Recursion-2

Recursion-2

Q1. Sum of digit using recursion

$N = 124$

$f(N)$

$1+2+4 = 7$

$\Rightarrow 4 + f(12)$

$\Rightarrow 4 + 2 + f(1)$

$\Rightarrow 4 + 2 + 1 + f(0)$ 0

$\Rightarrow$ Steps of Recursion:

1. Assumption
2. Main Logic
3. Base Condition

```python
def sum_of_digits(N):
    if N==0:
        return 0
    return (N%10) +
        sum_of_digits(N//10)
```

$N = 439632$

$N \% 10 \Rightarrow 2 + \text{sum\_of\_digits}(N//10)$

$N//10 \Rightarrow 43963$

$N = 43963\boxed{2}$

1. $N \% 10 = 2$

2. $N//10 = 43963\,6$

3. $N \% 10 = 6$

4. $N//10 = 4396\,3$

Q2. **Power Function**

Given a and n, calculate a^n. (positive; $a > 0$)

× a**n

× math.pow

Negative n is out of scope for now.

Write your own recursive function.

$\boxed{a^0 = 1}$

$3^3 = 27$

1. Assumption
2. Main Logic
3. Base Condition

$3^5 = 3^4 \cdot 3$

$= 3^3 \cdot 3 \cdot 3$

$= 3^2 \cdot 3 \cdot 3 \cdot 3$

$= 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3$

Input:

$n >= 0$

```
def power(a, n):
        if n == 0:
            return 1
        return pow(a,n-1) * a
```

$a^n = a^{n-1} \cdot a$

power(3,0)

→ power(3, -1) * 3

→ power(3, -2) * 3

```
res = 1
for i in range(n):
    res *= a
```

TC: $O(N)$

SC: $O(1)$

=> Dry Run

a=3 n=3

```
def power(a, n):
    if n==0:
        return 1
    return pow(a,n-1) * a
```

= 9 * 3 = 27

$\underbrace{\text{pow}(3,3)}$

27

a=3 n=2

```
def power(a, n):
    if n==0:
        return 1
    return pow(a,n-1) * a
```

9

= 3*3
= 9

~~a=3~~ n=1

```
def power(a, n):
    if n==0:
        return 1
    return pow(a,n-1) * a
```

3

= 1 * 3 = 3

a=3, n=0

```
def power(a, n):
    if n==0:
        return 1
    return pow(a,n-1) * a
```

1

	SC.
Pow(3,0)	2
Pow(3,1)	2
Pow(3,2)	2
Pow(3,3)	2
	8

n+1

Stack.

Space = 2(n+1)
= O(n)

Implicit stack space

Time Complexity Analysis

```
def sum(n):
    if n == 0:
        return 0
    return n + sum(n-1)
```

$T(N)$

c_1

$\Rightarrow T(0) = 1$

$\Rightarrow T(N)$ is time taken for input N.

$\Rightarrow T(N-1)$

add $\Rightarrow c_2$

$$T(N) = c_1 + c_2 + T(N-1)$$

① Recursive Relation.

$$\boxed{T(N) = T(N-1) + c}$$, where c is constant.

$= T(N-2) + c + c \quad = T(N-2) + 2c$

$N-1$

Substitute $(N-1)$ in above.

$\Rightarrow T(N-1) = T(N-1-1) + c$

$$\boxed{T(N-1) = T(N-2) + c}$$

② Substitution Method

$= T(N-3) + c + 2c \quad = T(N-3) + 3c$

$= \boxed{T(N-4) + c} + 3c \quad = T(N-4) + 4c$

③ Generalise K

$$\boxed{T(N) = T(N-K) + K * C}$$

$\hookrightarrow$ right side should be all known.

$K=1$	$T(N) = T(N-1) + c$
$K=2$	$T(N) = T(N-2) + 2c$
$K=4$	$T(N) = T(N-4) + 4c$

find

$$\boxed{T(N) = T(N-K) + K * C}$$

(4) Find the K value using base condition.

We know that $T(0) = 1$ ⟹ Base Condition

$N - K = 0$

⟹ $K = N$ ⟹ will be the space complexity.

(5) Put the K value

⟹ $T(N) = T(0) + N * C$

$= 1 + N * C$

What is Time Complexity ?

$$\boxed{T(N) = O(N)}$$

=> Optimised Power Idea.

```
def power(a, n):
    if n==0:
        return 1
    return pow(a,n-1)*a
```

=> $T(N)$

=> ① $T(N) = c + T(N-1)$

② $T(N) = T(N-K) + K*c$

③ $K = N$

④ $T(N) = T(0) + N.c = 1 + N.c$

$T(N) = O(N)$

$S.C. = K = O(N)$

$a^{16} = a^8 \cdot a^8$

$a^4 \cdot a^4$

$a^2 \cdot a^2$

$a \cdot a$

$\neq \cancel{2 \cdot a^8}$

$a^m \cdot a^n = a^{m+n}$

$a^{17} = \boxed{a^{16}} \cdot a$

$19-1=18$

$a^{10} = a^{5} \cdot a^{5}$ $\qquad$ $a^{19} = \cancel{a^{18} \cdot a}$ $\qquad$ $19//2 = 9$

$5//2 = 2$ $\qquad$ $\cancel{a^{4} \cdot a}$ $\qquad$ $18//2 = 9$

$4//2 = 2$ $\qquad$ $a^{2} \cdot a^{2} \cdot a$ $\qquad$ $a^{9} \cdot a^{9} \cdot a$

$a \cdot a$ $\qquad$ $9-1=8$ $\qquad$ $\cancel{a^{8} \cdot a}$ $\qquad$ $9//2 = 4$

$a^{4} \cdot a^{4} \cdot a$ $\qquad$ $8//2 = 4$

$a^{2} \cdot a^{2}$

$a \cdot a$

Main Logic

Generalized expression

$$a^{n} = \begin{cases} a^{n//2} \cdot a^{n//2}, & \text{if } n \text{ is even} \\ a^{n-1} \cdot a, & \text{if } n \text{ is odd.} \end{cases}$$

is even.

$$\boxed{a^{n} = \begin{cases} a^{n//2} \cdot a^{n//2}, & \text{if } n \text{ is even} \\ \underbrace{a^{n//2} \cdot a^{n//2}} \cdot a, & \text{if } n \text{ is odd} \end{cases}}$$

$a^{7} = a^{3} \cdot a^{3} \cdot a$

$\underbrace{a \cdot a} \cdot a$

$a^{9} = \underbrace{a^{4}} \cdot a^{4} \cdot a$

$a^{2} \cdot a^{2}$

$a \cdot a.$

$$a^n = \begin{cases} a^{n//2} \cdot a^{n//2}, & \text{if } n \text{ is even} \\ a^{n//2} \cdot a^{n//2} \cdot a, & \text{if } n \text{ is odd} \end{cases}$$

pow = optimized power,

```
def pow(a, n):                          --> T(N)
    if n == 0 :
        return 1
    if n % 2 == 0 :
        return pow(a, n//2) * pow(a, n//2)
    else :
        return a * pow(a, n//2) * pow(a, n//2)
```

$T(N/2)$ $T(N/2)$ a^{n-1}

Ashish

20d ↳ Zaheer $a^{N/2}$

20d ↳ Afifa $a^{N/2}$

$$\boxed{T(N) = 2T(N/2) + 1}$$

$T(0) = 1$

$\Rightarrow \boxed{T(N) = O(N)}$

HW: Solve it

Final Optimized Code.

```
def pow(a, n):                → T(N)
    if n==0 :
        return 1
    t = pow(a, n//2)          → T(N/2)
    if n % 2 == 0 :
        return t * t
    else :
        return a * t * t
```

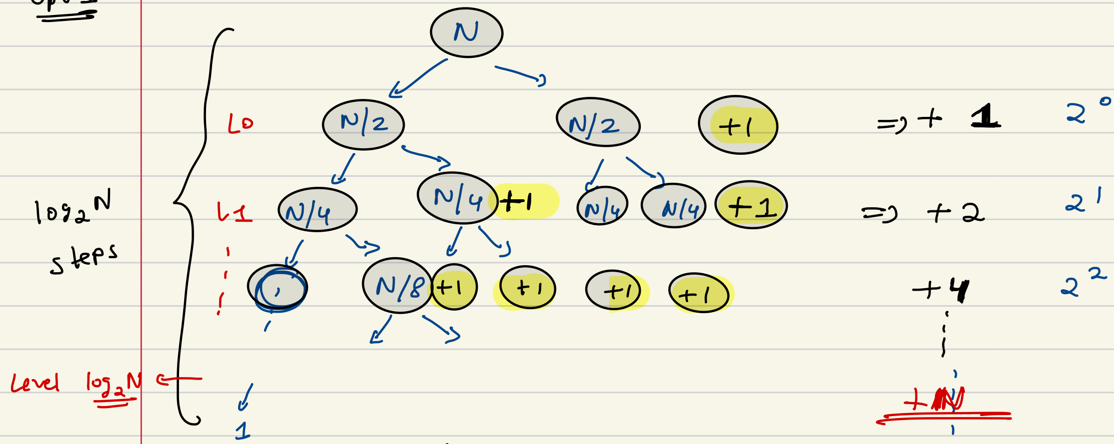
$$T(N) = T(N/2) + 1$$
$$T(0) = 1$$

} HW: Solve it

$\Rightarrow \quad T(N) = O(\log N)$

$K = \log_2 N = S.C.$

Opt 1



$$T(N/2) = 2T(N/4) + 1$$

$$2^{\log_2 N}$$

$$= N + \frac{N}{2} + \frac{N}{4} + \cdots + 1 + \cdots$$

a

$$GP = \frac{a}{1-r}$$

$$= \frac{N}{1 - \frac{1}{2}} = \underline{2N}$$

$r = 1/2$

$$\boxed{O(N)}$$

$$= \frac{N}{1/2} = 2N.$$

Opt 2

$$\begin{array}{l} N \\ \downarrow \end{array}$$

$$\left.\begin{array}{ll} N/2 & +1 \\ \downarrow & \\ N/4 & +1 \\ \downarrow & \\ N/8 & +1 \\ \downarrow & \\ \vdots & \vdots \\ \downarrow & \\ 1 & = 1 \end{array}\right\} O(\underline{\log_2 N})$$

"abcd"

d +

```
def reverse(s):
    if len(s)==0:
        return ''

    return s[-1] + reverse(s[0:-1])
```

'welcome'

target = 'el'

```
def check(s, t):

    if len(s)==0
        return 0

    if s[:len(t)] == t :
        return 1 + check(s[1:], t)
    else:
        return check(s[1:], t)
```

s = 'ababab'

t = 'aba'

s = 'aaaaa'

t = 'aaa'