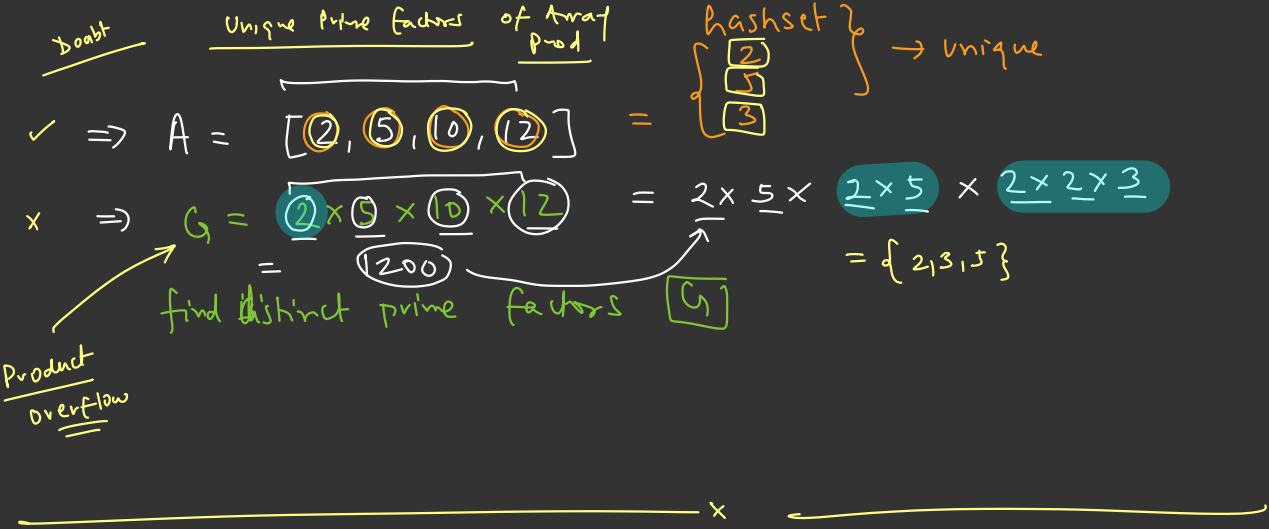

Doubt
Unique Prime factors of Array prod
hashset
2
5
3
→ unique
✓ ⇒ A = [2, 5, 10, 12] =
x ⇒ G = 2 x 5 x 10 x 12 = 2 x 5 x 2 x 5 x 2 x 2 x 3
= 1200
= {2,3,5}
find distinct prime factors
G
Product overflow

Recursion

Big Problem → Subproblems

$f(N)$

Magic Rule

1. Find out the smallest problem — Base Case
2. Assumption: Assume subproblem ~~$f(K)$~~ can be solved recursively for all $K < N$
3. Express $f(N) \rightarrow f(K)$ where $K < N$ — Recursive Case

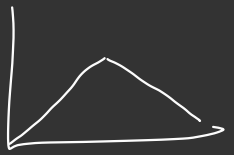
Factorial

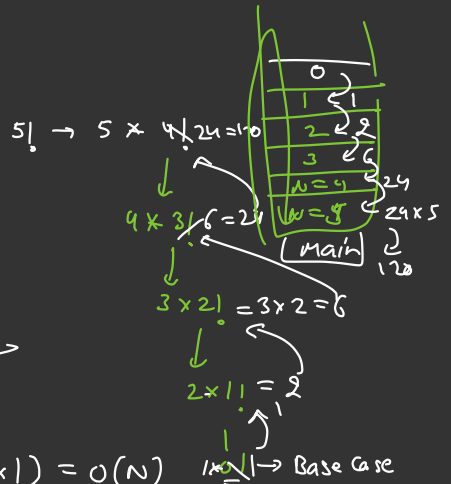
$5! = 5 * (4!)$

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

$f(0) = 1$

```
f(n){
    if(n==0)
        return 1
    return n * f(n-1)
}
```





→ $O(N \times 1) = O(N)$

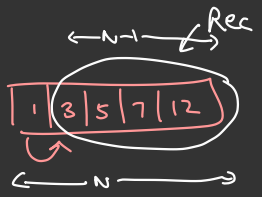
Time → Calls * work in each call

Space → Max stack Depth * space used in each call

$= O(N \times 1)$

$= O(N)$

Given an array, of size, check if it Sorted (inc order)




```
bool f(A, N) {
    Base case [ if(N==0 or N==1) {
                    return True
                }
    return A[0] < A[1] && f(A', N-1)
}
```

1<3 && 3 5 7 12 (A')

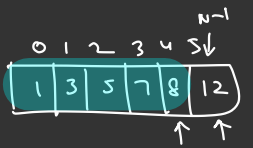
Create a new array O(N)

T 3<5 && A: 5 7 12 = T

T 5<7 && 7 12

T 7<12 && 12 ← N=1

T 12




```
bool f(A, N) {
    if(N==0 or N==1) {
        return True
    }
    return A[N-1] > a[N-2] && f(A, N-1);
}
```

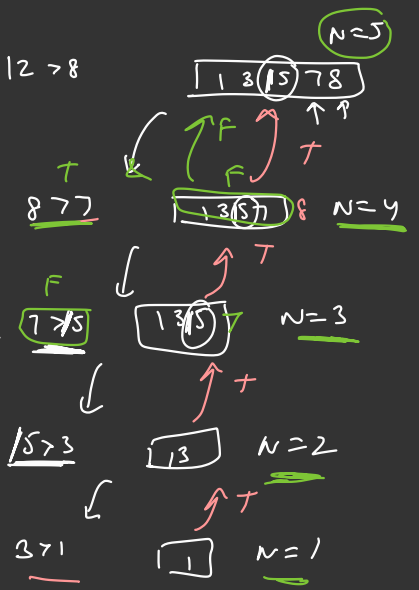
Base case

generate false value

N-1 as index to track what position to compare

same array

An array with N-1 elements



Time → O(N)

Space → Stack space due to multiple calls

↳ O(N)

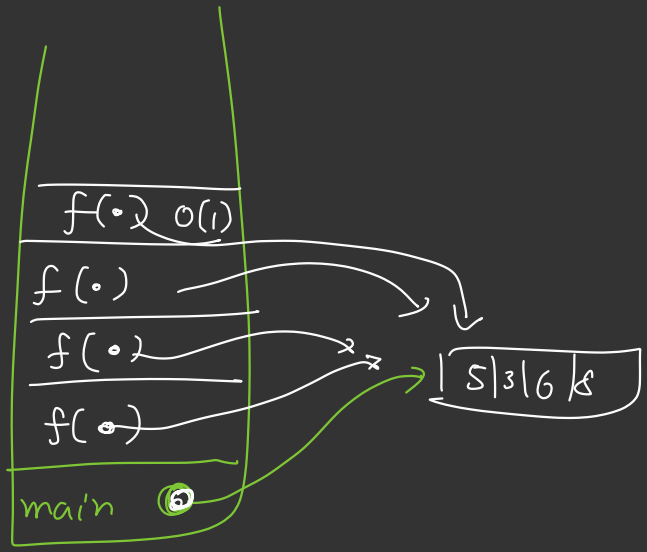
Recursion ALWAYS come with extra space overhead

N = 1

N = 2

N = 3

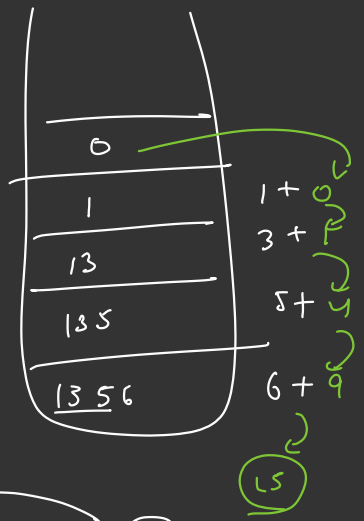
N = 4



Q Given a N find sum of it digit

N = 1 3 5 6

Expected = 15




```
int f (N) {
    if (N == 0)
        return 0
    return N%10 + f(N/10)
}
```

another way

```
if(N<10)
    return N
```

Base case

1356 → 1 3 5 6

1356%10 = 6 + f(N/10)

(9) N friends are going for a party. Each friend can go Solo or as a couple. Find out the total ways in which they can go.

N = 3 A, B, C

(AB) C
(AC) B
(BC) A
(A) (B) (C)

} 4 ways.

N = 4 A, B, C, D

solo + Couple

$$f(N) = 1 \cdot f(N-1) + (N-1) \cdot f(N-2)$$

(A · $8f$)

A B . . . . N friends

Party

$f(N-1)$

$f(1) = 1$

$f(2) = 2$

Base Case

(A)

1.

(BC) D, E

(BD) (CE)

many ways

N-1

B C D E F

Pick 1 out N-1

↳ N-1 choices

N-2 ppl

↓

$f(n-2)$ ways

A B

(A) (B)

(A, B)

2 ways

(A D)

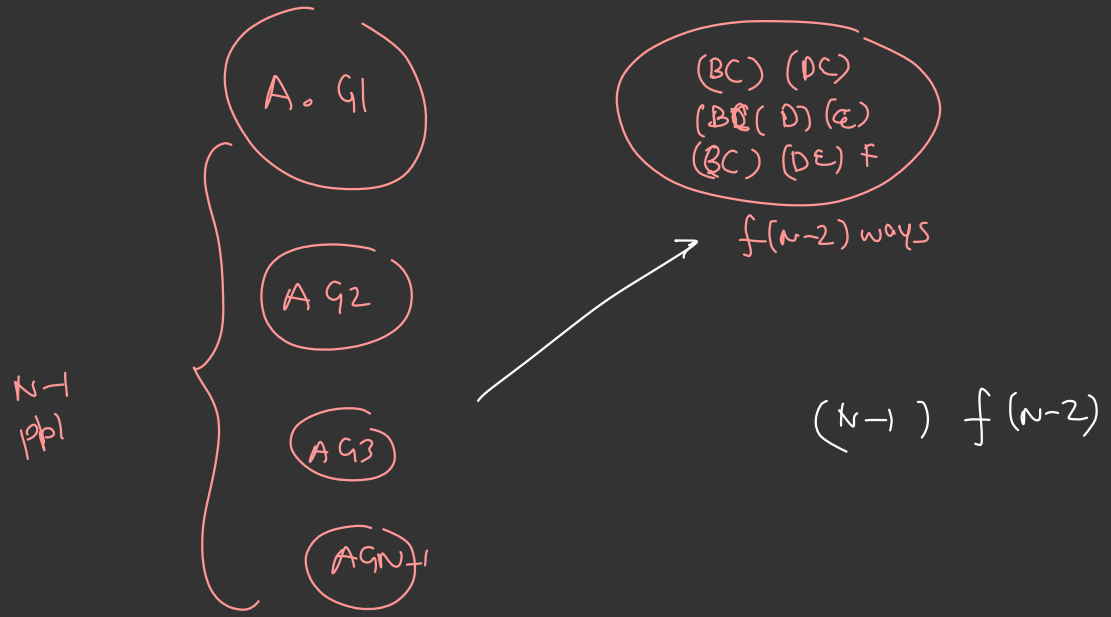
$$f(3) = f(2) + 2 \cdot f(1)$$

$$= 2 + 2 \cdot 1 = 4$$

$$f(4) = f(3) + 3 \cdot f(2)$$

$$= 4 + 3 \cdot 2$$

$$= 10$$


A. G1
A G2
A G3
AGN-1
N-1 ppl
(BC) (DC)
(BC) (D) (E)
(BC) (DE) f
f(N-2) ways
(N-1) f(N-2)

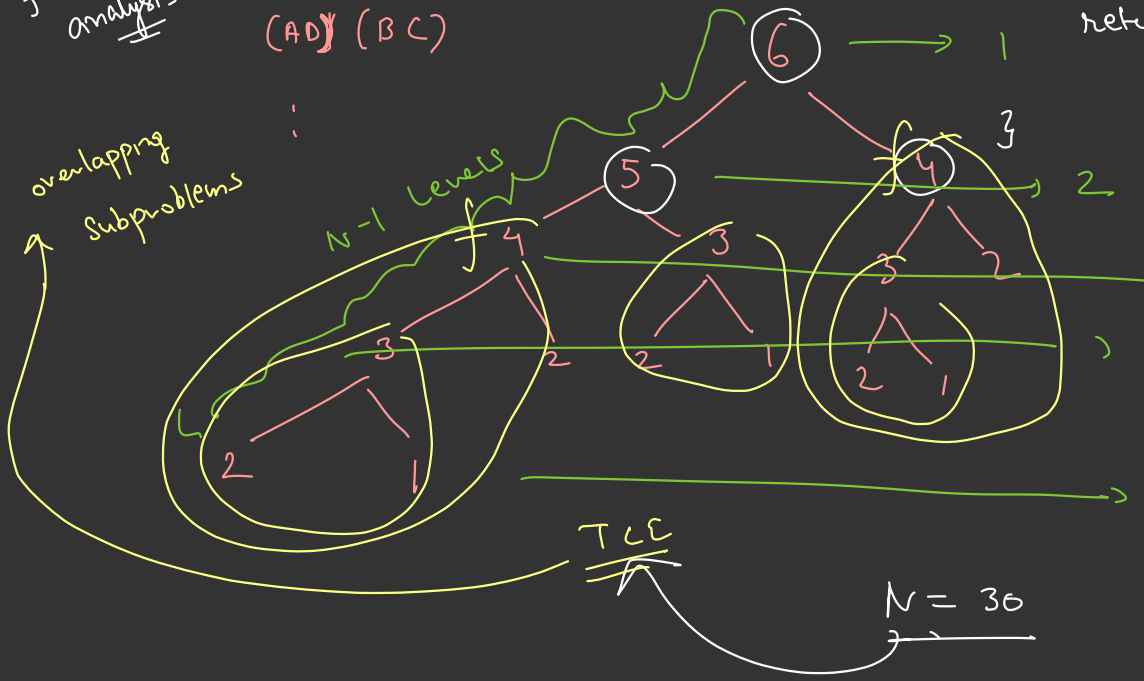
⇒ A, B, C, D

✓, ✓, ✓, ✓

(AB) (CD)
(AC) (BD)
(AD) (BC)
⋮

Revisit
fibonacci
analysis

```
int f(N) {
  if (N <= 2)
    return N
  return f(N-1) + f(N-2)
}
```



$1+2+4+8+\cdots 2^{n-2}$

$= O(2^n)$

Time

$2^{30} = (10^3)^3$

$= 10^9$ steps

② Money Change

Indian currency = [1, 2, 5, 10, 20, 50, 100, 200, 500, 2000]

denom []

₹ 276 → get a change using min Notes / coins.

= 200 + 50 + 20 + 5 + 1 } Min

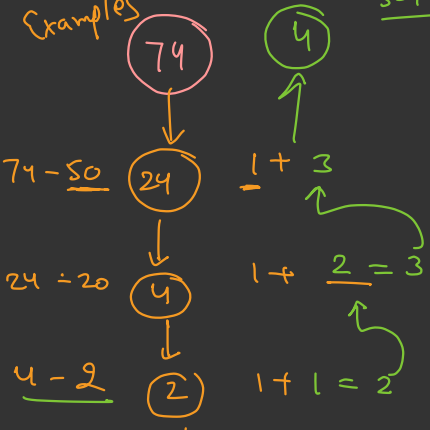
= 276

N money 5 Notes / coins

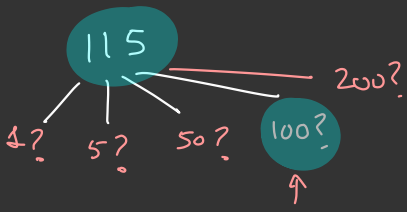
50, 20, 2, 2

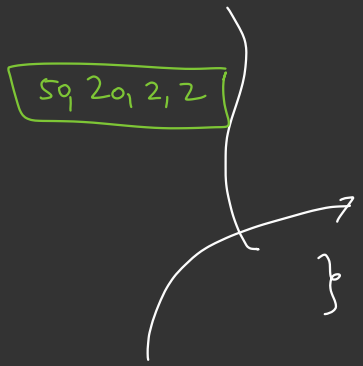
Examples

```
int f (money) {
    if (money == 0) {
        return 0;
    }
    // the
```



15 Mins





int note = findLargest(money)

print(note);

total_notes = 1 + f(money - note);

return total_notes

1 + 0 = 1

O(1)

Largest No ≤ money

115

1+ -100

15

-10

5

-5

0

O(No of terms) → Linear Search, O(log terms) Binary Search

[1, 2, 5, 10, 20, 50, 100, 200, 500, 2000]

Constant Size

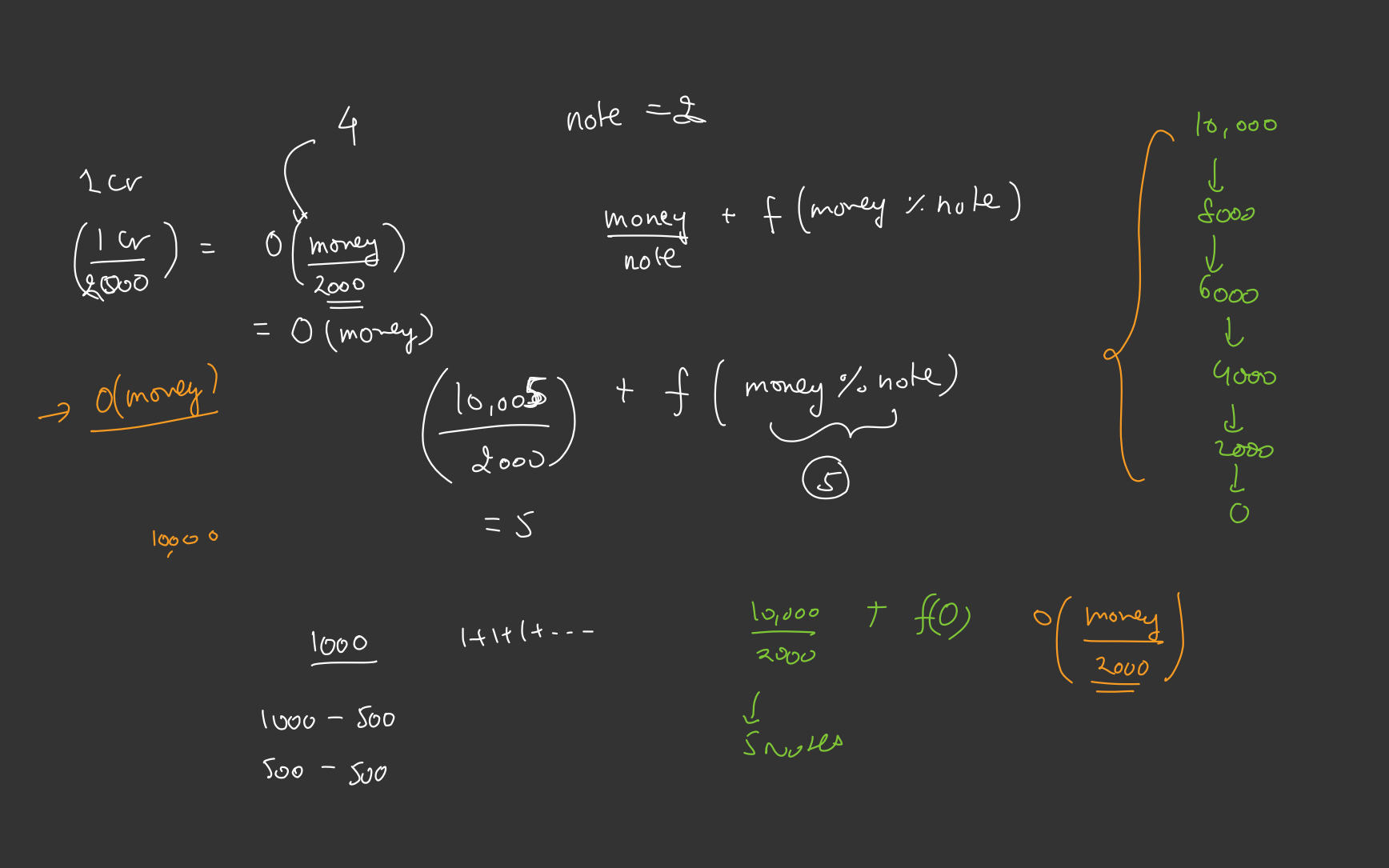
4240 → 2000

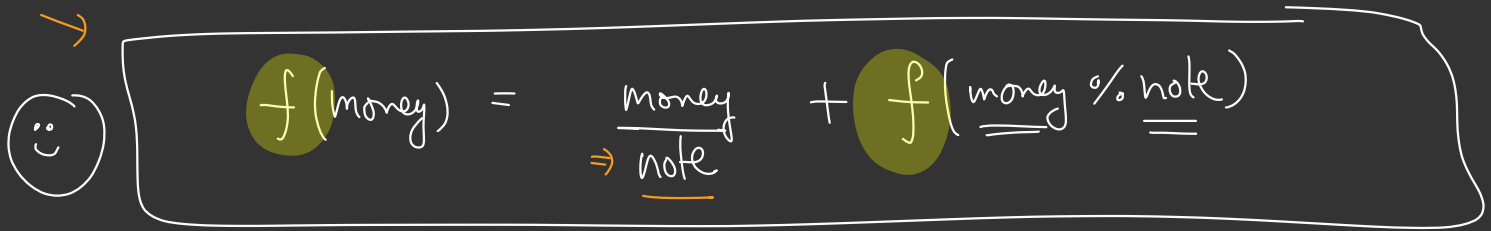
156 → 100

224 → 200

688 → 500

Time ⇒ 10,000


4
note = 2
1 cr
$\left(\frac{1\,cr}{2000}\right) = O\left(\frac{money}{2000}\right)$
$= O(money)$
$\frac{money}{note} + f(money \% note)$
10,000
5000
6000
4000
2000
0
$\rightarrow O(money)$
$\left(\frac{10,000}{2000}\right) + f(money \% note)$
5
$= 5$
10,000
1000
1+1+1+---
1000 - 500
500 - 500
$\frac{10,000}{2000} + f(0)$
5 notes
$O\left(\frac{money}{2000}\right)$



Every note type can be picked once

↓

O(1)

74 / 50 + f(24) = 1+3 = 4

24 / 20 + f(4) 2 = 1+2 = 3

[4 / 2] + f(0) = 2

f(0) = 0

money

0	cr

-2000
-2000
-2000

var → money Steps

const → 2000

O(money)

$\frac{10cr}{2000} + f(0)$

0

7 → 2 → 0

1

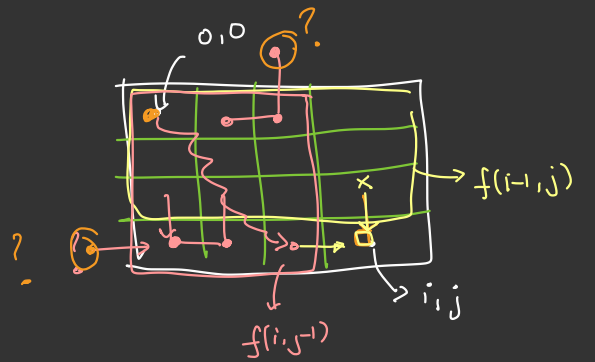
1

(5) + (2) =

1 + 1 = 2 steps

{ O(money)
O(1) = O(Notes Array) }

(9)

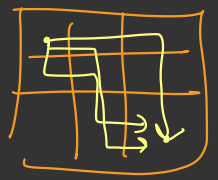


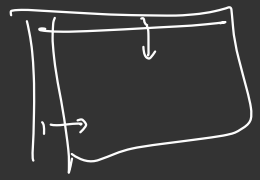
→ Right

↓ Down

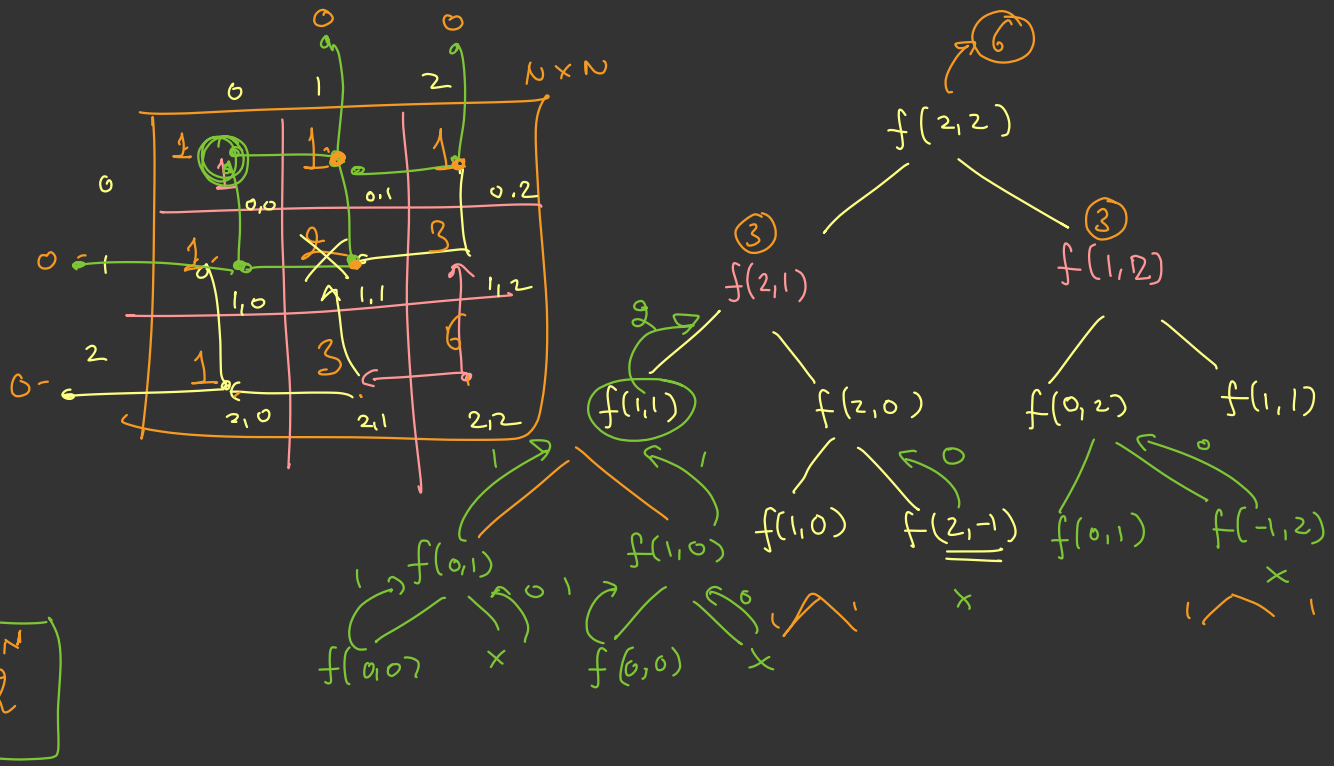
at any cell

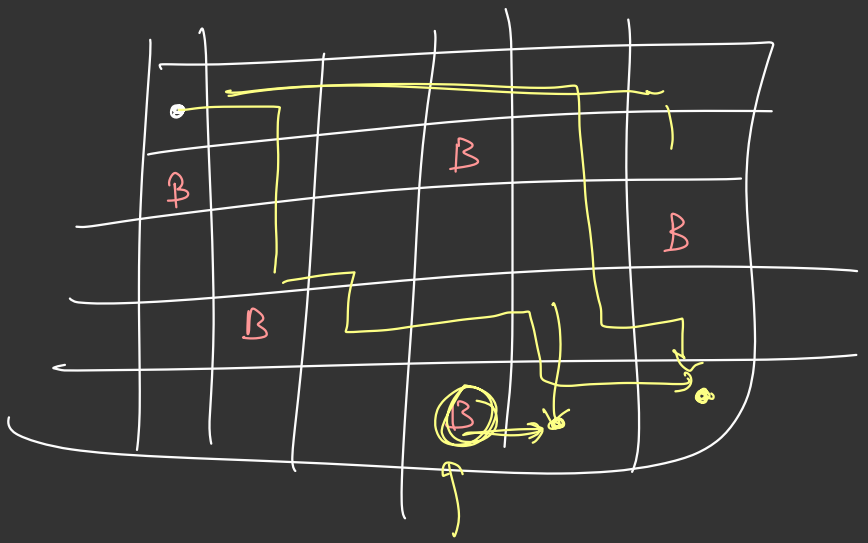
How many ways I can go cell i,j

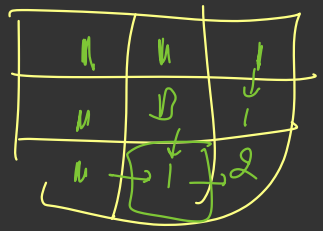




```
int f ( i , j ) {
    if(i==0 || j==0)
        return 1;
    if (out of Bounds) {        // or cell is blocked i,j
        return 0;
    }
    return f(i-1,j) + f(i,j-1);
}
```


N x N
0,0
0,1
0,2
1,0
1,1
1,2
2,0
2,1
2,2
f(2,2)
f(2,1)
f(1,2)
f(1,1)
f(2,0)
f(0,2)
f(1,1)
f(1,0)
f(2,-1)
f(0,1)
f(-1,2)
f(0,1)
f(0,0)
2^N


B
B
B
B
B
f(blocked) = 0


1
1
1
1
1
1
1
2
B