Date:

Black Board

Data Structures

Lecture # 11

Topics:

Recursion, Recursive Algorithms

① Factorial

$$n! = n * (n-1) * (n-2) \cdots 1$$

$$1! = 1$$

$$0! = 1$$

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

Recursive

Expressing a function in terms to a smaller instance of itself is called recursion

```
int fact(int n){
  if(n == 0 || n == 1)
      return 1
  else{
    int P = 1;
    for(int K = 2; K <= n; K++)
        P *= K;
    }
    return P;
}
```

→ Recursion: when a problem reduces to itself.

6, 2, 8, 1, 5

→ min find

1, 2, 8, 6, 5

1, 2, 5, 6, 8

Example of Reduction (not recursion)

Selection sort reduces to → find min

```
int fact (int n){
    if(n==0)
        return 1;
    else{
        return n * fact(n-1);
    }
}
```

base call

recursive call

fact(5)

5 * fact(4)

4 * fact(3)

3 * fact(2)

2 * fact(1)

1 * fact(0)

base case

??

recursive chain of calls

General Structure of a Recursive function

```
— foo ( — )
{
```

base call → directly return some answer.

recursive call → make some combination of recursive calls + some more work.

```
}
```

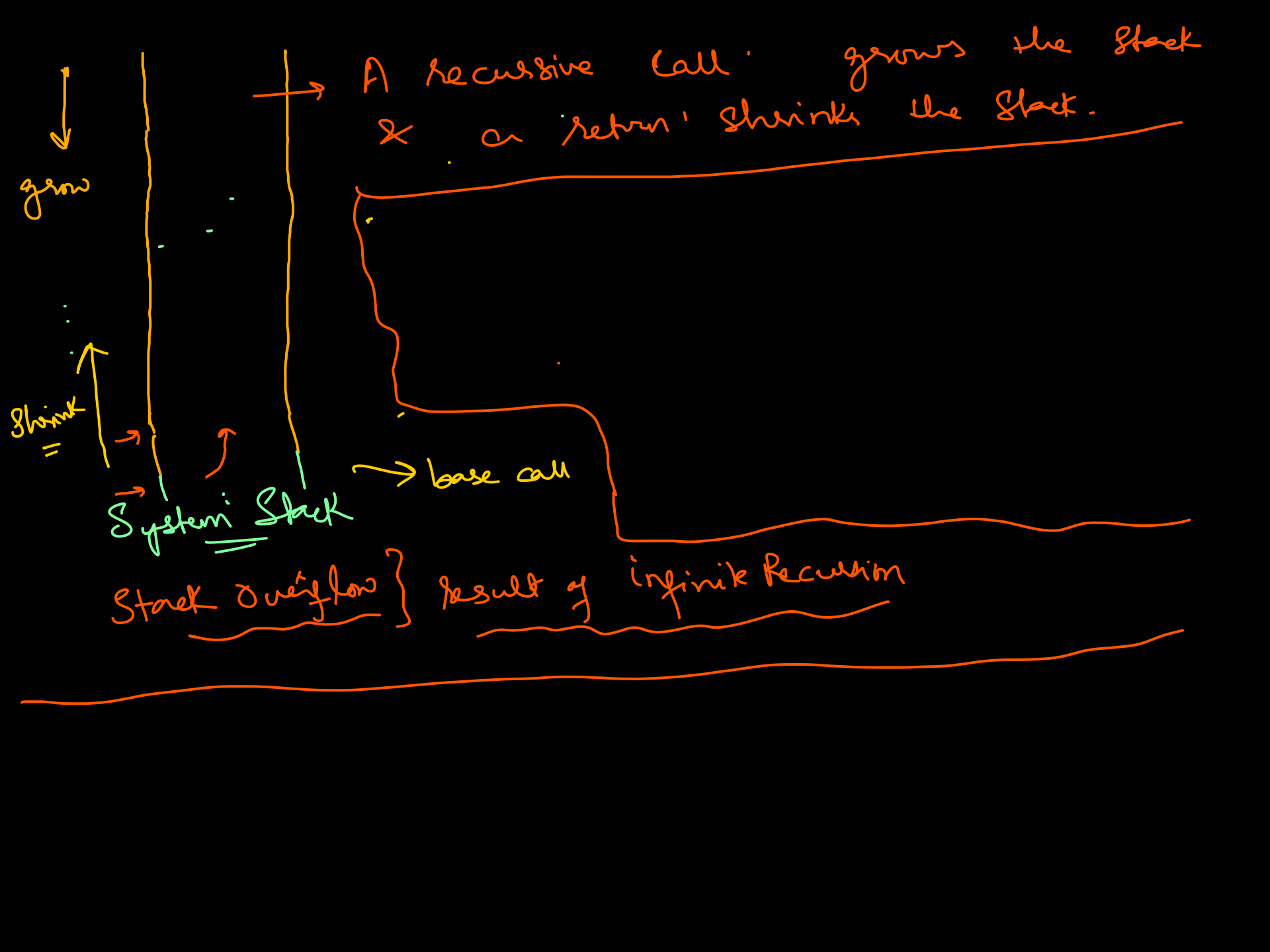
How is fact(5) computed

main

```
int x = fact(5);
```

→ 120

Recall: Every function call gets a fixed sized area on the system stack. (Activation Record)


A recursive call grows the Stack
& a return shrinks the Stack.
grow
Shrink
base call
System Stack
Stack overflow } result of infinite Recursion

(2) Fiboracci Numbers

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

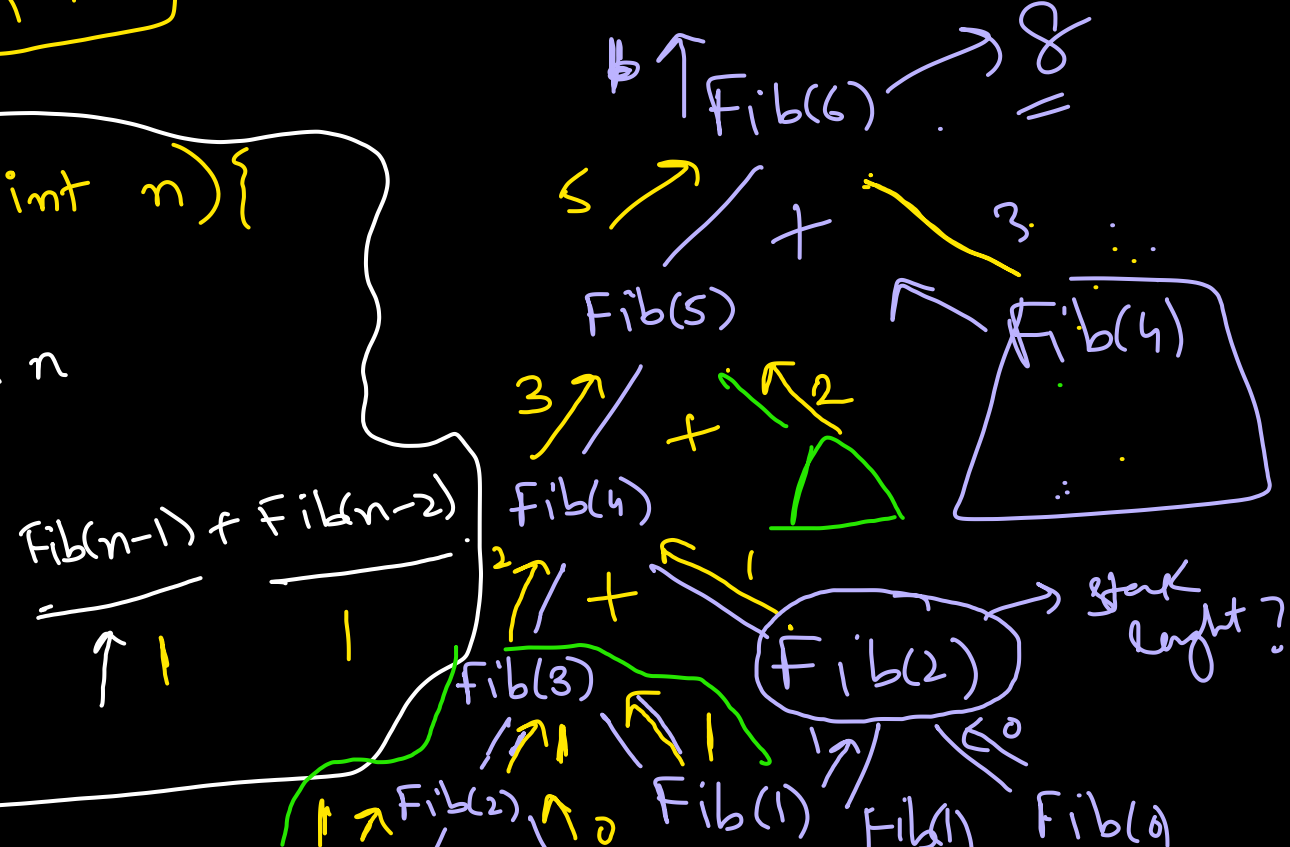
F_0 F_1 F_2 F_3 F_4 F_5 ... F_6

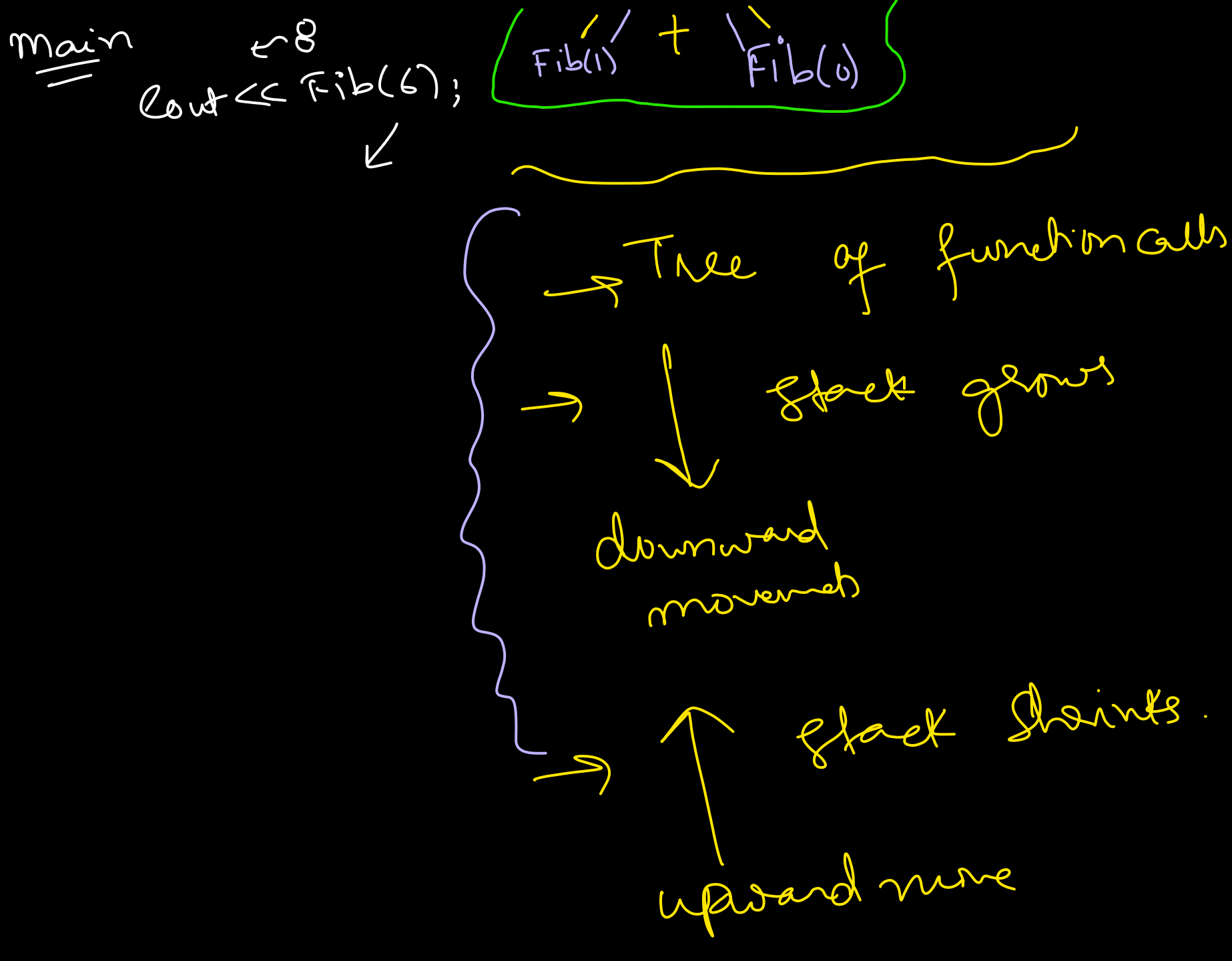
$F_n = F_{n-1} + F_{n-2}$

$F_0 = 0$, $F_1 = 1$

→ The F_n reduces to $F_{n-1} + F_{n-2}$

```
int Fib (int n){
  base call      { if(n<=1)
                     return n
                 else{
  recursive call     return Fib(n-1) + Fib(n-2)
                 }
  }
}
```




Main
cout << Fib(6);
Fib(1) + Fib(0)
Tree of function calls
stack grows
downward
movement
stack shrinks.
upward move

(3) FAST Power.

Pow $(x, n) \longrightarrow x^n$

$p = 1$

For $i \leftarrow 1$ to n

$p \leftarrow p * x$

Return p

$T(n) = O(n)$ Simple power.

↳ $2^{32} = \overbrace{2^1 \cdot 2^{31}}^{} =$

↳ $2^{32} = \underbrace{2^{16} \cdot 2^{16}}_{\uparrow} = (2^{16})^2$

Both are Recursion

$$2^{32} = (2^{16})^2 = ((2^8)^2)^2 = (((2^4)^2)^2)^2$$

$$= ((((2^2)^2)^2)^2)^2$$

$$= (((((2)^2)^2)^2)^2)^2$$

5 - squaring.

5 mults.

$2^{27} = 2 \cdot (2^{13})^2$

a^n → n odd → $a \cdot (a^{n/2})^2$

a^n → n even → $(a^{n/2})^2$

Recursive Algo.

$a^0 = 1$

```
int FastPower( int a, int n){
    if(n==0)
        return 1
    else
        int ans = FastPow(a, n/2)
        if(n%2==0)
            return ans*ans
        else
            return a*ans*ans
}
```

base

recursion

$T(n) = O(?)$

Answer this Question