Agenda

→ More problems on recursion

→ T.C. / S.C. analysis

Q1) Given a number n, write a recursive code to find the sum of digit of n.

$n > 0$

sum(1234) = 1+2+3+4

= 10

sum(1234) = sum(123) + 4

= sum(234) + 1

[Extract 1 number, call sum(remaining number), add extracted number]

n → Last digit = (n%10)

sum(n) = sum(rem. number after extracting last digit) + last digit

⇒ sum(n) = sum(n/10) + (n%10)

[SUBPROBLEM]

```
int sumOfDigits(int n){
    // Approach :- This returns the sum of digits,
    //                given any n ≥ 0

    // Base case
    if(n==0){
```

```
    return 0; }
    // Main logic
    return sumOfDigits(n/10) + (n%10);
}
```

Integer division

Q2) Implement the power function

$pow(a, b) \longrightarrow a^b$ $\quad a>0, b\geq 0$.

$a^b = \boxed{a*a*a*\ldots} *a$ (b times)

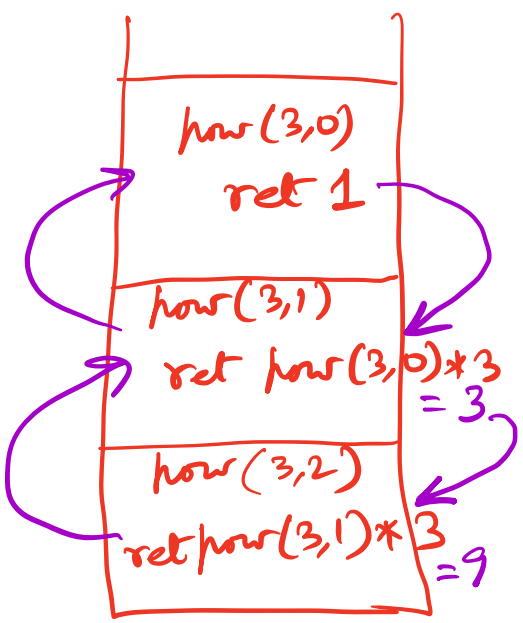
$\longrightarrow a^{b-1} \equiv pow(a, b-1)$

$4^5 \Rightarrow 3^5$

$pow(a,b) = pow(a, b-1) * a$

```
int pow(a, b) {
    // Base case
    if(b==0){
        return 1; }
    // Main logic
    return pow(a, b-1) * a;
}
```

a = 3
b = 2



2 recursive calls.

$a^b \longrightarrow$ b recursive calls

O(b) steps ≡ time

Approach-2

$a^6 = a^3 * a^3$

$a^8 = a^4 * a^4$

$a^{2k} = a^k * a^k$

$y = a^k$ ⟸

ret (y * y);

$a^{13} = a^{12} * a$

$a^6 * a^6$

a^{16} ⇒ 16 steps as per Approach 1.

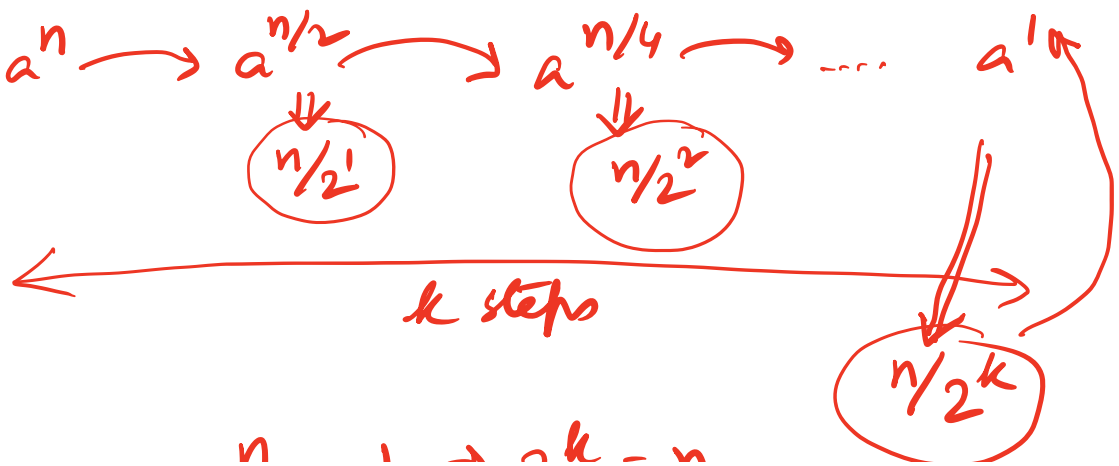
$a^{16} \rightarrow a^8 * a^8$

$a^4 * a^4$

$a^2 * a^2$

$a^1 * a^1$

4 recursive calls only.



$n \rightarrow 2^k = n$

$\frac{n}{2^k} = 1 \Rightarrow 2^k = n$

$\Rightarrow \boxed{k = \log_2 n}$

Main Logic

a^n → $a^{n/2} * a^{n/2}$ if n is even

a^n → $a^{n/2} * a^{n/2} * a$ if n is odd.

$y = a^{n/2}$

$y * y$

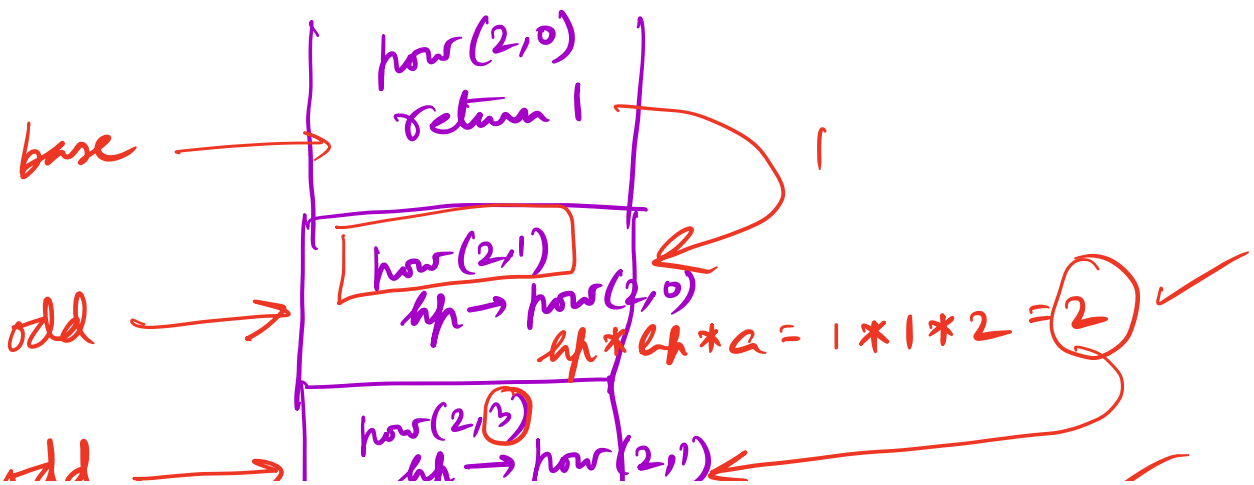
$y * y * a$

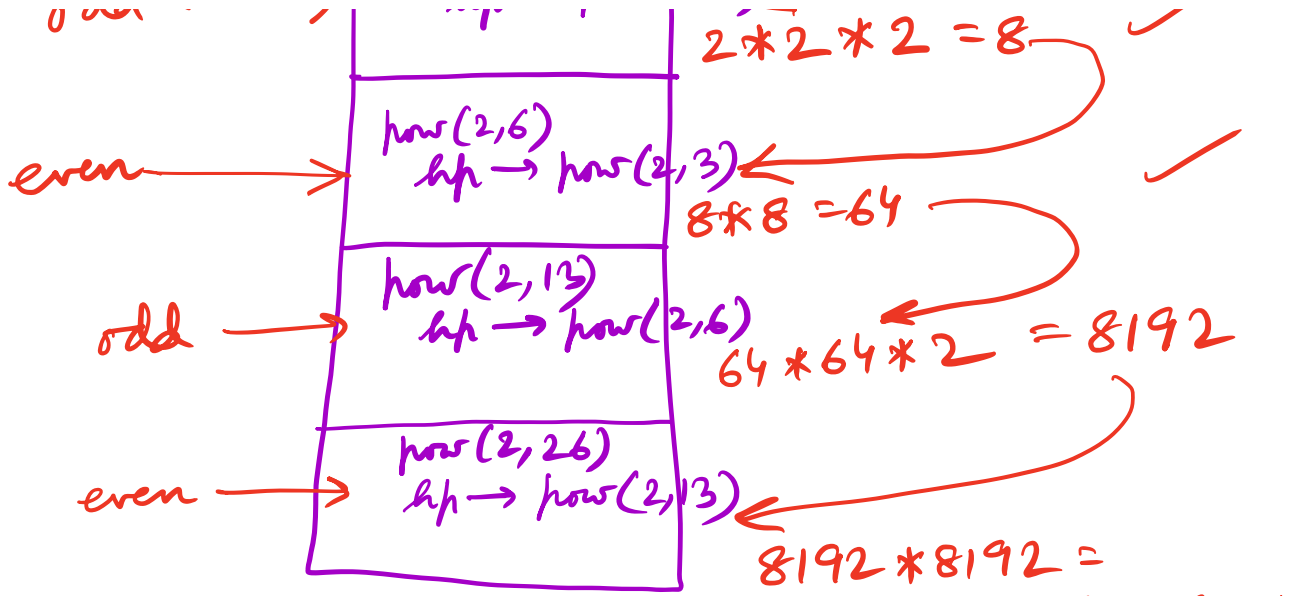
```
int pow(int a, int b){
    if(b==0)
        return 1;
    int halfpow = pow(a, b/2);
    if(b%2==0)
        return (halfpow * halfpow);
    else
        return (halfpow * halfpow * a);
}
```

$n = 26 \longrightarrow (11010)_2$

a = 2

b = 26





$\Rightarrow (11010)_2$

$((((2)^2 * 2)^2)^2 * 2)^2$

$4 * 2 \rightarrow 8$

$8^2 \rightarrow 64$

$64^2 \times 2 \rightarrow 8192$

1 0 0 0

$(((*a)^2)^2)^2 \Rightarrow a^8$

1 0 1 0

$(((*a)^2)^2 * a)^2$

a^2

$b = 5 = (101)_2$

$a = 3 \quad 3^5$

$((1 * a)^2)^2 * a$

$= a^5$

Time Complexity Analysis :-

eg 1: Sum of first n natural nos.

sum(N) = sum(N-1) + N ⇒

T(N) = Time (no. of steps) for computing sum(N).

(N-1) → T(N-1)

T(n) on the basis of T(n-1)

Recurrence Relation ⇒ $T(n) = \underbrace{T(n-1)}_{\text{how much time the subproblem takes}} + \underbrace{1}_{\text{extra time that sum(N) takes}}$

$T(n) = T(n-1) + 1$ — (1)

$= (T(n-2) + 1) + 1$

$= T(n-2) + 2$ — (2)

$= T(n-3) + 3$ — (3)

k^{th} step → $T(n-k) + k$

✓ $T(0) = 1$

For which k, shall k^{th} step have T(0)?

$T(n-k) + k = T(n)$

$k = n$

$$T(n) = T(n-k) + k$$
$$= T(0) + n = 1 + n$$
$$= O(n)$$

Diff. argument $\Rightarrow$ $T(n) = T(n-1) + 2$

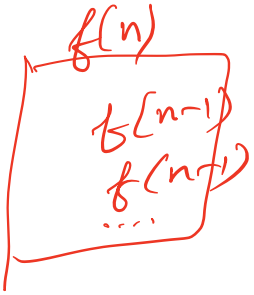
$$= [T(n-2) + 2] + 2$$
$$= T(n-2) + 4$$
$$= T(n-3) + 6$$
$$\vdots$$
$$T(n-k) + 2k$$

$T(0) = 1$

$k = n \Rightarrow 1 + 2n \equiv O(n)$

eg 2:- $T(n) = 2T(n-1) + 1$, $T(0) = 1$.

$$= 2 * [2T(n-2) + 1] + 1$$
$$= 4T(n-2) + 3$$
$$= 8T(n-3) + 7$$



$k = 3 \Rightarrow$ 2^3 $\quad 3 \quad$ $2^3 - 1$

k^{th} step $= 2^k T(n-k) + (2^k - 1)$

Which value of k gives us $T(n-k)$?

$n = k \Rightarrow T(n-k) = T(0)$

$= 1$

$$T(n) = 2^k T(n-k) + (2^k - 1)$$

$$= 2^n \cdot T(0) + 2^n - 1$$

$$= 2 \cdot 2^n - 1 = O(2^n).$$

eg 3 :- $T(n) = T(n/2) + 1$

$T(1) = T(1/2) + 1$

$= T(0) + 1$

$= 1 + 1 = 2$

$= (T(n/4) + 1) + 1$

$= T(n/4) + 2$

$= T(n/8) + 3$

$2^3 \quad 3$

k^{th} step $\Rightarrow$ $T(n/2^k) + k$

$T(0) = 1 \Rightarrow T(1) = 2$

```
fun(a, n)
{
  fun(a, n/2)
}
```

$T(1) = 2$

$\frac{n}{2^k} = 1 \Rightarrow n = 2^k$

$\log_2 n = \log_2 2^k$

$= k \log_2 2$

$= k$

$k = \log_2 n$

$T(n) = T(n/2^k) + k$

$k = \log_2 n \Rightarrow T(1) + \log_2 n$

$= 2 + \log_2 n = O(\log n)$

$$\frac{n}{2^{\log_2 n}} = \frac{n}{n^{\log_2 2}} = \frac{n}{n} = 1$$

$$a^{\log_c b} = b^{\log_c a}$$

eg :- $T(n) = 2 \times T(n/2) + 1, \quad T(1) = 1$

$= 2[2T(n/4)+1]+1$

$k=2 \longrightarrow = 4T(n/4) + 3$

$= 8T(n/8) + 7$

$k=4 \longrightarrow = 16T(n/16) + 15$

k^{th} step $\longrightarrow 2^k T(n/2^k) + (2^k - 1)$

$T(1)$

$k = \log_2 n$

$2^{\log_2 n} * T(1) + (2^{\log_2 n} - 1)$

$= n + (n-1)$

$= O(n)$

$2^{\log_2 n}$

```
f(n) {
    f(n/2)
    f(n/2)
}
```

If we do the half four twice, then this happens.

eg :- 5 :- $T(n) = 2 \cdot T(n/2) + O(n)$

$T(1) = 1$

$T(n) \leq 2 \cdot T(n/2) + c \cdot n$

$= 2[2T(n/4) + cn/2] + cn$

$T(n/2)$

$= 4T(n/4) + 2cn$

$= 4[2T(n/8) + cn/4] + 2cn$

$= 8T(n/8) + 3cn$

k^{th} step $\longrightarrow$ $2^k T(n/2^k) + c \cdot kn$

$k = \log_2 n$

$2^{\log_2 n} * T(1) + c \cdot (\log_2 n) \, n$

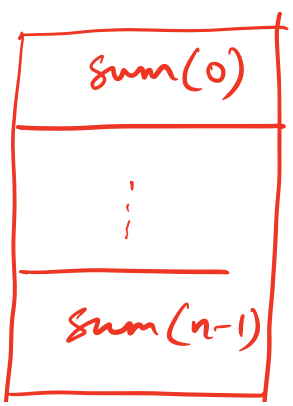
$= n + cn \log_2 n$

$= O(n \log n)$.

Space Complexity

Max. size of call stack at any moment.

eg:- $sum(n) = sum(n-1) + n$



no. of stack frames $= n+1$

$\Downarrow$

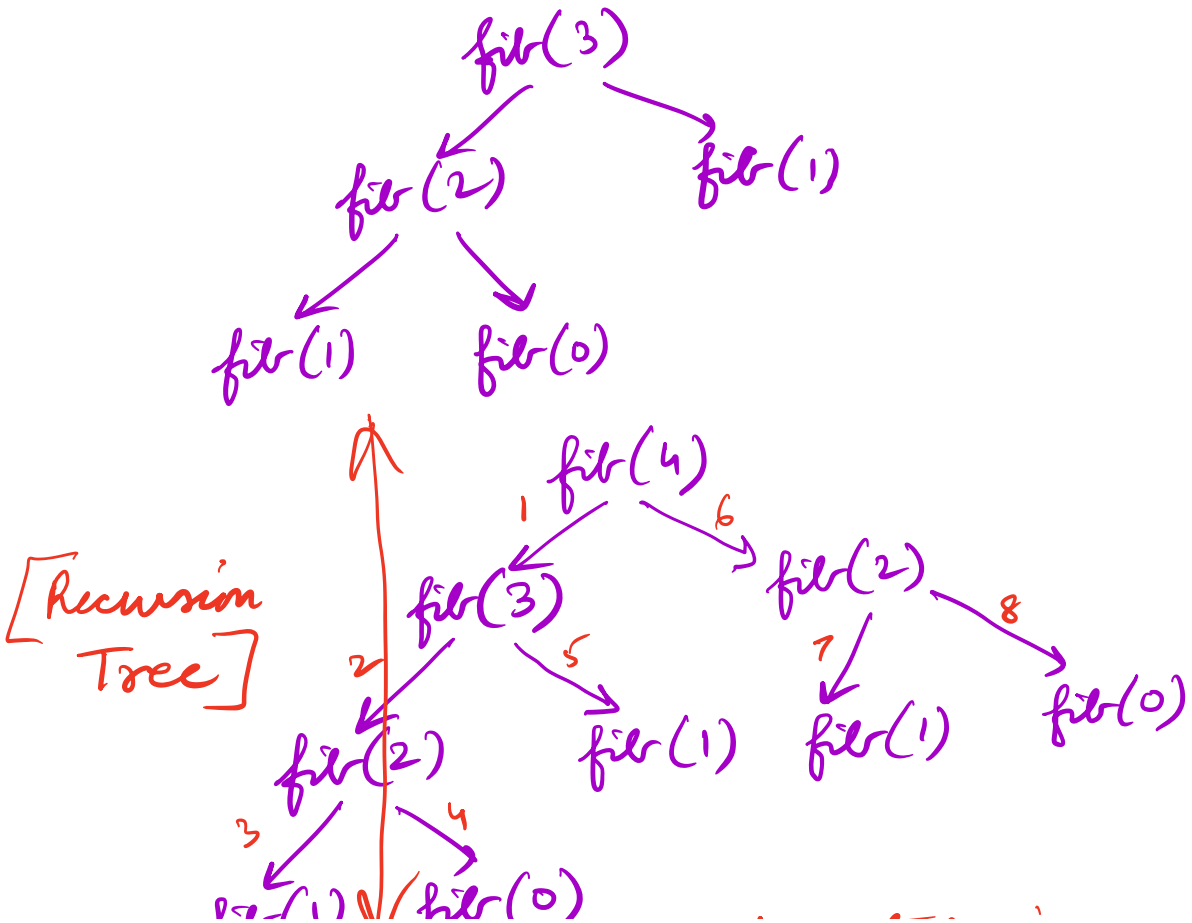
$O(n)$

sum(n)

eg 2: $fib(n) = fib(n-1) + fib(n-2)$

$n = 3$

Space complexity is $O(n)$.



Max stack size here → 4.

S.C. = Depth of the recursion tree

pow(a, b)

```
sp = pow(a, b/3);
value = sp * sp * sp;
if (b%3 == 0)
    return value;
if (b%3 == 1)
    return value * a;
return value * a * a;
```

2 conditions checked

$a^5 = (a^1)^3 * a * a$ (sp)

While we used b/2 ⇒ 1 condition.

b → b/2

$T(n) = T(n/2) + 1$

$= T(n/2^k) + k$

$= 1 + \log_2 n$

$= (1 + \log_2 n)$

$\log n - \log n$

b → b/3

$T = T(n/3) + 2$

$= T(n/9) + 4$

$= T(n/3^3) + 6$

$= T(n/3^k) + 2k$

$k = \log_3 n$

$(1 + 2\log_3 n)$

$\frac{1}{\log 2} = \frac{1}{0.301}$

Base 10 :-

$\log 2 = 0.301$

$2\log_3 n = \frac{\log n}{\log 3} * 2$

$= \log n * \frac{2}{0.477}$

$\frac{1}{0.301} < \frac{2}{0.477}$

$\Rightarrow \boxed{\frac{\log n}{\log 2}} < \frac{2\log n}{\log 3}$

$\longrightarrow$ Best.

Both are $O(\log n)$.