Lecture ÷ Recursion 2

Agenda

- pow(a, n)
- Time complexity of recursive function.
- Space complexity of recursive function.

Psp → 90%

Attendance > 80%

↓

contenst.

Before advanced module

Assignments — must

Qu1 Given a, n. find a^n using recursion

constraints:- $n >= 0$

Example: $a = 2$ $n = 5$, $2^5 = 32$

$a = 4$, $n = 4$, $4^4 = 256$

```
int pow(int a, int n) {
1.    if(n==0) {
          return 1;
      }
2.    int sa = pow(a, n-1);

3     return a * sa;
}
```

Assumption

Given a, n, find and return a^n.

Main logic

$3^5 = 3 * 3 * 3 * 3 * 3$

$3^5 = 3^4 * 3$

$2^{18} = 2^{17} * 2.$

$$\begin{tikzcd}
a^n \arrow[d] \\
sa = a^{n-1} \arrow[u, bend right=60, "a * sa."']
\end{tikzcd}$$

Dry run: $a = 2$, $b = 5$

$$\begin{tikzcd}
\text{1. } pow(2,5) \quad 1\ 2 \arrow[d, shift right] \arrow[r, "16*2=32."] & {} \\
\text{2. } pow(2,4) \quad 1\ 2 \arrow[u, shift right, "2*8=16"'] \arrow[d, shift right] \\
\text{3. } pow(2,3) \quad 1\ 2 \arrow[u, shift right, "4*2=8"'] \arrow[d, shift right] \\
\text{4. } pow(2,2) \quad 1\ 2 \arrow[u, shift right, "2*2=4"'] \arrow[d, shift right] \\
\text{5. } pow(2,1) \quad 1\ 2 \arrow[u, shift right, "2*1=2"'] \arrow[d, shift right] \\
\text{6. } pow(2,0) \arrow[u, shift right, "1"']
\end{tikzcd}$$

Base case

$n == 0$, return 1.

$2^0 = 1$

$2^1 = 2^0 * 2 = 2$

$2^2 = 2^1 * 2 = 4$

No of func calls:- $n + 1$

Approach2: Reduce no of func calls.

$3^4 \Rightarrow 3^3 * 3$ $\longrightarrow$ Approach 1.

$3^4 \Rightarrow 3^2 * 3^2$

$3^8 \Rightarrow 3^7 * 3$ $\longrightarrow$ Approach 1

$3^8 \Rightarrow 3^4 * 3^4$

$3^5 \Rightarrow 3^3 * 3^2 \Rightarrow 3^2 * 3^2 * 3$

$3^9 \Rightarrow 3^4 * 3^4 * 3.$

Approach1: $3^8 = \underbrace{\boxed{3^7}}_{\text{subproblem}} * 3$ $[\ a^{n-1}\ ]$

$3^8 = 3^4 * 3^4$

$3^9 = 3^4 * 3^4 * 3$

$[\ a^{n/2}\ ]$

1. if n is even –

$a^n = a^{n/2} * a^{n/2}$

2. if n is odd –

$a^n = a^{n/2} * a^{n/2} * a$

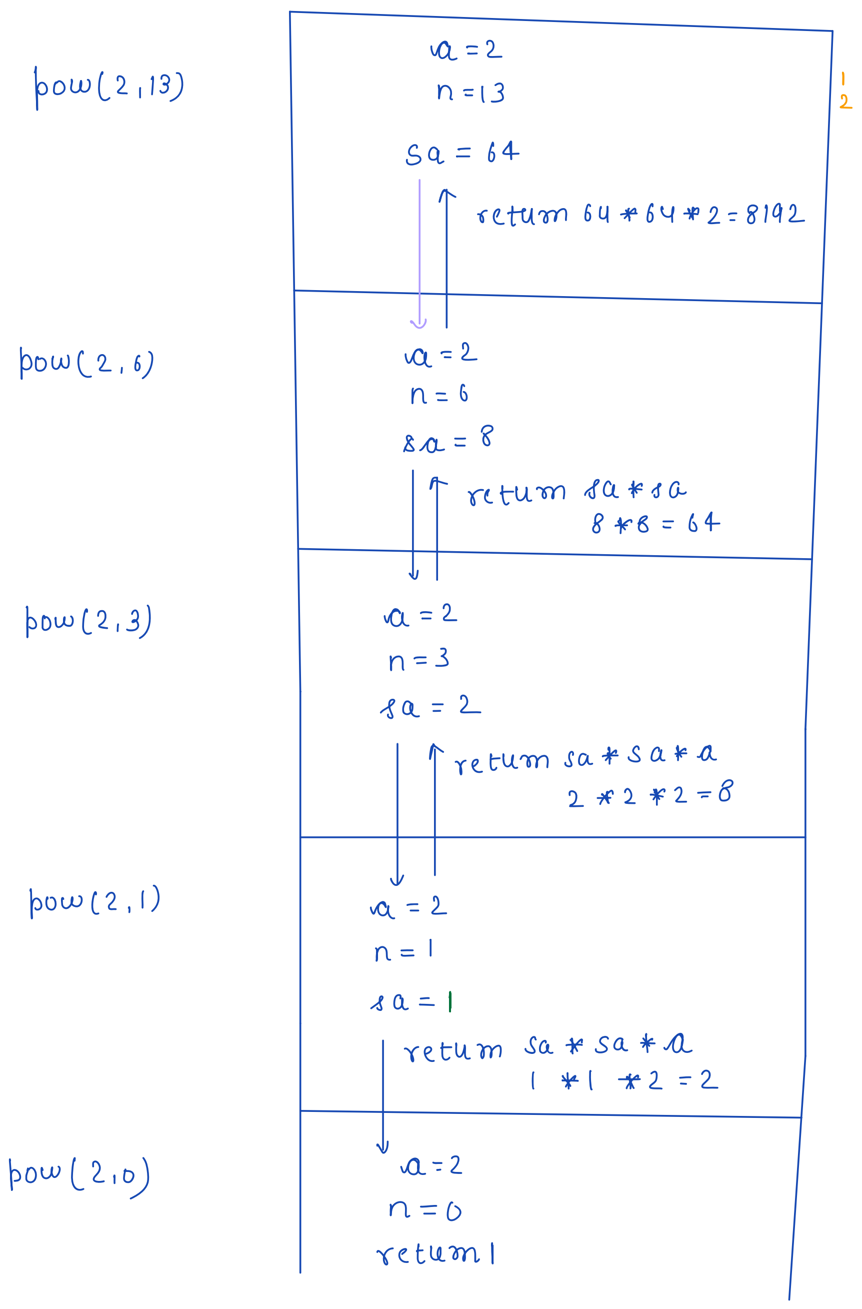
```
int  pow( int a, int n) {
   1  if ( n == 0) {
          return 1;
      }
  2  int  sa = pow( a, n/2);

  3  if ( n % 2 == 0) {

         return  sa * sa;
     }

 4  return  sa * sa * a;
}
```

Approach1. [14 function calls]

$$\begin{tikzcd}
\text{pow}(2,13) \arrow[d] \\
\text{pow}(2,12) \arrow[d] \\
\text{pow}(2,11) \arrow[d] \\
\text{pow}(2,10) \arrow[d, dotted] \\
\vdots \arrow[d] \\
\text{pow}(2,0)
\end{tikzcd}$$

Approach2 [5 function calls]

$$\begin{tikzcd}
 & \text{pow}(2,13) \arrow[r] \arrow[d, shift right] & 64 * 64 * 2 = 8192 \\
\text{even} & \text{pow}(2,6) \arrow[u, shift right, "8 * 8 = 64"'] \arrow[d, shift right] & \\
 & \boxed{\text{pow}(2,3)} \arrow[u, shift right, "2 * 2 * 2 = 8"'] \arrow[d, shift right] \arrow[r, dash] & \text{odd} \\
 & \text{pow}(2,1) \arrow[u, shift right, "1 * 1 * 2 = 2"'] \arrow[d, shift right] \arrow[r] & \text{odd} \\
 & \text{pow}(2,0) \arrow[u, shift right, "1"'] &
\end{tikzcd}$$



```
int pow( int a, int n) {
  1  if ( n == 0) {
        return 1;
     }
  2  int sa = pow(a, n/2);
  3  if ( n % 2 == 0) {
        return sa * sa;
     }
  4  return sa * sa * a;
}
```

Math.pow(a, n) ⇒ Approach 2.

$\downarrow$

TC ≠ O(1)

```
int pow(a,n) {
    if (n==0) {
        return 1;
    }
  if (n%2==0) {                          sa
     return pow(a,n/2) * pow(a,n/2);
  }
  return pow(a,n/2) * pow(a,n/2) *a;
}
```

n Even — 2 fun calls
pow(a,n/2) *
pow(a,n/2)

n odd — 2 fun calls.
pow(a,n/2) *
pow(a,n/2) *
a

Qu Given a, n, m . Calculate $a^n \% m$

constraints: $1 <= a <= 10^9$

$1 <= n <= 10^9$

$1 <= m <= 10^9$

```
  long
  ~~int~~ pow(int a, int n, int m) {
1    if (n==0) {
         return 1;
  long }
2  ~~int~~ sa = pow(a,n/2,m);        => sa < m <= 10^9
3  if (n % 2 == 0) {
       return (sa * sa) % m;
                10^9  10^9
   }
4  return (sa * sa * a) % m;          long temp = (sa * sa) % m
          10^9  10^9  10^9            return (temp * a) % m;
}
```

Wrong ⇐ $(\underbrace{sa\%m}_{10^9} * \underbrace{sa\%m}_{10^9} * \underbrace{a\%m}_{10^9})\%m$

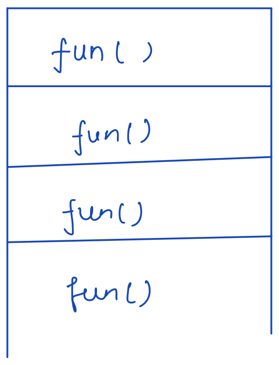
TC of recursive code

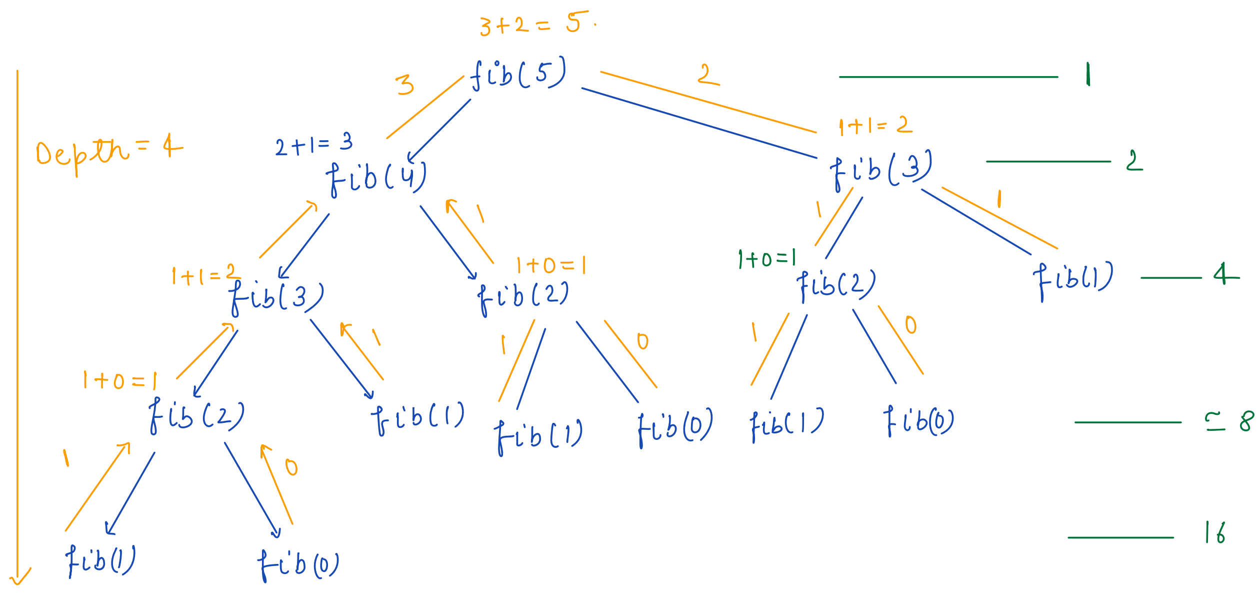
Recursive code: multiple instance of a funcⁿ.

T.C. = $x * y$

Step1: How many times you are generating the function inside call stack? x

Step2 for every funcⁿ block, what is the T.C? y





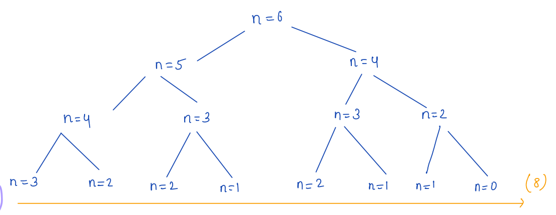
$x \Rightarrow 1 + 2 + 4 + 8 + 16 \cdots\cdots$

sum of GP: $\dfrac{a(r^n - 1)}{r-1}$

$a = 1$

$r = 2$

n = n value given in question (approx)



$x = \dfrac{1(2^n - 1)}{2-1} = 2^n - 1$

$y = O(1)$

TC: $2^n - 1 * O(1) \simeq O(2^n)$

```
fib(n) {
    if(n==0||n==1){
        return n;
    }
sa1 = fib(n-1);
sa2 = fib(n-2)
sa3 = fib(n-3);

return sa1 + sa2 + sa3
}
```

h/w

```
int pow( int a , int n) {
  1.  if( n==0) {
          return 1;
      }
  2. int sa = pow(a, n-1);

  3 return a * sa;
}
```

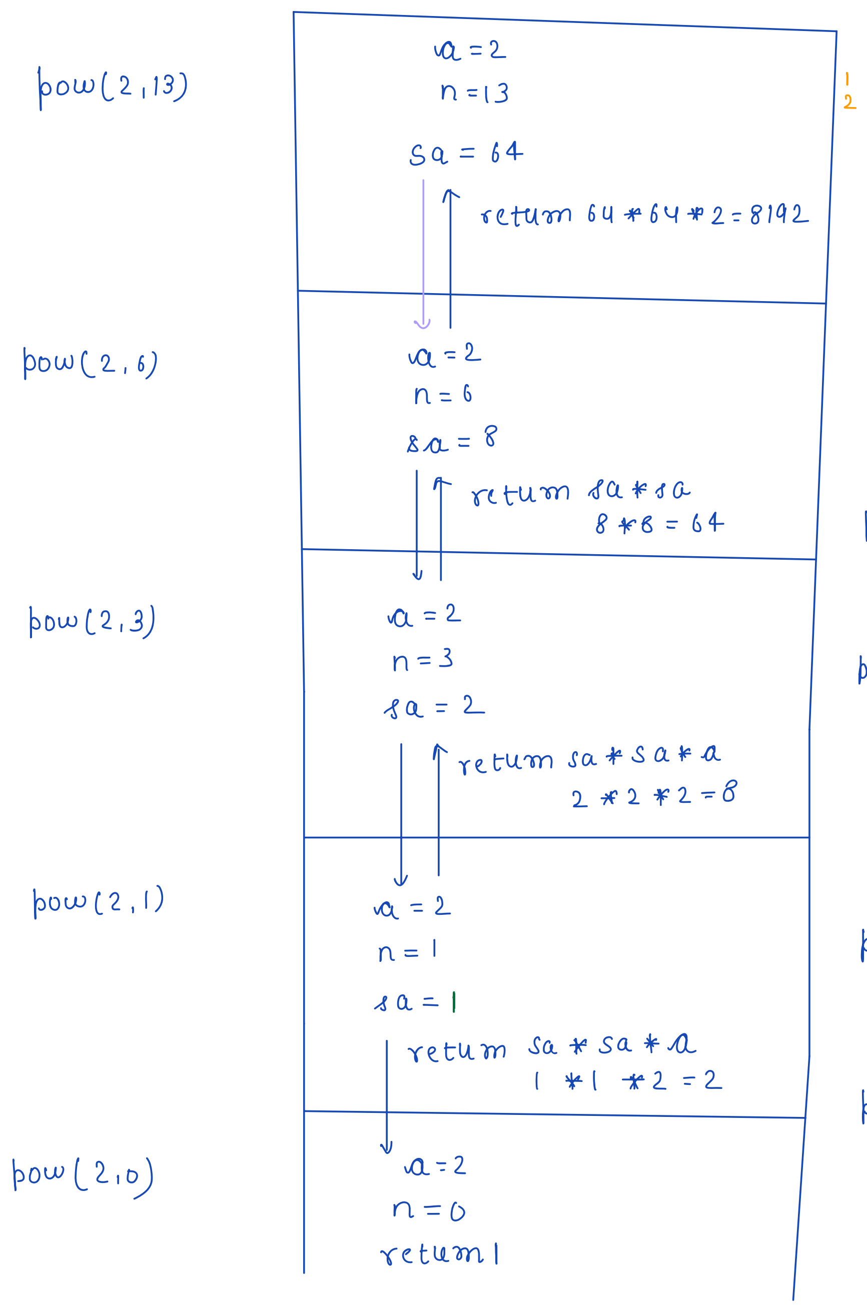
Dry run: $a = 2, \ b = 5$ $16 * 2 = 32.$

$$\begin{tikzcd}
1.\ pow(2,5)\quad 1\ 2 \arrow[d, shift right] \\
2.\ pow(2,4)\quad 1\ 2 \arrow[d, shift right] \arrow[u, shift right, "2*8=16"'] \\
3.\ pow(2,3)\quad 1\ 2 \arrow[d, shift right] \arrow[u, shift right, "4*2=8"'] \\
4.\ pow(2,2)\quad 1\ 2 \arrow[d, shift right] \arrow[u, shift right, "2*2=4"'] \\
5.\ pow(2,1)\quad 1\ 2 \arrow[d, shift right] \arrow[u, shift right, "2*1=2"'] \\
6.\ pow(2,0) \arrow[u, shift right, "1"']
\end{tikzcd}$$

$x = n+1 \simeq n.$

$y = O(1)$

$TC = x * y = O(n) * O(1)$
$= O(n).$



```
int pow( int a, int n) {
  1  if ( n == 0) {
         return 1;
     }
  2  int sa = pow(a, n/2);
  3  if ( n % 2 == 0) {
         return sa * sa;
     }
  4  return sa * sa * a;
}
```

$$\begin{tikzcd}
pow(2,13) \arrow[d] & n \arrow[ddd, "\log_2 n"] \\
pow(2,6) \arrow[d] & n/2 \\
pow(2,3) \arrow[d] & n/4 \\
pow(2,1) \arrow[d] & n/8 \\
pow(2,0) & \vdots
\end{tikzcd}$$

$x = \log_2 n.$

$y = O(1)$

TC : $x * y = \log_2 n * O(1) = \log_2 n.$

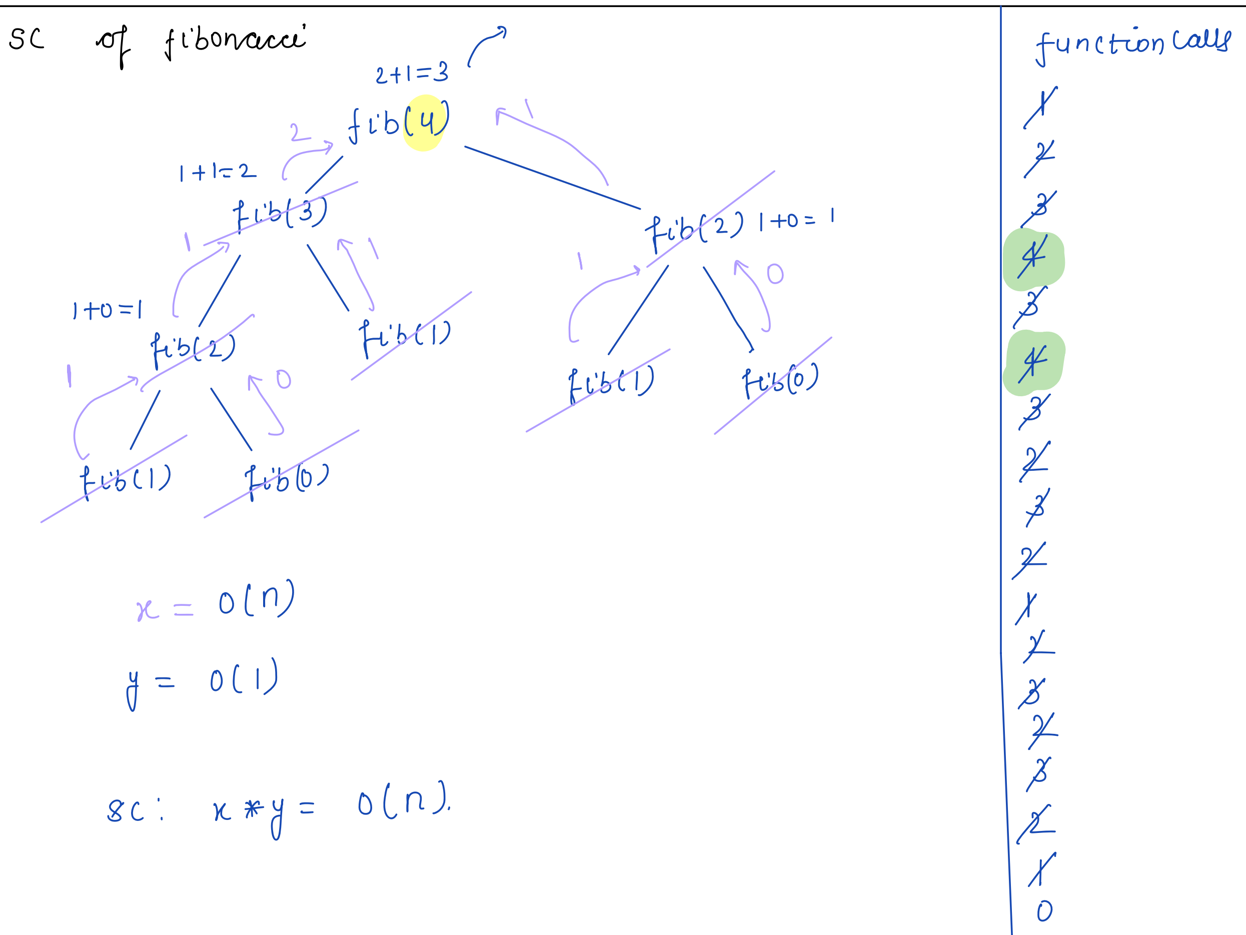
Space complexity of recursive func^n

x = max no of function calls at a given time in stack space

y = Inside one func^c block, how many space?

SC : $x * y$.

SC of fibonacci



$x = O(n)$

$y = O(1)$

SC : $x * y = O(n)$.

Assignment: Try out TC & SC for every recursion problem you solve in assignments / homeworks

Thankyou ☺

Doubts:

Given a, n, m. Calculate $a^n \% m$

constraints: $1 <= a <= 10^9$

$1 <= n <= 10^9$

$1 <= m <= 10^9$

```
long
~~int~~ pow( int a, int n, int m) {
1   if ( n == 0) {
        return 1;
    }
    long
2   ~~int~~ sa = pow(a, n/2, m);        =>  sa < m <= 10^9
3   if ( n % 2 == 0) {
4       return (sa * sa) % m;          [no need to change]
                 10^9  10^9
    }
5   ~~4~~ return (sa * sa * a) % m;      long temp = (sa * sa) % m
                  10^9  10^9  10^9      return (temp * a) % m;
}
```

Wrong $\Leftarrow$ $(\underbrace{sa \% m}_{10^9} * \underbrace{sa \% m}_{10^9} * \underbrace{a \% m}_{10^9}) \% m$

$[0, m-1]$ sa — line 4 $[0, m-1]$, line 5 $[0, m-1]$; sa $\uparrow$ m

$(sa * sa * \overset{a}{\cancel{a}}) \% m$ — 10^9, 10^9, 10^9 — $10^{27} \% m$

long temp = (sa * sa) % m

return (temp * a) % m

$(10^9 * 10^9) \% m$

$(sa * sa) \% m$ — $10^9 * 10^9$ — 10^{18}

✗ $(\underbrace{sa \% m}_{[0, m-1]} * \underbrace{sa \% m}_{[0, m-1]} * \underbrace{a \% m}_{[0, m-1]}) \% m$

m m m

$10^9 * 10^9 * 10^9$