Day 1

21 November 2022 19:33

Dynamic Programming

1. Function → Python, Java, C++

2. Recursion → optimization → Dynamic Programming

C++

void → return नहीं करना

func_name(,) {

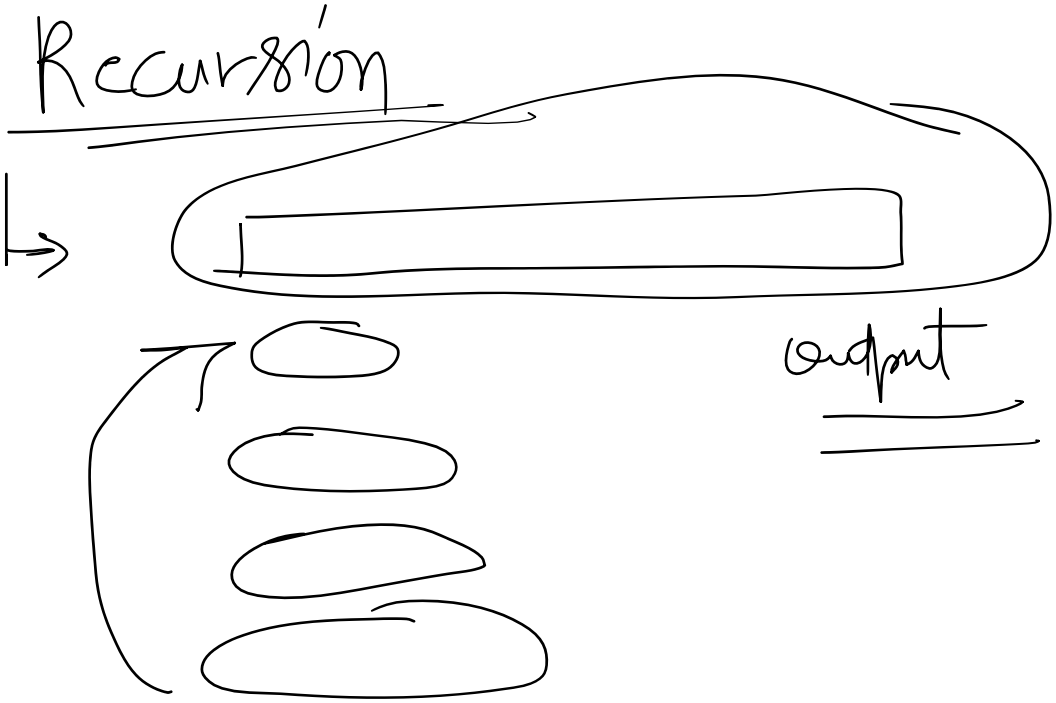
return type

{

int func_name (int para1, int para2){

return xi;

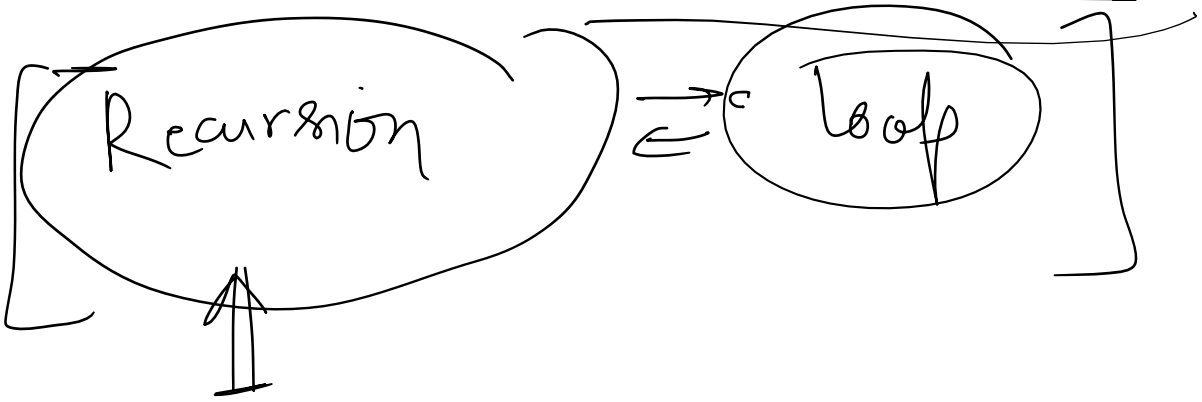
}



1. Factorial (through recursion)

$n! = n \times n-1 \times n-2 \cdots \times 1$

$5! = 5 \times 4 \times 3 \times 2 \times 1$



Factorial

$n \times$ Factorial $(n) \rightarrow n \times n-1 \cdots \times 1$

Factorial $(n-1) \rightarrow n-1 \times \cdots 1$

subproblem

Big problem solved.

$1! = 1$

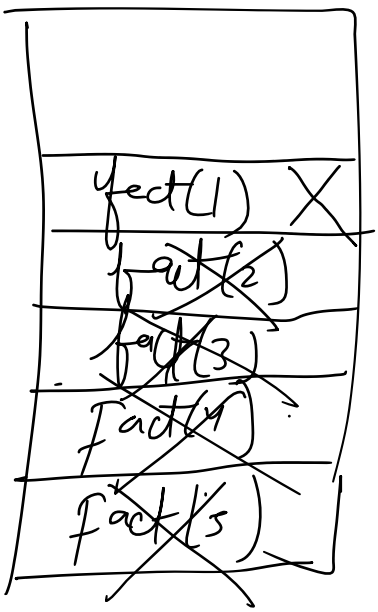
```
int factorial (int n){
    if(n==1) return 1;
return n x factorial(n-1);
}
```

return n * [illegible]

}

//op1

Stack space



functions in Activation Record

factorial(5) 120

5 x factorial(4) (24)

4 x fact(3) (6)

3 x fact(2) 2

(2) 2 x fact(1)

1

5! = 120

(1) Fibonnacci Series

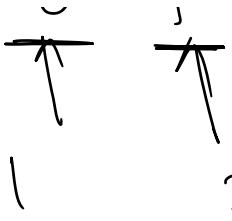
0 1 1 2 3 5 8 13 21

0th 2nd 3rd 4th 5th

Fib(5)

(3)

0 1



fibo(n) → fibo(n-1) + fibo(n-2)

fibo(5) → fibo(4) + fibo(3)

fibo(4) → fibo(2) + fibo(3)

fibo(3) → fibo(1) + fibo(2)

fibo(2) → fibo(1) + fibo(0)

1

0

Subproblem → larger problem

Smaller problem

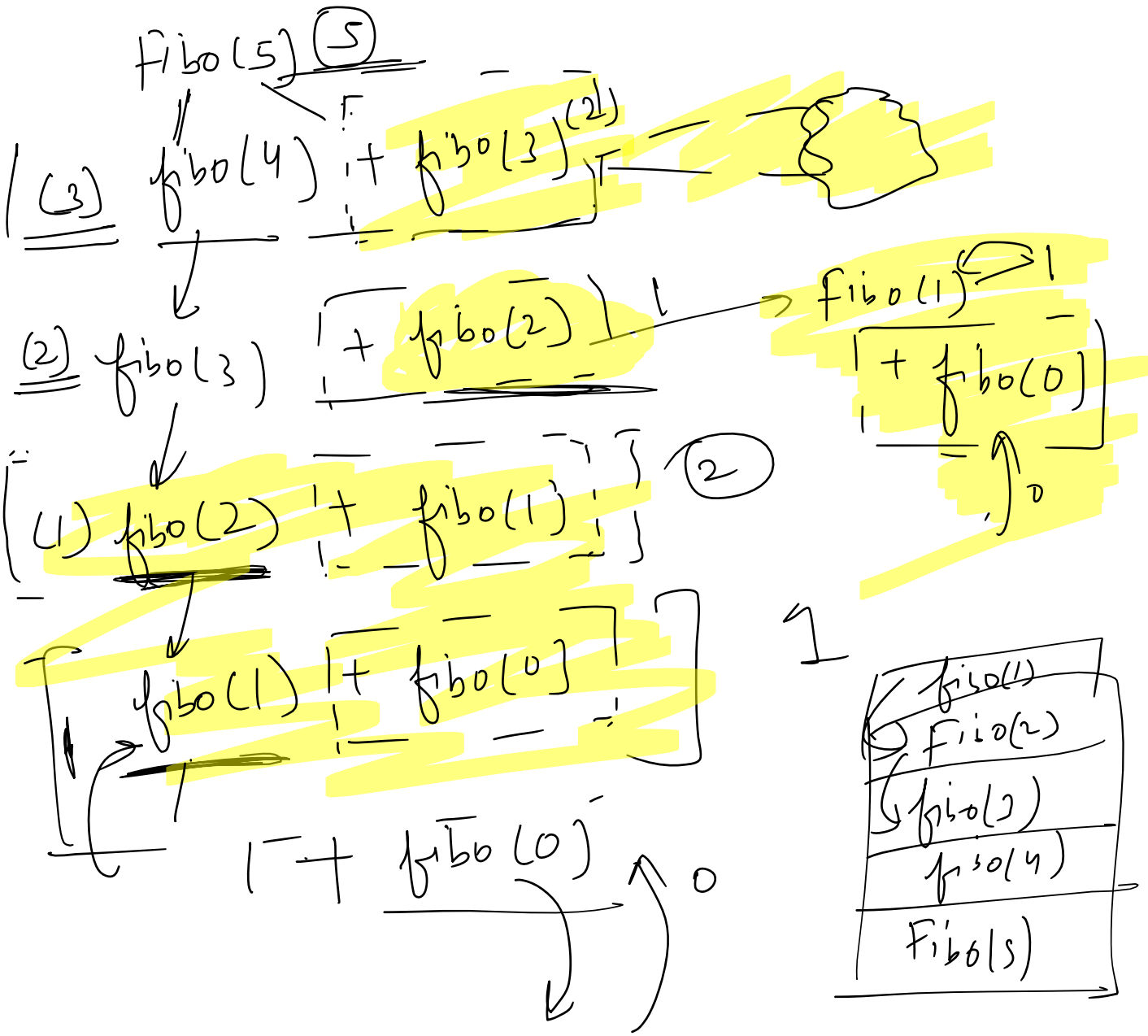
Solution

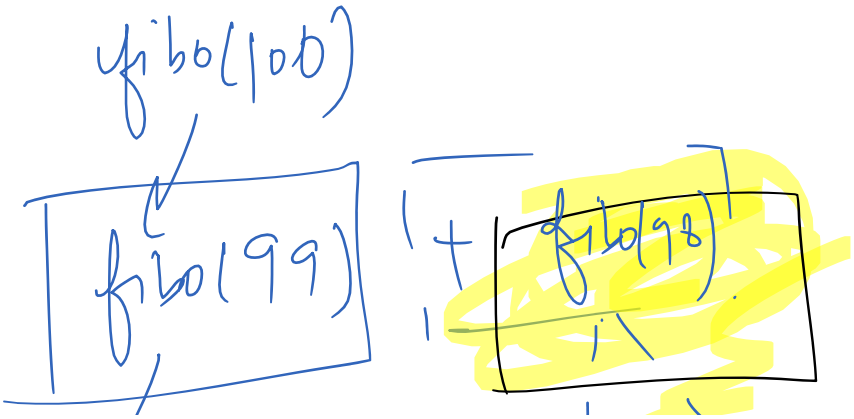
T.C – O(2^N)

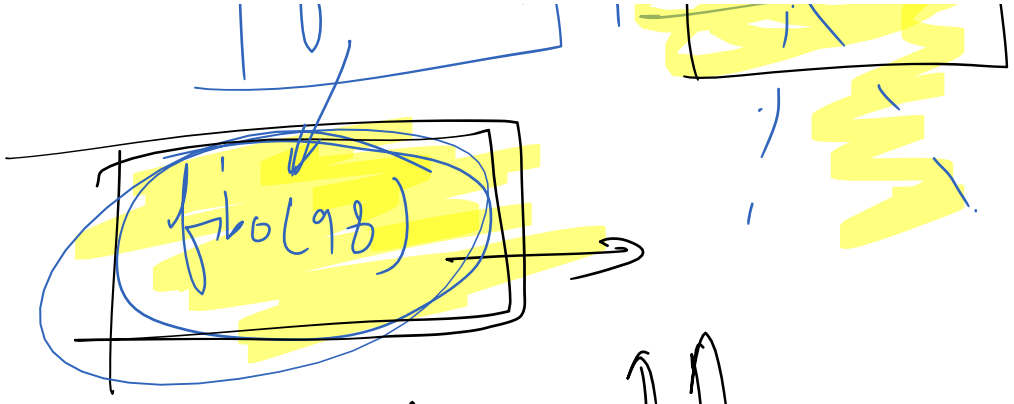
S.C → O(N)

int fibo (int n) {
if (n == n) return n;

```
int fibo(int n){
    if(n==0) return 0;
    if(n==1) return 1;
    return fibo(n-1) + fibo(n-2);
}
```





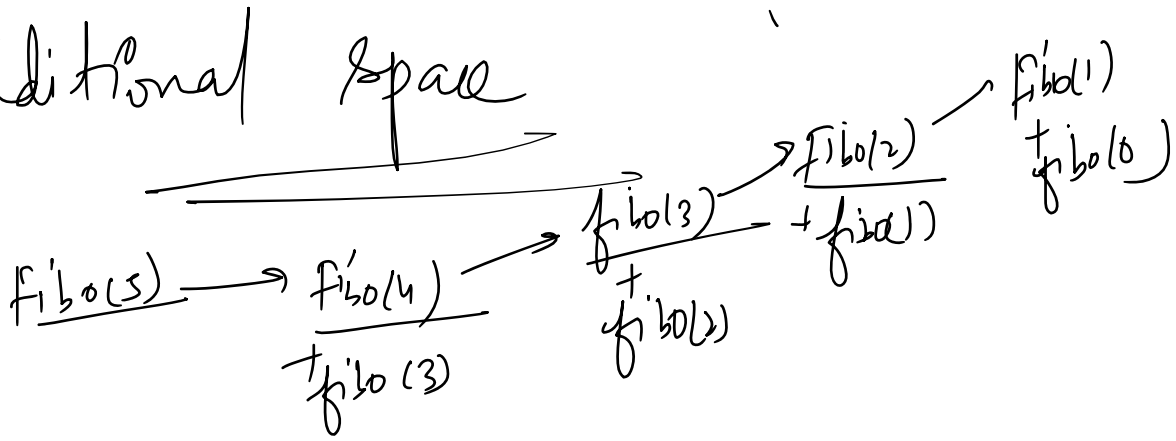


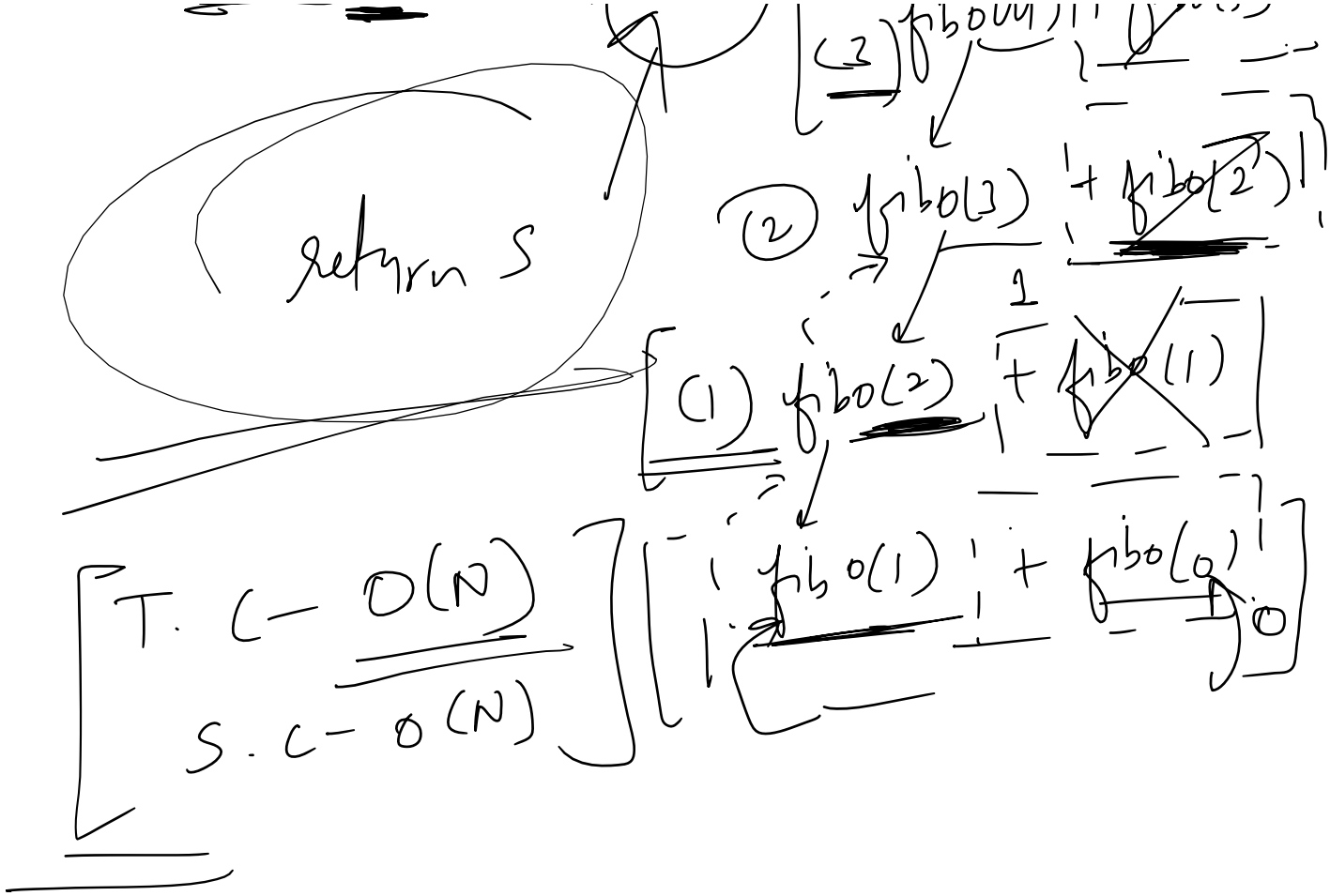
Time Complexity !!!

Dynamic Programming

```
int fibo (int n){
  [ if (n ≤ 1) return n;
  [ return fibo(n-1) + fibo(n-2);
}
```

Additional Space






```
int main(){
  int n;
  cin>>n; // Input
  vector<int> dp(n+1,-1);
  cout<< fibo(n,dp) <<endl;
}
```


```
int fibo(int n, vector<int>& dp){
  if(n<=1) return n; // ✓
  if(dp[n] != -1){
```

```
if( dp[n] != -1 ){
    return dp[n]; // Answer
}
dp[n] = fibo(n-1, dp) + fibo(n-2, dp);   // Computation
return dp[n];
}
```

T.C - O(2^N) ⇒ T.C - O(N)

Memo. → Top-Down DP

(75) (3) DP