3. PROBLEM DECOMPOSITION BY RECURSION

In problem decomposition, we first decompose the problem into smaller problems and recompos the results of those smaller problems to get result of the main problem.

A recursive function always consists 2 major conditions.

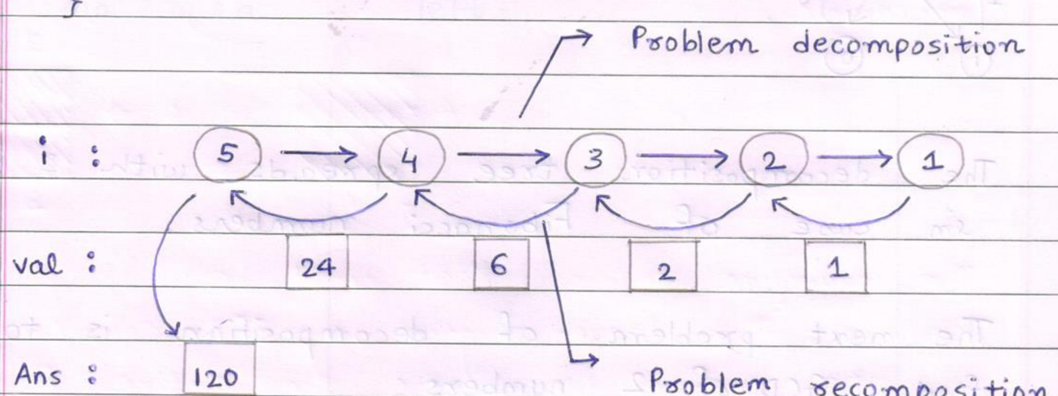
a) Base condition - to end the recursion
b) Recursive condition - the main logical part

We look at a problem which we have solved earlier - Fact (n).

```
int fact (int n)
{
    int val ;

    if (n==1)  val = 1 ;              // Base condition
    else       val = n* fact (n-1) ; // Recursive condition

    return val ;
}
```



The decomposition tree is a straight line sequence in case of factorial. This might not always be the case. Consider the next example :-

Fibonacci Numbers :- Fib (n)

Base condition : $(n \leq 0) \rightarrow val = 0$

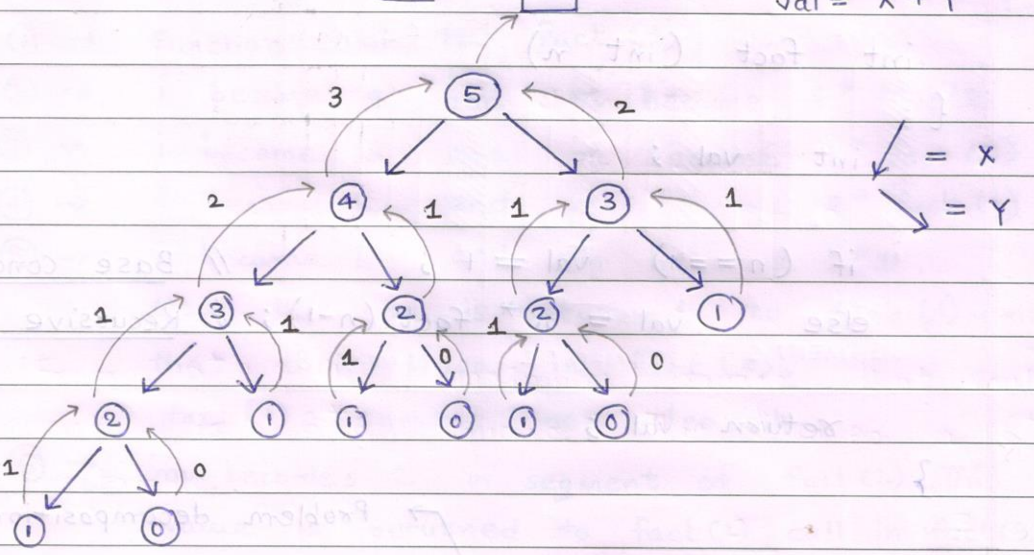
$(n == 1) \rightarrow val = 1$

Recursive condition : $(n > 1) \rightarrow X = fib(n-1)$

$Y = fib(n-2)$

$val = X + Y$

Ans. $\boxed{5}$



The decomposition tree spreads with 2 branches in case of Fibonacci numbers.

The next problem of decomposition is to find GCD of 2 numbers.

GCD of 2 numbers :- GCD (a, b)

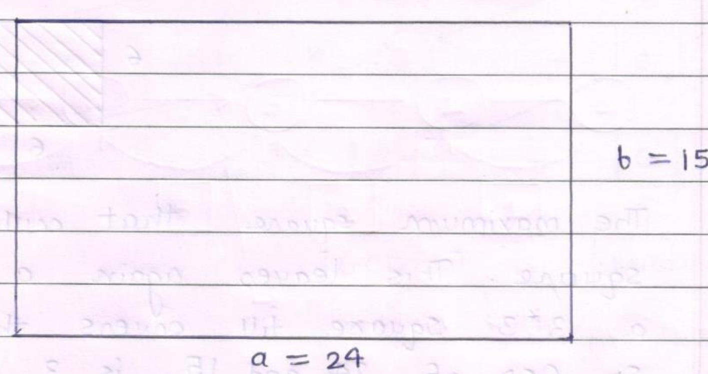
Before writing the base and recursive condition, we must understand how to think of GCD ?

GCD (Greatest Common Divisor) is the biggest number that divides both a and b. Physically, what GCD means is -

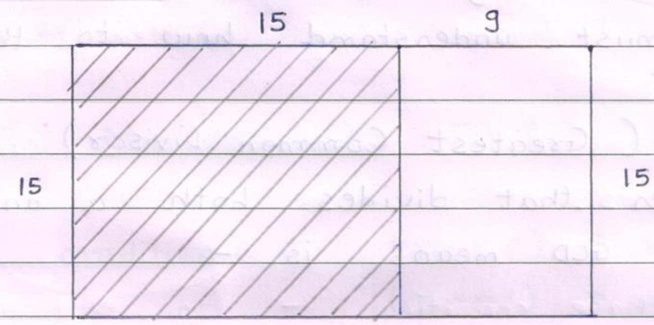
Suppose there is a floor of length a and breadth b (taking larger as a), then GCD is the dimension of the largest square tile that perfectly covers the entire area of the floor.

If the two numbers a and b are primes, then too the floor can be entirely covered by 1 x 1 square tiles and the number of such tiles required will be a * b.

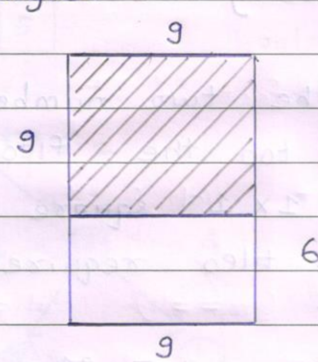
Consider a = 24 and b = 15. Our approach to find GCD will be in a way that we will keep on removing the largest square possible till no area is left.



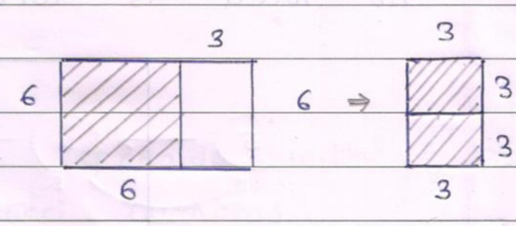
The largest square that could be cut from this floor is 15 * 15. This still leaves a 9 * 15 rectangle.



From this remaining 9 * 15, the maximum square that could be cut is 9 * 9 square. This still leaves a 6 * 9 rectangle.



From 6 * 9, the maximum square that could be cut is 6 * 6 square. This leaves a 6 * 3 rectangle.



The maximum square that could be cut is 3 * 3 square. This leaves again a 3 * 3 square. So a 3 * 3 square till covers the entire floor area. So GCD of 24 and 15 is 3.

The algorithm of finding GCD is -

(1) Write the bigger of a and b in form,

Bigger = Quotient * Smaller + Remainder

i.e. Dividend = Quotient * Divisor + Remainder

Dividend	Quotient	Divisor	Remainder	
a	a/b	b	a%b	... (a > b)

(2) In next step, Divisor becomes dividend and remainder becomes divisor.

(3) Step (2) is repeated till remainder is zero. Once remainder is zero, Divisor is GCD.

* Illustration :- gcd (24, 15)

Dividend	=	Quotient	*	Divisor	+	Remainder
24	=	1	*	15	+	$\boxed{9}$
15	=	1	*	9	+	$\boxed{6}$
9	=	1	*	6	+	$\boxed{3}$
6	=	2	*	$\boxed{3}$	+	$\boxed{0}$
				GCD		Remainder zero

GCD of a and b :- gcd (a, b)

Base condition :- (b == 0) → GCD = a .

Recursive condition :- (b > 0) → gcd (b, a % b)

From our previous illustration, we observe that :-

gcd (24, 15) = 3.
Also gcd (15, 9) = 3
Also gcd (3, 6) = 3.

So the problem decomposition is as follows :-

$\boxed{GCD = a = 3}$ ← gcd (24, 15) a = 24 & b > 0

a = 3 ↓ ↓

gcd (15, 9) a = 15 & b > 0

a = 3 ↓ ↓

gcd (9, 6) a = 9 & b > 0

a = 3 ↓ ↓

gcd (6, 3) a = 6 & b > 0

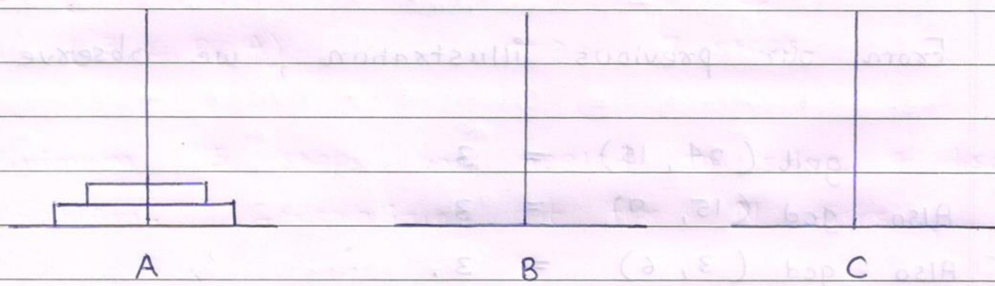
a = 3 ↓ ↓

gcd (3, 0) a = 3 & condition failed

The next problem after GCD is 'Tower of Hanoi' problem.

* TOWER OF HANOI Problem

The problem is there are 3 pegs. A, B and C with discs in peg A which can move into & out the peg.

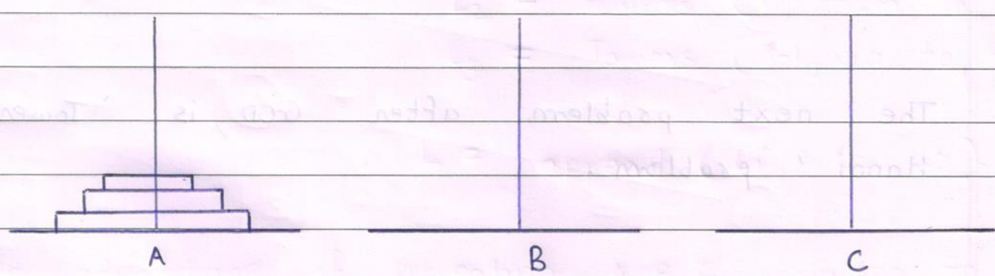


We have to move these 2 discs to B such that no smaller disc should come below larger disc anytime. During the shifting course, we can lift one disc at a time.

We follow, A → C , A → B , C → B .

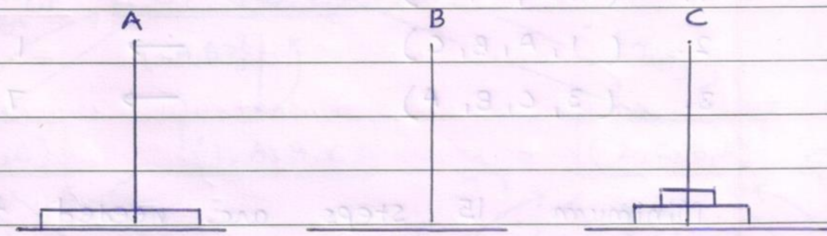
These 3 are minimum number of steps required to shift discs from A to B using C.

Now lets say we have 3 discs.

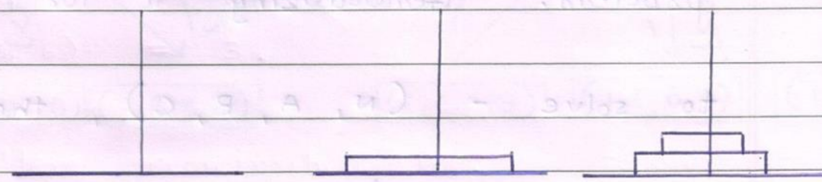


We solved problem of 2 discs by moving the larger disc to B and putting back the smaller one from C to B. Here again in 3 discs, our aim is to put the largest in B. For that above 2 discs must be in C. So the steps are :-

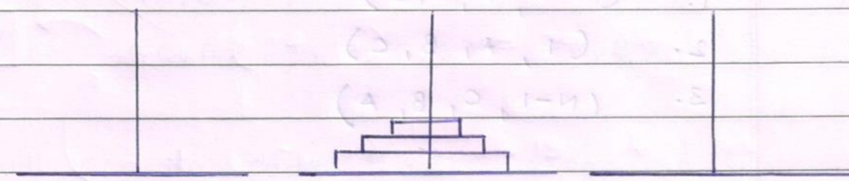
1) A → B
2) A → C
3) B → C



4) A → B



5) C → A
6) C → B
7) A → B



It seems we took 7 steps to do so but actually we have solved it in 3 steps.

Let the question be - Solve (No. of discs, from, to, via) and to solve (3, A, B, C).

Step 1 :- We solved 1st a 2 disc problem from A to C using B. i.e. (2, A, C, B).

Step 2 :- We moved 1 largest disc from A to B. i.e. (1, A, B, C).

Step 3 :- We moved 2 discs from C to B using A i.e. (2, C, B, A)

Also we can see 7 steps (3 + 1 + 3) required are minimum to solve 3 disc problem. If we want to solve - (4, A, B, C), then solⁿ is -

1. (3, A, C, B) ⟶ 7 steps
2. (1, A, B, C) ⟶ 1 step
3. (3, C, B, A) ⟶ 7 steps.

Minimum 15 steps are needed to solve 4 discs problem. Generalizing it for N discs,

to solve - (N, A, B, C), then solⁿ is -

1. (N-1, A, C, B)
2. (1, A, B, C)
3. (N-1, C, B, A)

Tower of Hanoi : Towers (n, from, to, via)

Base condition :- $(n == 1) : L = \{ \langle from, to \rangle \}$

Inductive condition : $(n > 1)$

L_1 = Towers (n-1, from, via, to)

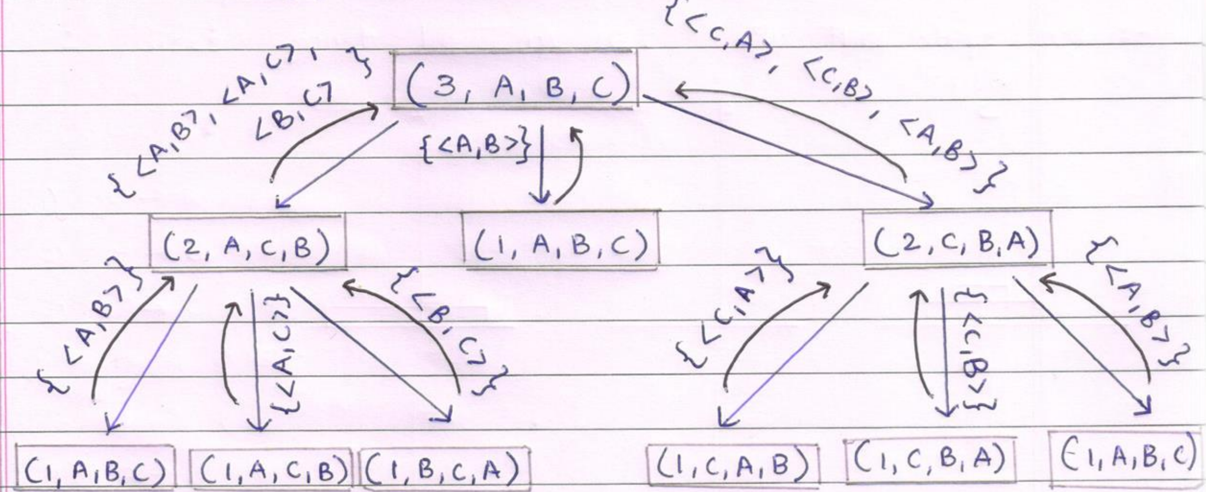
L_2 = Towers (1, from, to, via)

L_3 = Towers (n-1, via, to, from)

L = append (L_1, L_2, L_3).

The advantage of converting A, B, C variables to parameters is that we can now solve problem generally, i.e move four discs from B to C using A. In this case we need not change the order of variables

Following is the decomposition and recomposition tree for 3 discs.



So, $L = \{ \langle A, B \rangle, \langle A, C \rangle, \langle B, C \rangle, \langle A, B \rangle, \langle C, A \rangle, \langle C, B \rangle, \langle A, B \rangle \}$

Now we will go to code these programs :-

a) Fibonacci Numbers
b) GCD of 2 numbers
c) Tower of Hanoi Problem