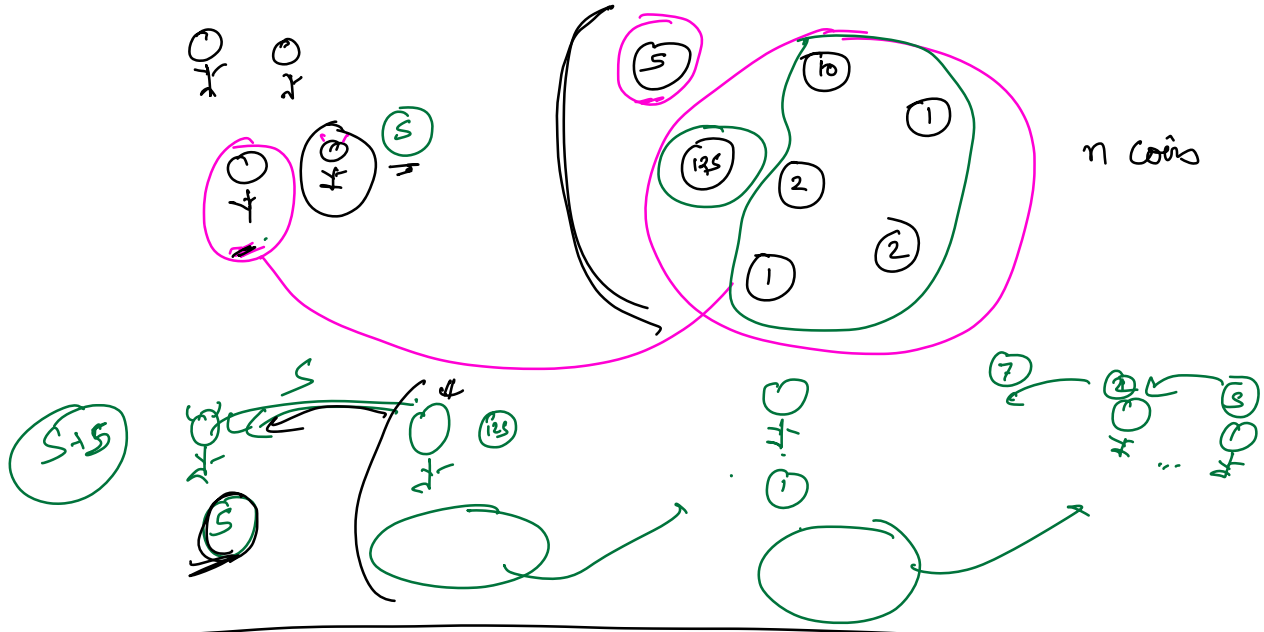
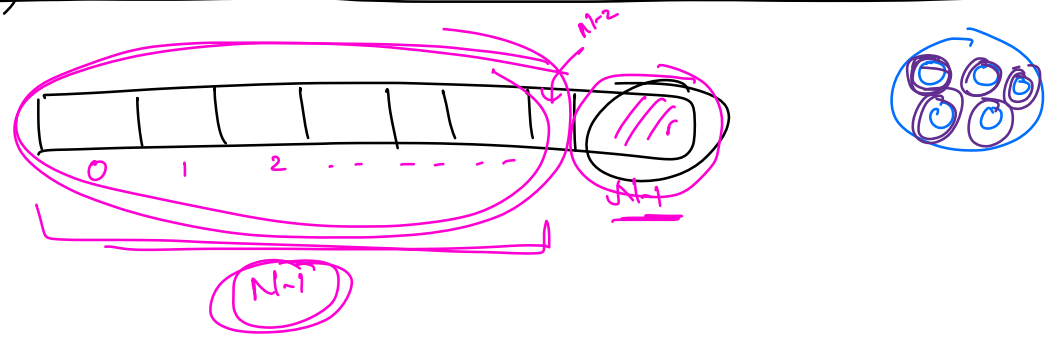



```
int SumCoins ( Coins[], N ) {
    if (N == 0) return 0;
    return Coins[N-1] + SumCoins(Coins[], N-1);
}
```

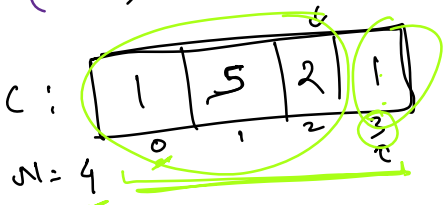
0-1

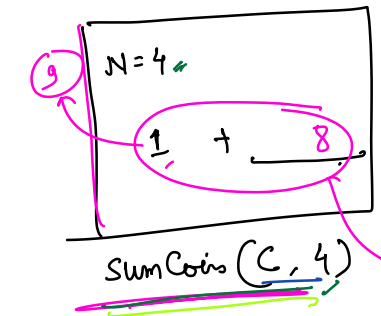


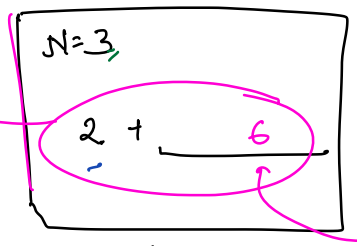
Recursion : A function calling itself.

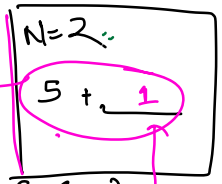
```
int SumCoins(Coins[], N) {
    if (N == 0) return 0;
    => return Coins[N-1] + SumCoins(Coins[], N-1);
}
```

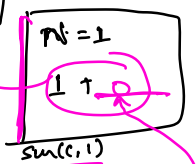
Assert/Trust

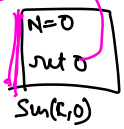


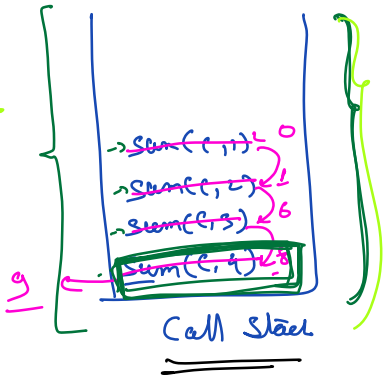












LIFO

Stack

Call-Stack / Recursion Stack

TC = Total No. of function calls × TC of one function call

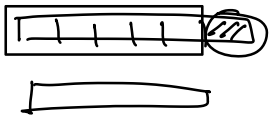
O(N) × O(1) = O(N)

SC = [The max size of stack × SC of every function call] ⇒ O(N) × O(1) = O(N)

→ Depth of rec tree.

Scope

= [coins[N-1] + sumCoin(coins, N-1); ↳ subproblem.]

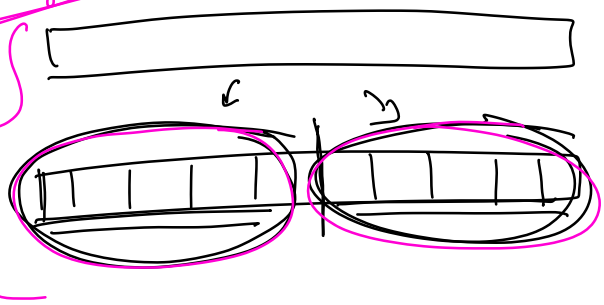


Subproblem > 1

No overlap

Merge Sort

Divide & Conquer



Fibonacci

0, 1, 1, 2, 3, 5, 8, 13, ----

Print N^{th} Fibonacci No.

fib(N) = fib(N-1) + fib(N-2);

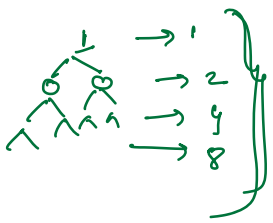
```
int fib(N) {
  if (N <= 1) ret N;
  ret fib(N-1) + fib(N-2);
}
```

L -

Dynamic Program

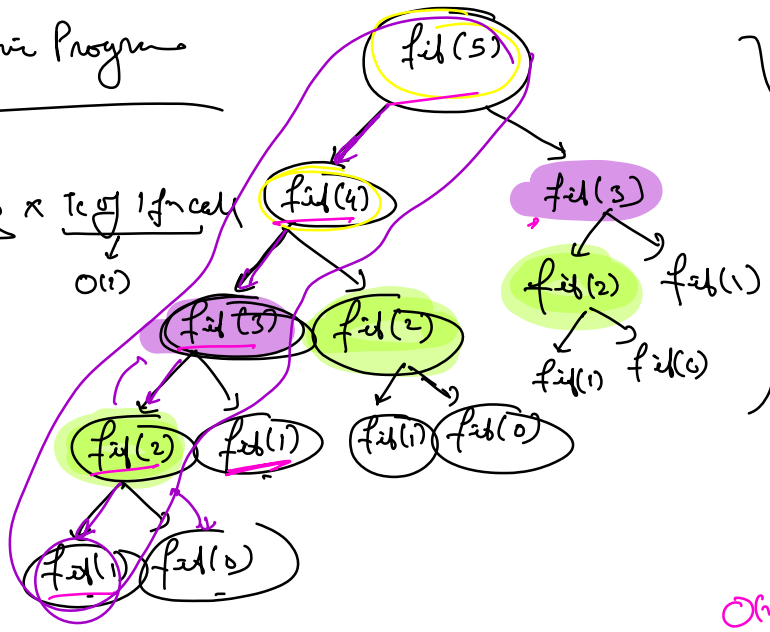
TC = No of fn calls × TC of 1 fn call

O(1)

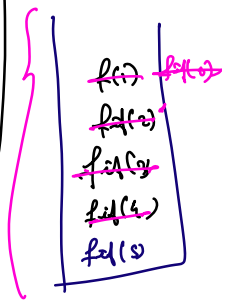


1 + 2 + 4 + 8 + 16 + 32 ...

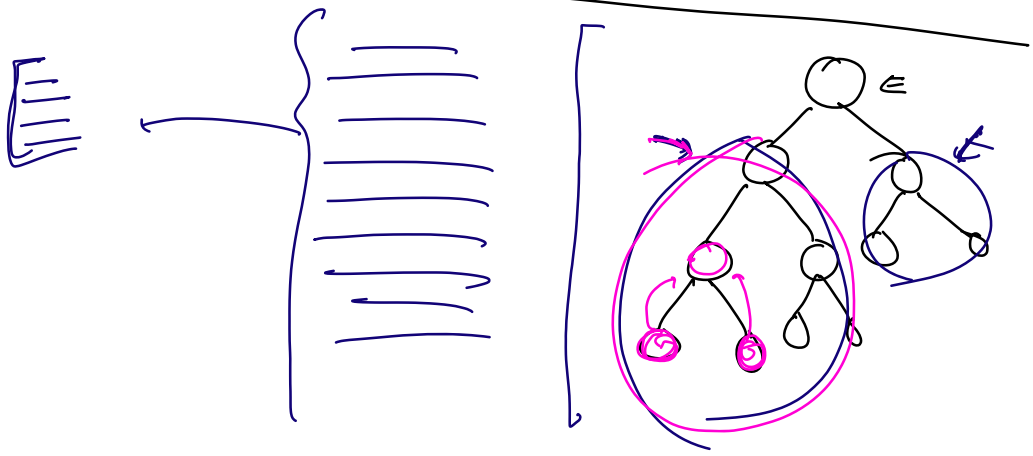
×2 ×2 ×2 ×2



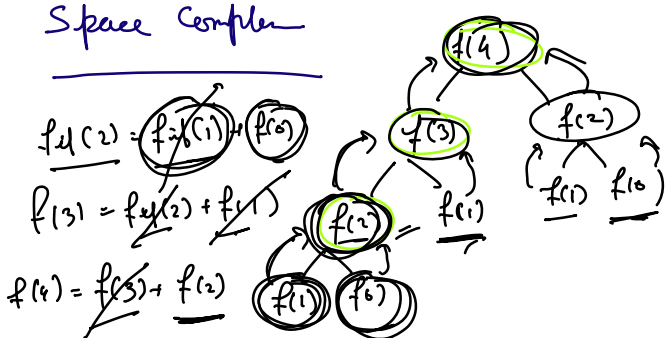
Overlapping sub probl



$$\text{Eqn} = \frac{a(r^n - 1)}{r - 1} = \frac{1(2^n - 1)}{2 - 1} = O(2^n)$$



Space Complexity



Seize : 4

Max Size : 4

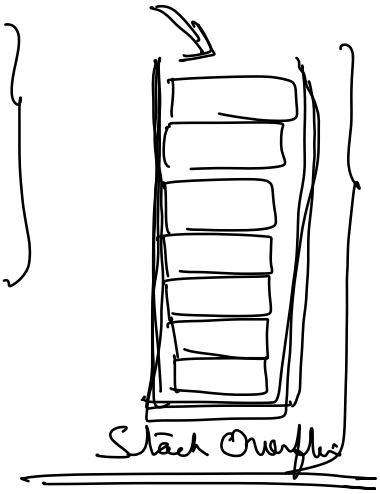
3 Steps

- Assume & trust : That recursion will work for smaller sub-problem.
- Recursive eq (logic) : Ret the ans of bigger problem using the ans of sub problem.
- Base / Termination Condition

```
while(1){
    print(0);
}
```

```
void PrintZero(){
    print(0);
    PrintZero();
}
```

0
0
0
0
0
0
...



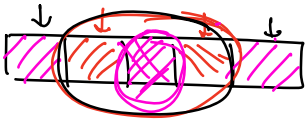
Q Given a string. Check if it is palindrome

* Recursively

- ARORA
- MADAM
- NITIN
- CTC

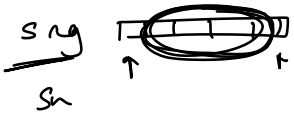
IsPalin (str, s, e)

Assume ; IsPalin (smaller str) will give correct output.

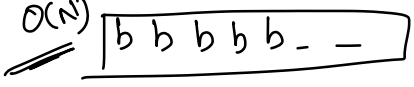


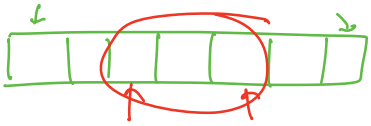
Logic / Recurrence :

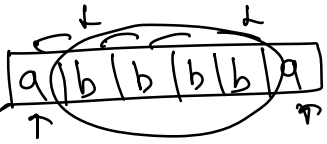
```
if ( first & last char are same)
        ret IsPal ( String b/w the)
else
    ret false.
```



O(N)

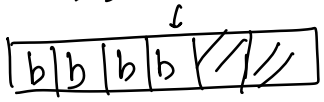




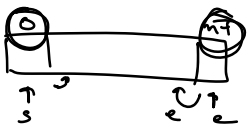


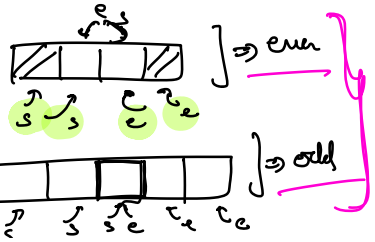
O(N)

b b b b



```
boolean isPal ( Stg, s, e) {
⇒ if ( s >= e) ret true ;
  if ( Stg[s] == Stg[e])
        ret isPal ( Stg, s+1, e-1);
  else
        ret false;
}
```

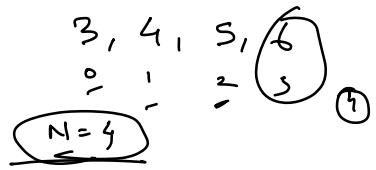


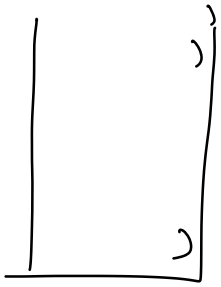


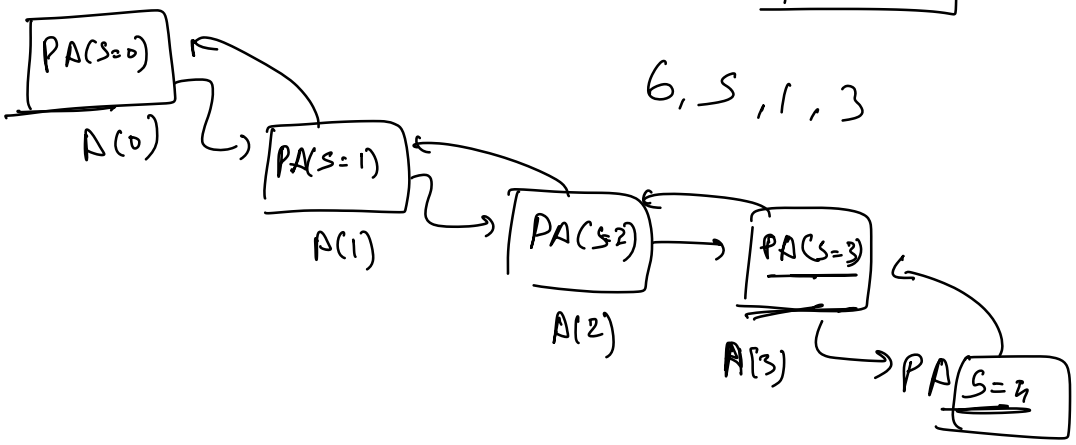
isPal (stg, 0, n-1)

Print all elements of an array

```
void PrintArr(A[], N, s) {
    if(s == N) ret;
    Prit (A[s]);
    PrintArr(A, N, s+1);
}
```



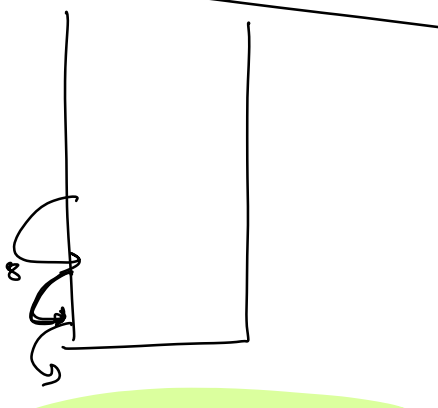


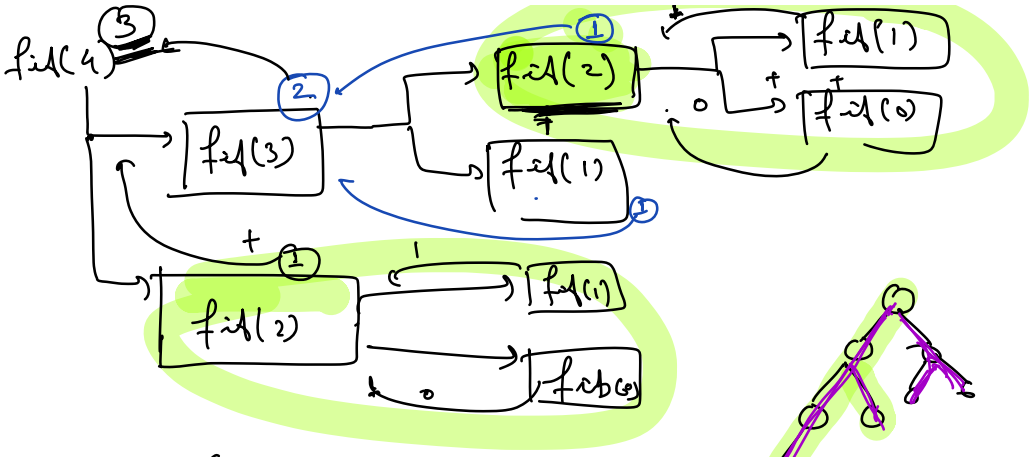


6, 5, 1, 3

```
int add(a, b){
    ret (a+b);
}

main() {
    Prit(add(5, 3));
}
```





0, 1, 1, 2, 3
0 1 2 3 4

DFS

---

fib(4) = fib(3) + fib(2)

fib(3) f(2)

fib(2) + f1

fi(1)

fib(3)
fib(4)