Recursion

A function that calls itself, with a failure condition.

- Any problem that can be solved recursively. Can also be solved iteratively.

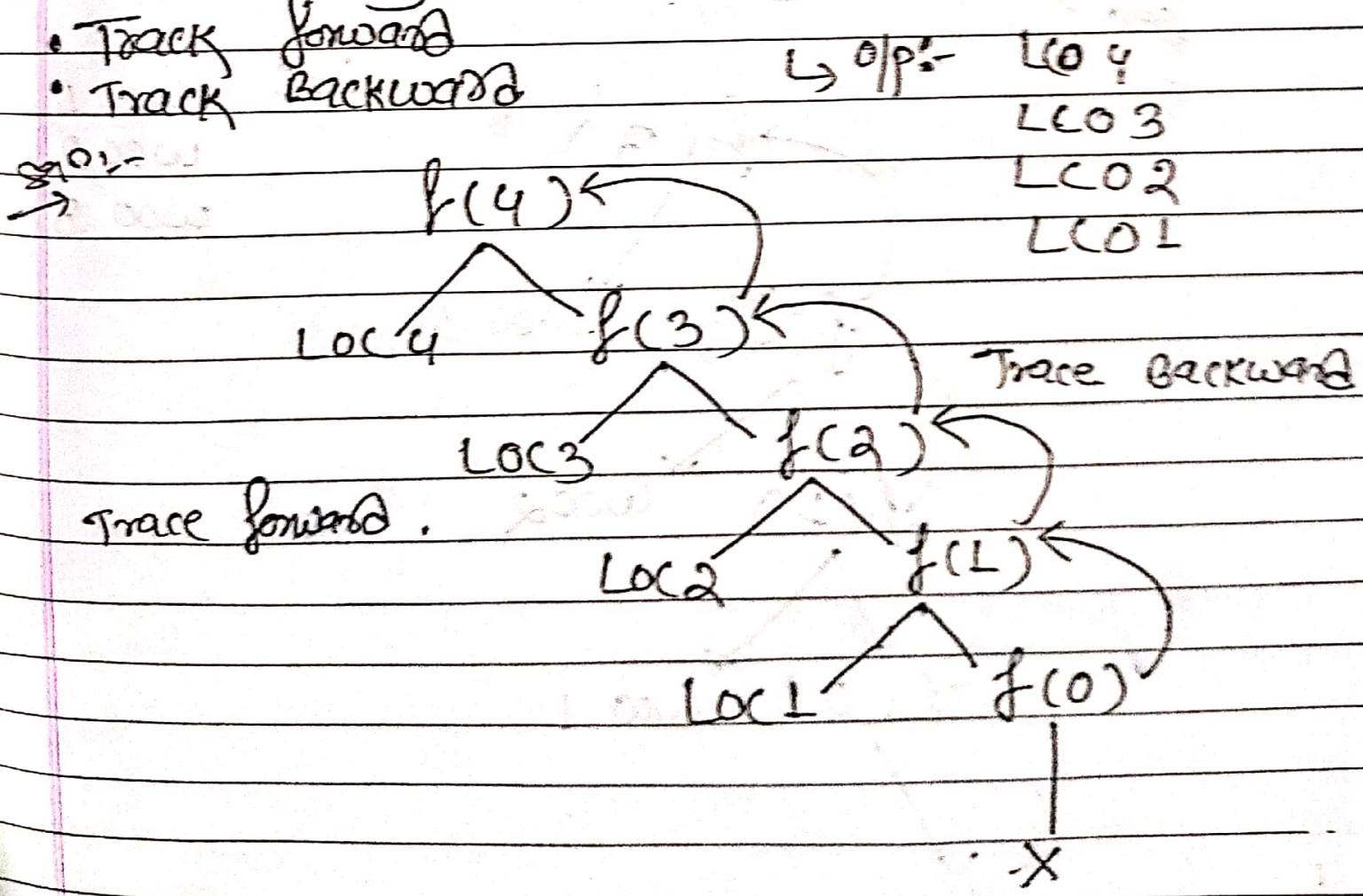
```
function (params) {
    if (some_condition) {
        - - - - -
        condition_change
        function (params)
        - - - - -
        - - - - -
        - - - - -
    }
}
```


```
function (n) {
    if (n>0) {
        print ("LCO, n")
        function (n-1)
    }
}

main () {
    n= 4
    function (n)
}
```

- Track forward
- Track Backward

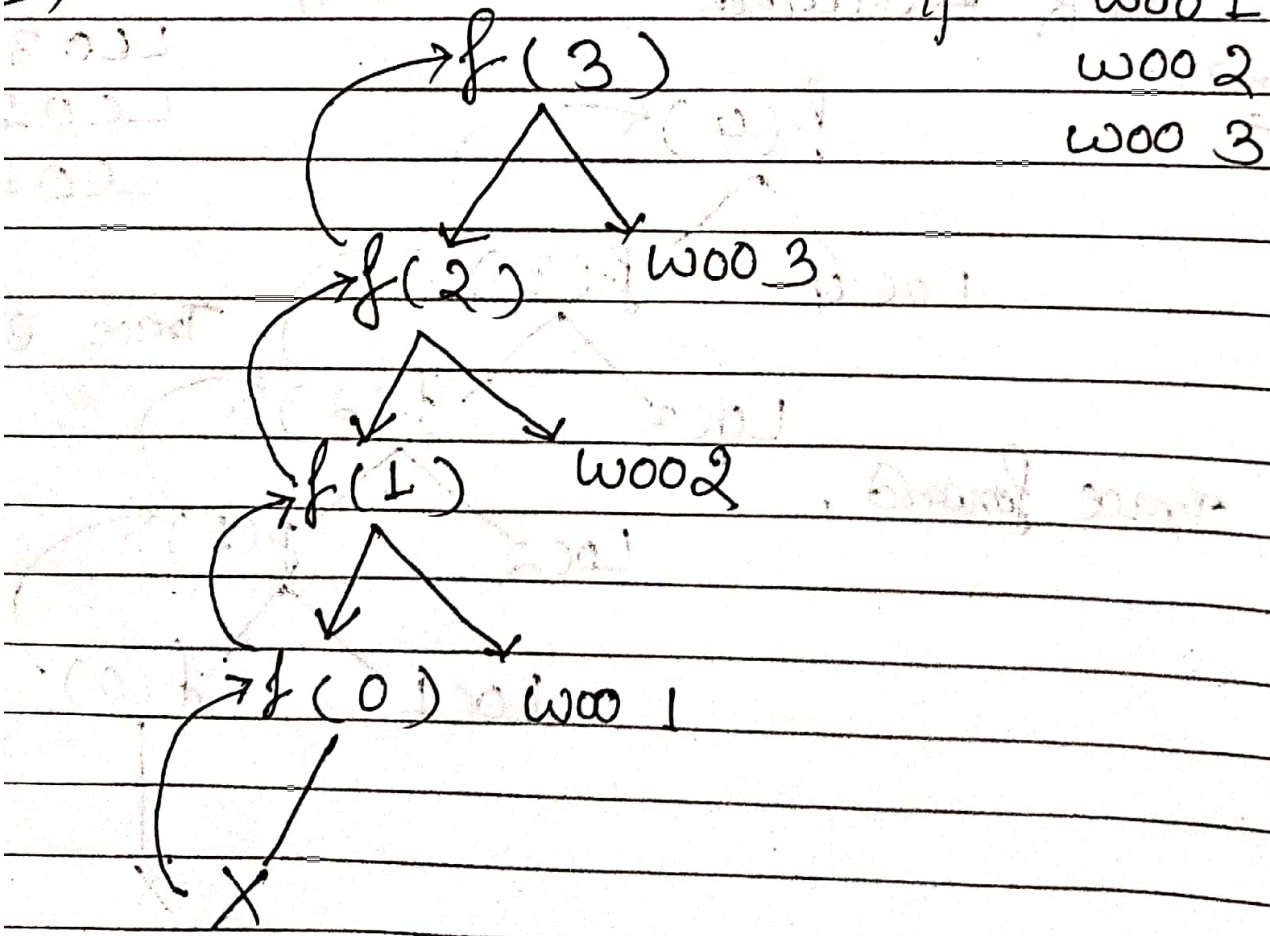
o/p:-



↳ o/p:- LCO 4
LCO 3
LCO 2
LCO 1

```
# function (n) {
    if (n > 0) {
        function (n-1)
        printf ("%d", n)
    }
}
main () {
    n = 3
    function (n)
}
```

o/p:-

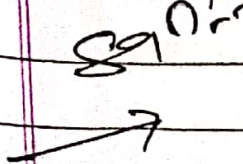


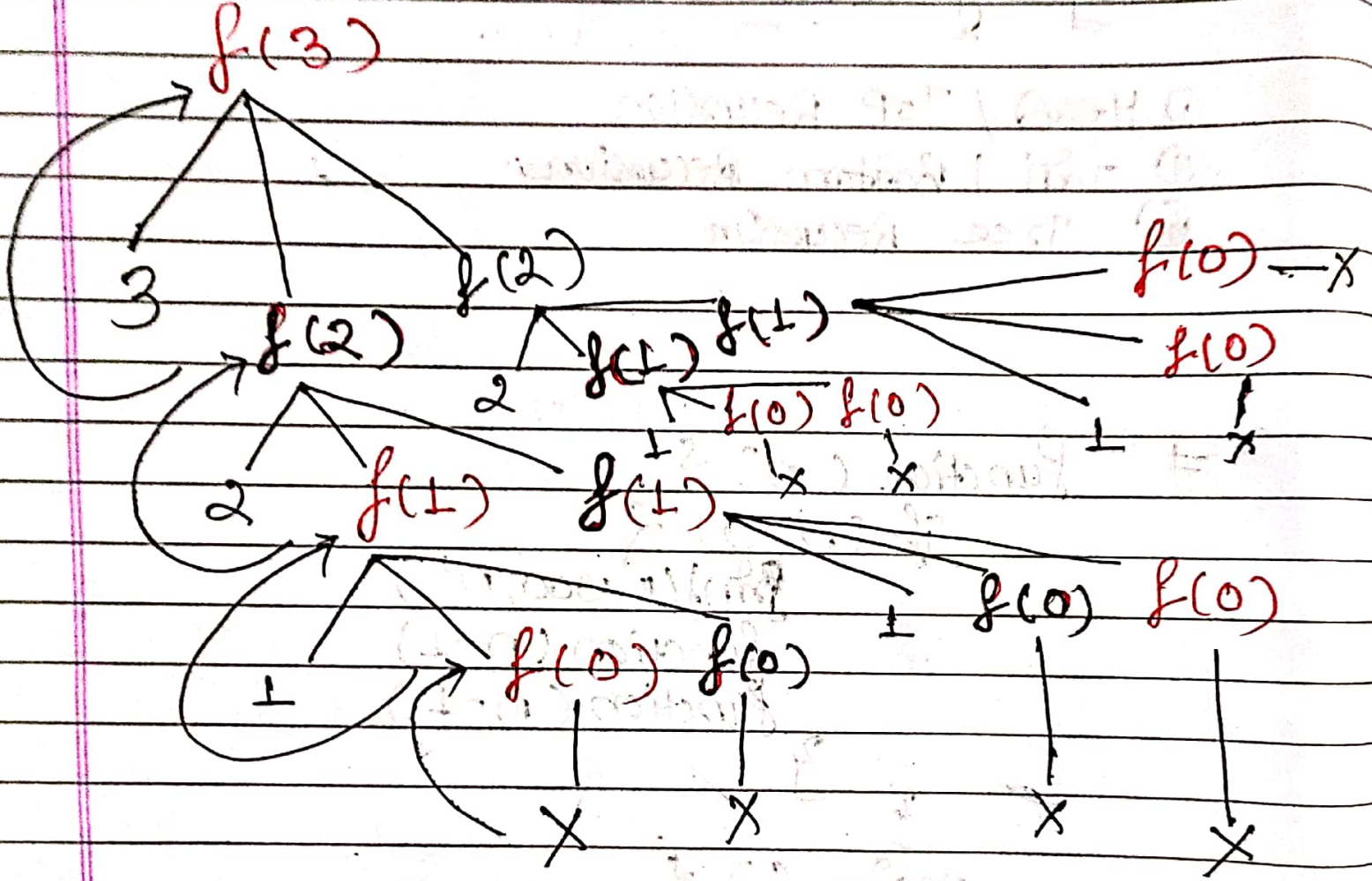
o/p: 1
2
3
3
2
1

Types of Recursion :-

(i) Head / Top Recursion
(ii) Tail / Bottom Recursion
(iii) Tree Recursion

```
# function (n) {
    if (n>0) {
        printf("%d", n)
        function (n-1)
        function (n-1)
    }
}
main () {
    n = 3
    function (n)
}
```





O/p :- 3, 2, 1, 1, 2, 1, 1

Q1

— previous program

```
function(n-1)
function(n-1)
print(n)
```

Ans:- 1,1,2,1,1,2,3

— previous program

Q2

previous program

```
function(n-1)
print(n)
function(n-1)
```

Ans:- 1,2,1,3,1,2,1

previous program

Factorial Theory

$n! = 1 \times 2 \times 3 \times 4 \times \ldots \ldots n!$

$4! = 1 \times 2 \times 3 \times 4$

$5! = 1 \times 2 \times 3 \times 4 \times 5$

$4! = 4 \times 3 \times 2 \times 1$

$f(0) = 1$

$f(1) = 1$

$f(-n)$ = factorial of -ve number doesn't exist.

$n! = n \times f(n-1)$

formula,

$$fact(n) = n * fact(n-1)$$

Fibonacci Sequence

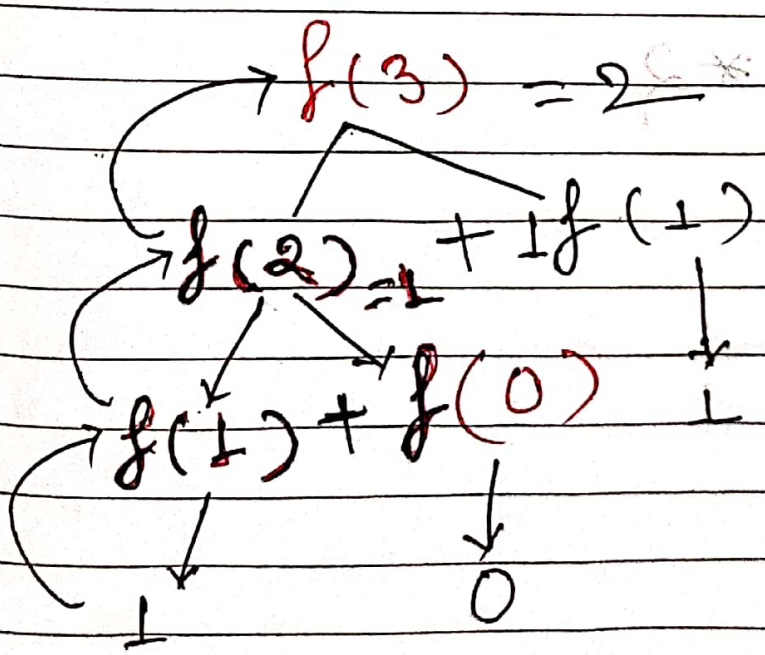
The fibonacci sequence is the series of numbers:
0, 1, 1, 2, 3, 5, 8, 13, 21, ------

The next number is found by adding up the two numbers before it.

ie. The 3 is found by adding the two numbers before it (1+2).

formula,

$$fib(n) = fib(n-1) + fib(n-2)$$



∴ f(3) = 2

let's check,

0, 1, 1, 2, 3, 5, ---
↓ ↓ ↓ ↓
f(0) f(1) f(2) f(3)

so, f(3) = 2

Power program

P(2,3)

$= 2^3 = 2 \times 2 \times 2 = 8$

P(b, p-1) * b

P(2,3)

P(2,2) * 2 = 8

P(2,1) * 2 = 4

P(2,0) * 2

1

x

Combination program

nC_r

$n \rightarrow$ total no. available

$r \rightarrow$ How many Combination

↳ ABCD

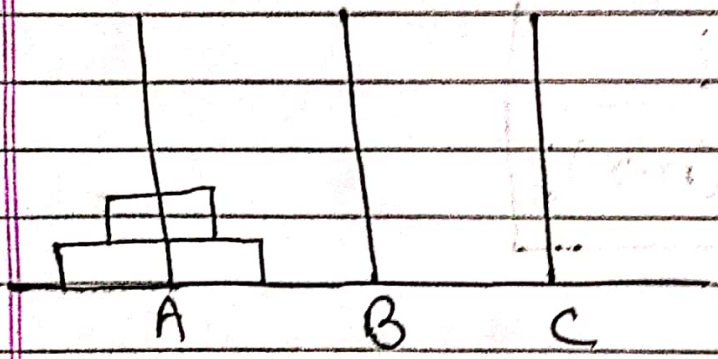
- AB
- BC
- CD
- AD
- BA not allowed

$$nC_r = \frac{n!}{r!\,(n-r)!}$$

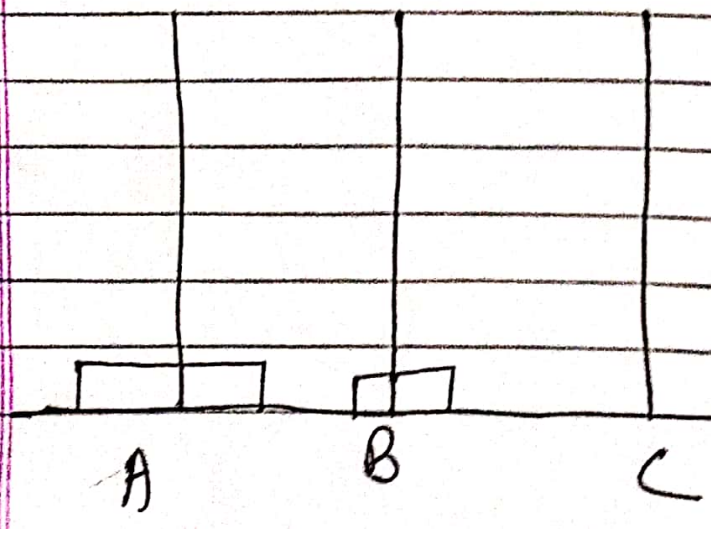
Tower of Hanoi

Condition :- (I) You can move only one disk at a time.
(II) We can't put smaller disk below the big disk.

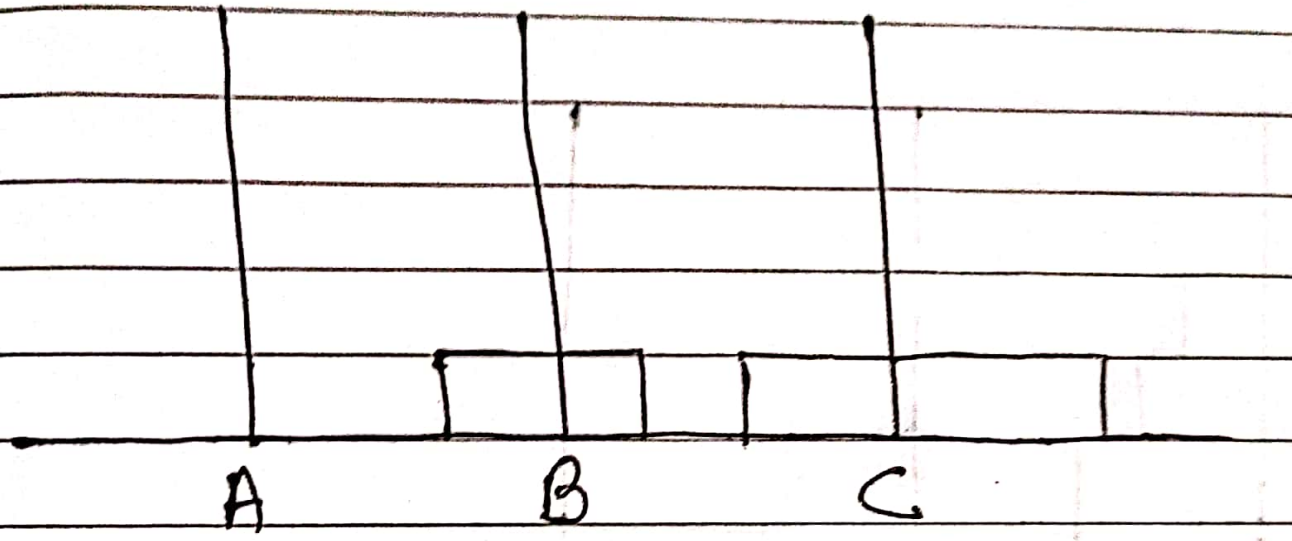
Tower of Hanoi with two disk



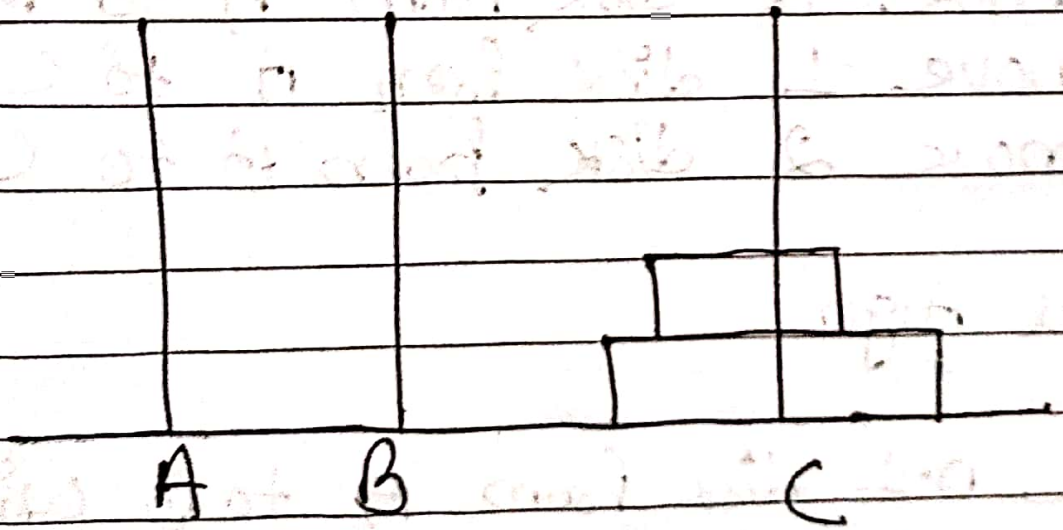
(I) move A to B using C.



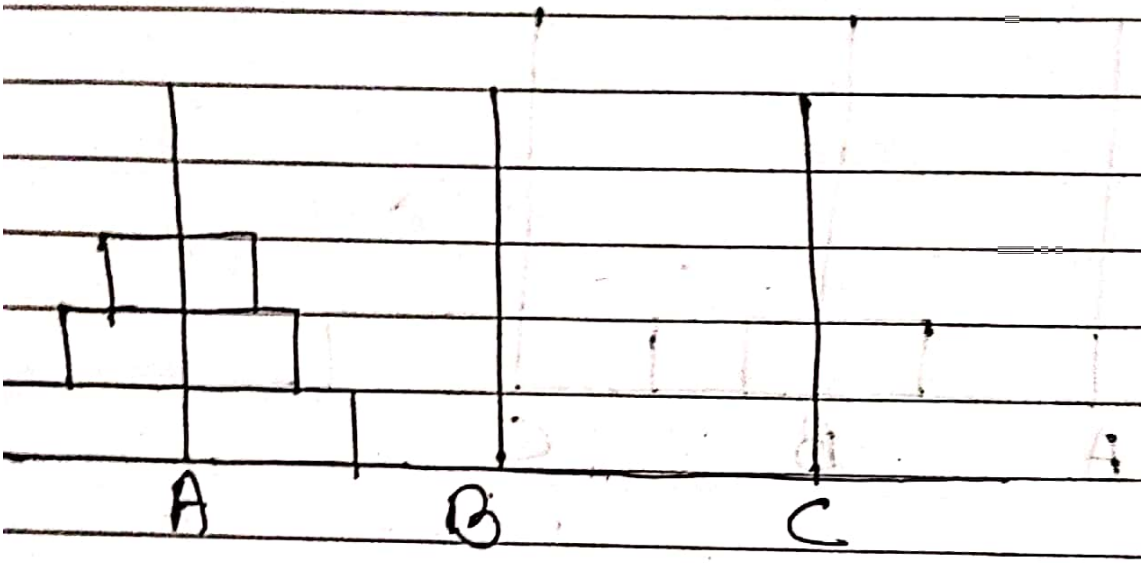
(ii) move A to C.



(iii) move B to C.



Tower of Hanoi for 3 disk



move 2 disk from A to B using C
move 1 disk from A to C
move 2 disk from B to C using A

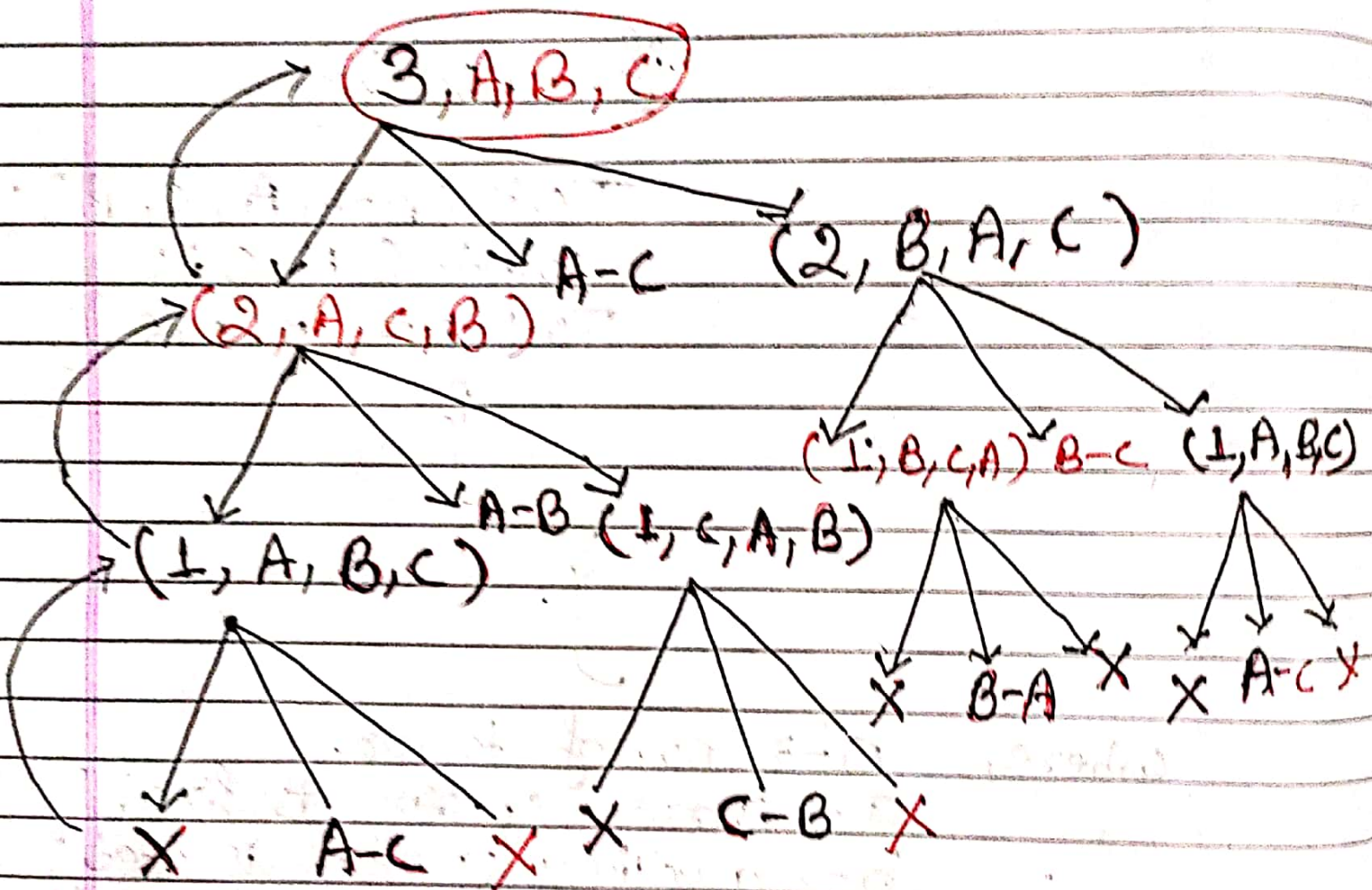
general Algo

(i) move n-1 disk from A to B using C.
(ii) move 1 disk from A to C.
(iii) move n-1 disk from B to C using A

```
void TOH (int n, int a, int b, int c){
    if (n>0){
        TOH (n-1, A, C, B);
        print (A, C);
        TOH (n-1, B, A, C);
    }
}
```

where, n → no. of tower.
a → source (position is fixed)
b → auxiliary (position is fixed)
c → destination (position is fixed)

Tracing tree of towers of Hanoi



O/P :- A-C, A-B, C-B, A-C, B-A, B-C, A-C