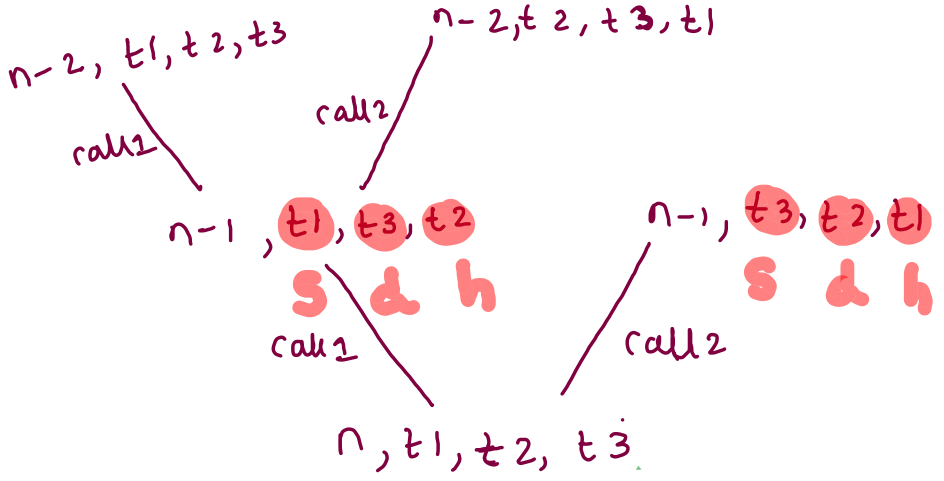
problem or subproblem → transfer some dishs from any source to any dest

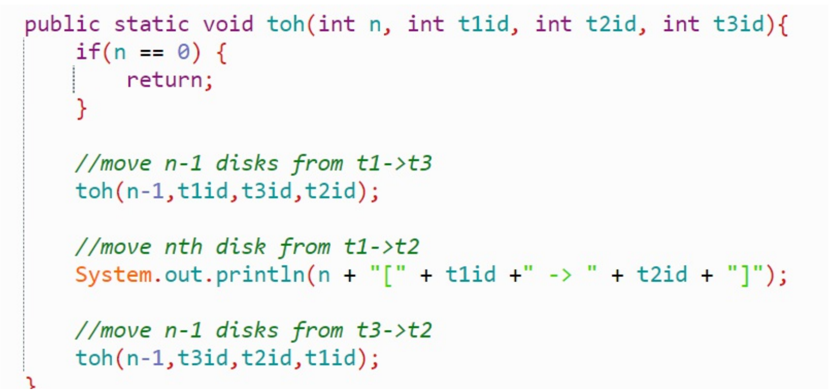


```java
public static void toh(int n, int t1id, int t2id, int t3id){
    if(n == 0) {
        return;
    }

    //move n-1 disks from t1->t3
    toh(n-1,t1id,t3id,t2id);

    //move nth disk from t1->t2
    System.out.println(n + "[" + t1id +" -> " + t2id + "]");

    //move n-1 disks from t3->t2
    toh(n-1,t3id,t2id,t1id);
}
```

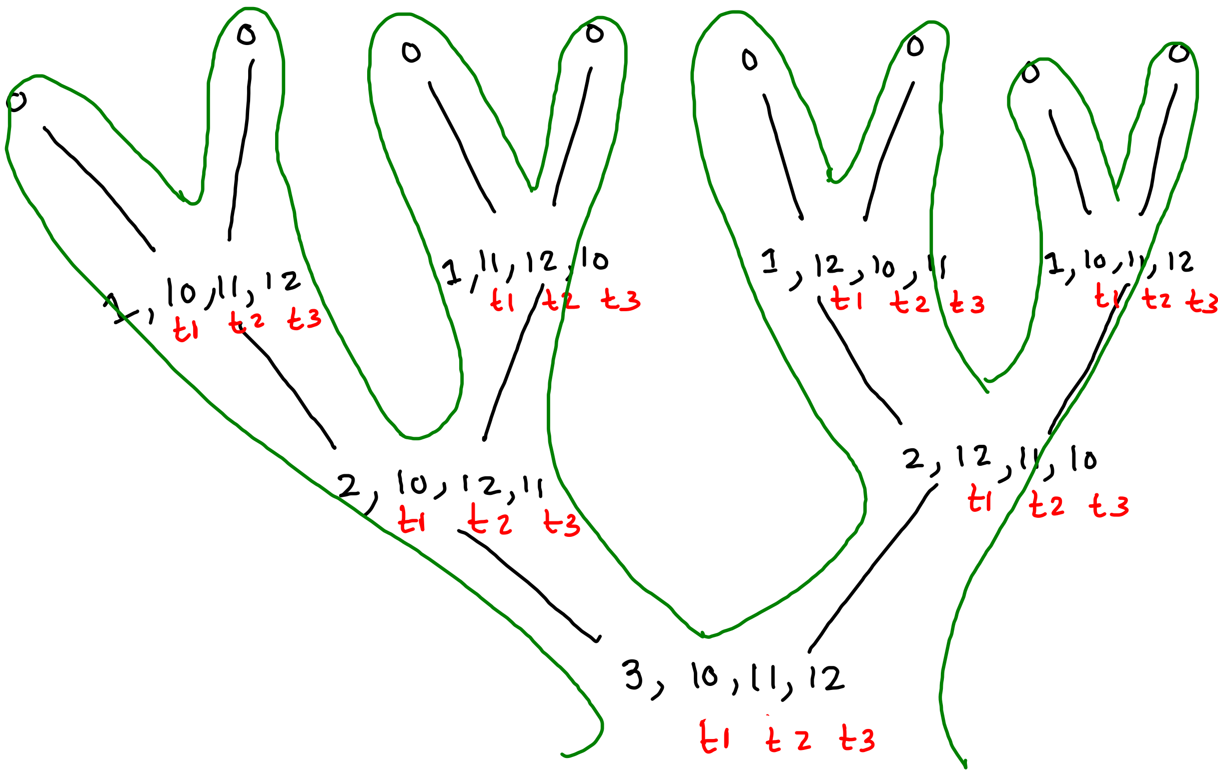


t1 -> t2





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

    //move n-1 disks from t1->t3
    toh(n-1,t1id,t3id,t2id);

    //move nth disk from t1->t2
    System.out.println(n + "[" + t1id +" -> " + t2id + "]");

    //move n-1 disks from t3->t2
    toh(n-1,t3id,t2id,t1id);
}
```



1 [10 -> 11]

2 [10 -> 12]

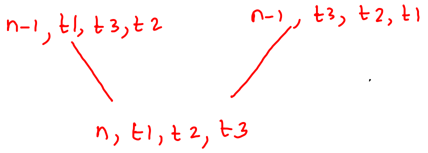
1 [11 -> 12]

3 [10 -> 11]

1 [12 -> 10]

2 [12 -> 11]

1 [10 -> 11]



$$\begin{tikzcd}[column sep=small]
n-1,\ t1,\ t3,\ t2 \arrow[dr, dash] & & n-1,\ t3,\ t2,\ t1 \arrow[dl, dash] \\
& n,\ t1,\ t2,\ t3 &
\end{tikzcd}$$

✓ 1 [10 -> 11]

✓ 2 [10 -> 12]

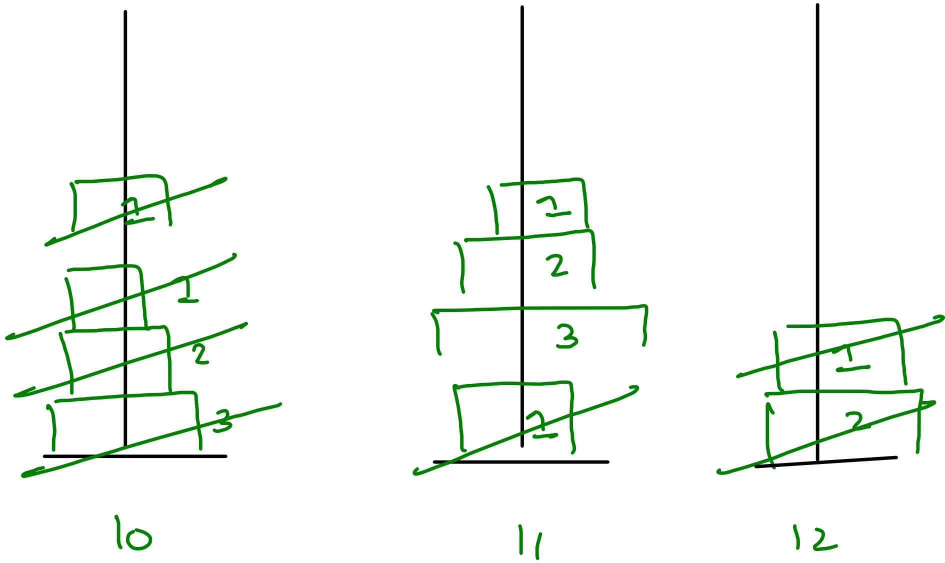
✓ 1 [11 -> 12]

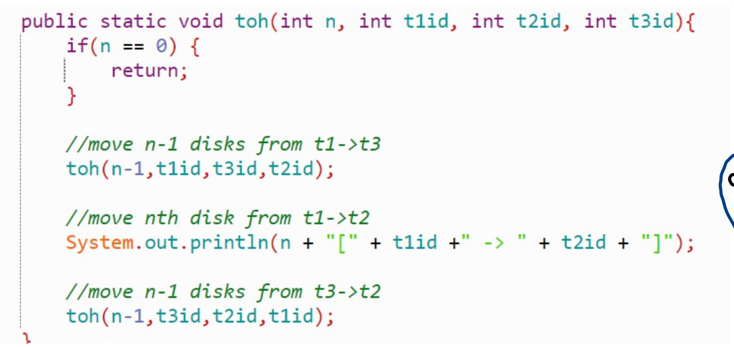
✓ 3 [10 -> 11]

✓ 1 [12 -> 10]

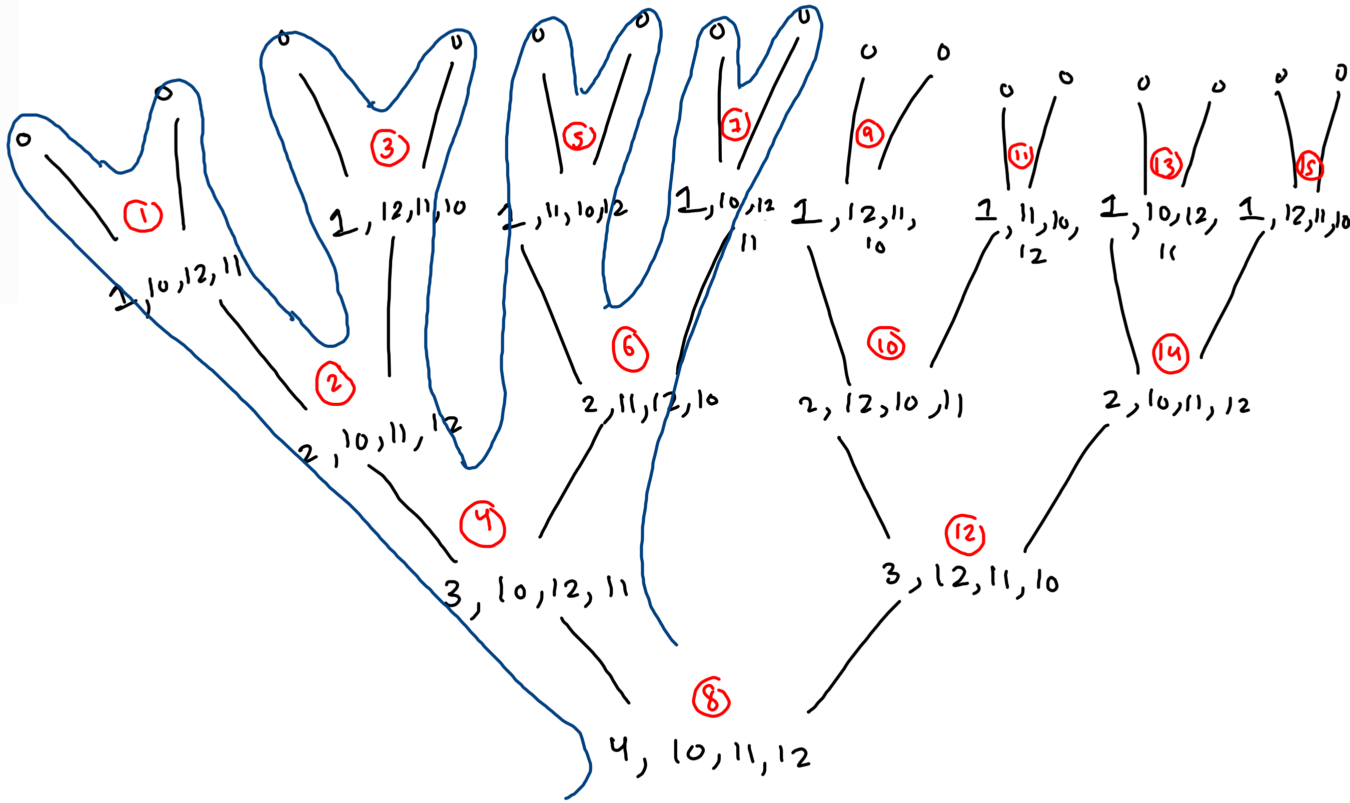
✓ 2 [12 -> 11]

✓ 1 [10 -> 11]





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

    //move n-1 disks from t1->t3
    toh(n-1,t1id,t3id,t2id);

    //move nth disk from t1->t2
    System.out.println(n + "[" + t1id +" -> " + t2id + "]");

    //move n-1 disks from t3->t2
    toh(n-1,t3id,t2id,t1id);
}
```



1.	1	[10 -> 12]
2.	2	[10 -> 11]
3.	1	[12 -> 11]
4.	3	[10 -> 12]
5.	1	[11 -> 10]
6.	2	[11 -> 12]
7	1	[10 -> 12]
8	4	[10 -> 11]