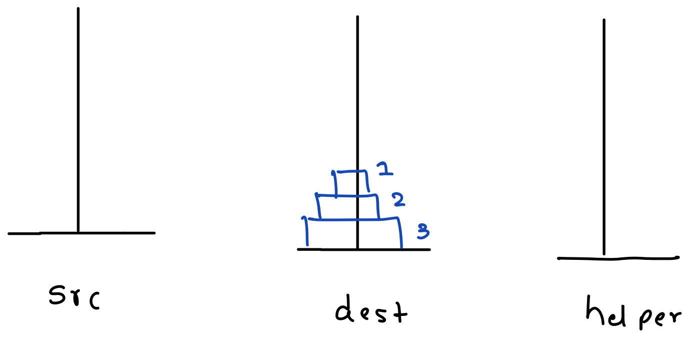
Q.1 Tower of Hanoe

Given n disks and 3 towers (src, dest and helper). Transfer all disks from src to destination and print instructions by keeping the following rules in mind :

1) we can move only one disk at a time.
2) big disk can't be placed over a small disk.
3) we can only move top-most disk from a tower.

n = 3



move 1 from s to d

move 2 from s to h

move 1 from d to h

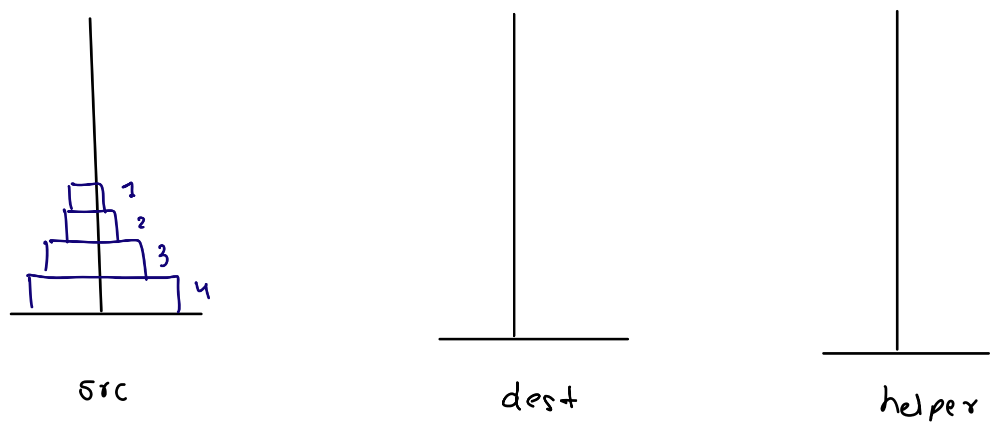
move 3 from s to d

move 1 from h to s

move 2 from h to d

move 1 from s to d

n=4



1st agenda : transfer 1st 3 disks from src to helper
↳ 7 steps

2nd agenda : transfer 4th disk from src to dest
↳ 1 step

3rd agenda : transfer 1st 3 disks from hel to dest.
↳ 7 steps

toh (n, s, d, h) ⇒ transfer n disks from s to d

└ transfer n-1 disks from s to h

```
toh (n-1, s, h, d)
```

└ transfer nth disk from s to d

```
sopln ("move" + n + "from" + s + "to" + d);
```

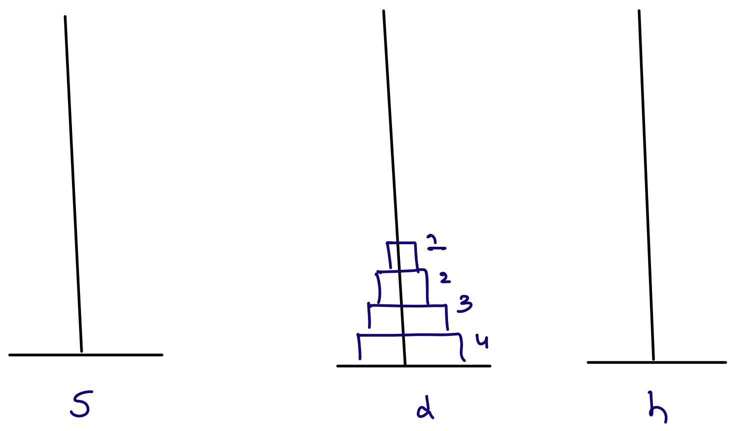
└ transfer n-1 disks from h to d

```
toh (n-1, h, d, s);
```

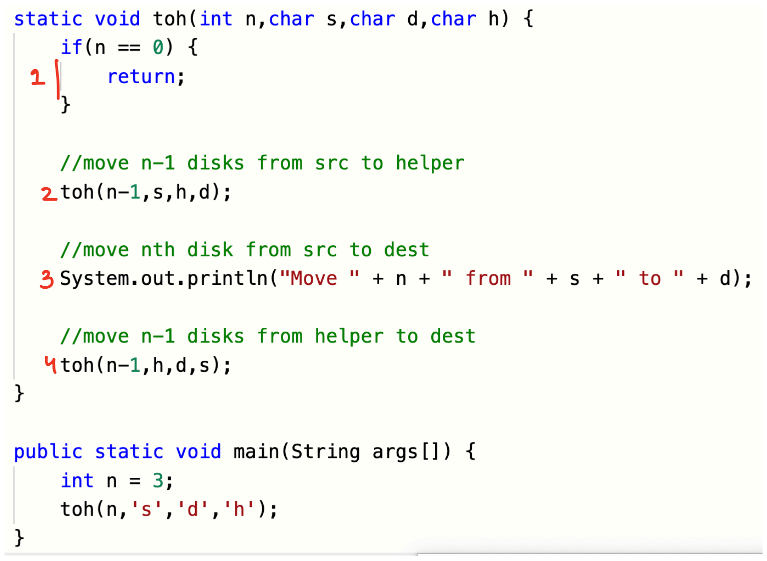
```
void  toh ( int n, char s, char d, char h ) {
    if (n==0) {
        return;
    }
    // move n-1 disks from src to helper
    toh (n-1, s, h, d);

    // move nth disk from src to dest
    sopln ("Move" + n + "from" + s + "to" + d);

    // move n-1 disks from helper to dest
    toh (n-1, h, d, s);
}
```

n = 4

```
Move 1 from s to h ✓
Move 2 from s to d ✓
Move 1 from h to d ✓
Move 3 from s to h ✓
Move 1 from d to s ✓
Move 2 from d to h ✓
Move 1 from s to h ✓
Move 4 from s to d ✓
Move 1 from h to d ✓
Move 2 from h to s ✓
Move 1 from d to s ✓
Move 3 from h to d ✓
Move 1 from s to h ✓
Move 2 from s to d ✓
Move 1 from h to d ✓
```



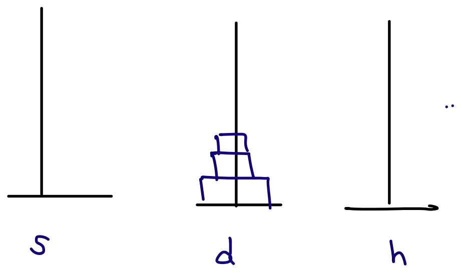
Dry run



```
static void toh(int n,char s,char d,char h) {
    if(n == 0) {
1 |     return;
    }

    //move n-1 disks from src to helper
  2 toh(n-1,s,h,d);

    //move nth disk from src to dest
  3 System.out.println("Move " + n + " from " + s + " to " + d);

    //move n-1 disks from helper to dest
  4 toh(n-1,h,d,s);
}

public static void main(String args[]) {
    int n = 3;
    toh(n,'s','d','h');
}
```

$n = 3$

$$\begin{tikzcd}
n-1, s, h, d & & n-1, h, d, s \\
& n, s, d, h \arrow[ul, dash] \arrow[ur, dash] &
\end{tikzcd}$$



move 1 from s to d

move 2 from s to h

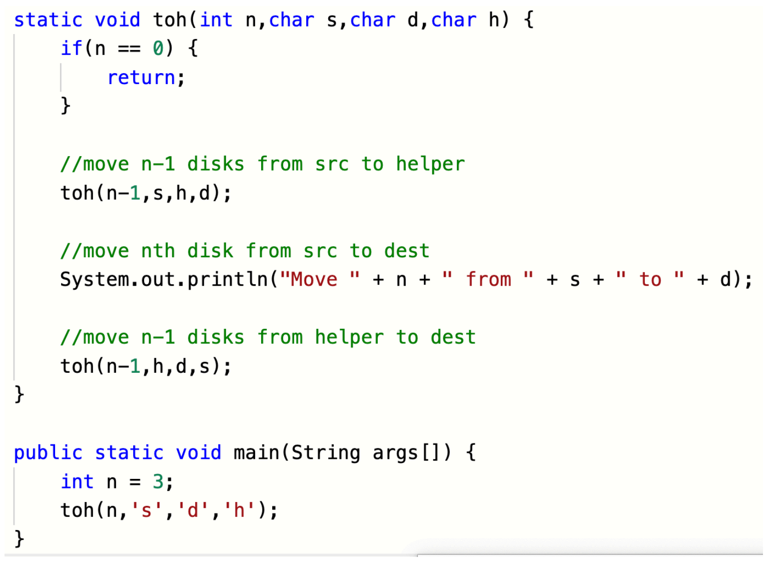
move 1 from d to h

move 3 from s to d

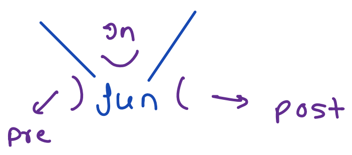
move 1 from h to s

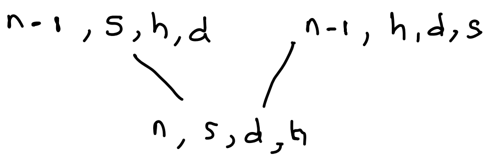
move 2 from h to d

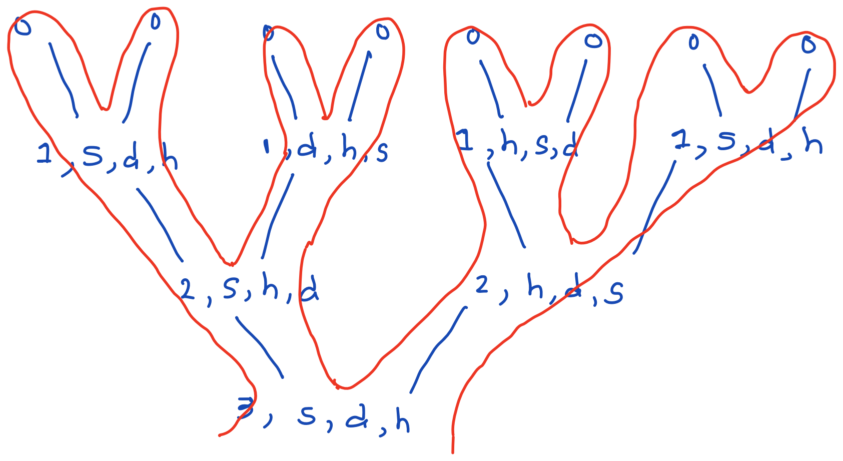
move 1 from s to d



```java
static void toh(int n,char s,char d,char h) {
    if(n == 0) {
        return;
    }

    //move n-1 disks from src to helper
    toh(n-1,s,h,d);

    //move nth disk from src to dest
    System.out.println("Move " + n + " from " + s + " to " + d);

    //move n-1 disks from helper to dest
    toh(n-1,h,d,s);
}

public static void main(String args[]) {
    int n = 3;
    toh(n,'s','d','h');
}
```







move 1 from s to d
move 2 from s to h
move 1 from d to h
move 3 from s to d
move 1 from h to s
move 2 from h to d
move 1 from s to d

todo: count of steps

space complexity

↳ function frames are getting stored inside call stack.

TC : (TC of single function * total no. of function calls)

SC : (SC of single function * max no. of function calls in call stack at any point of time)

Examples

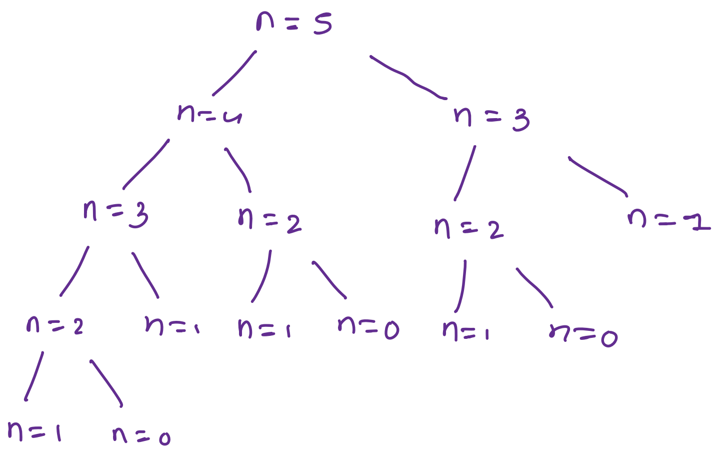
```
int fact (int n) {
    if (n==0) {
        return 1;
    }
    int temp= fact (n-1);
    return temp*n;
}
```

$$\begin{tikzcd}
n=5 \arrow[d] \\
n=4 \arrow[d] \\
n=3 \arrow[d] \\
n=2 \arrow[d] \\
n=1 \arrow[d] \\
n=0
\end{tikzcd}$$

TC: $O(n)$

SC: $O(n)$

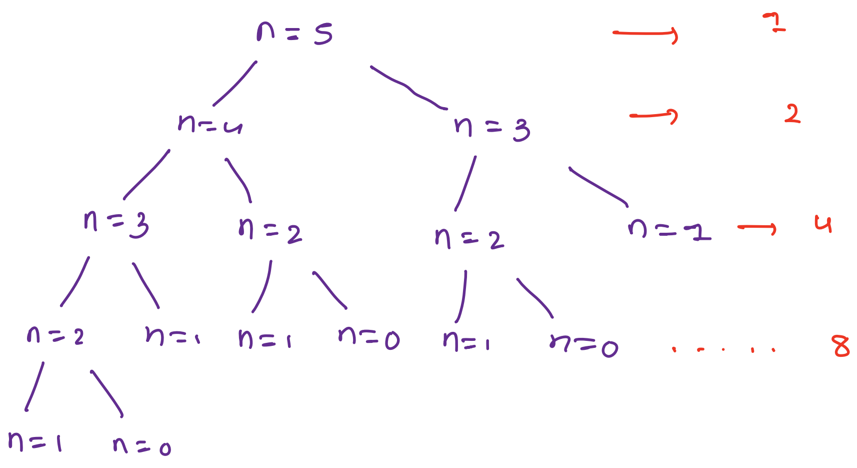
```
int fib (int n) {
    if( n==0 || n==1) {
        return n;
    }
    int temp1= fib(n-1);
    int temp2= fib(n-2);
    return temp1 + temp2;
}
```



max no. of function calls in call stack at any point of time) $\Rightarrow$ length of longest branch

TC: $O(2^n)$

SC: $O(n)$



total calls = 1 + 2 + 4 + ...

$$S_t = \frac{a(r^t - 1)}{r - 1}$$

$a = 1$
$r = 2$
$t = n$

$$= \frac{1(2^n - 1)}{1} \approx 2^n$$

```
int pow (int a, int n) {
    if (n = = 0) {
        return 1;
    }
    int temp = pow(a, n-1);
    return temp*a;
}
```

$$\begin{tikzcd}[row sep=small]
3,4 \arrow[d] \\ 3,3 \arrow[d] \\ 3,2 \arrow[d] \\ 3,1 \arrow[d] \\ 3,0
\end{tikzcd}$$

TC : O(n)

SC : O(n)

```
int pow (int a, int n) {
    if(n==0) {
        return 1;
    }
    int temp = pow(a, n/2);
    if (n % 2 = = 0) {
        return temp * temp;
    }
    else {
        return temp * temp * a;
    }
}
```

$$\begin{tikzcd}[row sep=small]
3,20 \arrow[d] \\ 3,10 \arrow[d] \\ 3,5 \arrow[d] \\ 3,2 \arrow[d] \\ 3,1 \arrow[d] \\ 3,0
\end{tikzcd}$$

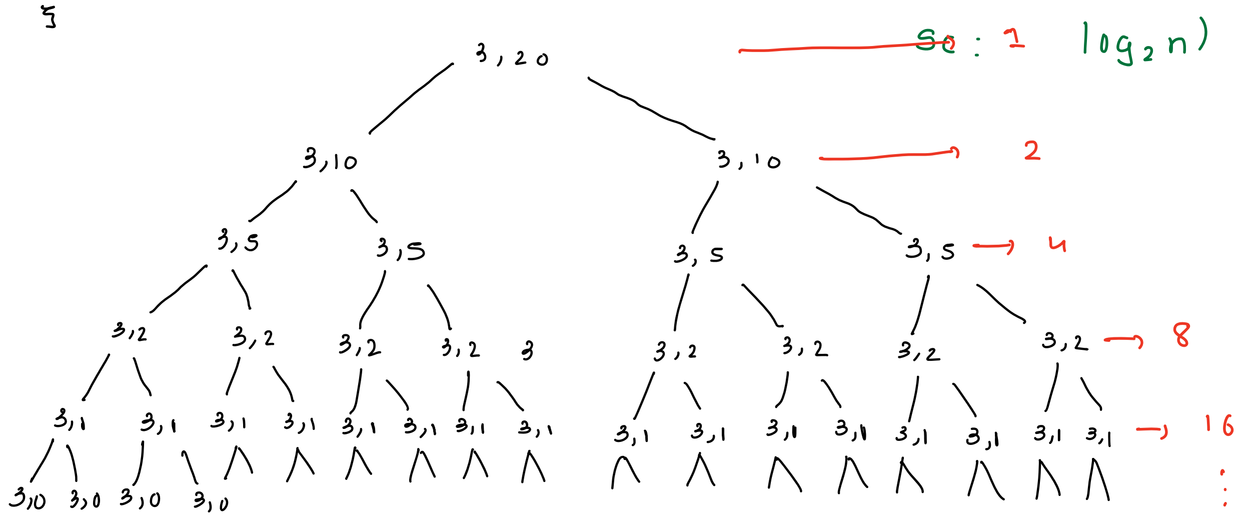
TC : $O(\log_2 n)$

SC : $O(\log_2 n)$

```
int pow (int a, int n) {
    if(n==0) {
        return 1;
    }

    if (n % 2 == 0) {
        return pow(a, n/2) * pow(a, n/2);
    }

    else {
        return pow(a, n/2) * pow(a, n/2) * a;
    }
}
```

TC : O(n)

SC : O($\log_2 n$)



total calls = 1 + 2 + 4 + 8 .....

$$S_t = \frac{a(r^t - 1)}{r - 1}$$

$a = 1$

$r = 2$

$t = \log_2 n$

$$= \frac{1(2^{\log_2 n} - 1)}{1} = 2^{\log_2 n} - 1 \quad \{2^{\log_2 n} = n\}$$

$$\approx n$$

Doubts

kth element problem

$0 \rightarrow 01$

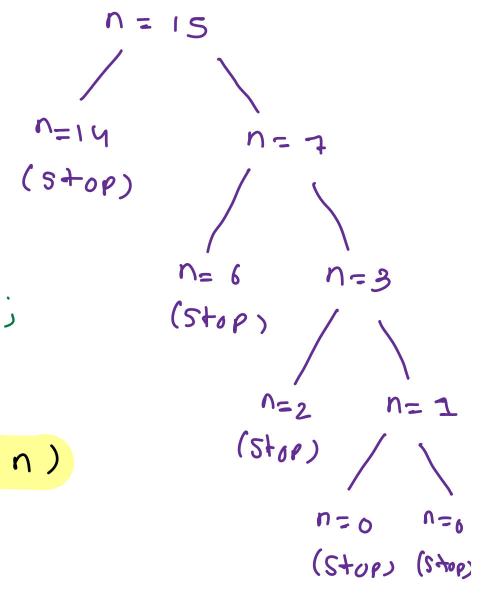
$1 \rightarrow 10$

$A = 4$

$$\begin{tikzcd}
A = 1 \arrow[dr, bend left] & \\
A = 2 \arrow[u] & temp = [0] \;,\; ans = [0,1] \\
A = 3 \arrow[u] & temp = [0,1] \;,\; ans = [0,1,1,0] \\
A = 4 \arrow[u] & temp = [0,1,1,0] \;,\; ans = [0,1,1,0,1,0,0,1]
\end{tikzcd}$$

```
int fun(int n) {
    if (n%2 == 0) {
        return 0;
    }
    return fun(n-1) + fun(Math.floor(n/2));
}
```

TC: $O(\log_2 n)$



n = 15
n=14
(stop)
n= 7
n= 6
(stop)
n=3
n=2
(stop)
n= 1
n=0
(stop)
n=0
(stop)