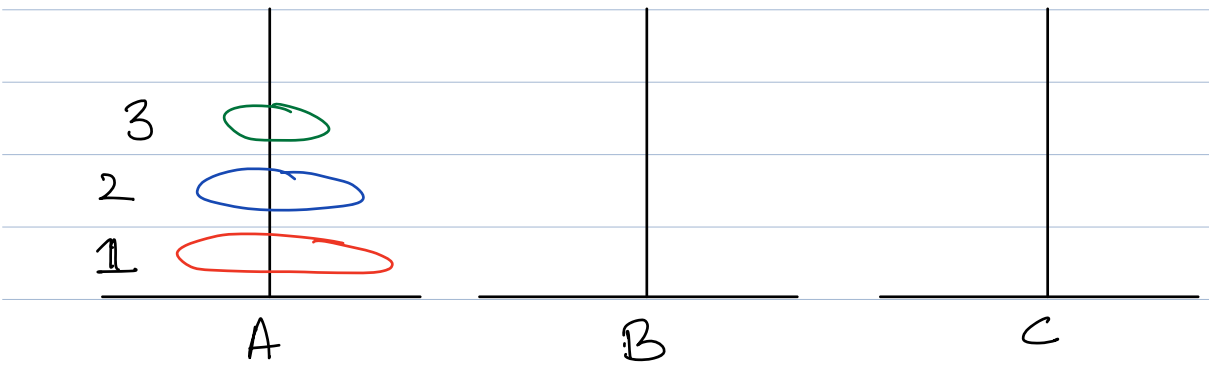
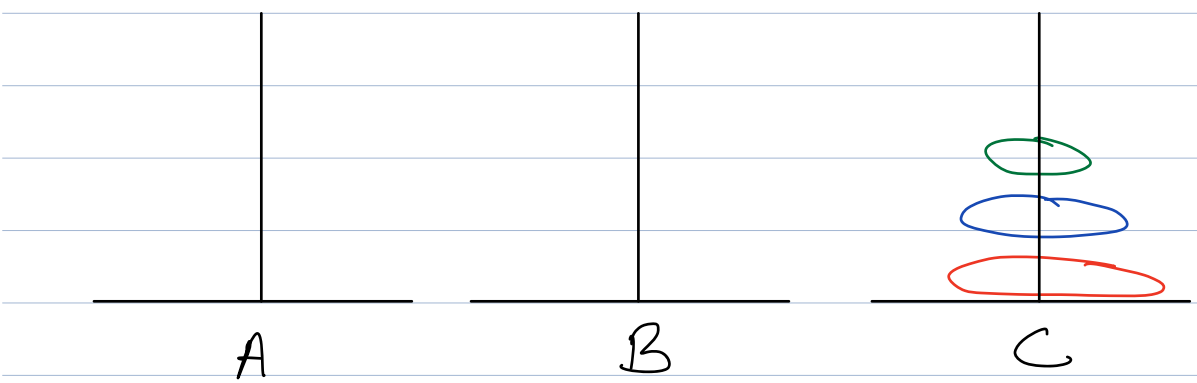
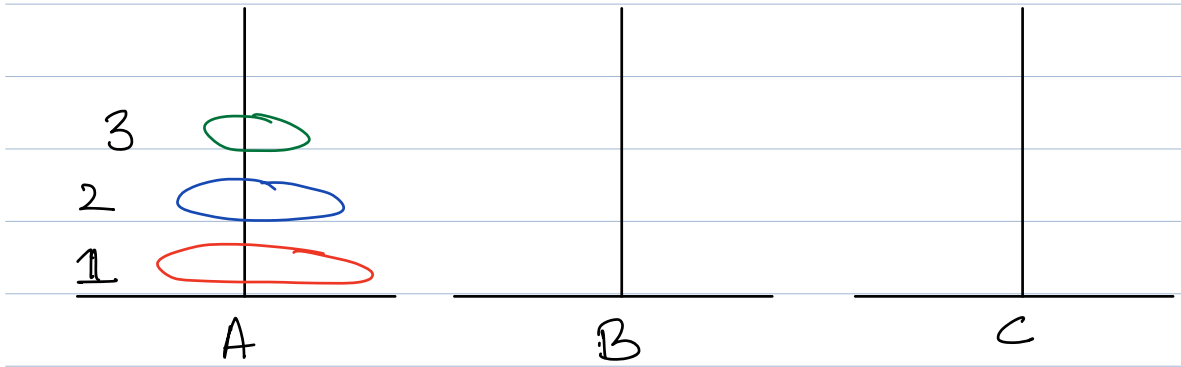
Tower of Hanoi

Q Shift the disc from A to C. You can use B.
Bigger disc cannot be on top of a smaller disc at any point. You can move only 1 disc at a time.

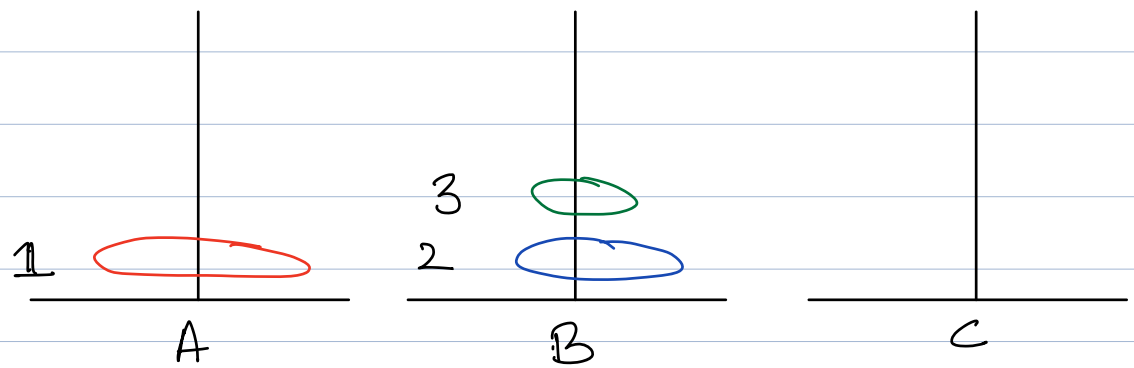




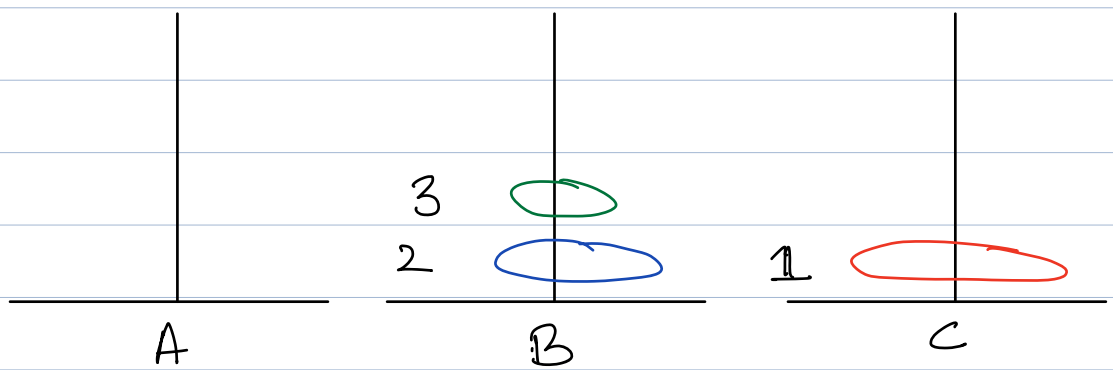
Approach 1



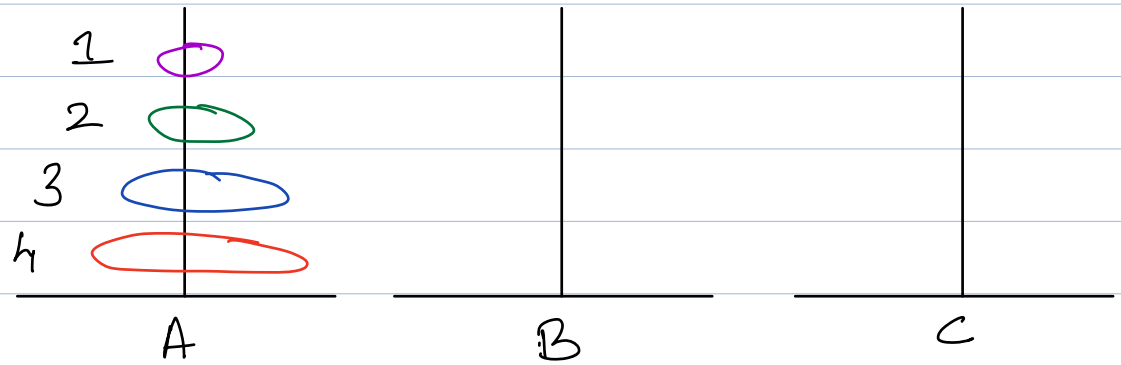
3 → AC , 2 A → B , 3 C → B

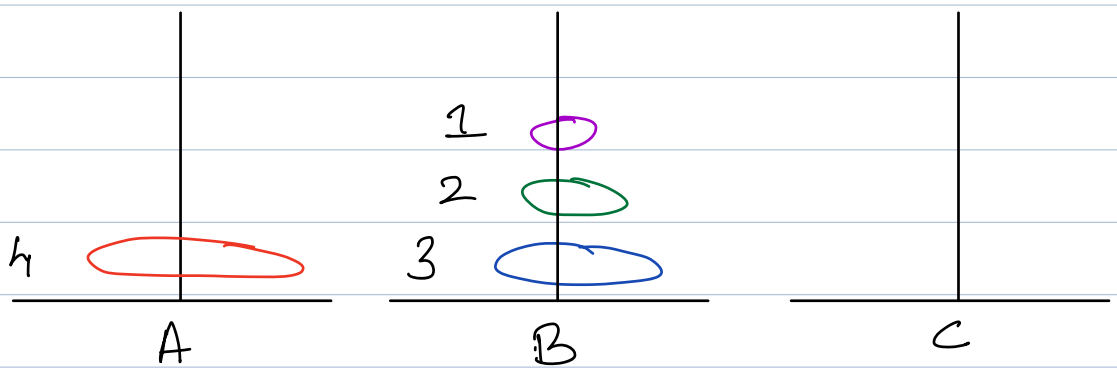


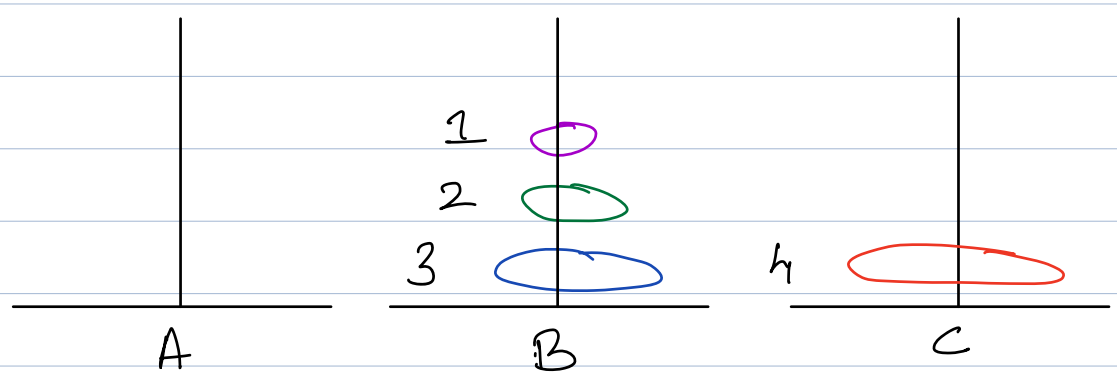
1 A → C

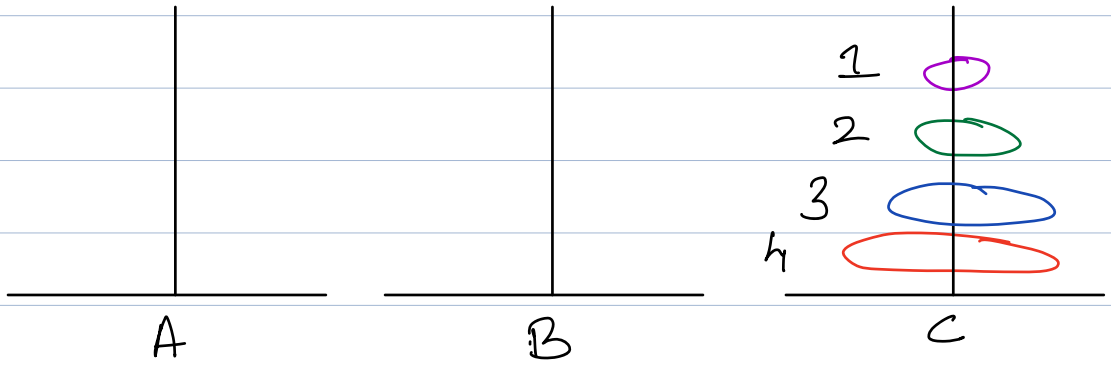


3 B→A, 2→B→C, 3A→C


1
2
3
4
A
B
C


4
1
2
3
A
B
C


1
2
3
4
A
B
C



Q n discs. You need to move from Tower A to Tower C using Tower B. Print all the steps.

1 2 3 4 5

1) Manifestation

tower (n) ⇒ solves for n discs ✗

tower (int n, String source, String destination, String helper)

2) Main logic

1) Move (n-1) rings/discs to helper.

tower (n-1, source, helper, destination)

2) Move n^{th} ring to destination.

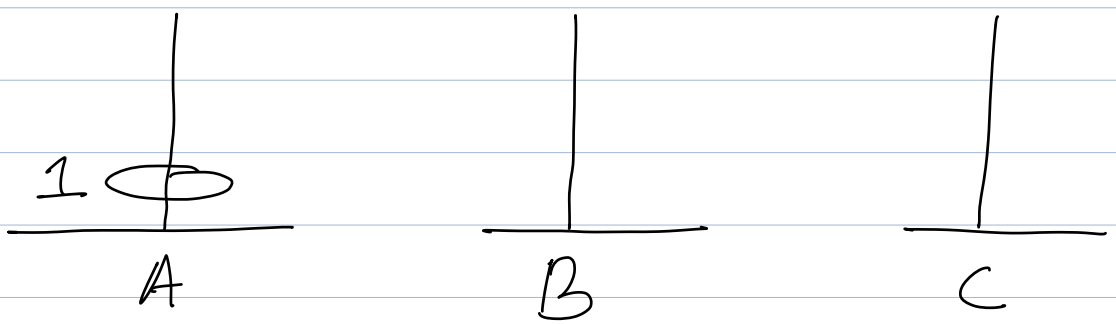
point (n: source destination)

3) Move (n-1) rings from helper to destination.

tower (n-1, helper, destination, source)

3) Base Case

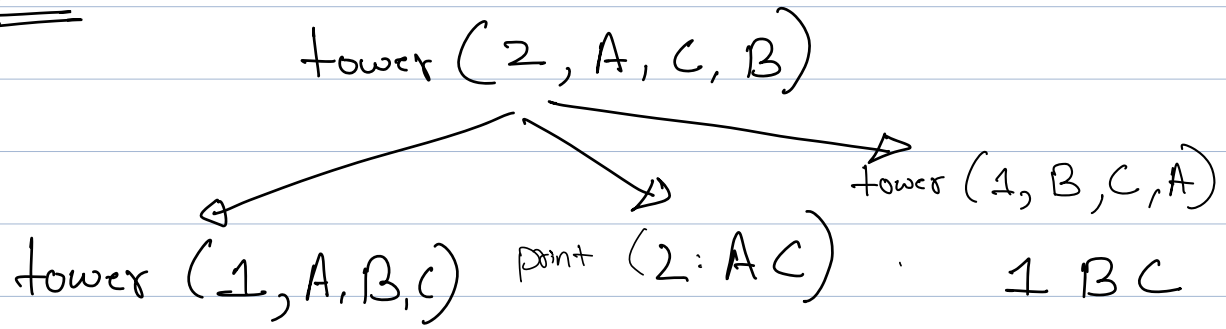
n = 1

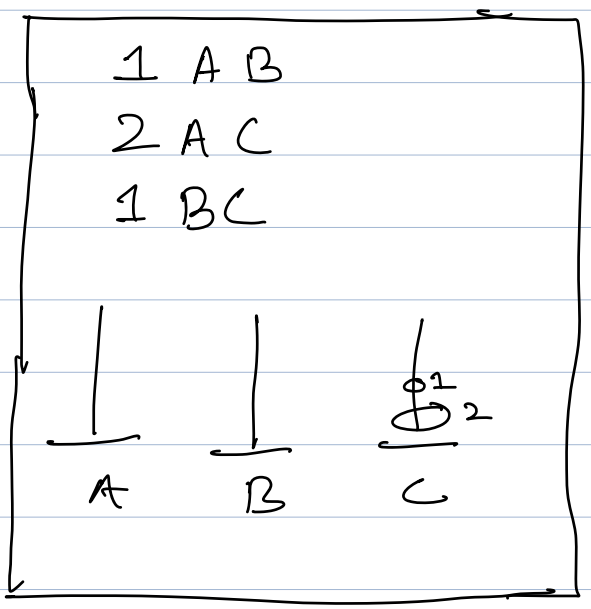


Point 1: A C

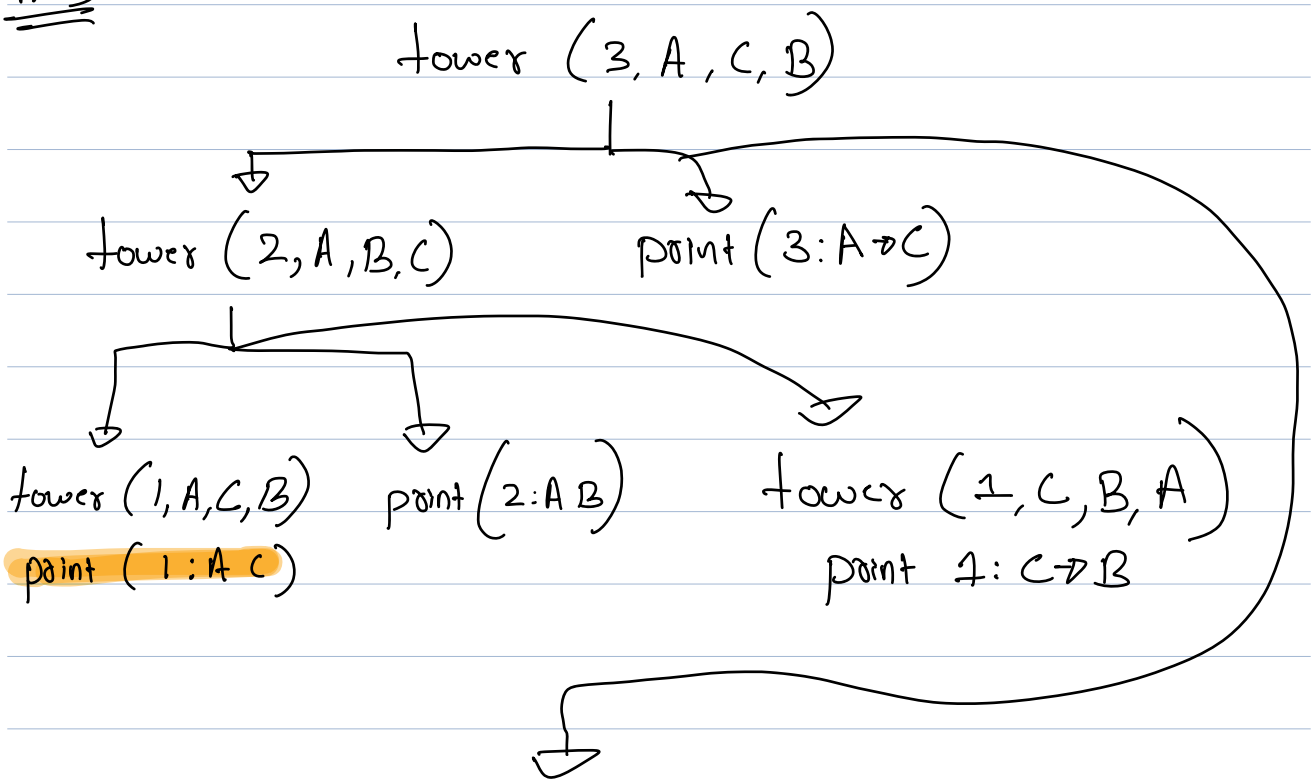
```
void tower (int n, String source, String destination,
                                          String helper)
{   if (n == 1) {
        print (1: source destination)      // Base
        return;                                Case
    }

    tower (n-1, source, helper, destination)

    print ( n: source destination)

    tower (n-1, helper, destination, source)

}
```

$n = 2$


tower (2, A, C, B)
tower (1, B, C, A)
tower (1, A, B, C)
print (2: A C)
1 B C


1 A B
2 A C
1 B C
1
2
A
B
C

n=3



1:AC , 2AB , 1C-B , 3A→C , 1B→A

2BC | 1AC

Time Complexity.

$T(n) \Rightarrow 1 + 2T(n-1)$

$T(n) \Rightarrow 1 + 2(1 + 2T(n-2))$

$\Rightarrow 3 + 4T(n-2)$

$T(n) \Rightarrow 7 + 8T(n-3)$

$T(n) \Rightarrow 2^k - 1 + 2^k T(n-k)$ $\quad T(1) = 1$

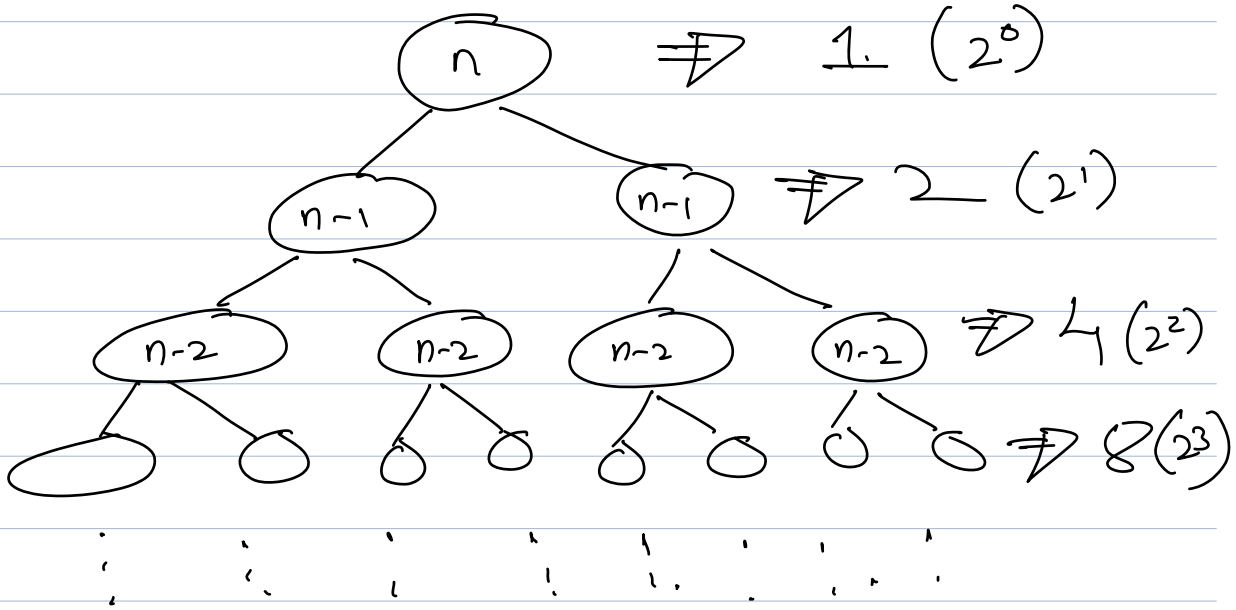
$n - k = 1$

$n - 1 = k$

$\Rightarrow 2^{n-1} - 1 + 2^{n-1}$

$\Rightarrow 2^n$

TC : $O(2^n)$



$\Rightarrow 1\ (2^0)$

$\Rightarrow 2\ (2^1)$

$\Rightarrow 4\ (2^2)$

$\Rightarrow 8\ (2^3)$

(1) (2) (3) (4) (5) (6) (7) $\Rightarrow 2^{n-1}$

$2^0 + 2^1 + 2^2 \ldots 2^{n-1} \Rightarrow 2^n - 1$

TC: $2^n \times O(1)$

$= 2^n$

Height of the tree $\Rightarrow n$

SC: $O(n)$

10:33pm

K^{th} Symbol

```
    0 1 2 3
1   0
2   0 1
3   0 1 1 0
4   0 1 1 0 1 0 0 1
    0 1 2 3 4 5 6 7
```

$0 \Rightarrow 01$

$1 \Rightarrow 10$

Q Return the value at n^{th} row and k^{th} column.

Ex:1 $n \Rightarrow 3, k \Rightarrow 2$

$\Rightarrow 1$

Ex2: $n \Rightarrow 4, k \Rightarrow 7$

$\Rightarrow 1$

Constraints

$1 \leq N \leq 10^5$

$0 \leq k \leq 10^{18}$

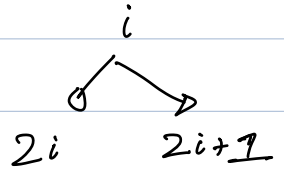
String(n) $\Rightarrow$ string(n-1) + toggle (string(n-1))

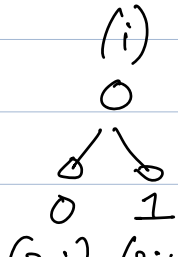
Observation

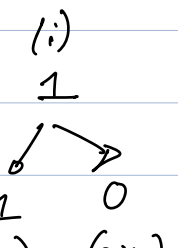
Val (n, k)

↓ parent

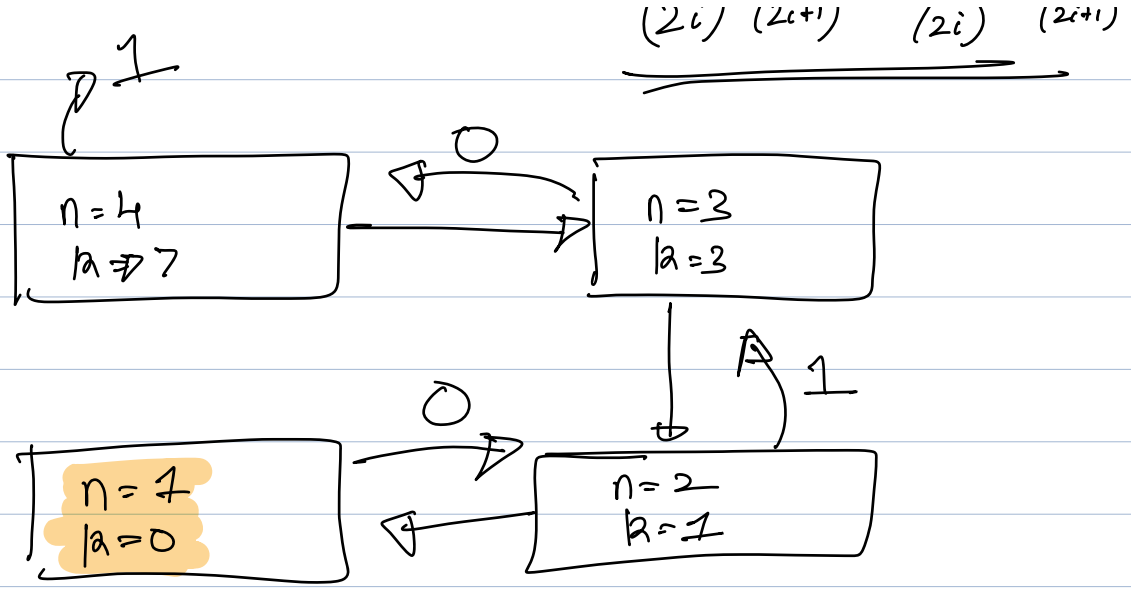
Val (n-1, $k/2$)







(2i) (2i+1) (2i) (2i+1)



1) Manifestation.

int solve (int n, int k)

↳ returns the value at nth row & kth column.

2) Main logic

1) Find parent

int parent = solve (n-1, k/2)

2) Find the value.

if (k%2==0) {

return parent;

} else {

return (1-parent);

}

1 ^ parent

~ parent

3 Base Case !

(i) if (n == 1)
    return 0;

(ii) if (k == 0)
    return 0;

```
int solve (int n, int k) {
    if (n==1 || k==0)
        return 0;

    int parent = solve (n-1, k/2);
    if (k % 2 == 0) {
        return parent;
    } else {
        return (1-parent);
    }

}
```

~~$T(n) \Rightarrow T(n-1) + 1$~~

$T(k) \Rightarrow T(k/2) + 1$

TC: $O(\log k)$

SC $\Rightarrow O(\log k)$

Ex 1

Time Comp $= O(n)$

$n = 10^5 / 10^6 / 10^7$
$k \Rightarrow 10$

$\Rightarrow O(n)$

$O(\log k)$