Recursion - II

Recursive Code

1. **Base Case** – smallest instance of the problem
   ↳ directly know the answer for base case
2. **Assumption** – Smaller **Subproblem** the solution exists
3. **Main Logic** – Express a big problem in terms of one or more subproblems

Q1. Factorial n

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

Assumption

Recursive Relation

$$f(0) = 1$$

Base

Q2. Tiling Problem

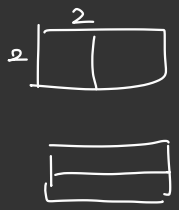
$$f(N) = f(N-1) + f(N-2)$$

Base Case

$$\begin{cases} f(1) = 1 \\ f(0) = 1 \end{cases}$$

OR

$$\begin{cases} f(2) = f(1) + f(0) = 2 \\ f(1) = 1 \end{cases}$$



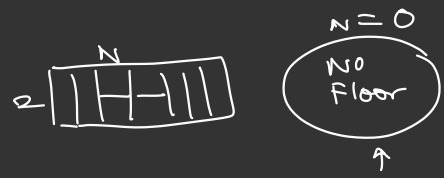
$f(3) = 2 + 1 = 3$

$f(4) = 3 + 2 = 5$

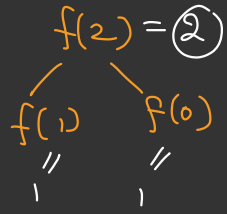
$f(5) = 5 + 3 = 8$

$f(4) + f(3)$

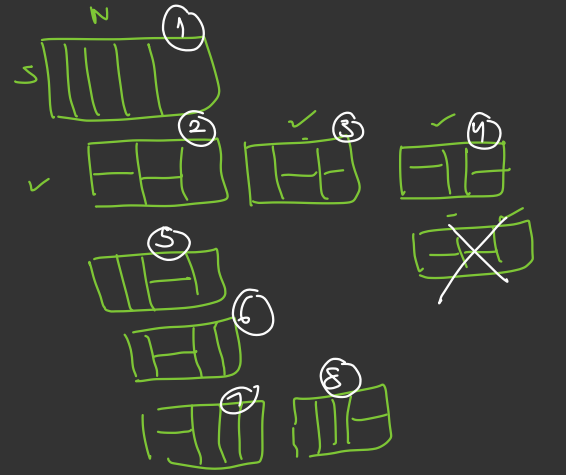
5 3



Not placing any tile is also a way.



Drawing



Rules for Space & Time Complexity in Recursive Programs

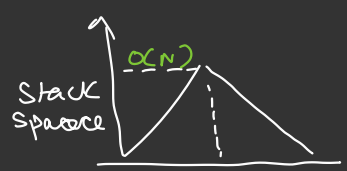
Time Complexity = Total calls * time in each call

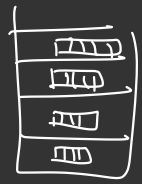
```
fn() {
    fn(n-1) + f(n-2)
    for(i=0 — N){
    =
    }
}
```

Space = Max Space Utilisation during any point execution

= Call Stack Depth × Space used by each stack frame / fn call.

Max when we hit the base case





Fibonacci Analysis

[Same for Tiling Problem]

Base Case

Series → 0, 1, 1, 2, 3, 5, 8, 13 .....

n-2 n-1

n^{th} term

0, 1, 1

Rec case [$f(n) = f(n-1) + f(n-2)$]

f1 f2

Base case [$f(0) = 0$, $f(1) = 1$]

CODE

```
int f(int n) {
L1→ if(n==0 or n==1) { return n; }
L2→ int f1 = f(n-1);
L3→ int f2 = f(n-2);
L4→ return f1+f2;
}
```

Two Subproblems

or return f(n-1) + f(n-2)

Call Stack & Recursion Tree Diagram

0, 1, 1, 2, 3, 5, ...

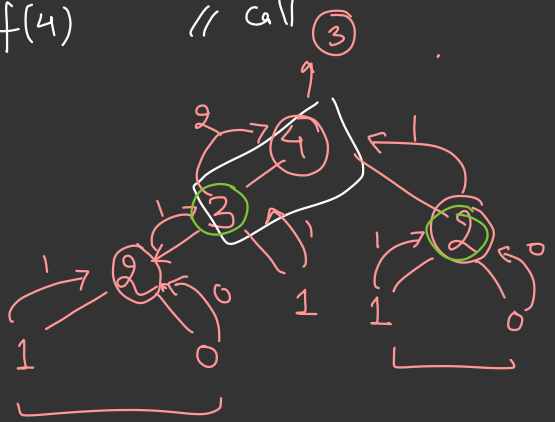
n=0 n=1 n=2 n=3 n=4 n=5

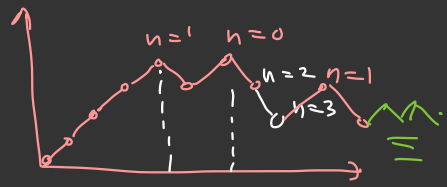
n = 4

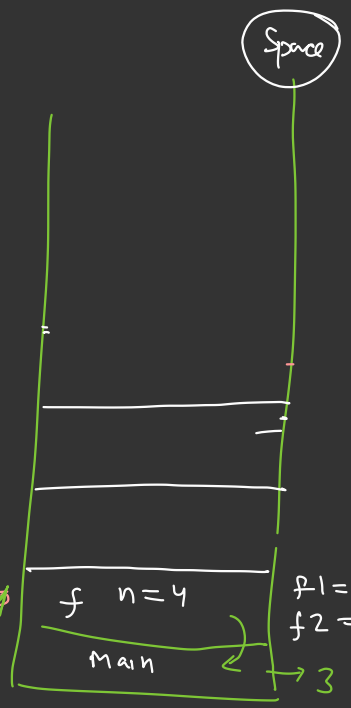
Main() {

f(4) // call

3







→ you don't need to draw the stack

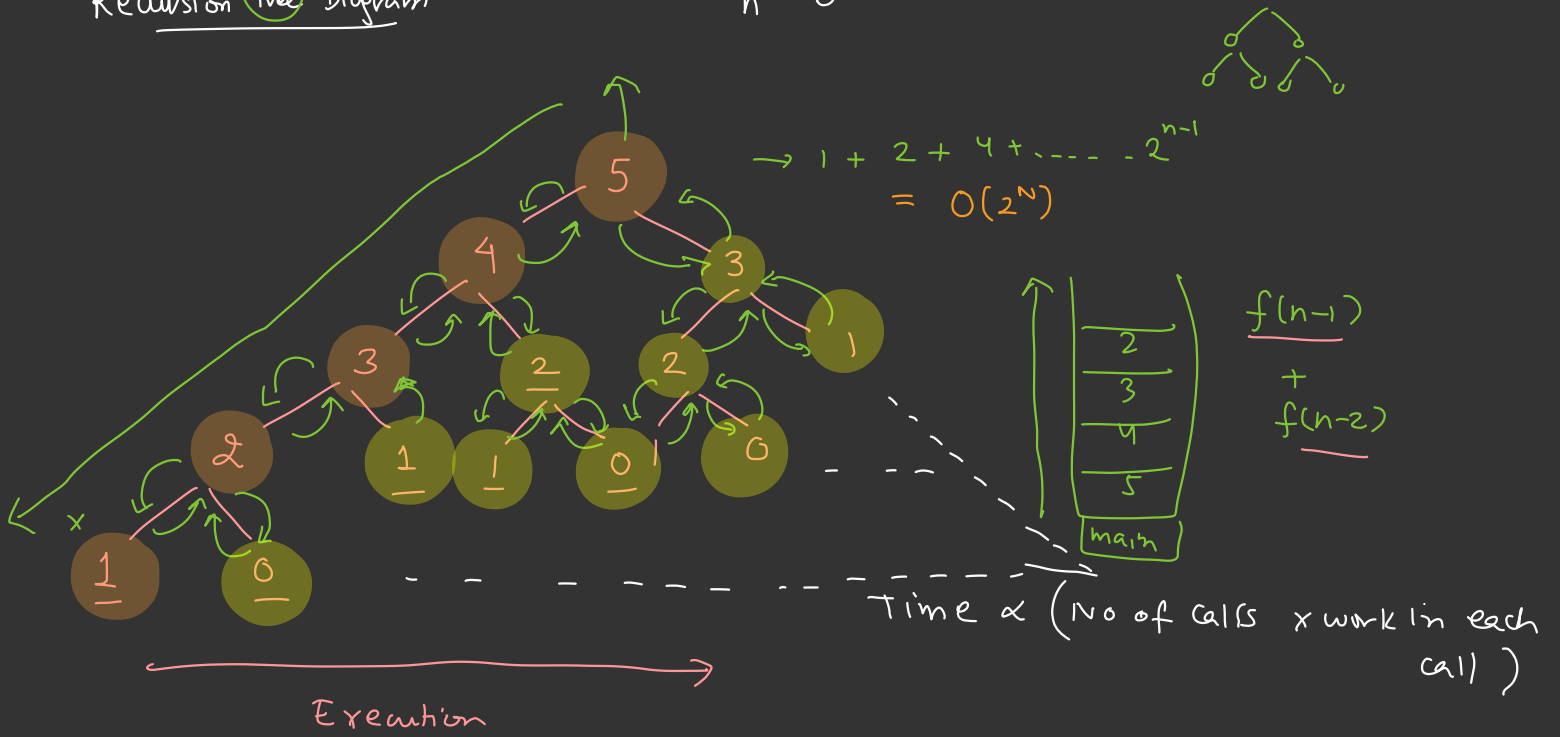
f1 = 2

f2 = 1

f(4-2) = f(2)

Recursion Tree Diagram

$n = 5$

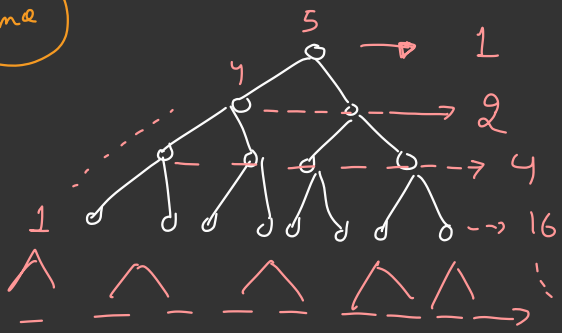


Space → (Max Depth of Stack = Height of Tree) x Space in each call

$O(N)$ — f1, f2 — $O(1)$

= $O(N)$ space

Time



$2^{1-1} = 1$

$2^{2-1} = 2$

$2^{3-1} = 4$

n level $= 2^{n-1}$ fn calls on N Level

Total Calls = $1 + 2 + 4 + 8 + \ldots\ldots 2^{n-1}$ [Geometric Prog]

$= a\,\frac{[r^n - 1]}{r-1}$ [Sum of GP]

$= 1\,\frac{[2^n - 1]}{2-1} = 2^n - 1$

Time = $2^n \times O(1)$ work in each call → Addition + Base case

$= O(2^N)$

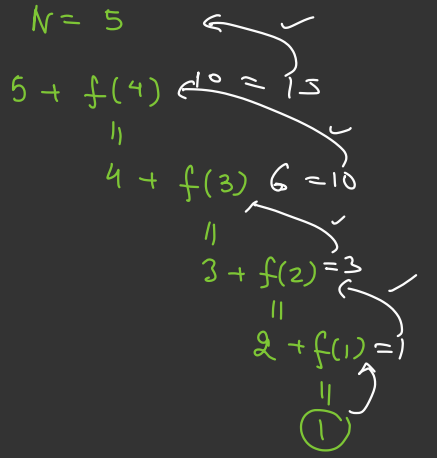
Bad for Fibonacci

⑥ Find Sum of first N (Natural) Numbers

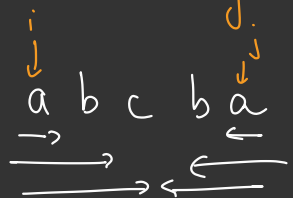
N = 5

1+2+3+4+5
= 15

```
int Sum ( int N) {
    if(n==1) return 1
    return n + f(n-1)
}
```



(9) Given a String, check Palindrome.



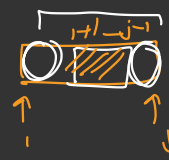
a b a d e
not a palindrome

a[i] == a[j] → Mismatch → return false
i++
j--

Iterative

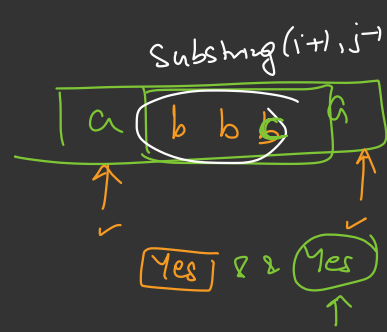
```
i = 0
j = n-1
while (i<j) {
    Compare a(i), a(j)
        ↳ True → i++, j--
        ↳ return false
}
Return True
```

```
bool chkPalindrome ( str, i, j ) {

    // Base Case
    if ( i >= j ) {
        return True;
    }

    // Rec Case
    return (str[i] == str[j]) && chkPalindrome(str, i+1, j-1);

}
```

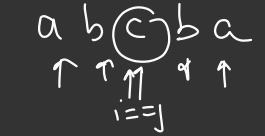


i j length → 1

i > j length - 0



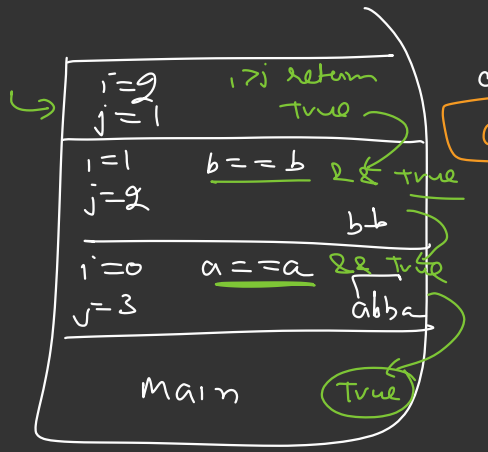
$$f(i,j) = a[i] == a[j] \;\&\&\; f(i+1, j-1)$$

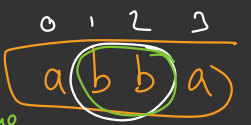


b b
a a

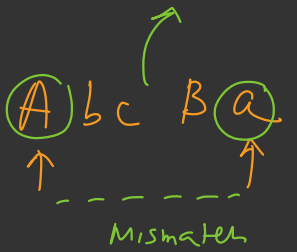
odd length

i==j





abc bca



Q Given a & n, you need to find a^n, use Recursion.

$a = 5$
$n = 3$

$5^3 = 125$

Solution - I

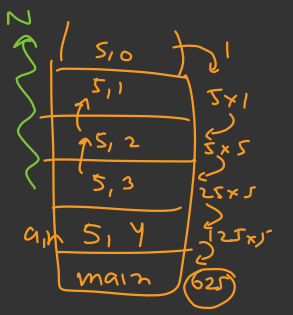
$5^3 = 5 . 5 . 5$
$= 5 . (5^2)$ ← subproblem

$a^n = a . a^{n-1}$
$\Rightarrow f(a,n) = a * f(a,n-1)$

```
int power(int a, int n) {
    if(n==0)
        return 1;
    return a * power(a,n-1)
}
```

Time → O(N) Space → O(N)

5^4
↓
$5 . 5^3$
↓
$5 . 5^2$
↓
$5 . 5^1$
↓
5^0



Solution-II

ans = 1

$\underbrace{a \times a \times \cdots \times a}_{N \text{ times}}$

```
for( —— N times){
    ans = ans * a
}
```

Time → O(N)
space → O(1)

Solution-III

$f(a, n) \curvearrowright f(a, n-1) \rightarrow O(N)$

$\Rightarrow f(a, n) \curvearrowright f(a, n/2) \rightarrow O(\log N)$

$a^{10} = (a^5)^2$

$a^5 = a(a^2)^2$

$a^2 = (a^1)^2$

$a^1 = a(a^0)^2$

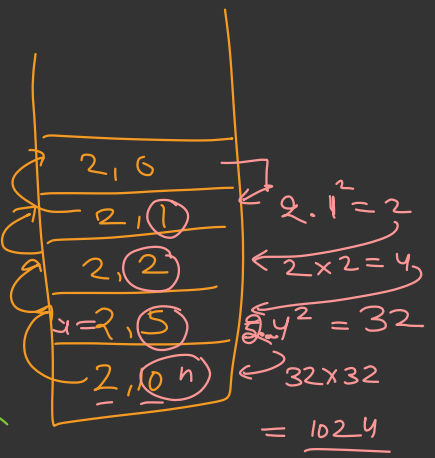
$a^n \downarrow a^{n-1} \downarrow a^{n-2} \downarrow \cdots \downarrow a^0$

$a^n \downarrow a^{n/2} \downarrow a^{n/4} \downarrow \cdots \downarrow a^0$

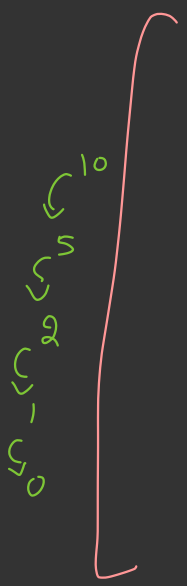
$$\Rightarrow \begin{cases} a^n = \left(a^{n/2}\right)^2 & \text{if } n \text{ is even} \\ a^n = a\left(a^{n/2}\right)^2 & \text{if } n \text{ is odd} \end{cases}$$

p

$2^5 = 32$




```
int power(a, n) {
    if(n==0) { return 1 }
    p = power(a, n/2);
    if (n is even)  return p*p
    else            return a*p*p;
}
```



3

log 10

Space → log N

Time → log N

$$5^{9} = 5 \cdot \left(5^{4}\right)^{2}$$

$$5^{\text{odd}} = 5 \cdot \left(5^{\frac{\text{odd}}{2}}\right)^{2}$$

Assignment

fast modulo Exponentiation long

$\Rightarrow a^n \% p$ → Step wise Mod

a → large

p is int → 10^9

n is a large No

Modulo → % at every step where addition/mult can lead to overflow

```
power ( int a, int n, int p) {
if(n==0) return 1
long sp = power(a, n/2, p);
if (n is even) {
    return (sp * sp) % p
}
else {
    ((a%p) * (sp*sp)%p ) %p ;
}
}
```

$sp = (a^{n/2}) \% p \rightarrow 10^9$

$10^9 \times 10^9 = 10^{18}$

10^9

$(25\%6)\%6$
$= 1\%6$
$= 1$

$(\ )\%p$
$\downarrow$
$(0-p-1)\%p$
$\downarrow$
ans

$$(a*b)\%m = (a\%m \times b\%m)\%m$$

Formula

Example

$a = 5$
$n = 4 \qquad p = 6$

$5^4 \% 6$

$(5 \times 5 \times 5 \times 5) \% p$ ← overflow

$= (625) \% 6$

$= 1$

$= ((5\%p) \times (5\%p))$

$(5 \times 5)\%p \times (5 \times 5)\%p$

$(4 \times 1)\%6$

$= 1$

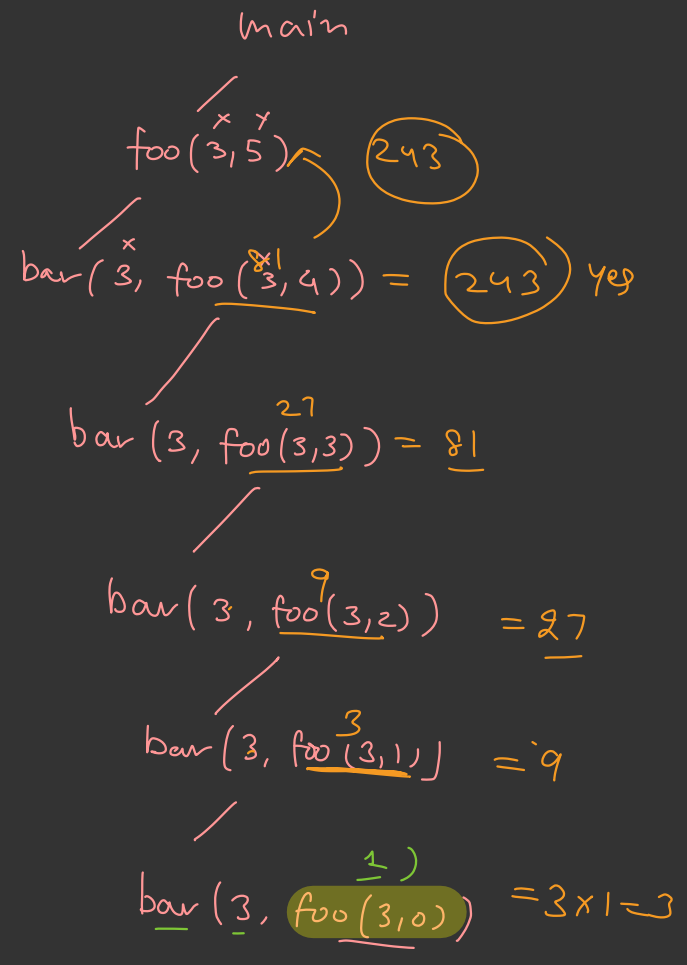
bar(x, y)
↳ Add x, y times
(x y)

Doubts

```
#include
using namespace std;

int bar(int x, int y){
    if (y == 0)   return 0;
    return (x + bar(x, y-1));
}

int foo(int x, int y){
    if (y == 0) return 1;
    return bar(x, foo(x, y-1));
}

int main(){
    cout << foo(3, 5);
}
```

multiply 3,1

bar(3,1)
= 3 + bar(3,0)
= 3



bar(3,2) = 3 x 2 = (6)

= 3 + bar(3,1)

= 3 + 3 + bar(3,0)

" 0

= (0)

Longest Int Problem

"20", "25", "7", "63", ...

Bit Tricky

↳ MergeSort

Comparison

Sorting

63 7 25 20

7, 63, 25, 20

[10, 5,]

(7)

10, 5

a[i] ↑ a[i+1] ↑

"10" + "5" < "5" + "10"

5 , 10

① 5 , 10 , 7

5 , 7 , 10

② 7 , 5 , 10

10 , 7 < 7 , 10

10 7 < 7 10

5 , 7 < 7 , 5

7 5 10

Sort + Comparator

(a+b > b+a)