PW SKILLS

Recursion → function call itself

1) Factorial

1) Base case condition
   ↳ terminate
2) Recursive calls

fact(n) {

if(n==0 or n==1) {

return 1;

}

return n * fact(n-1);

0! = 1

1! = 1

n=5

5! = 5 x 4 x 3 x 2 x 1 = 120

fact(5)

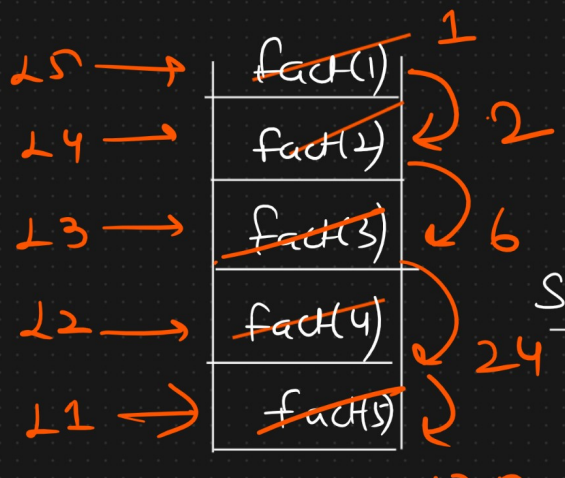
5 x fact(4) = 120

4 x fact(3) = 24

3 x fact(2) = 6

2 x fact(1) = 2

1



Store function call

Stack

↳ LIFO

O(n) - Space complexity

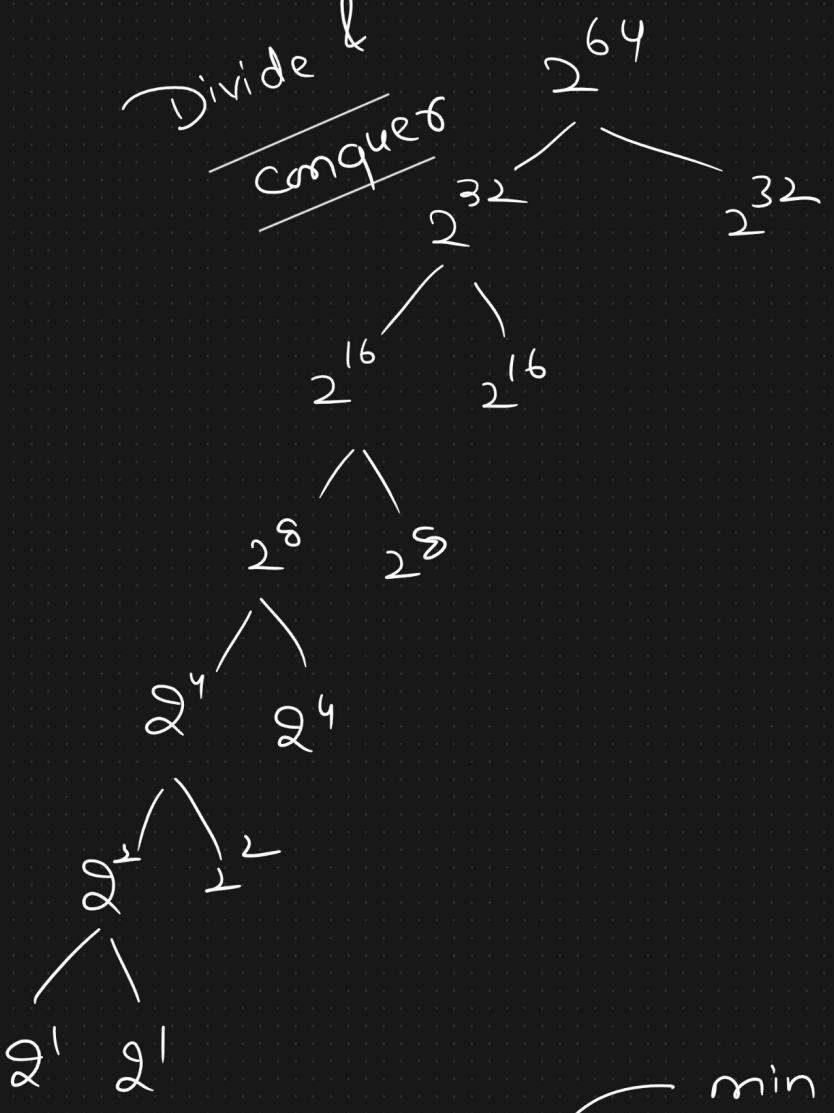
Recursion (Real time usage)

Advantages

1) Complex tasks $\xrightarrow{\text{split}}$ simpler task

Power function

Divide & conquer



2) Short & elegant — min time complexity & space complexity

3) Interview — 50:00 min
   ↳ 2 Problems

Disadvantage

1) Time Limit Exceeded

0	1	2	3	4	5
0	1	1	2	3	5

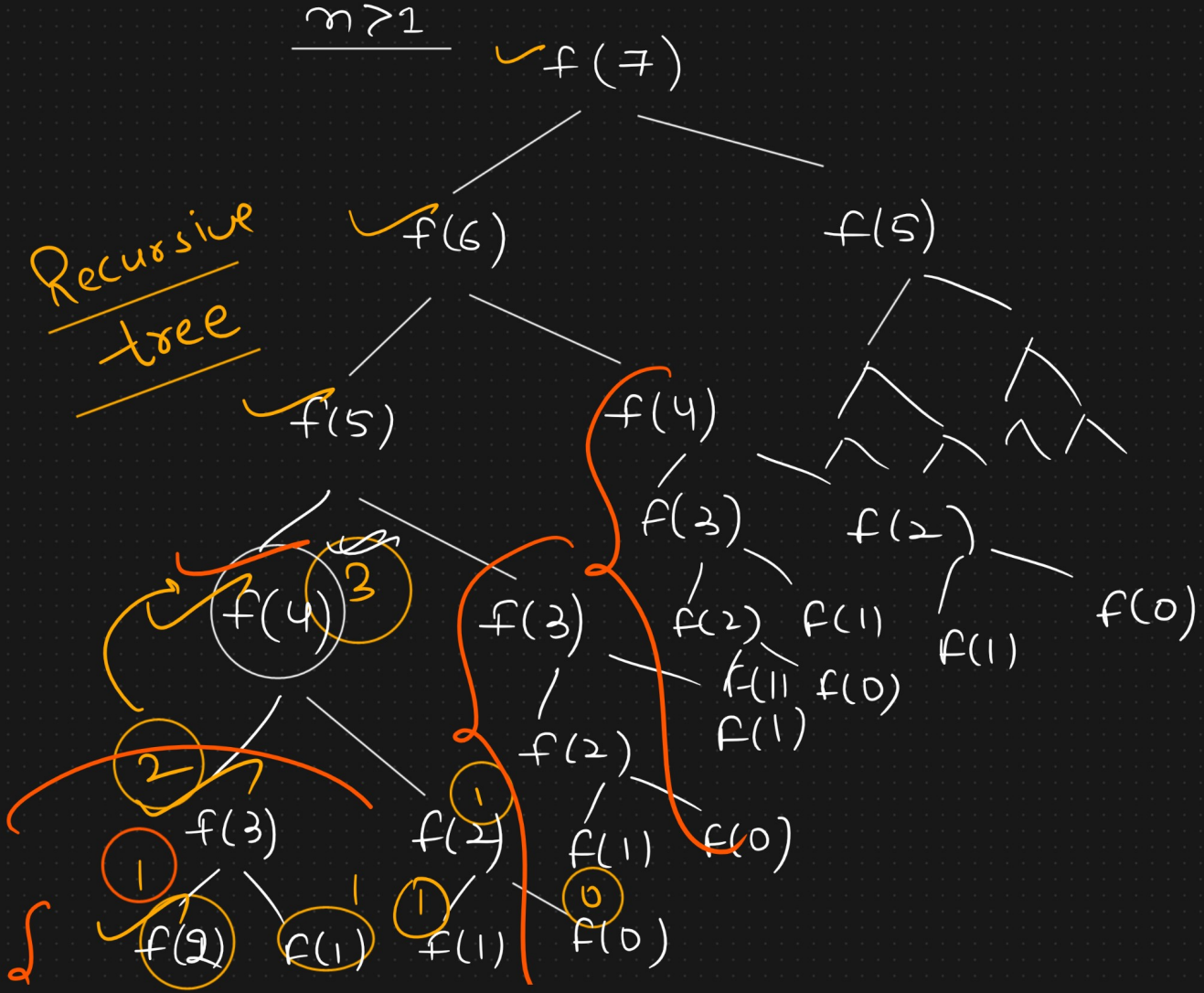
$$f(n) = \begin{cases} n & ; n \leq 1 \\ f(n-1) + f(n-2) & ; n > 1 \end{cases}$$

$$f(6) = f(5) + f(4)$$

$$= 5 + 3$$

$$= 8$$

$n > 1$



↳ Count of number of stairs

↳ Binary search — Recursion

↳ Multiplies

↳ k = 5, num = 4

4, 8, 12, 16, 20

↳ Sum of Digits

num = 12345

Sum = 1 + 2 + 3 + 4 + 5

= 15

Base case: num == 0 ↳ 0

Recursive call: num % 10 + sum(num/10)

-1 + 3 - 4 - 1

1 - 2 + 3 - 4 + 5 - 6 = -3

n = 6

$n = 5$

① Alternate sign series

$1 - 2 + 3 - 4 + 5 = 3$

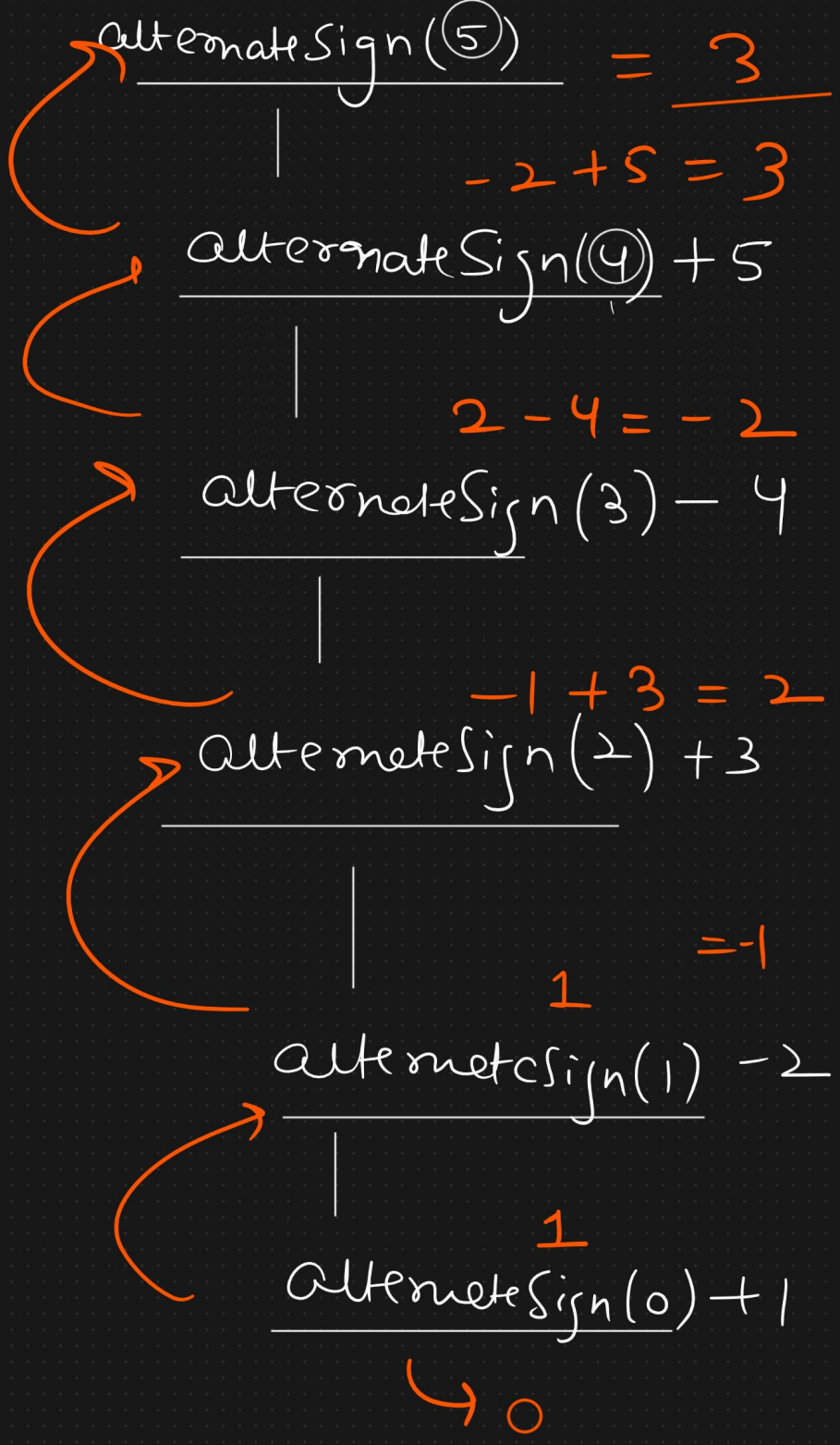
+ sign — Odd num

```
alternateSign(n) {          // n even
  if(n==0){ return 0; }
  if(n%2==0){
     alternateSign(n-1) - n
  }
  else {
     alternateSign(n-1) + n
  }
```

- sign — even

odd num

Recursive call



Hint:

n = matrix[0].length

m = matrix.length — # Rows

Binary Search

$\begin{cases} m = 3 \\ n = 4 \end{cases}$

arr(1)(1)

	0	1	2	3
0	1	3	5	7
1	10	11	16	20
2	23	30	34	60

Map

virtual array

Binary Search

0	1	2	3	4	5	6	7	8	9	10	11
1	3	5	7	10	11	16	20	23	30	34	60

Sorted array

right

low = 0, high = 11 — m×n − 1

while(low <= high){

midIdx = low + (high − low)/2;

midEle = arr[midIdx/n] (midIdx%n)
row col

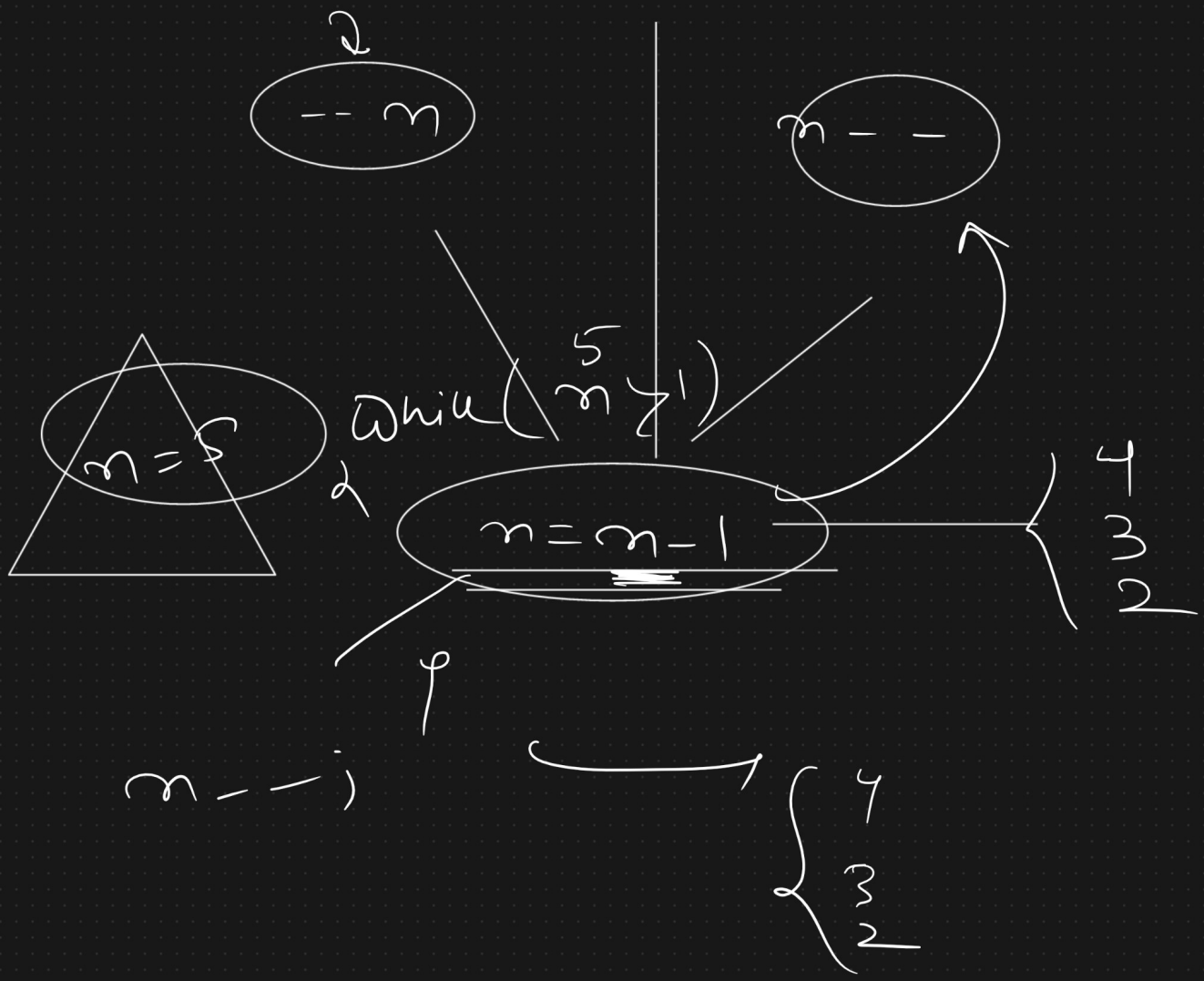
n — # columns

rowNum = 5/4 = 1

ColNum = 5%4 = 1

```
if(midEle == target){

    return true;
}
else if(midEle < target){
    right
        low = mid+1;
    }
    else {
        left
        high = mid-1;
    }
}
return false
}
```


2
--n
n--
5
while(n>1)
n=5
2
n=n-1
4
3
2
4
n--;
4
3
2