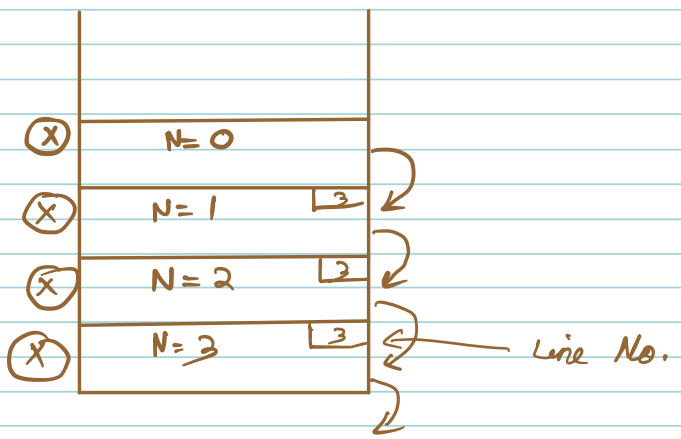
Recursion-2

Quiz-1 : what is the output of the following code for N = 3?

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

OUTPUT

1 2 3

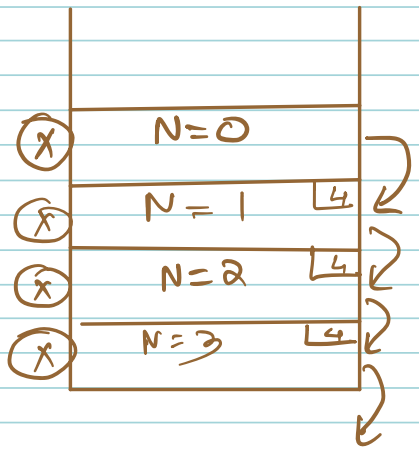


Quiz 2 : what is the output of the following code for N=3?

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

Output

3 2 1



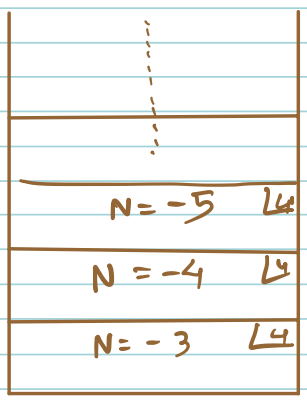
Quiz 3 :- what is the output of the following code for N = -3?

```
Void Solve (int N){
    if (N==0)
        return;
    print (N);
    Solve (N-1);
}
```

3

OUTPUT

-3 -4 -5



Error : Stack overflow.

Question :- what will happen when recursive won't end.

↳ In case of loop you must have got TLE.

↳ In case of Recursion before Reaching to a state of TLE our code end up taking more space than allocated, Hence we end up with Stack overflow.

Question :- TOWER OF HANOI

There are n disks placed on tower A of different sizes.

GOAL

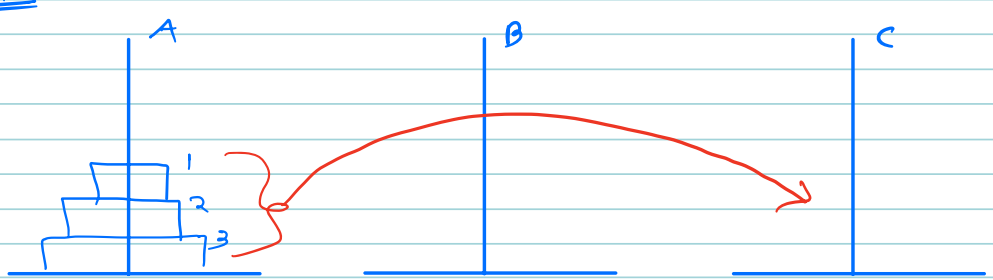
↳ Move all disks from tower A to C using tower B if needed.

CONSTRAINTS

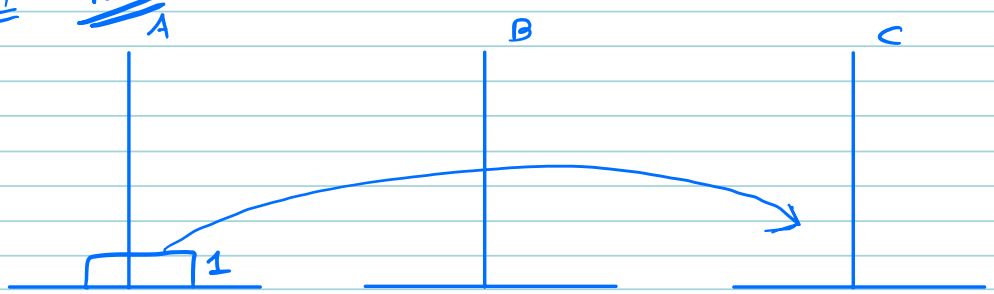
↳ ① only 1 disk can be moved at a time.
② Larger disk can not be placed on a small disk at any step.

Print the movement of disks from A to C in minimum steps

TASK

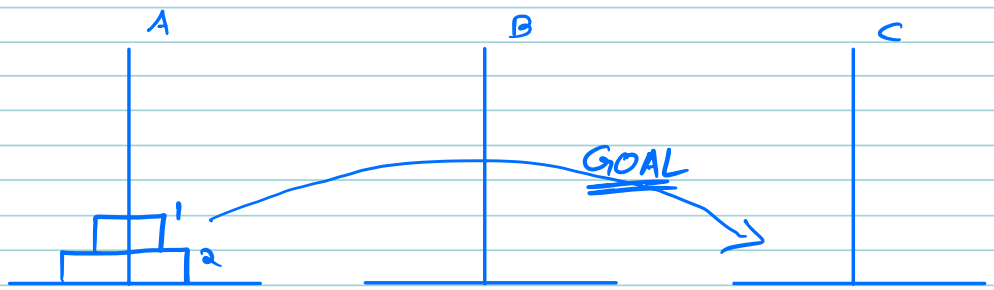


Ex-1 N=1

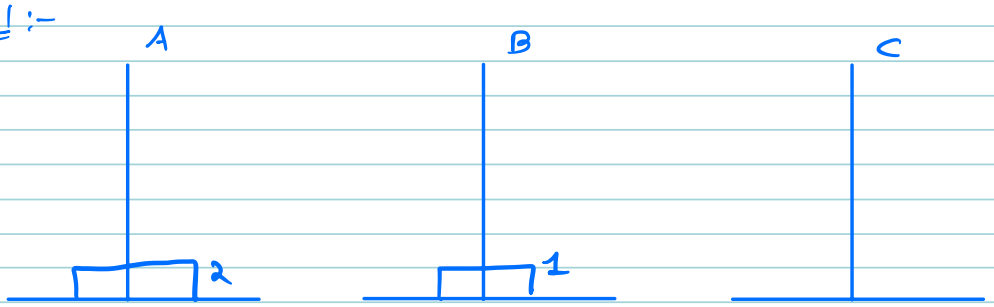


1 : A → C

Ex-2 N=2

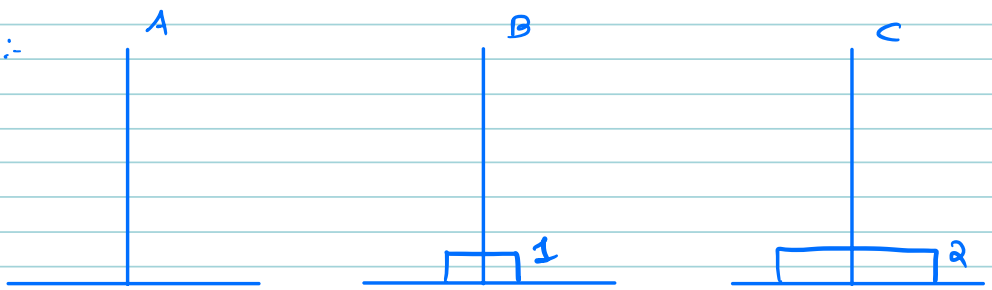


Step 1 :-



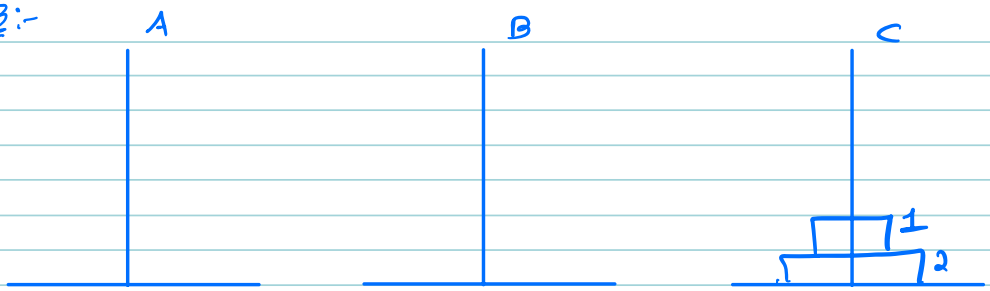
1 : A → B

Step 2 :-



2 : A → C

Step3 :-



1 : B → C

Ex-3

N = 3

→ Think in terms of recursion.

→ IDEA is base on how we will break it into subproblem & how we can use the 2 disks movement

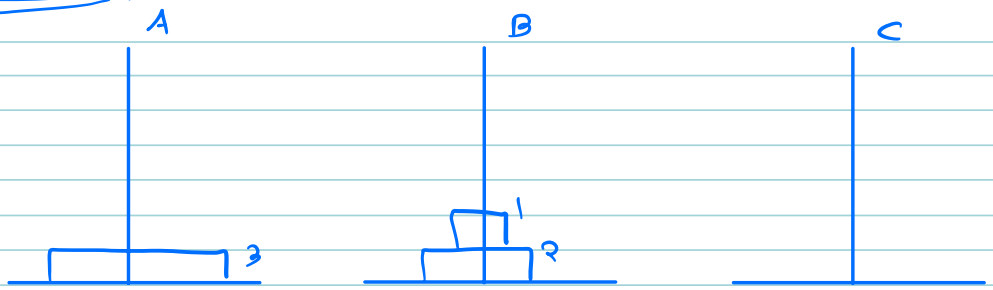
In the Direction of Recursion

current problem :- ① 2 disks from A to B

② 3rd Disk from A to C

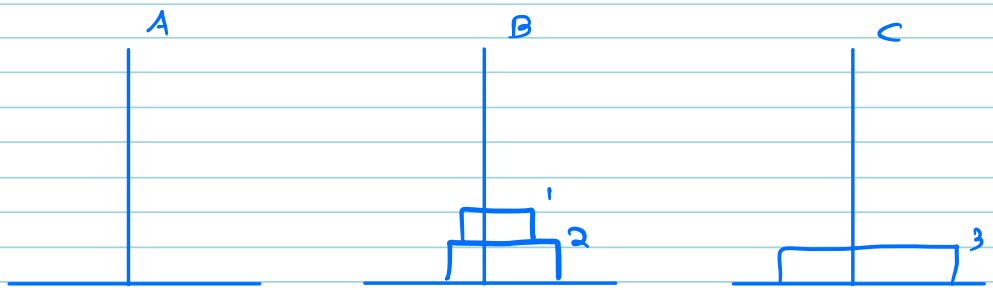
③ 2 disks from B to C

① First step : 2 Disks from A to B



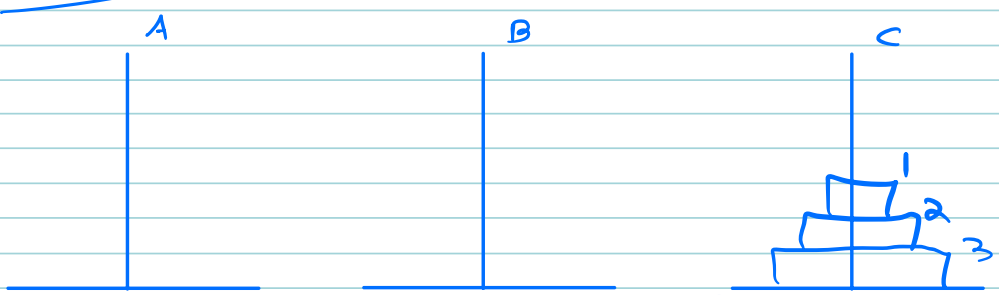
1 : A → C
2 : A → B
1 : C → B

② Step Second : 3rd Disk from A to C



3 : A → C

③ Step third : 2 Disks from B to C

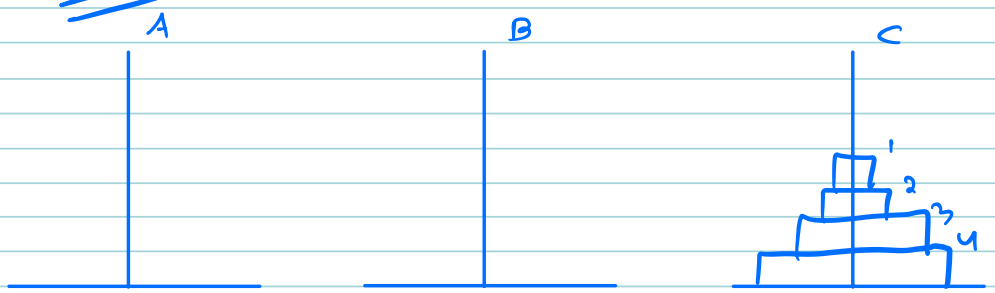


1 : B → A
2 : B → C
1 : A → C

OUTPUT:-

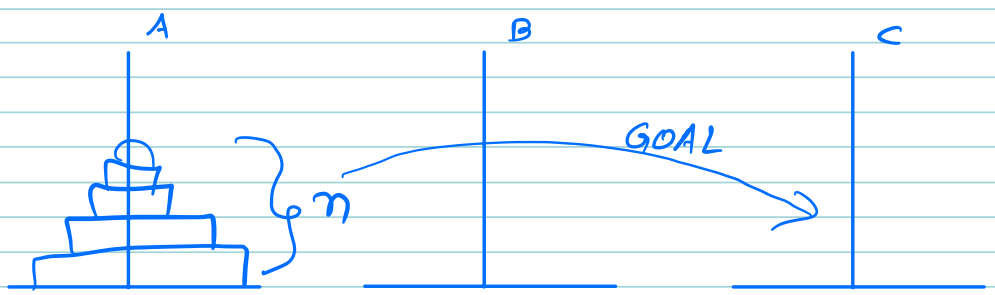
1 : A → C
2 : A → B
1 : C → B
3 : A → C
1 : B → A
2 : B → C
1 : A → C

Ex 3:- N = 4

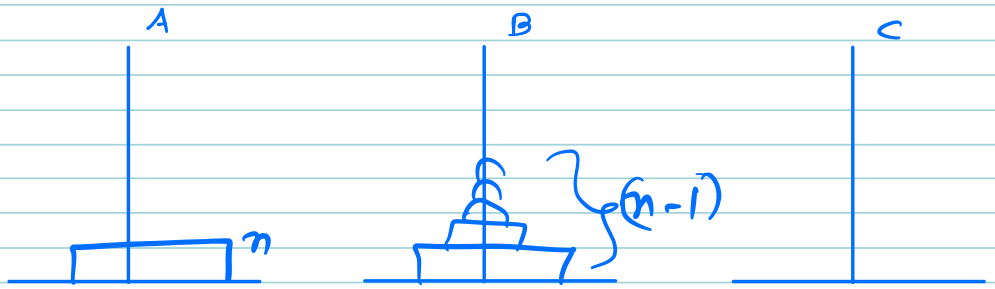


Sub-problem → ① 3 disks from A to B
② 4th Disk from A to C.
③ 3 disks from B to C

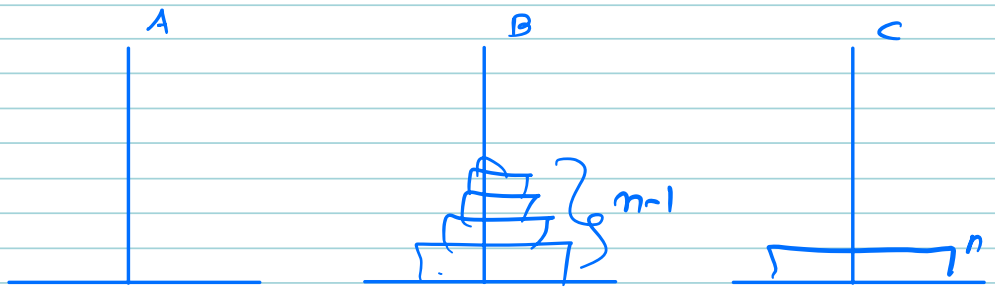
Generic



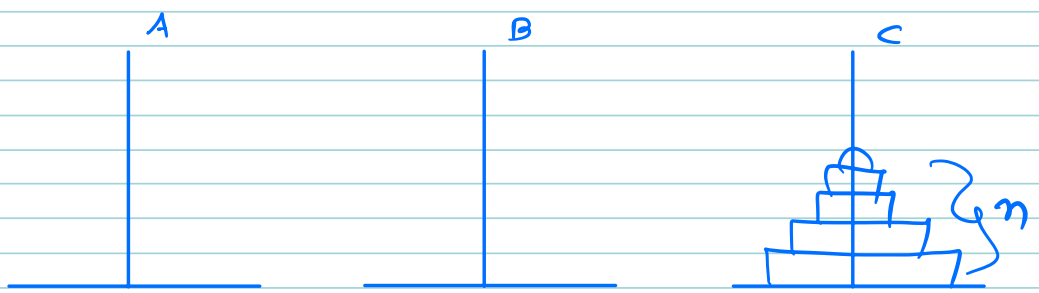
IDEA:- ① Move (n-1) Disks from A to B



② Move n^{th} Disk from A to C



③ Move (n-1) Disks from B to C



PSEUDO CODE

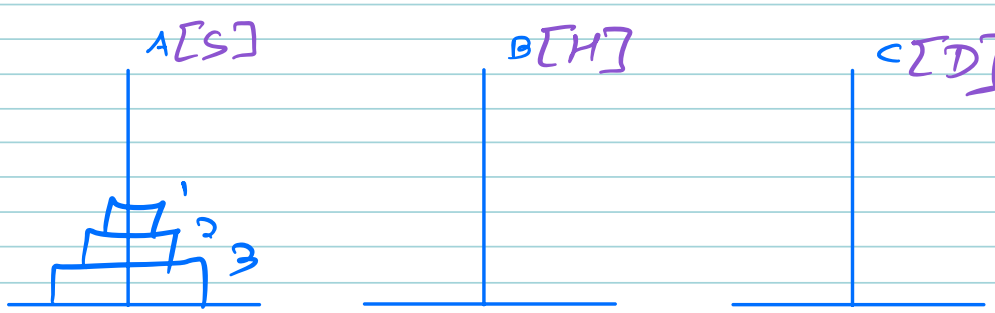
A → Source, B → Helper, C → Destination

```
void TOHC ( N , char S, char H, char D )
    if ( N == 0 ) { return; }
    TOHC ( N-1 , S, D, H );
    Print ( N : S → D );
    TOH ( N-1, H, S, D );
}
```

=> Any Recursive solution.

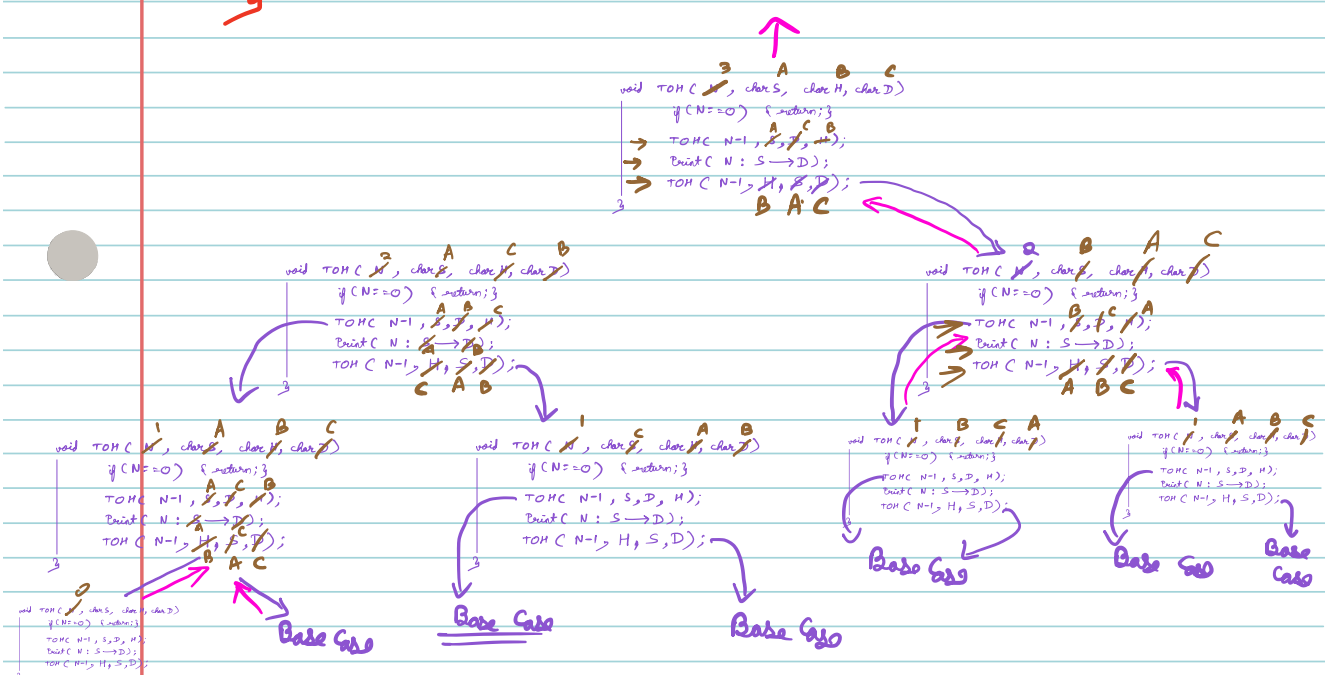
- High level understanding → [3 Step of Recursion]
- Low level understanding → [Dry Run]

DRY RUN



PSEUDO CODE

```
void TOH ( N , char S, char H, char D)
    if (N==0) { return; }
    TOHC N-1 , S, D, H);
    Print( N : S → D);
    TOH ( N-1, H, S, D);
}
```



1: A–C
2: A–B
1: C–B
3: A–C
1: B–A
2: B–C
1: A–C

1 : A → C
2 : A → B
1 : C → B
3 : A → C
1 : B → A
2 : B → C
1 : A → C

$T(0) = 1$

Time complexity

1st way :- Recursive Relation

$$T(N) = 2T(N-1) + 1$$

Maths

$$T(N) = 2T(N-1) + 1$$

$T(N-1) = 2T(N-2) + 1$

$$\Rightarrow T(N) = 2[2T(N-2) + 1] + 1$$

$$= 4T(N-2) + 3$$

$T(N-2) = 2T(N-3) + 1$

$$\Rightarrow T(N) = 4[2T(N-3) + 1] + 3$$

$$= 8T(N-3) + 7$$

Generic

$$T(N) = [2^K T(N-K)] + (2^K - 1)$$

where $K = N$

$$\Rightarrow T(N) = 2^N T(N-N) + (2^N - 1)$$

$$= 2^N \cdot T(0) + 2^N - 1$$

$T(0) = 1$

$$= 2^N + 2^N - 1$$

$$= 2 \cdot 2^N - 1$$

$$= 2^{N+1} - 1$$

$TC \rightarrow O(2^N)$

$SC \rightarrow O(N)$

2nd Way

TC = Time taken by one func. call * No of function call

$= 1 * [2^0 + 2^1 + 2^2 + \ldots + 2^N]$

H.W

$= 2^{N+1}$

$TC = O(2^N)$

$SC = O(N)$

question Print all valid parenthesis of len 2N.

Eg, N = 1, ⟶ ()

N = 2, ⟶ ()(), (())

N = 3, ⟶ ()()(), ((())), ~~)))(((~~
(())(), ()(()), (()())

N = 4, ⟶ ()(())(), (((())))), ...

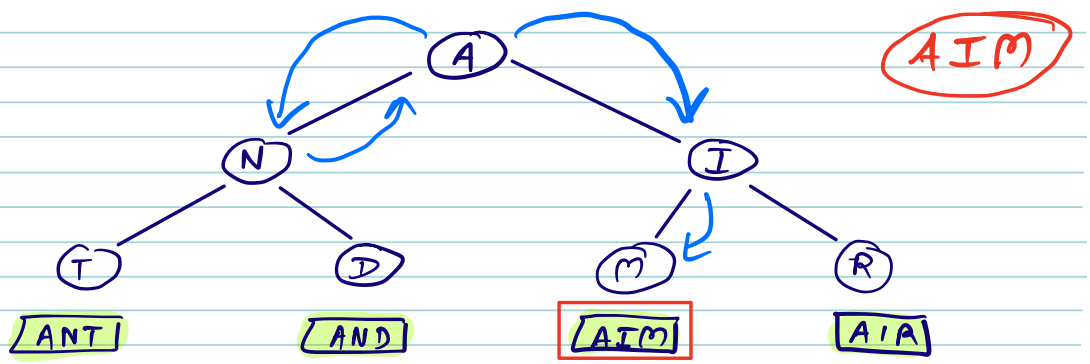
IDEA

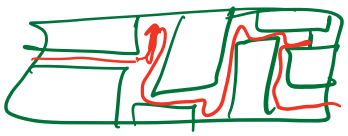
↳ String 2N, check if it is valid or not

↳ N=3, ()) .....

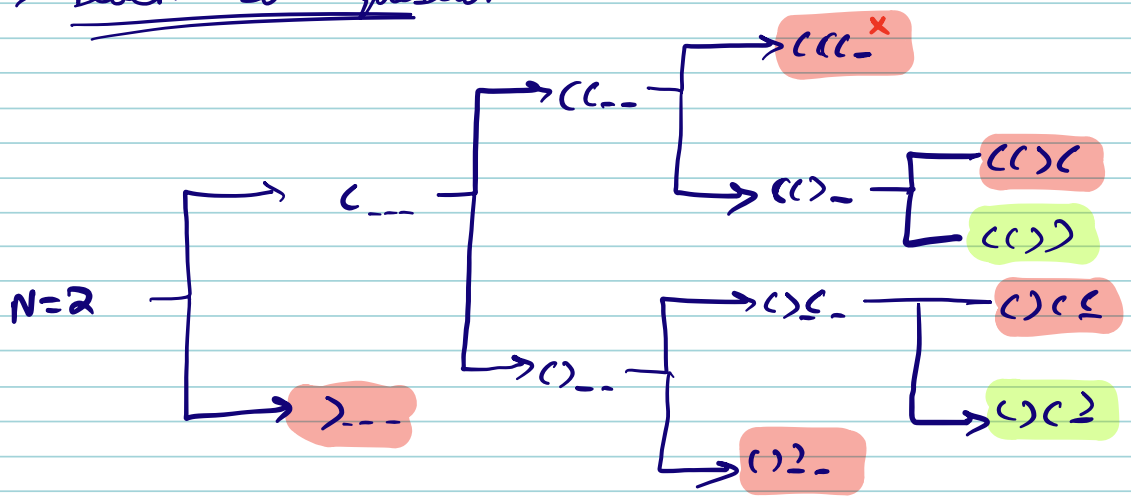
↑ while Creating further if String becomes invalid. Don't proceed if

Backtracking → Try all the possible Ideas of Recursion.





=> Back to question



Backtracking :- (Try all possible paths)

question :- when to say a string is valid?

Let, → open bracking count (open_br)
→ closing bracket count (closing_br)

Observation

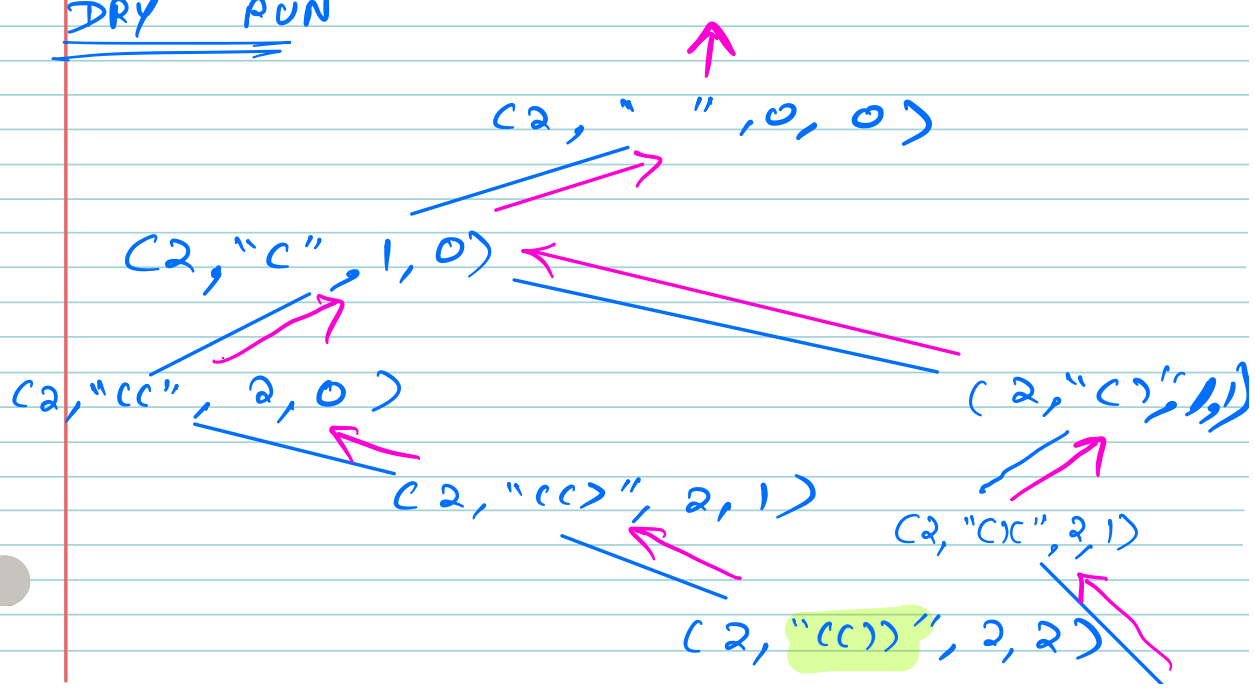
① open_br <= N

② open_br >= closing_br

PSEUDO CODE

```
void solve( N, str, open_br, closing_br)   // str = ""
{
    if ( str.len == 2N ) {
        Print(str)
        return;
    }
    if ( open_br < N ) {
        solve( N, str + "(", open_br + 1,
               closing_br);
    }
    if ( closing_br < open_br) {
        solve( N, str + ")", open_br,
               closing_br+1);
    }
}
```

DRY RUN



(2, "()()", 3, 3)

TC → worst Case, If you are able to make 2 calls from each Node

$O(2^N)$

SC → $O(2N) \approx O(N)$