RECURSION 2

* Agenda :

- Sum of digits
- Power of a number
- Optimize power function
- Time complexity analysis (substitution, Recursive tree)

---

* Sum of digits :

n = 124

I need all digits of the above mentioned number.

sum = 1 + 2 + 4 => 7

Quiz :

n = 123
n % 10 => remainder => 3

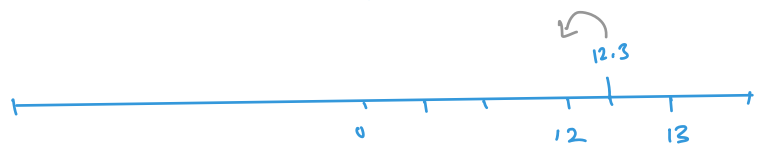
10 $\sqrt{123}$ 12
10

23
20
---
3 Ans

Quiz :

n = 123
n // 10 → floor value

n / 10 ⇒ $\frac{123}{\cancel{10}}$ ⇒ 12.3



floor value = 12

n = 124

⇒ 7 4 + (12)

2 + (1)

3

1 1

def add(124):

7

4 + add(12)

3

2 + add(1)

1

1 + add(0)

0

add(0) → Base condition

Base condition:

```
if n == 0:
    return 0
```

Recurrence relation:

⇒ n = 124

⇒ n % 10 + add(n // 10)

⇒ 4 + add(12)

return n % 10 + add(n // 10)

Code:

```
def    add (n) :

    # Base Condition
    if  n  ==  0 :
        return 0

    return  n % 10  +  add(n//10)
```

```python
def sum_of_digits(n):
    # base condition
    if n == 0:
        return 0

    return n % 10 + sum_of_digits(n // 10)
```

=> n = 1203

6

3 + sum_of_digits(120)

3 0 + sum_of_digits(12)

3 2 + sum_of_digits(1)

1 1 + ~~sum_of_digits(0)~~ 0

★ **Power of a number :**

power (2, 3) ⇒ power (base, pwr)

8

power (2, 2) 2×2^2

4

power (2, 1) ⇒ 2×2^1

2

power (2, 0) # base condition

8

$2 \times$ power (2, 2) 2×2^2

4

$2 \times$ power (2, 1) ⇒ 2×2^1

2

$2 \times$ power (2, 0) # base condition

Base condition :

```
if pwr == 0 :
    return 1
```

recurrence relation

power (base, pwr)

base × power (base, pwr − 1)

★ **Optimized power function :**

⇒ power (2 , 16)

power(2, 15)

power (2 , 14)

power (2, 13)

⋮

power (2, 0)

Here it will take 16 recursive calls.

number of calls ∝ Time Complexity

In above case we will be doing almost n recursive calls. (if power = n).

power = 128 ⇒ 128 recursive calls
power = n ⇒ n recursive calls

$T(c) = \boxed{O(n)}$

$\Rightarrow \quad a^{16} \quad \Rightarrow \quad a^8 \times a^8 \qquad$ (i)

$$\downarrow$$

$a^4 \times a^4 \qquad$ (ii)

$$\downarrow$$

$a^2 \times a^2 \qquad$ (iii)

$$\downarrow$$

$a^1 \times a^1 \qquad$ (iv)

$$\downarrow$$

$a^0 \Rightarrow 1$

$T(c) = \boxed{O(\log n)}$

$a^{128} \rightarrow 64 \rightarrow 32 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$

(7 calls)

★ $a^{19} = a \times a^{18} \Rightarrow a \times a^9 \times a^9$

$$\downarrow$$

$a^9 \cdot a^9$

$$\downarrow$$

$a \times a^8$

$$\downarrow$$

$a^4 \times a^4$

$\vdots$

Recurrence relation :

$a^n \Rightarrow a^{n//2} \times a^{n//2}$ (n is even)

$a^n \Rightarrow a \times a^{n//2} \times a^{n//2}$ (n is odd)

$a^{17} = a \times a^8 \times a^8$

```python
def opt_power(a, n):
    # base condition
    if n == 0:
        return 1

    half = opt_power(a, n // 2)

    # Check for even odd
    if n % 2 == 0:
        return half * half
    else:
        return a * half * half
```

power (2, 16)

half = 2^8

256 $2^8 \times 2^8$ → half ⇒ 2^4

16 $2^4 \times 2^4$ → half ⇒ 2^4

4 $\quad 2^2 \times 2^2 \to$ half $\Rightarrow 2^1$

2 $\quad 2^1 \times 2^1 \to 2^0$

$2 \times 2^0 \times 2^0$

★ Length of a string

- Iterative
  - TC ⇒ O(N)
  - S ⇒ O(1)
- Recursion
  - TC ⇒ O(N)
  - O(N)

\# Graphs, trees, BST ⇒ Recursion

★ Time Complexity Analysis :

i) Substitution :

\# def add(n) :
  if n == 1 :
    return 1 } O(1) → C1

  return n + add(n-1)
  ↳ O(1) ↳ T(n-1)

$\hookrightarrow c_2$

$T(n) = T(n-1) + \overbrace{c_1 + c_2}^{c}$

$\boxed{T(n) = T(n-1) + c}$ — (1)

Recursive relation

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

Substitute value of 2 in eqn 1

$T(n) = T(n-2) + c + c$
$T(n) = T(n-2) + 2c$ — (3)

$T(n) = T(n-3) + 3c$

⋮ -- k recursive calls -- ⋮

$T(n) = T(n-k) + kc$ — (3)

after k recursive call base condition hits

$\therefore$ $n - k = 1$

$\therefore$ $k = n - 1$

Substitute value of k in eq (3)

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

$= T(n-n+1) + nc - c$

$= \cancel{T(1)} + n\cancel{c} \cancel{-c}$

$\boxed{T(c) = O(n)}$

ii) Recursive Tree :

$\Rightarrow$ add (5)

add (4)

add (3)

add (2)

add (1)

add (0)

Height of a tree = 5

if n = 5, height = 5

if n, height = n

$T(c)$ = Height of tree

$= \boxed{O(n)}$