SCALER

Recursion - 2

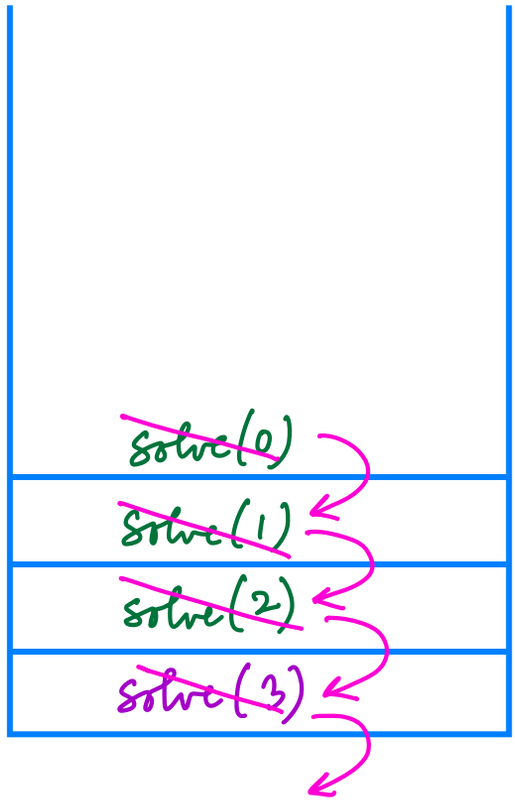
TABLE OF CONTENTS

- Power computation
- All indices of array

1

```
void solve (int N){
    if(N==0) {return}
    solve(N-1);
    print(N);
}
```

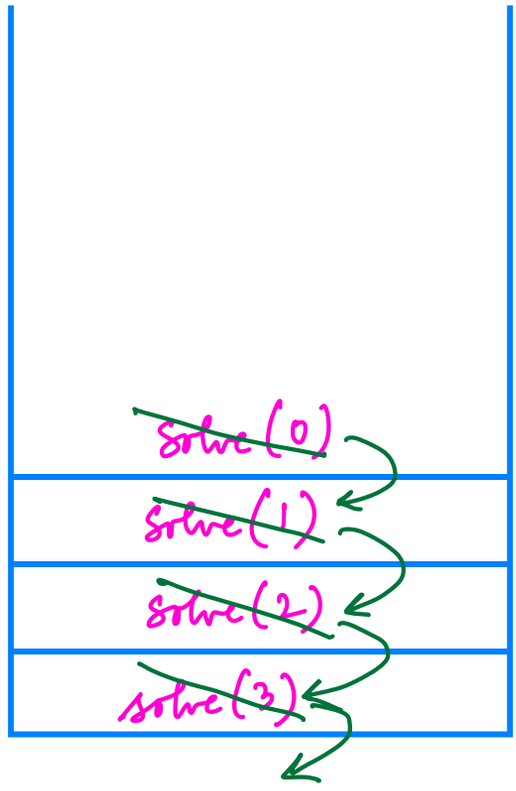
1 2 3



2

```
void solve (int N){
    if(N==0) {return}
    print(N);
    solve(N-1);
}
```

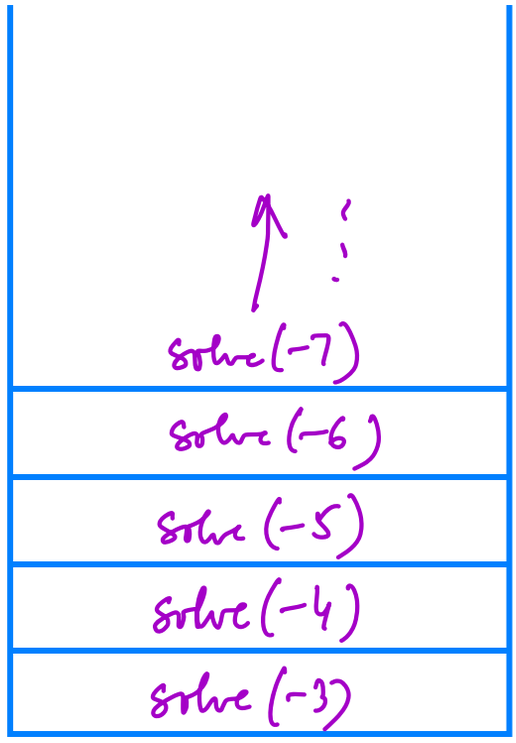
3 2 1



3

```
void solve (int N){
    if(N==0) {return}
    print(N);
    solve(N-1);
}
```

-3 -4 -5 -6 ....



Stack Overflow error
(Runtime Error)

Q) Given a and n (two integers, $n \geq 0$). find a^n using recursion.

$$a^n = a * a^{n-1}$$

$$pow(a,n) = a * pow(a,n-1)$$

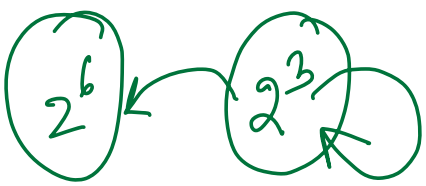
$$pow(a,0) = 1 .$$

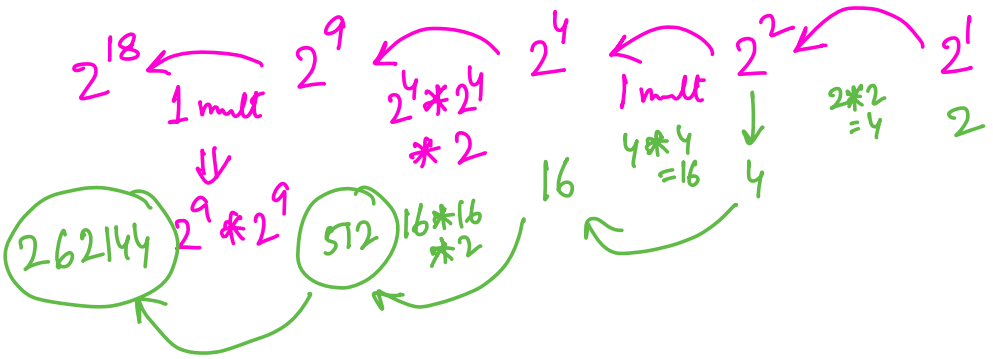
```
function pow(a, n) {
    if(n==0)
        return 1
    return a * pow(a,n-1)
}
```

O(n) T.C.
O(n) S.C.

$2^3 \xrightarrow{*2} \xrightarrow{*2} \xrightarrow{*2} 2^6$

$2^3 \xrightarrow{*2^3} 2^6$.



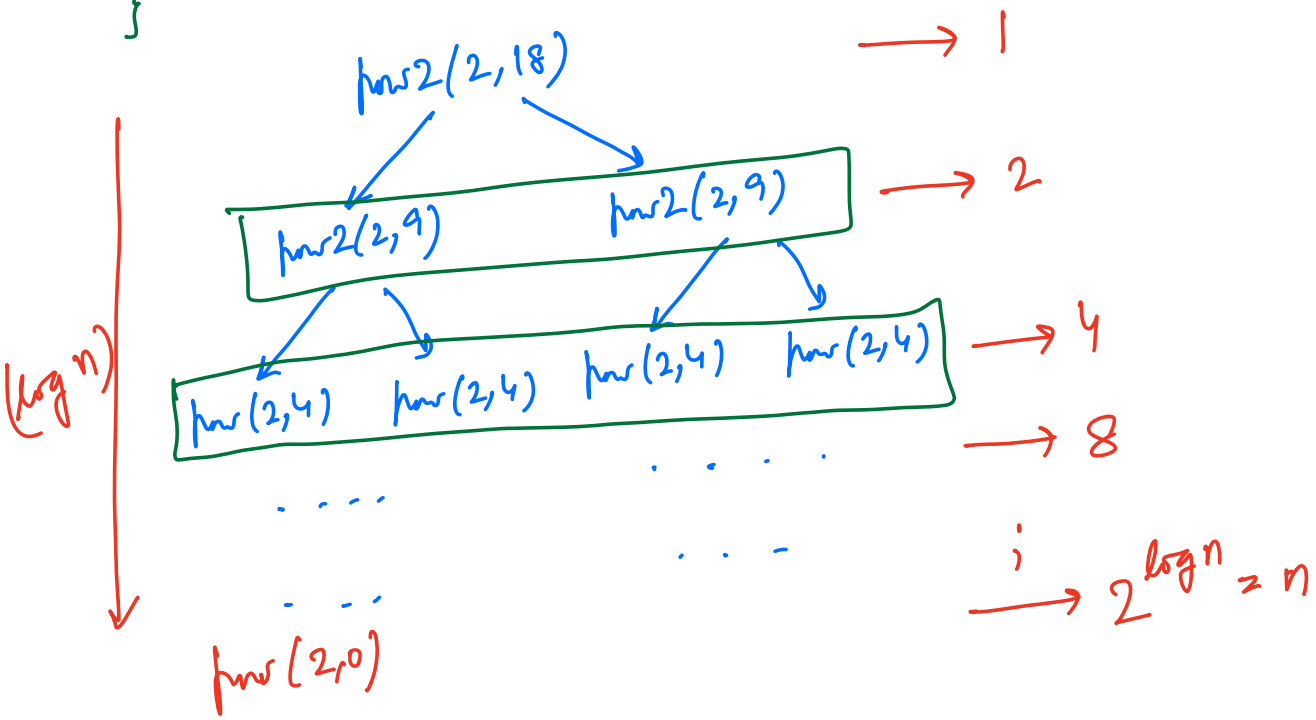


~ log(18) steps

pow(a,n) = pow(a, n/2) * pow(a, n/2) if n is even

pow(a, n) = pow(a, n/2) * pow(a, n/2) * a if n is odd

```
fn pow2(a,n){
    if (n==0)
        return 1
    if (n%2==0)
        return pow(a, n/2) * pow(a,n/2)
    return pow(a, n/2) * pow(a, n/2) * a
}
```



1 + 2 + 4 + 8 + ... + n

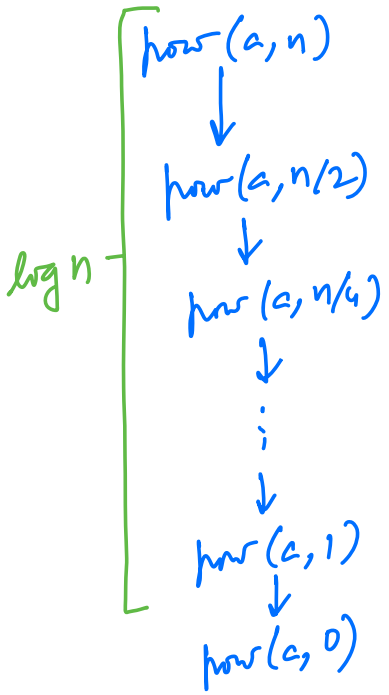
= O(n) T.C.

O(log n) S.C

pow(a,n) = pow(a, n/2) * pow(a, n/2) if n is even

pow(a,n) = pow(a, n/2) * pow(a, n/2) * a if n is odd

```
fn pow2(a,n){
    if (n==0)
        return 1
    half_pow = pow2(a, n/2)
    if (n%2==0)
        return half_pow * half_pow
    return half_pow * half_pow * a
}
```

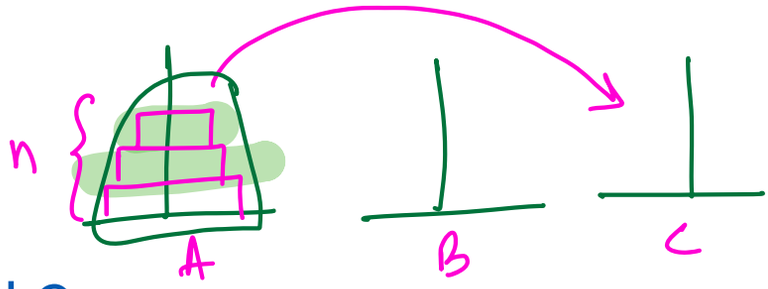


O(log n) T.C.

O(log n) S.C.

SCALER

Tower of Hanoi



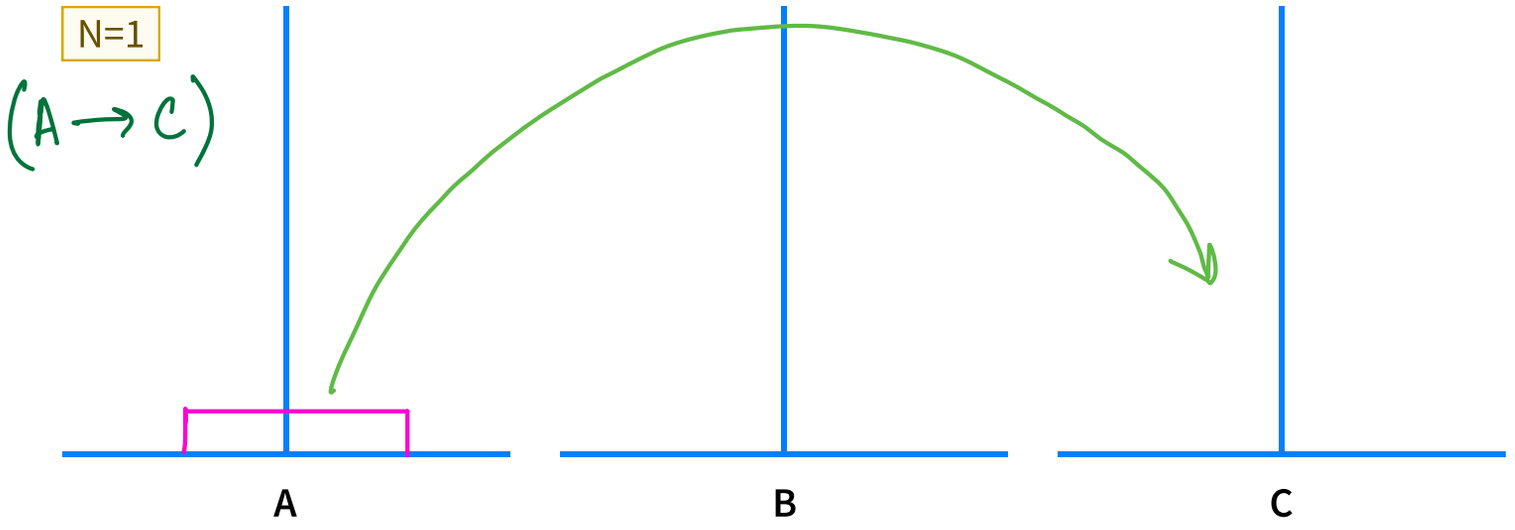
- Given 3 Towers A, B and C
- There are n-disks placed on tower A
- Move all the disks from A to C {using B}

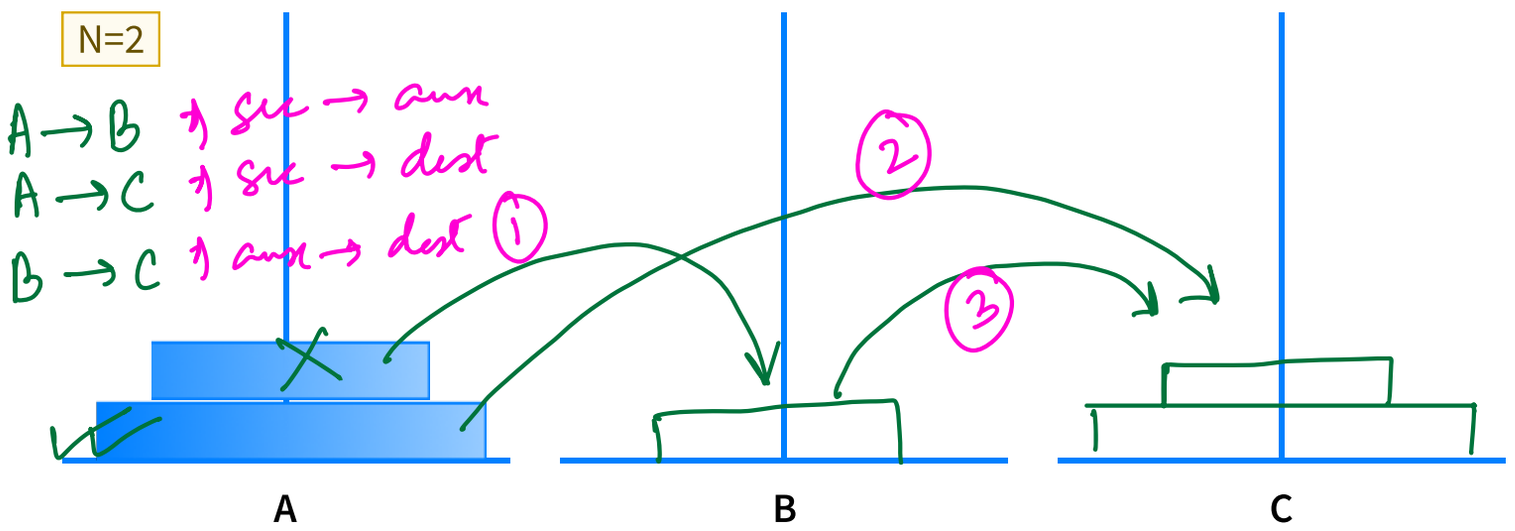
Note:

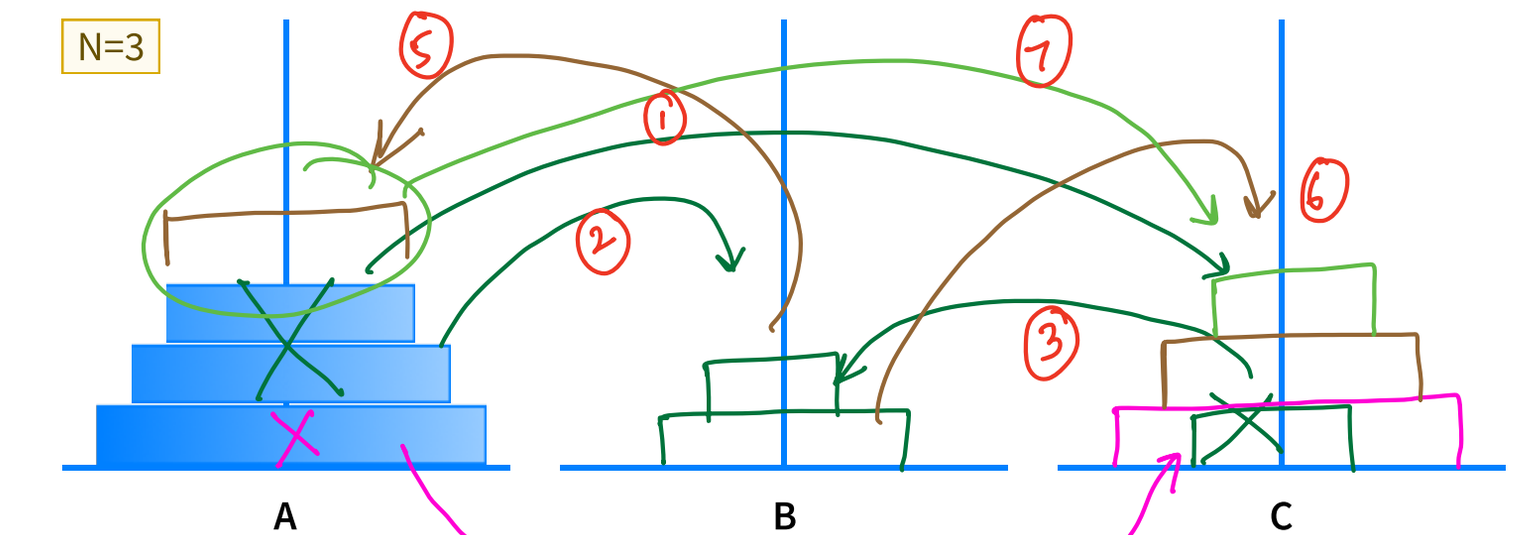
1. Only ~~ne~~ one disk can be moved at a time. (Topmost)
2. Larger disk can't be placed on a smaller disk.

Question

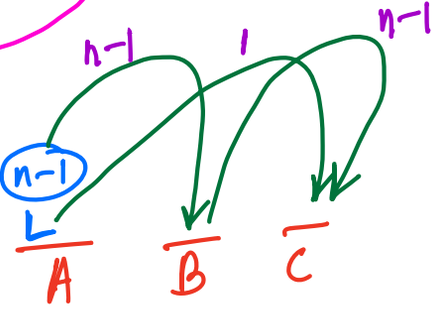
Print movement of the disks. [minimum steps]



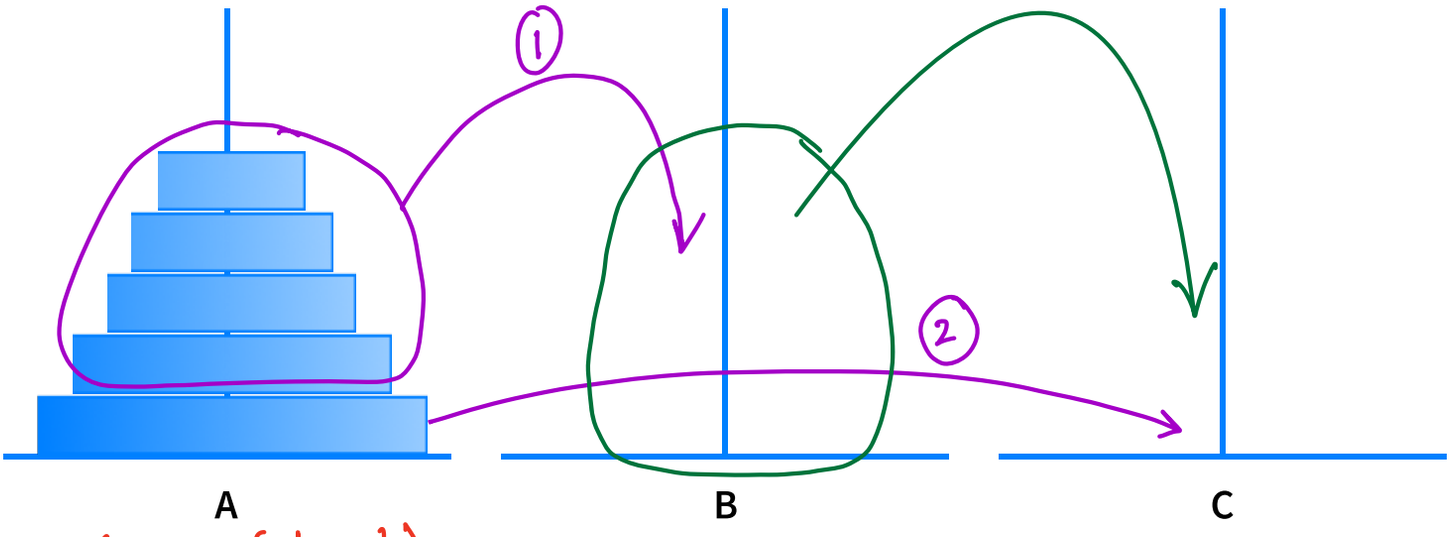




1. A → C
2. A → B
3. C → B
4. A → C
5. B → A
6. B → C
7. A → C



Approach for N disks



toh(N, 'A', 'C', 'B')

Move N-1 disks from ? ⟶ ? src(A) to aux(B) toh(N-1, 'A', 'B', 'C')

Move Nth disk from ? ⟶ ? src(A) to dest(C) print("A → C")

Move N-1 disks from ? ⟶ ? aux(B) to dest(C) toh(N-1, 'B', 'C', 'A')

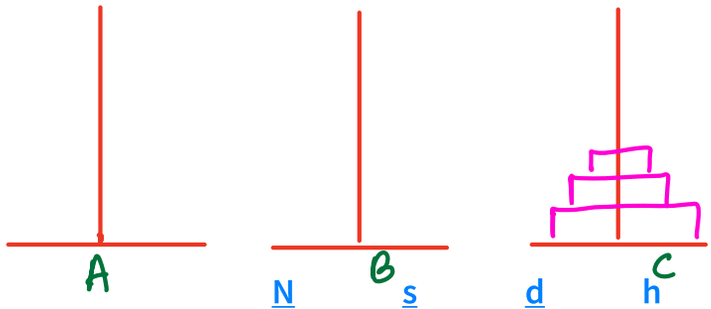
Assumption :- toh(n, s, d, h) will print the movement of steps to take n disks from s to d.

```
void toh(int N, char s, char d, char h){
    if(n==0)
        return
    toh(N-1, s, h, d)
    print(s + "->" + d)
    toh(N-1, h, d, s)
}
```

[Break till 10:46 PM]

SCALER

#dryrun



toh (3 , A , C , B)

- if(n==0) return
- ✓ toh(N-1, s, h, d)
- ✓ print(s + "->" + d)
- ✓ toh(N-1, h, d, s)

toh(2, A, B, C) — s, d, h

- if(n==0) return
- ✓ toh(N-1, s, h, d)
- ✓ print(s + "->" + d)
- ✓ toh(N-1, h, d, s)

toh(1, A, C, B) — s, d, h

- if(n==0) return
- ✓ toh(N-1, s, h, d)
- ✓ print(s + "->" + d)
- toh(N-1, h, d, s)

toh(0, A, B, C)

- if(n==0) return
- toh(N-1, s, h, d)
- print(s + "->" + d)
- toh(N-1, h, d, s)

toh(0, B, C, A)

toh(1, C, B, A)

toh(2, B, C, A) — s, d, h

- toh(N-1, s, h, d)
- print(s + "->" + d)
- toh(N-1, h, d, s)

toh(1, B, A, C)
B → A

toh(1, A, C, B)

A → C
A → B
C → B
A → C
B → A
B → C
A → C

T.C Analysis

n	# steps	
1	1	2^1-1
2	3	2^2-1
3	$3+1+3=7$	2^3-1
4	$7+1+7=15$	2^4-1
5	31	2^5-1
6	63	2^6-1
7	127	2^7-1
⋮		
n		2^n-1

$TC \rightarrow O(2^n)$

$SC \rightarrow$ max stack size

$O(n)$.

Q) Given an arr[n] and target B, find all indices at which B occurs in the array.

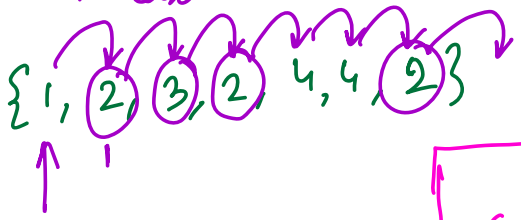
arr → {4, 5, 3, 1, 5, 4, 5}
(1, 4, 6)

B = 5

o/p → {1, 4, 6}.

index (idx) → 1 param of rec. function.

idx = 0 → idx = 1 → idx = 2 → ... → idx = n-1



B = 2

{1, 3, 6}

recur(arr, B, idx, list)

recur(arr, B, idx+1, list)

Base case
end of array → if(idx == n)
return;

Print Valid Parenthesis

→ Given N. Print all valid parenthesis of length 2N

→ No. of opening & closing brackets are same
→ balanced

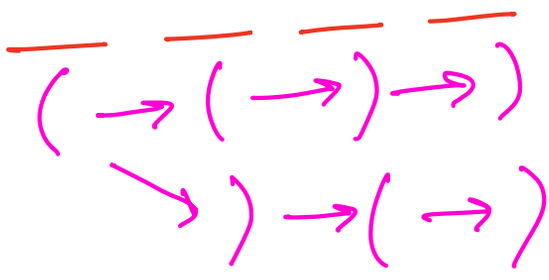
N=1 ( ) ,) (

N=2 ( )( ) , (( )) ( )) ( ))(( )

N=3 ( )( )( ) , (( )( )) , (( ))( ) , (( )) , ( )(( ))

$\frac{(2n)!}{n!\,(n+1)!}$ C_n

✦ Idea →



1. If open brackets are finished
open == n ⇓
cannot use '('.

2. If open == close till now
⇓
can't use ')'.

open_count
close_count
n

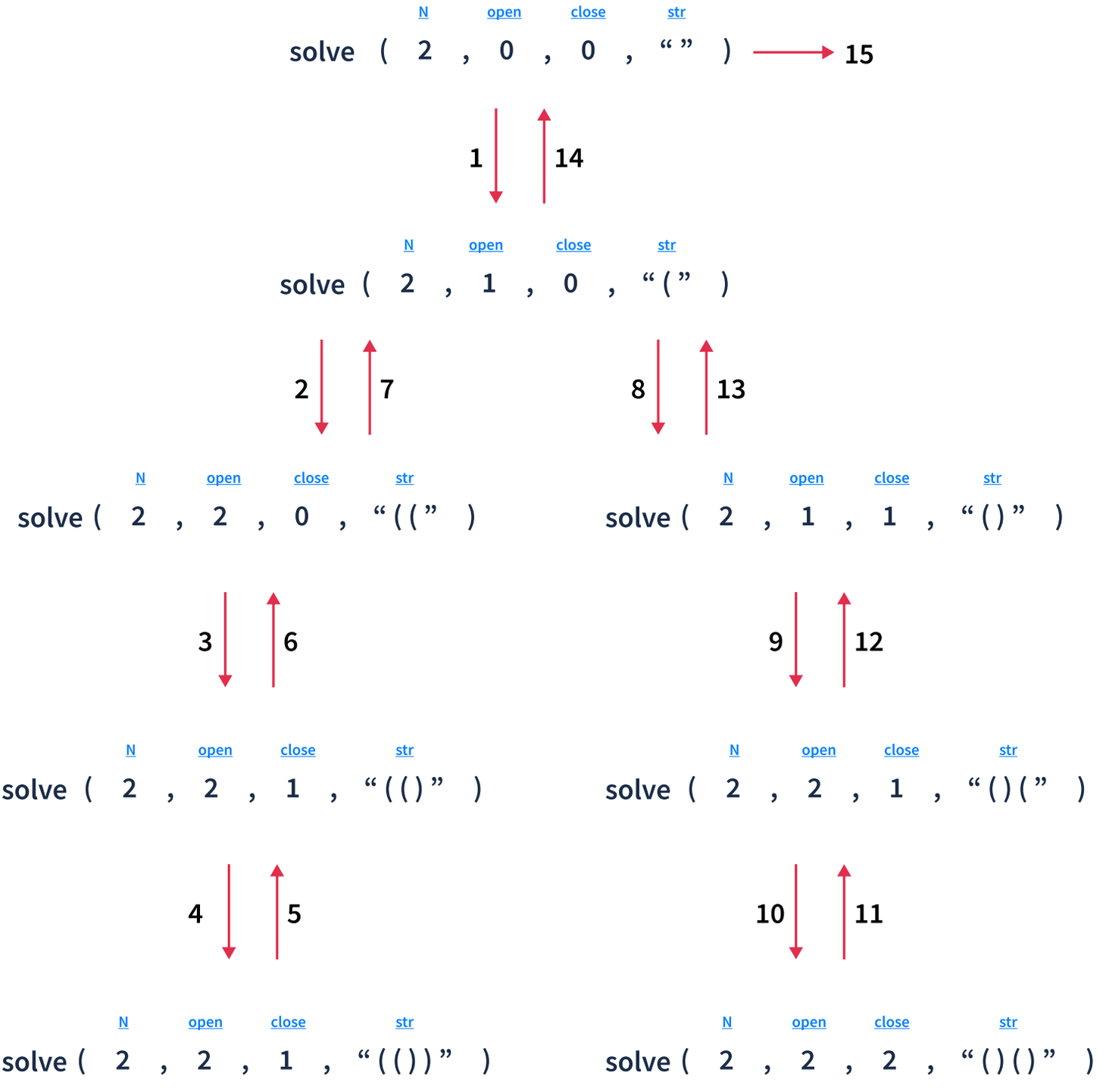
solve(n, 0, 0, "")

char[]

```
void solve(int N, int opening, int closing, string str){
        idx = opening + closing
        if(opening > closing){
                str[idx] = ')'
                solve(n, opening, closing+1, str)
        }
        if(opening < N){
                str[idx] = '('
                solve(n, opening+1, closing, str)
        }
}
```

if(idx == 2*N) → or closing == N
    Print(str)

#dryrun


N open close str
solve (2 , 0 , 0 , " ") → 15
1
14
N open close str
solve (2 , 1 , 0 , "(")
2
7
8
13
N open close str
solve (2 , 2 , 0 , "((")
solve (2 , 1 , 1 , "()")
3
6
9
12
N open close str
solve (2 , 2 , 1 , "(()")
solve (2 , 2 , 1 , "()(")
4
5
10
11
N open close str
solve (2 , 2 , 1 , "(())")
solve (2 , 2 , 2 , "()()")