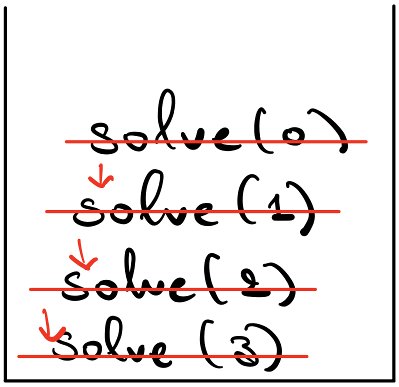
Quiz 1

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

N=3



1, 2, 3

Quiz 2

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

N=3

3, 2, 1

N = -3

Quiz 3

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

N = 3

-3 → -4 → -5 → -6 .....

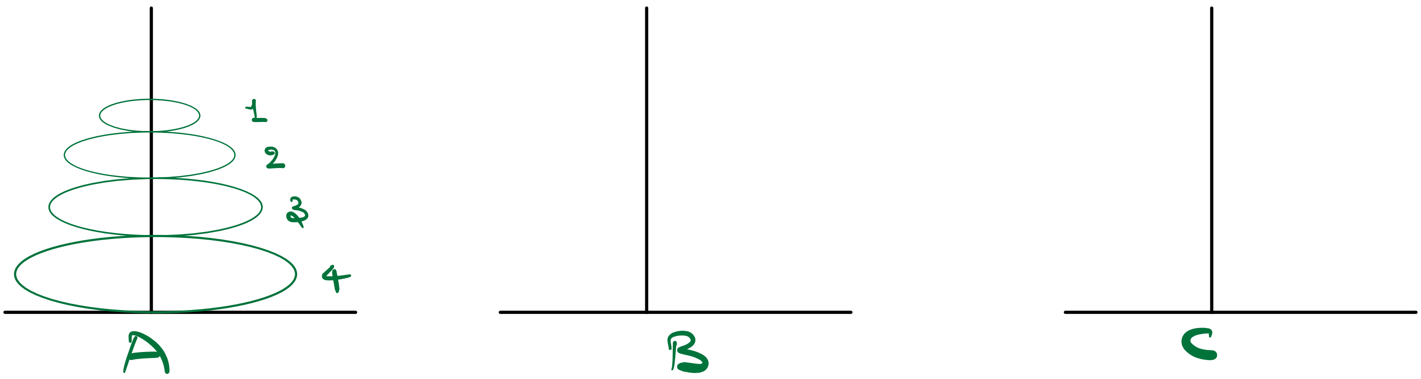
Tower of Hanoi

3 towers A, B & C

A ↓ source

C ↓ destination.

There are N disks of different sizes present on the source tower (A) arranged like smeller on top of bigger



Conditions. :-

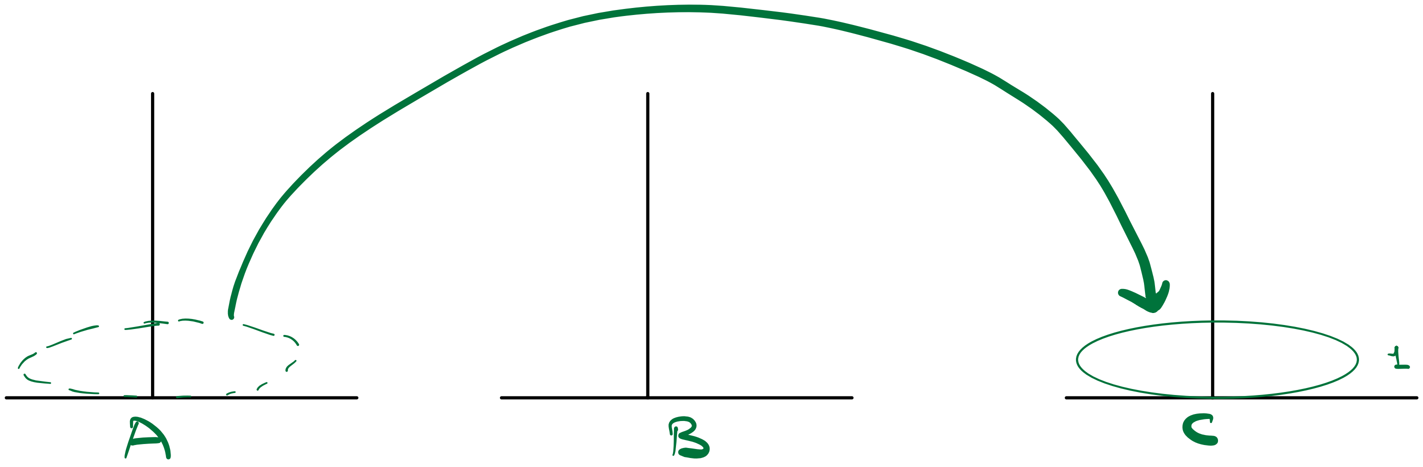
1) Only one disk can be moved in a step.

2) Larger disk cannot be placed on top of a smaller disk.

Print the movement of disks from A to C in minimum no. of steps.

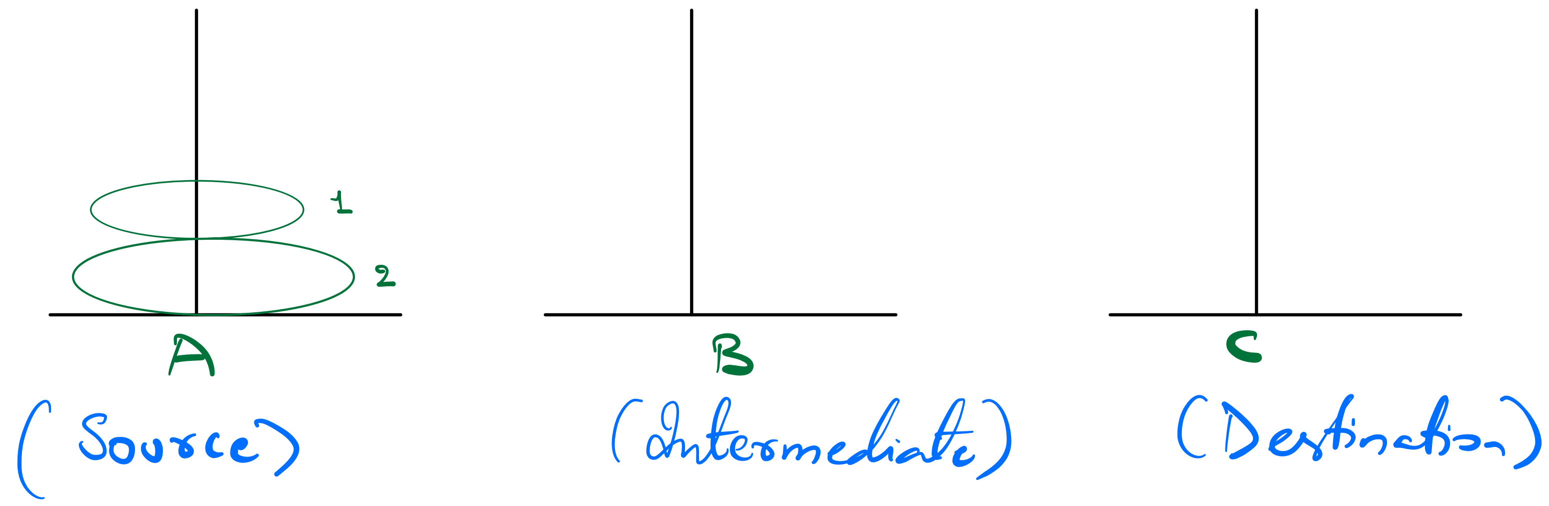
Sol^n

1) N = 1

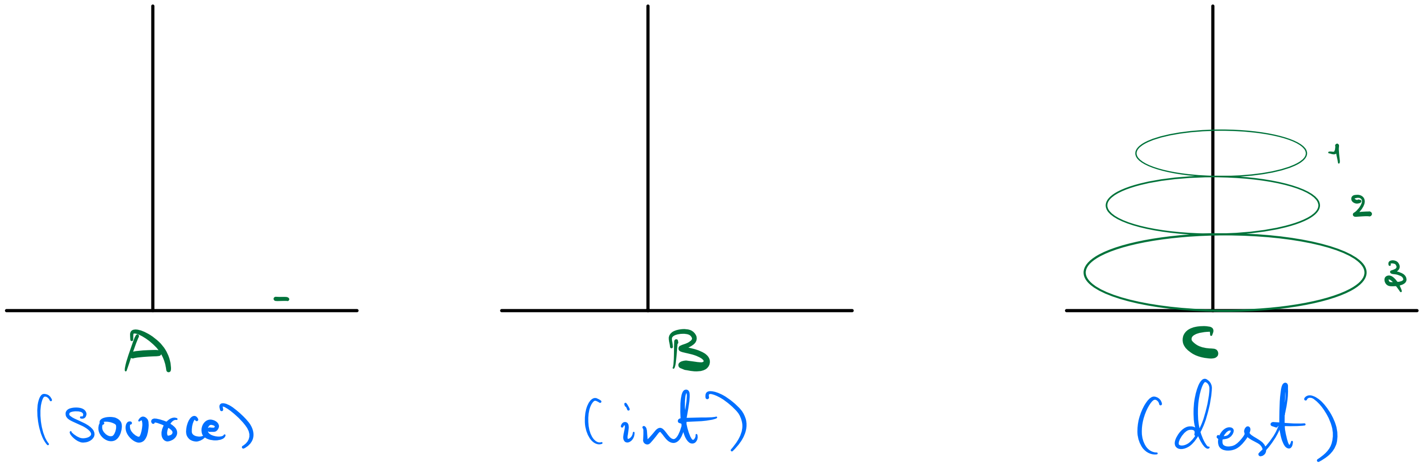


Ans = A → C

N=2



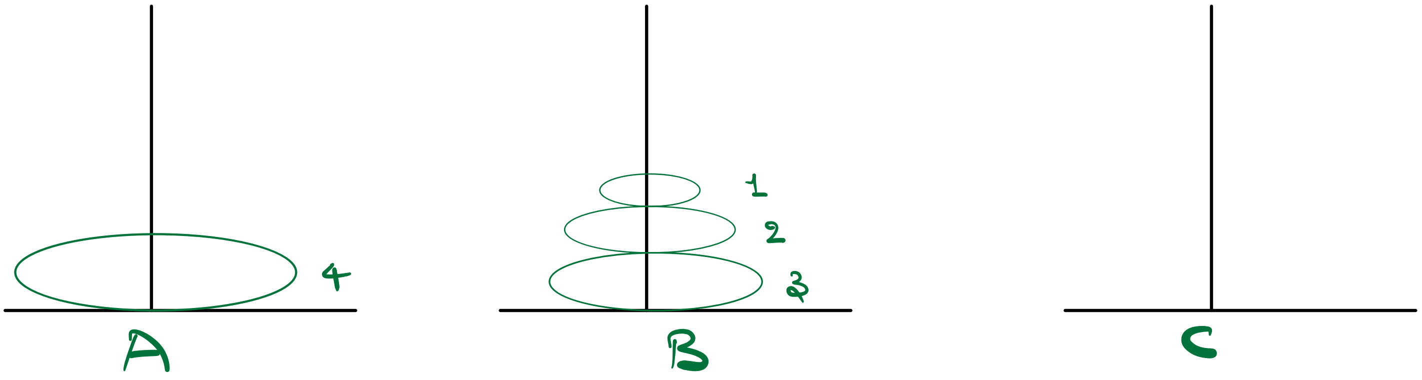
(src → int) A → B
(src → dest) A → C } 3 steps
(int → dest) B → C



src des int
3 rings from A to C using B.
⇓
src des int
(3) 2 rings from A to B using C
(1) 3rd ring from A to C using B
(3) 2 rings from B to C using A

7

N = 4



(N) 4 rings from $\overset{\text{src}}{A}$ to $\overset{\text{dest}}{C}$ using $\overset{\text{int}}{B}$

$\Downarrow$

(7) $\overset{(N-1)}{3}$ rings from $\overset{\text{src}}{A}$ to $\overset{\text{int}}{B}$ using $\overset{\text{dest}}{C}$

(1) 1 ring from $\overset{\text{src}}{A}$ to $\overset{\text{dest}}{C}$ using $\overset{\text{int}}{B}$

(7) $\overset{(N-1)}{3}$ rings from $\overset{\text{int}}{B}$ to $\overset{\text{dest}}{C}$ using $\overset{\text{src}}{A}$

15

No. of steps to move N rings from src to dest $= 2^N - 1$

Code

1) Assumption:

```
void TOH(N, src, dest, int)
```

TOH will print the correct steps to move N disks from src to dest using int.

2) Main Logic

```
TOH (N-1, src, int, dest)
print (" src → dest");
 TOH (N-1, int, dest, src);
```

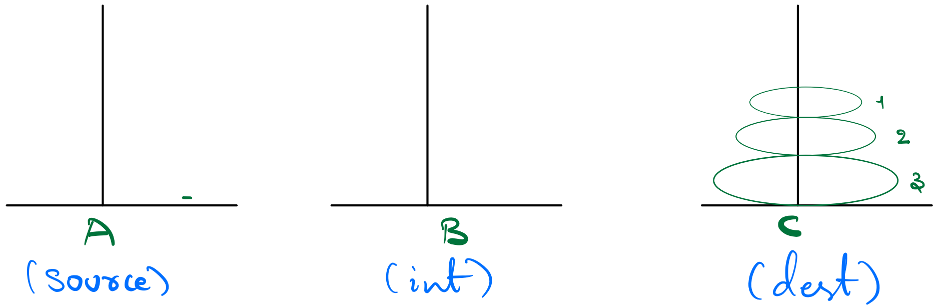
3) Base Case

```
if (N == 0) {
            return;
}
```

Code

```
void  TOH( N , src, dest,  int) {
    if (N == 0) { return ; }
    TOH (N-1, src, int, dest);
    print (N : src → dest);
    TOH (N-1, int, dest, src);
}
```

```
TOH (3, A, C, B)
    TOH (2, A, B, C)
        TOH (1, A, C, B)
            TOH (0, _ _ _)
            print (A → C);
            TOH (0, _ _ _)
        print (A → B)
        TOH (1, C, B, A)
            TOH (0, _ _ _)
            print (C → B);
            TOH (0, _ _ _)
    print (A → C);
    TOH (2, B, C, A)
        TOH (1, B, A, C)
            TOH (0, _ _ _)
            print (B → A);
            TOH (0, _ _ _)
        print (B → C)
        TOH (1, A, C, B)
            TOH (0, _ _ _)
            print (A → C);
            TOH (0, _ _ _)
```



✓ A → C

✓ A → B

✓ C → B

✓ A → C

✓ B → A

✓ B → C

✓ A → C

N ⇒ $(2^N - 1)$

T.C. = $O(2^N)$

S.C. = $O(N)$

Q) Print all valid parenthesis of length 2N for a given value N. ( )

Valid parenthesis means equal no. of opening & closing bracket in correct order.

Eg

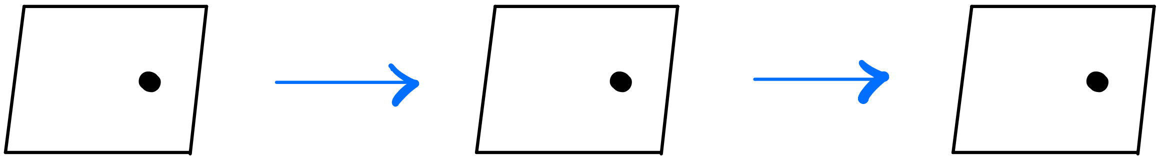
N = 1 ⇒ ( ), ~~)(~~

N = 2 ⇒ (()), ()(), ~~)()(~~, ~~...~~ . . . . .

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

Back Tracking

Brute force (All possibilities)



Open (Do)
Search
Close (Undo)

Open (Do)
Search
Close (Undo)

Open (Do)
Search
Close (Undo)

$N = 3$

(6) $\underline{2} \times \underline{2} \times \underline{2} \times \underset{\text{Choice}}{\underline{2}} \times \underline{2} \times \underline{2}$

( )

Total possible sequence of par. (valid + invalid) $= 2^{2N}$

(Notion of Choice)

Ex// Paper ⇒ MCQ ⇒ T/F

3 Questions.

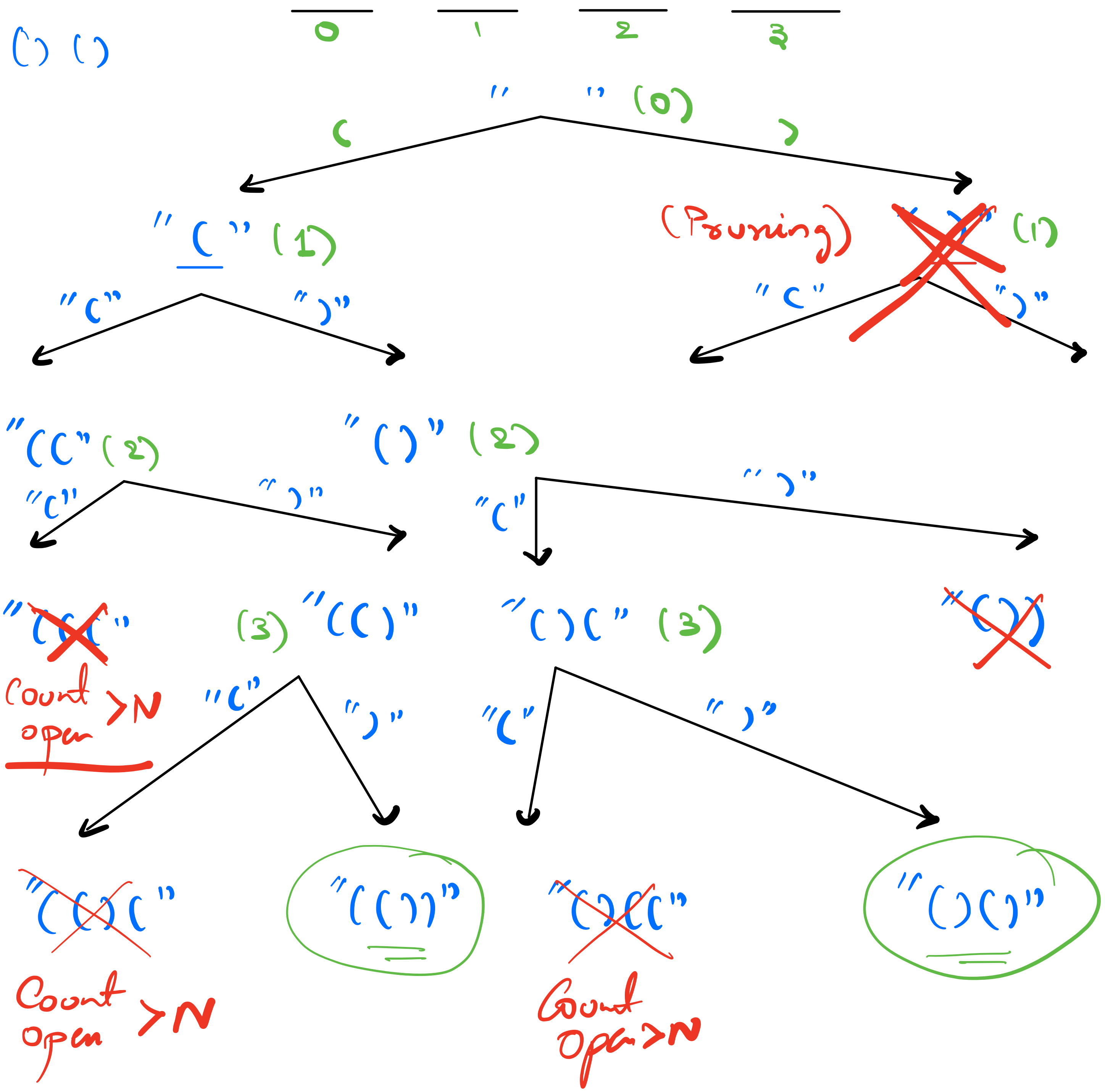
$\underline{2} \times \underline{2} \times \underline{2} = 2^3 = 8$

T	T	T
T	T	F
T	F	T
T	F	F
F	T	T
F	T	F
F	F	T
F	F	F

} 8

N = 2

Till every index ⇒ count open >= count close



Code

```
                  ""           0        0
                  ↑            ↑        ↑
void generate ( str, N, open, close) {

     if ( str.length() == 2N) {
              print (str);
```

```
            return;
    }

    if (open < N) {
        generate( str+'(', N, open+1, close);
    }

    if (close < open) {
        generate ( str+')', N, open, close+1);
    }
}
```

H.W. How to save space ??

N pairs = 2N length

char Array of length 2N