Recursion-2

How to Think Recursively - - > Transforming into Recursive Code

Base case is very important

① Time Complexity :- of recursion program is directly proportional to the number of functions calls that are made before terminates

② Space Complexity :- of recursion program is directly proportional to the height of stack

$O(n^2)$ { $\cong 25$ ← $n=5$ → 6 fn calls (5 ") } $O(n)$

$\cong 100$ ← $n=10$ → 10/11

approximately

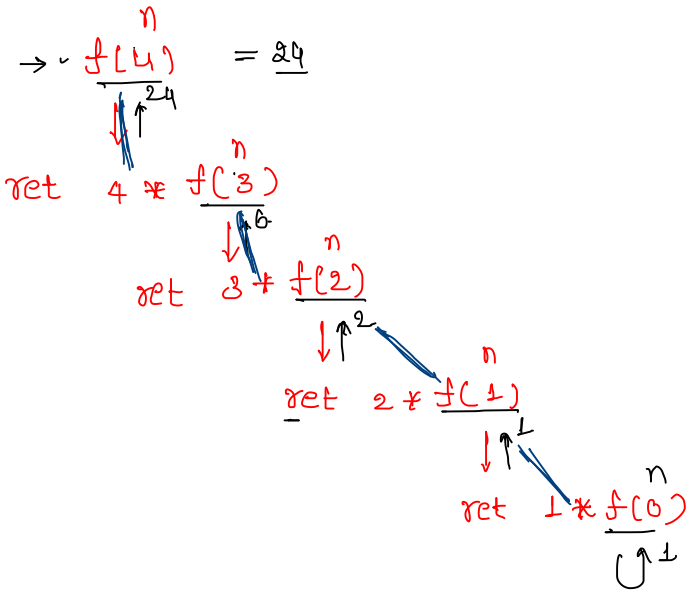
$n=4$ → 16 fn (2^4)

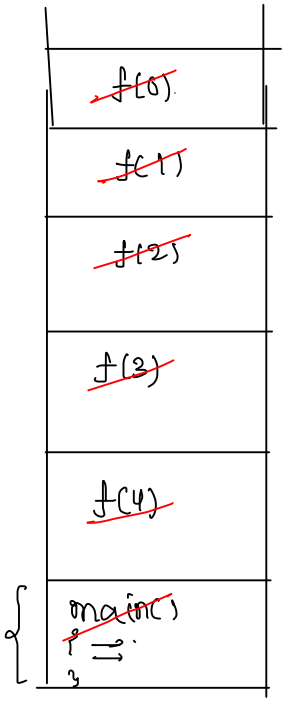
$n=5$ → 32 fn (2^5) } $O(2^n)$ exponential T.C

$n=10$ → 1024 fn calls (2^{10})

1) Factorial program

```
int fact(int n)
{
        if(n==0) // stop
        {
                return 1;
        }
        else
        {
                return n*fact(n-1);
        }

}
```





Approximately n activation records

so S.C: O(n)

max depth of rec. tree

for every function call, there will be one activation record will be push into the stack

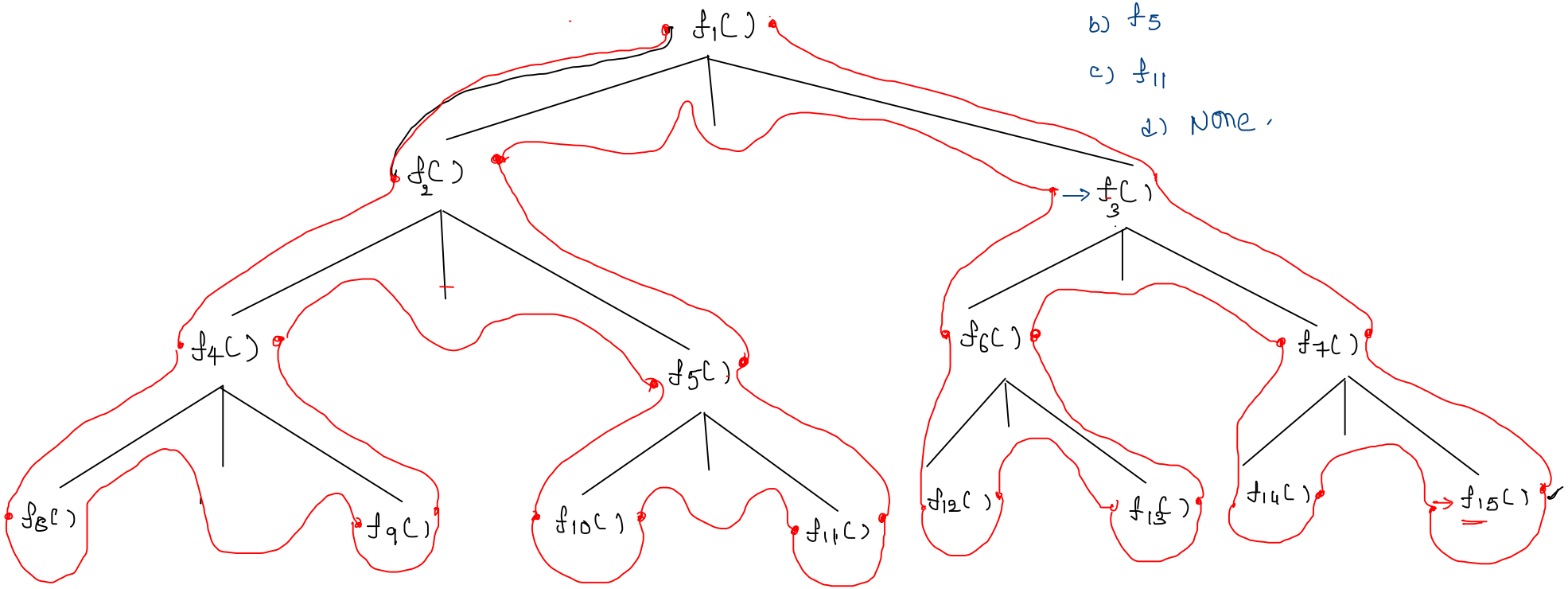
for some program (lets not worry)

✓ a) f_2 ✓

b) f_5

c) f_{11}

d) None.

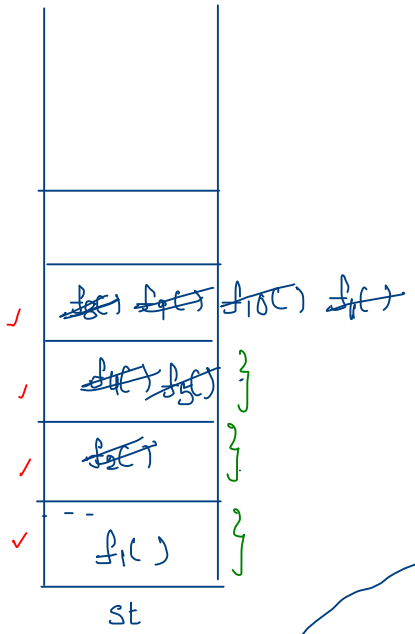


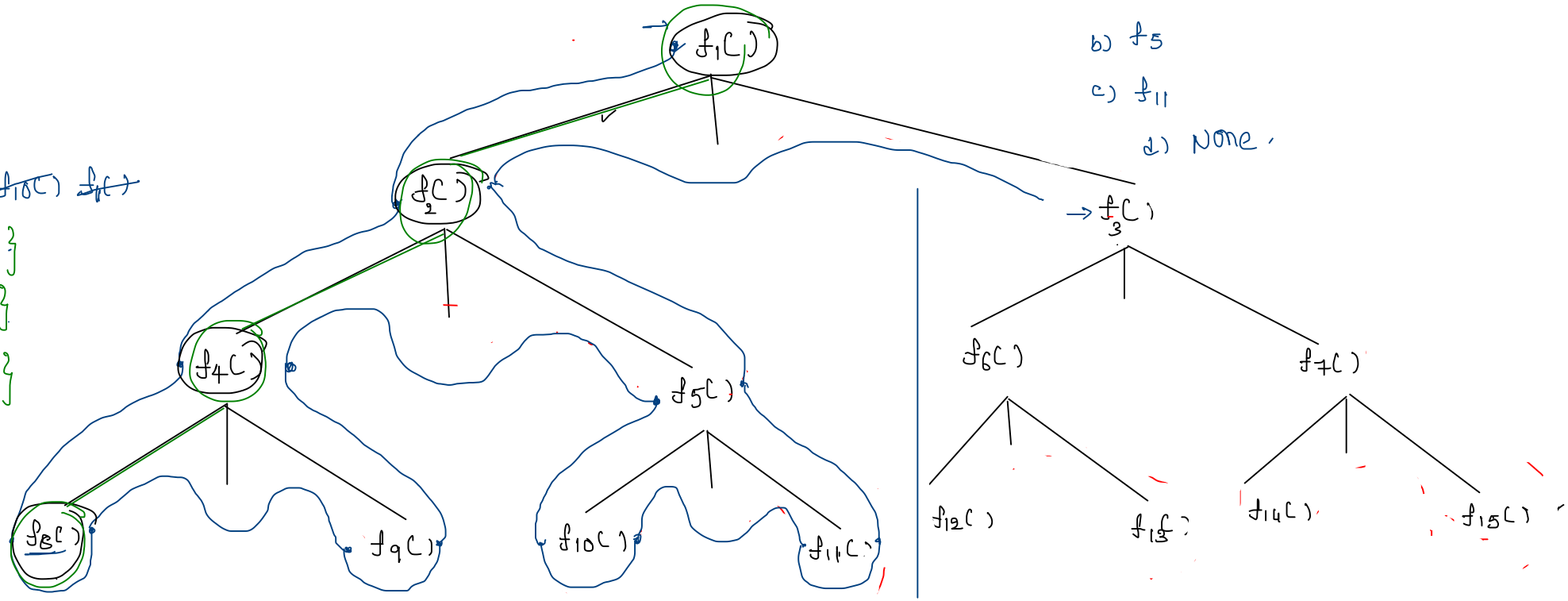
recursion tree

top to down

Left to Right

for some program (lets not worry)





a) f_2
b) f_5
c) f_{11}
d) None.

3+1 = 4 cells

max (Height) of stack we used

(or)

max depth of recursion tree

recursion tree

top to down

Left to Right

2) Implement pow(a,b) function [a>=1 and b>=0] :-

pow(2,3) = 8 ↳ 2x2x2

pow(3,0) = 3 ↳ $3^0 = 1$

pow(3,4) = 81

a b
pow(2,5) = 2^5 = 32

= (2) x 2x2x2x2
a ↳ pow(2,4)
a b-1

pow(2,5) = 2 * pow(2,4)

pow(2,4) = 2 * 2^3 = 2 x pow(2,3)

pow(2,0) = 1

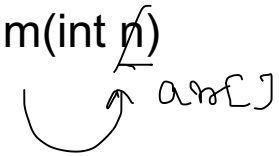
```
int pow(int a, int b) // a power b
{

        if(b==0)
                return 1;

        else
                return a*pow(a,b-1);
}
```

```
int pow(int a, int b)
{
        if(b==0)
                return 1;

        else
        {
                return a*pow(a,b-1);
        }


}
```

3) Find the sum of all digits of given number

n=12345 == > 1+2+3+4+5 = 15 [output]

```
int sum(int n)
{

}
```



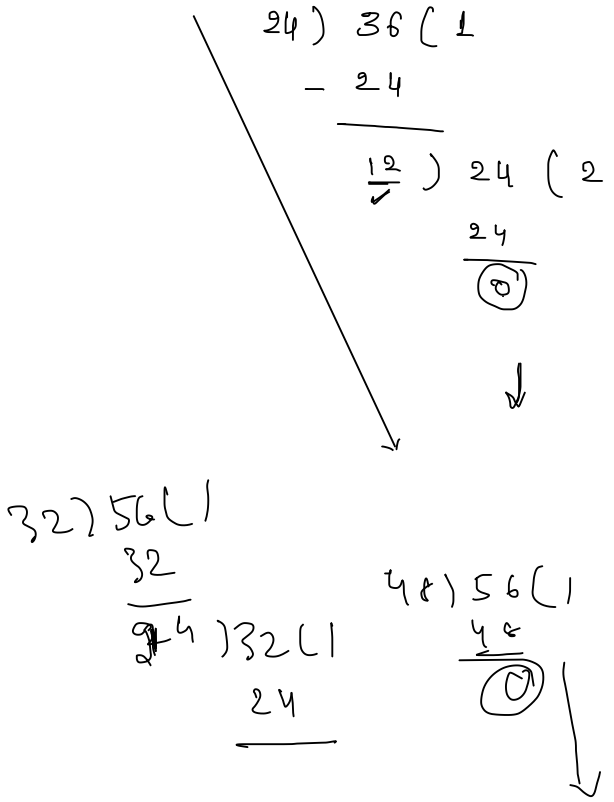
Greatest Common Divisor

(OR)

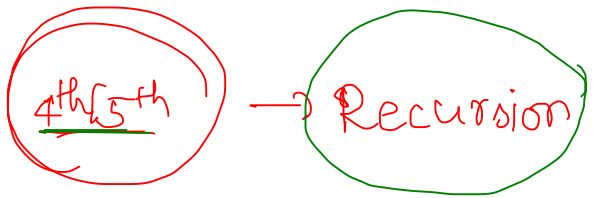
HCF

4) Implement GCD(a,b) :-

GCD(24,36) = ?



Studied ✓



prime factorization

24 = 2 x 2 x 2 x 3

36 = 2 x 2 x 3 x 3

} 2x2x3 = 12 ✓

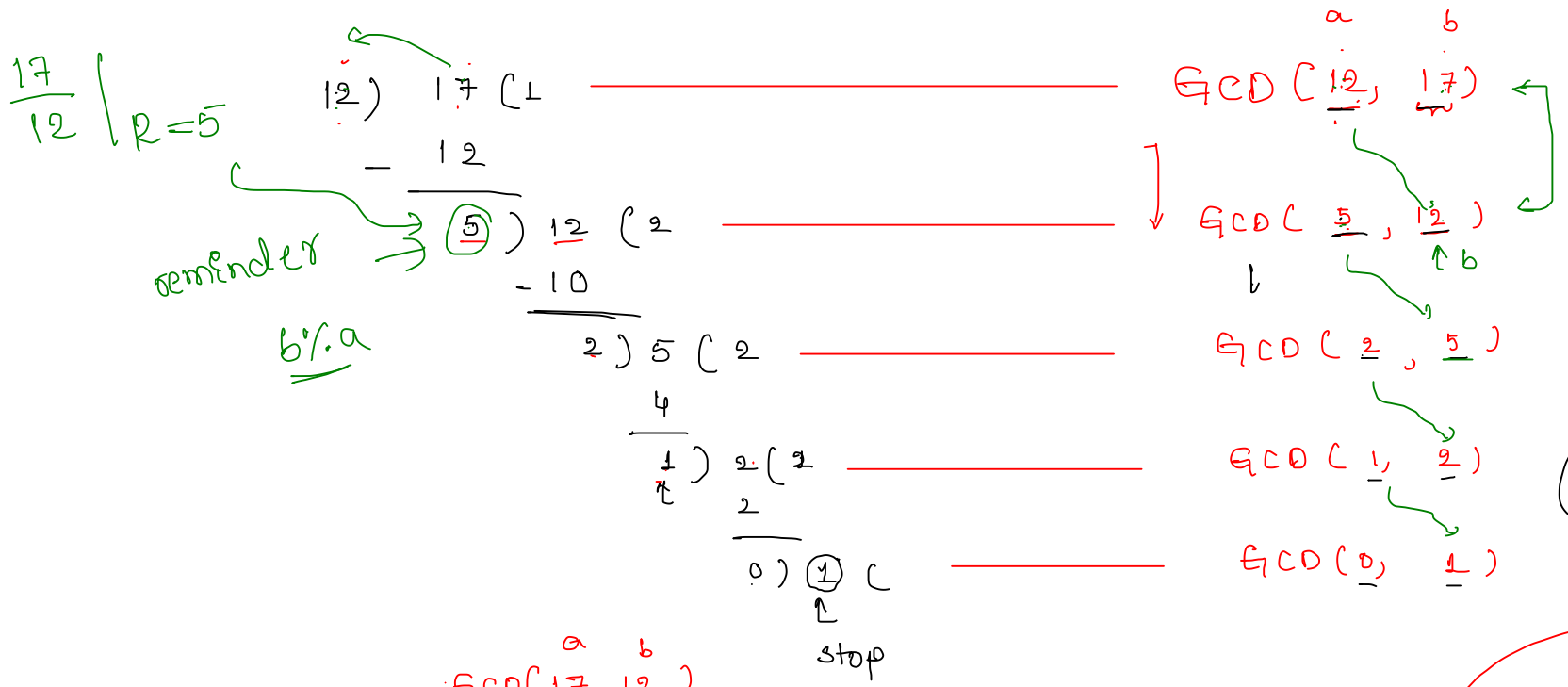
2 x 2 x

44 =

48

```
int GCD(int a, int b)
{



}
```

GCD(12, 17) = 1



GCD(17, 12)

↓↓

ret GCD(12, 17)

↓

Normal flow

12%17 = 12

GCD(12, 17)

↓

ret GCD(, a)

GCD(b-a, a)

```
int GCD(int a, int b)
{
        if(a==0)
                return b;

        return GCD(b%a,a);
}
```

Euclid

Mathematician

Algo

any two fibonacci numbers consecutive are w.c i/p to the Algo

GATE CSE 2021 Set 2 | Question: 23

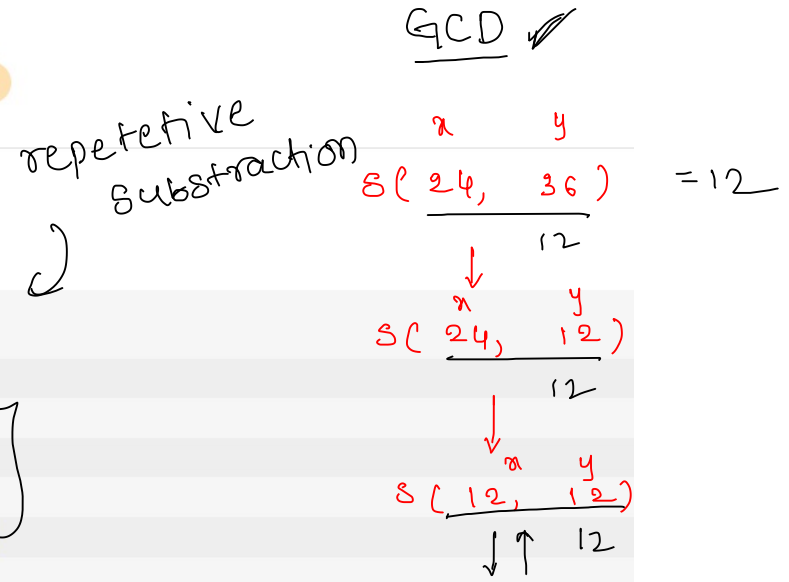
asked in **Algorithms** Feb 18, 2021 • retagged Nov 30, 2022 by **Lakshman Bhaiya**

9

Consider the following ANSI C function:

```
int SomeFunction (int x, int y)
{
    if ((x==1) || (y==1)) return 1;
    if (x==y) return x;
    if (x > y) return SomeFunction(x-y, y);
    if (y > x) return SomeFunction (x, y-x);

}
```

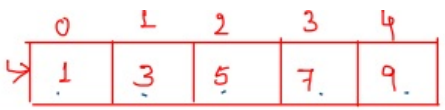
The value returned by $\text{SomeFunction}(15, 255)$ is _____



Array based recursion problems

5) Find sum of all elelements of array

```
int sum(int arr[], int n)
{
        if(n==1) return arr[0];

        return arr[n-1]+sum(arr,n-1);
}
```



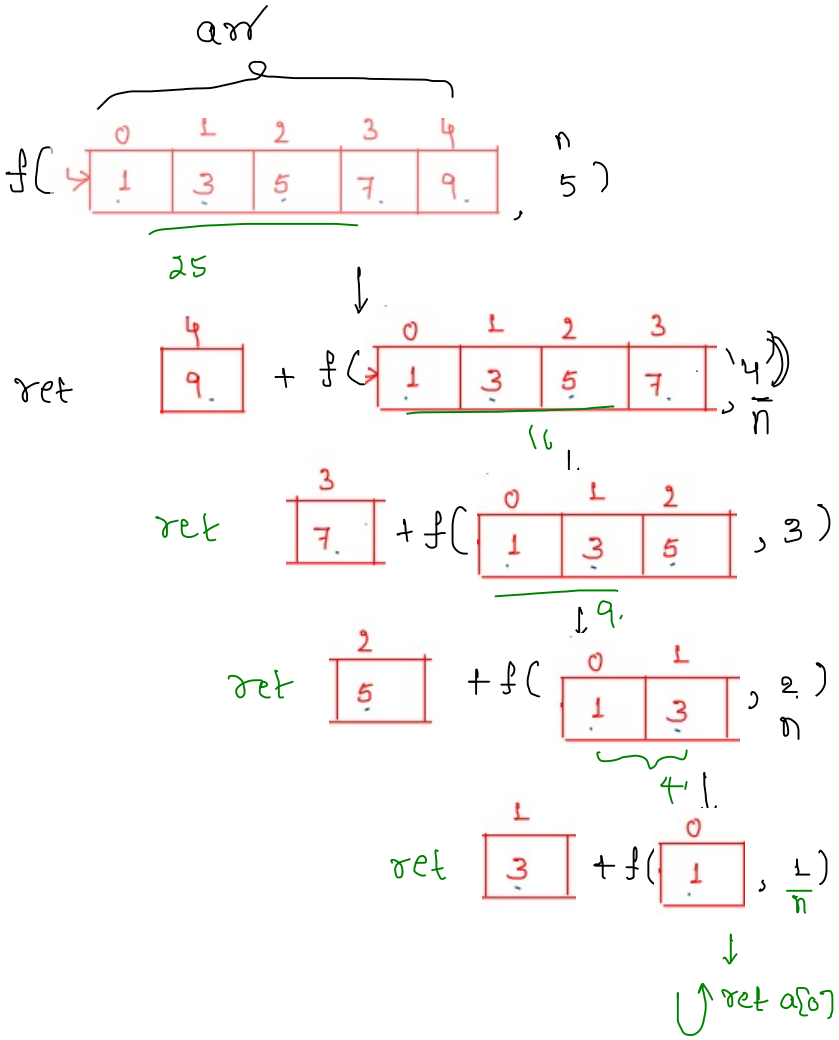
s(a, 5) ← # of eles

↓

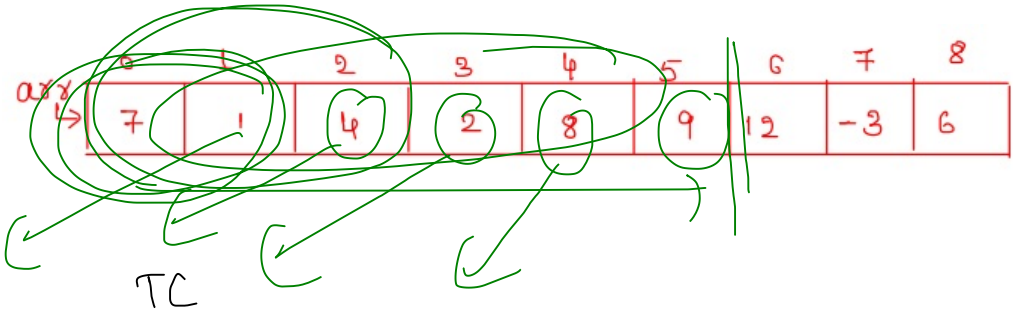
{ 9 + s(a, 4)

↺

a[n-1] + s(a, n-1)



6) Find maximum element of an array using recursion




```
int findMax(int arr[], int n)
{
        int res=arr[0];
        for(int i=1;i<n;i++)
        {
                res=Math.max(arr[i],res);
        }
        return res;
}
```

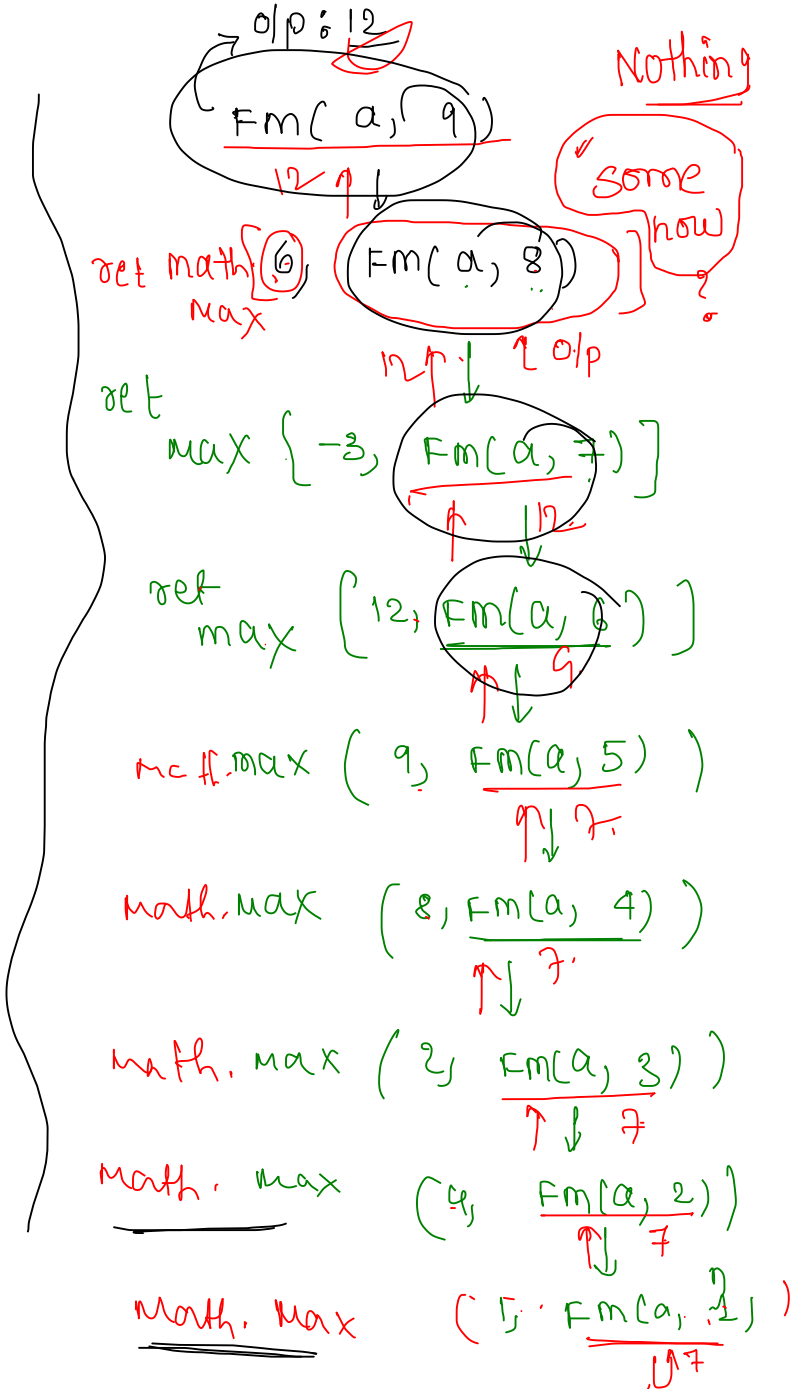
⇒ O(n) TC

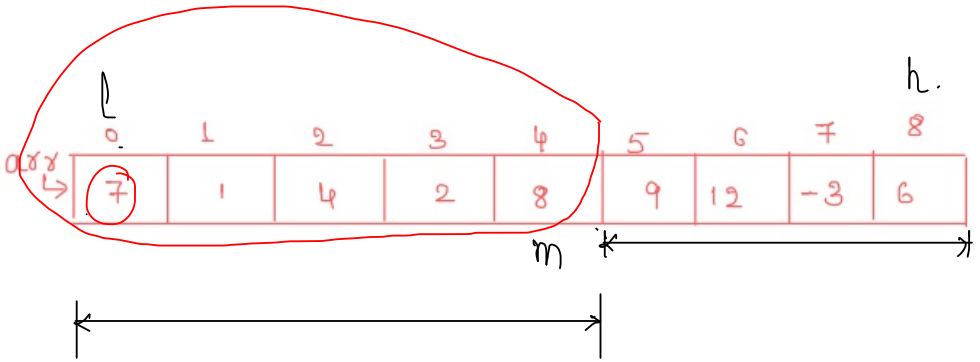
SC : O(1)

```
int findMax(int arr[], int n)
{
        if(n==1)
                return arr[0];
        else
                return Math.max( arr[n-1], findMax(arr,n-1) );
}
```

⇒ O(n) TC

SC : O(n) [Recursive stack space]





return Math.max(FM(arr,low,mid) , FM(arr,mid+1,high));

t_1 t_2

```java
public int fun(int arr[], int low, int high)
{
    if(low==high)
        return arr[low];
    if(low<high)
    {
        int mid=low+(high-low)/2;
        return Math.max(fun(arr,low,mid), fun(arr,mid+1,high));
    }
    return Integer.MIN_VALUE;
}
```