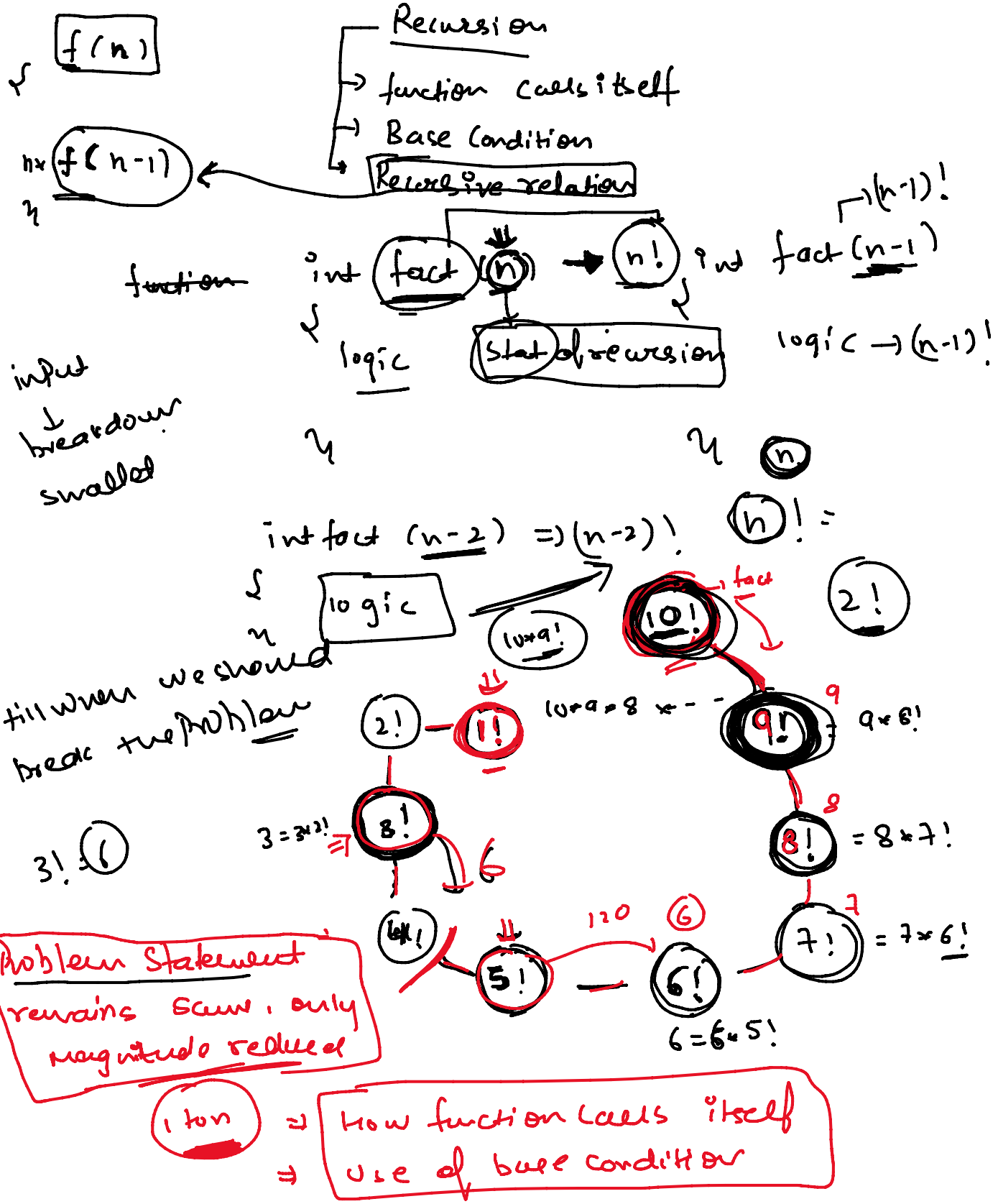
Class 45

29 March 2023 21:06

Class 45



* Reverse array using recursion

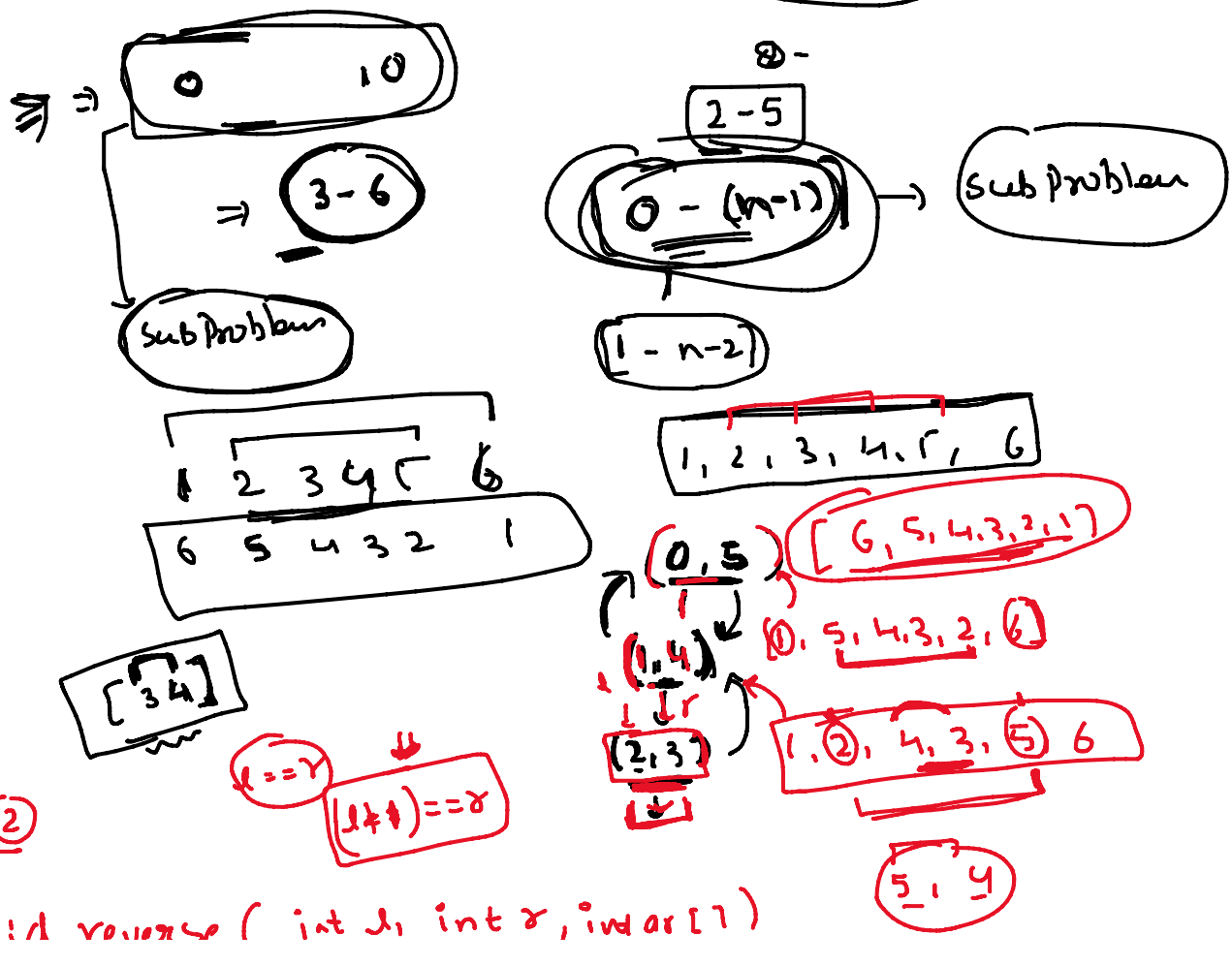
For loop

```
int s = 0;
int e = n-1;
while (s<=e)
{ swap (a[s], a[e]);
  s++; e--;
}
```

Problem Statement :- Reverse array (0 - n-1)

(0 - n-1)

Small problem → Subproblem



(5, 4)

```
void reverse ( int l, int r, int ar[] )
{        if ( l > r ) return;
   => f ( l+1, r-1 );
      swap ( a[l], a[r] );
}
```

=> subordinate and will always do work perfectly

[ ]

swap (a[l], a[r])

l r
[2, 3]

l r
[1, 2]
l r
1, (2) 3

swap (a[l], a[r])

(l+1, r-1)

Swap (a[l], a[r])

(l, r)

l r
[2, 3]
(1, 0)

(0, 1) -> swap

(1, 0) (Invalid range)

(0, n-1) -> O(1)
(1, n-2) -> O(1)
(2, n-3) -> O(1)
⋮
(l, l) -> O(1)

n/2

n

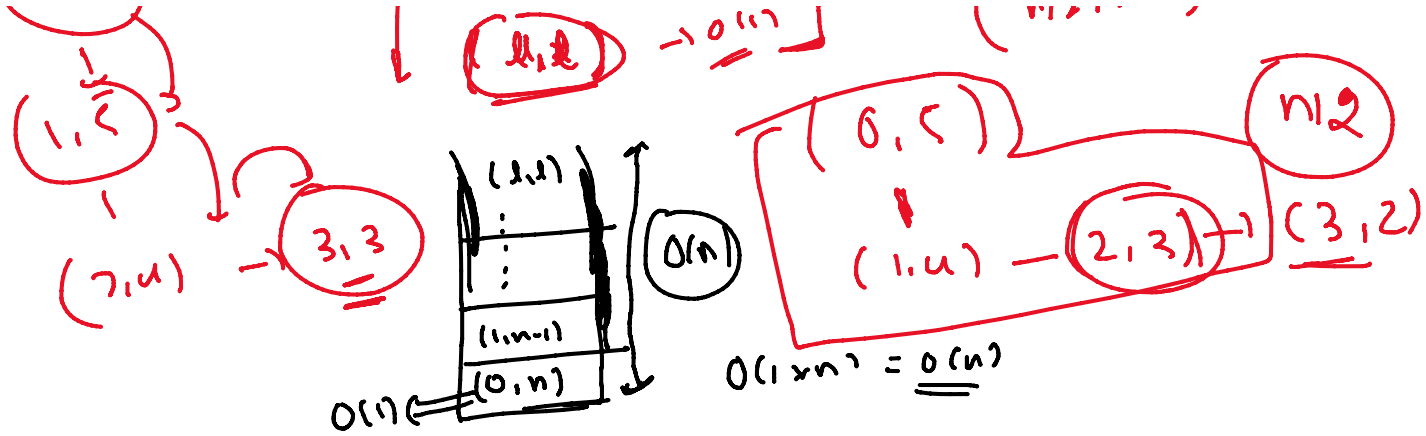
T.C => no of stacks × work done in each stack

n/2 × O(1) => O(n)

S.C = amount of space in each stack × Total stacks in stack

(0, n-1)
(1, n-2)
(n/2, n/2)

(0, 6)

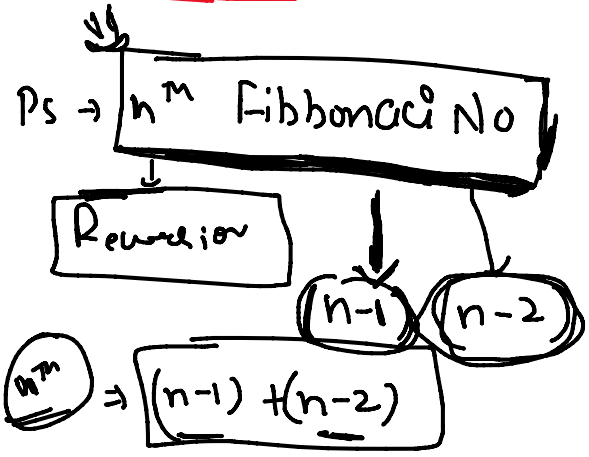


Function → single recursion call

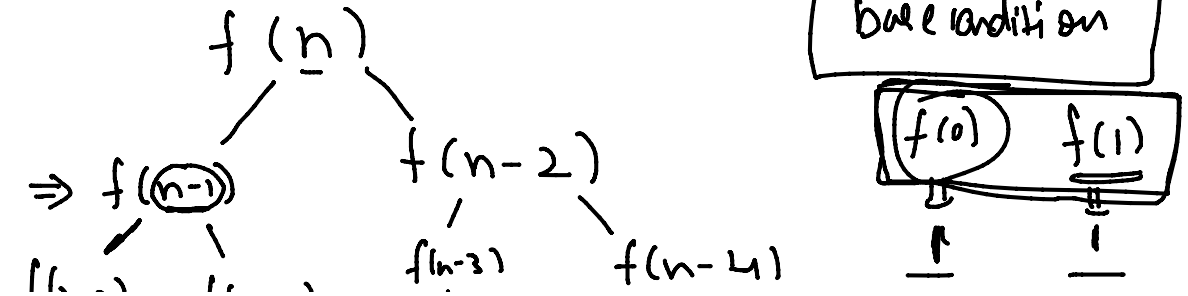
(2) Multiple recursion calls

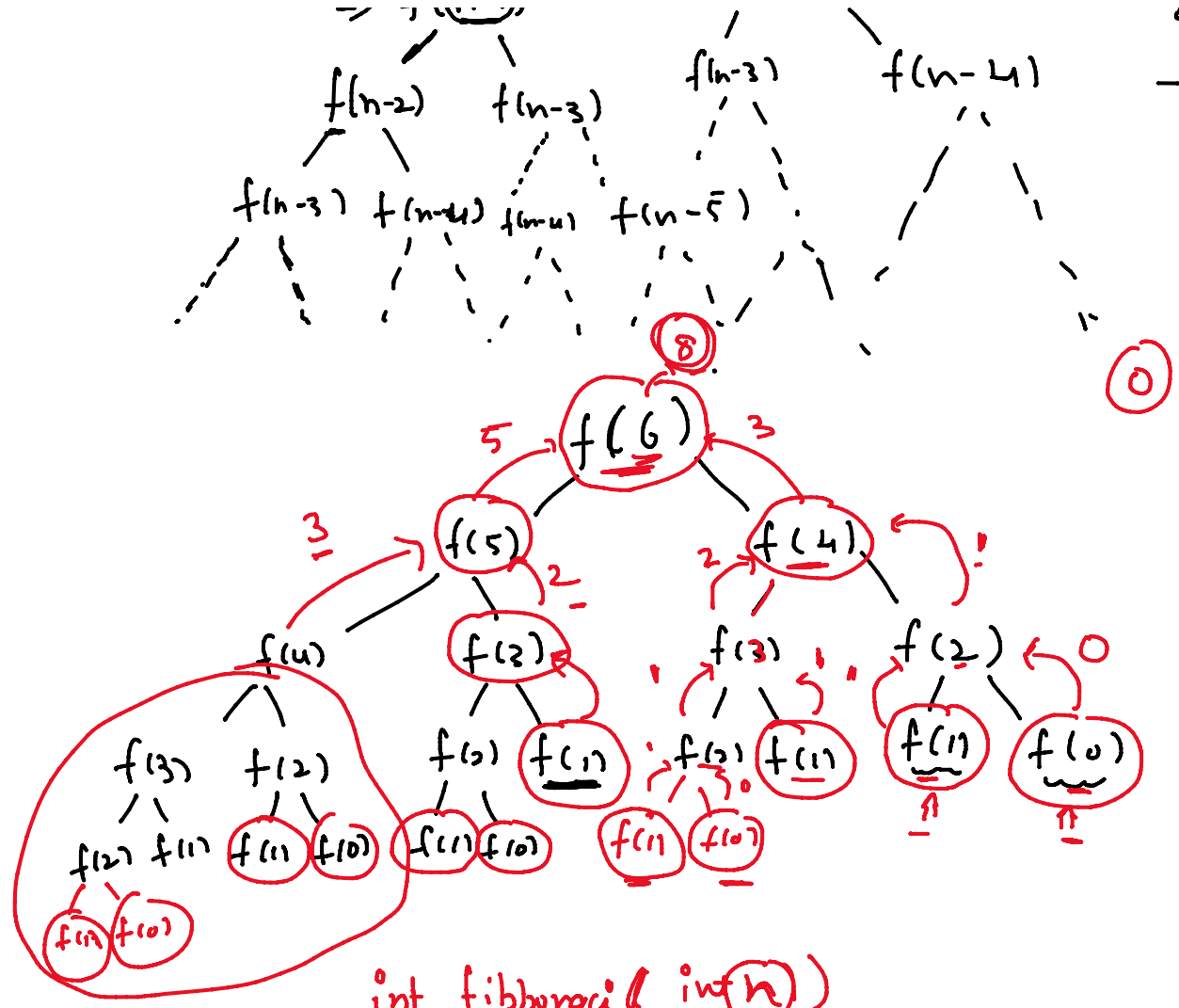
Fibbonaci series

0 1 1 2 3 5 8 13 21 - - .




```
For loop   int Preva = 0 ; int Prevb = 1 ;
For ( i = 2 ; i <= n ; i++)
{  int c = Preva + Prevb ;  c = n = 1
   Preva = Prevb ;          = 1
   Prevb = c ;              = 1
}
cout << Prevb ;
```





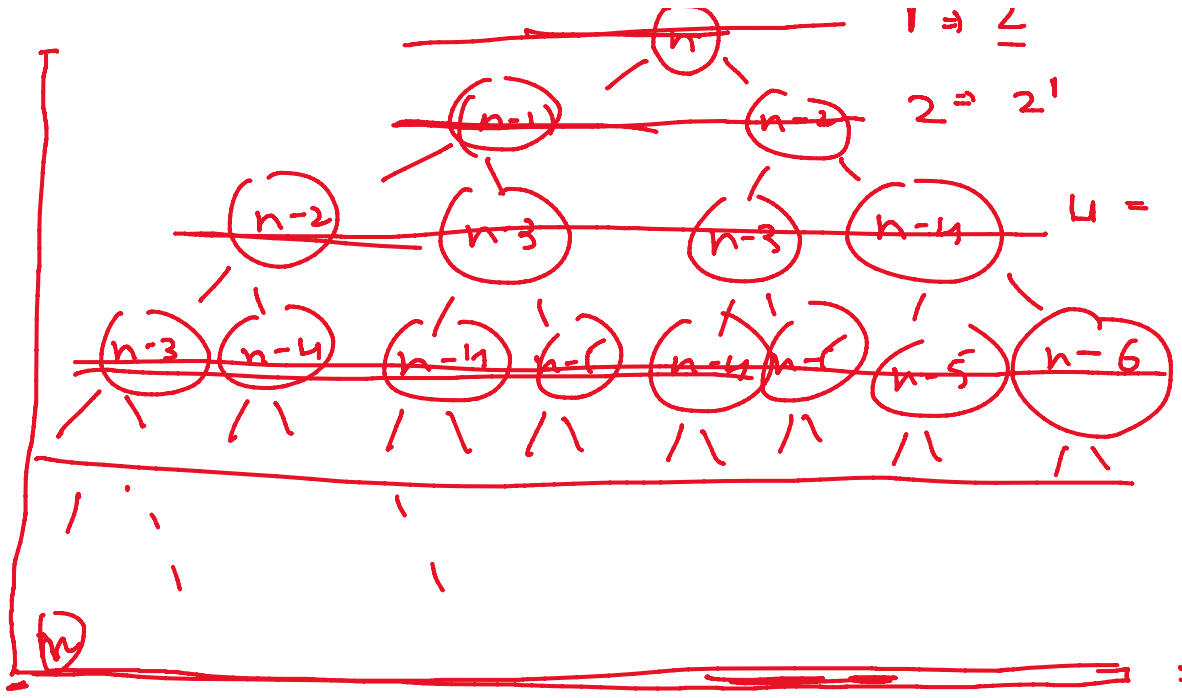

```
int fibbonaci( int n)
{   if(n==0)
      return 0;
    if(n==1)
      return 1;
                                  // Base condition
    int subo1 = fibbonaci(n-1);
    int subo2 = fibbonaci(n-2);
    return subo1 + subo2;
}
```

T.C = ?

T.C = No of steps in worst invocation

$1 \Rightarrow 2^0$

$n \Rightarrow 2^n$



$1 \Rightarrow 2^0$

$2 \Rightarrow 2^1$

$4 = 2^2$

$8 = 2^3$

$16 = 2^4$

$32 = 2^5$

$\Rightarrow 2^n$

Total Steps =

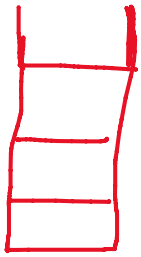
$2^0 + 2^1 + 2^2 + 2^3 + \cdots + 2^n \Rightarrow 2^n$

Exponential

$* O(1)$

$= O(2^n)$

$1\left[\frac{2^{n+1}-1}{2-1}\right] = 2^{n+1}$ $= 2 * 2^n = 2^n$



S.C = O(n)

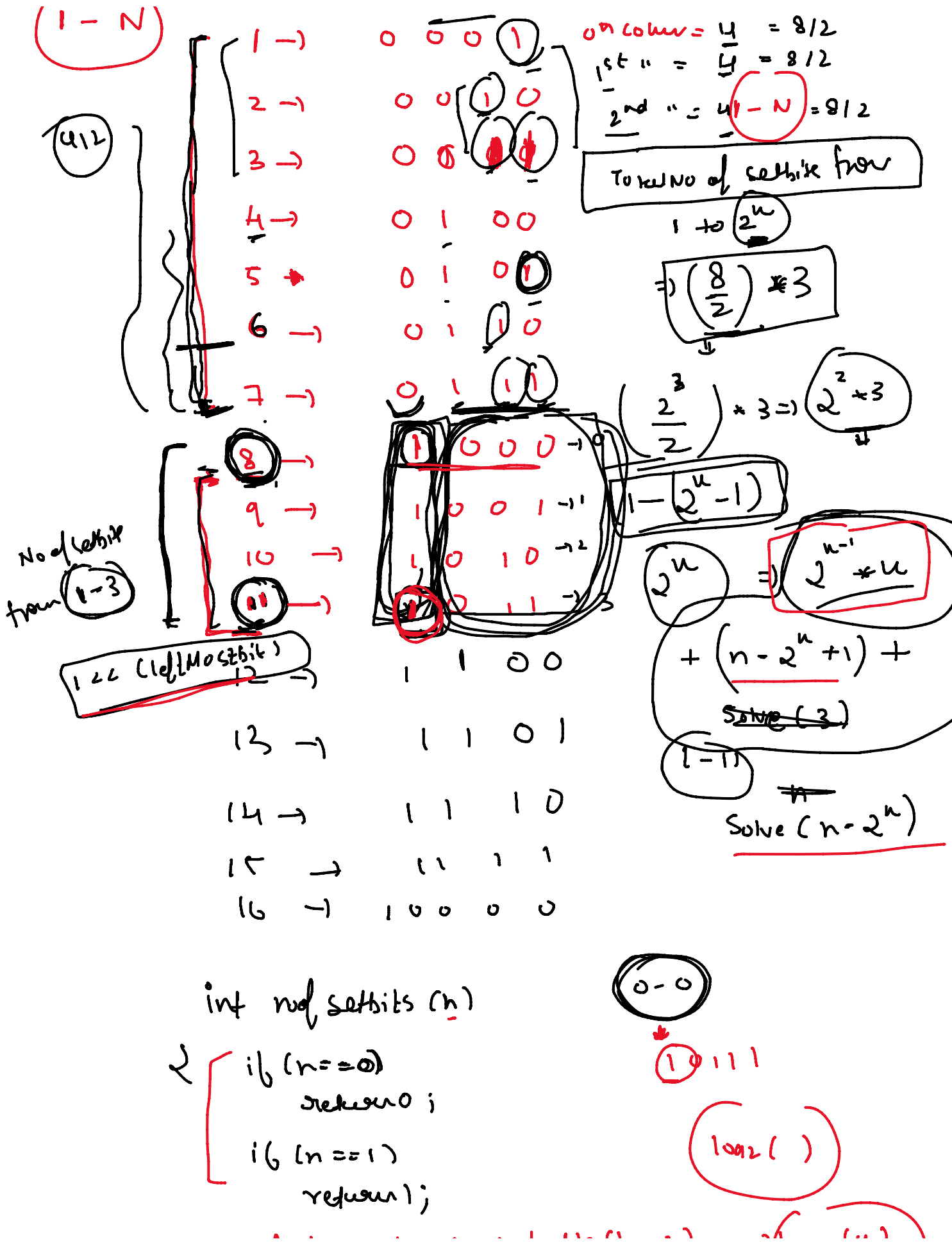
S.C = O(n)

Count No of Setbits

1 - N

$\lceil (1-) \quad 0 \quad 0 \quad 0 \quad (1) \rceil$ on count $= \frac{4}{4} = 8/2$

$\text{set} = \frac{4}{4} = 8/2$


1 - N
1 → 0 0 0 1
2 → 0 0 1 0
3 → 0 0 1 1
4 → 0 1 0 0
5 → 0 1 0 1
6 → 0 1 1 0
7 → 0 1 1 1
8 → 1 0 0 0 → 0
9 → 1 0 0 1 → 1
10 → 1 0 1 0 → 2
11 → 1 0 1 1 → 3
12 → 1 1 0 0
13 → 1 1 0 1
14 → 1 1 1 0
15 → 1 1 1 1
16 → 1 0 0 0 0
on column = N/2
1st " = N/2
2nd " = N - N/2
Total No of setbits from 1 to 2^n
(8/2) * 3
(2^3/2) * 3 => 2^2 * 3
1 - (2^n - 1)
2^n = 2^(n-1) * n
+ (n - 2^n + 1) +
Solve (3)
[-1]
Solve (n - 2^n)
No of setbits from 1-3
1 << (leftMostbit)
q12
int noofsetbits (n)
{
if (n==0)
return 0;
if (n==1)
return 1;
0-0
1 0 1 1 1
log2( )

return 1;

int lmb = log(n1) double(log2) $\Rightarrow \log_2(u)$

int ans = lmb * (1 << (lmb - 1));

ans += n - (1 << u) + 1;

ans += no of setbits (n - (1 << u))

return ans;