Misc Problems

Agenda

→ Reverse a number
→ Harmonic Sum
→ Space complexity
→ Interview problem

★ Reverse a number

⇒ n = 125
ans = 521
return ans

find last digit of a number
% operator (n % 10)

5 , 2 , 1 ⇒ 521

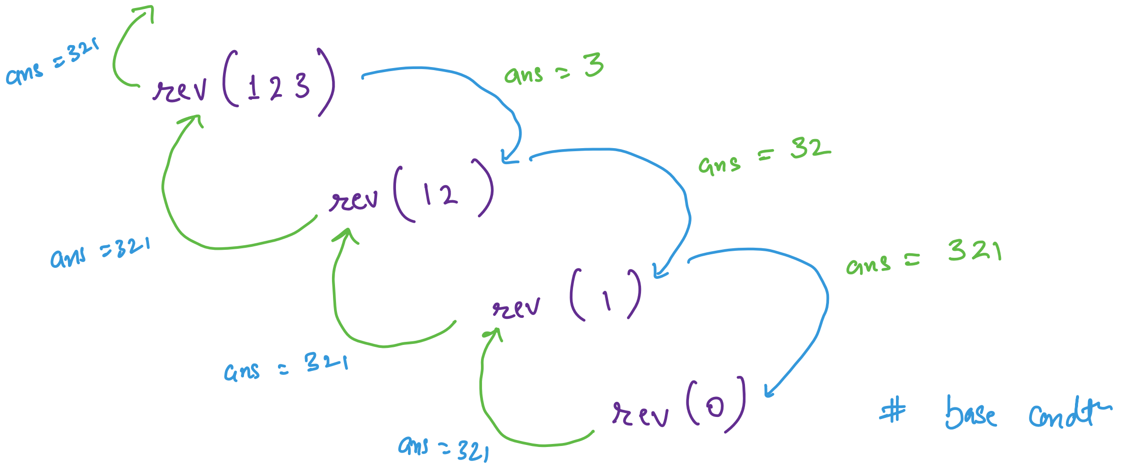
ans = 0
ans = 50 + 2
ans = 520 + 1

ans = 0

$\boxed{ans = ans \times 10 + n \% 10}$

return rev(n//10 , ans)

★ Recursive Tree



TC = O(len(n))

★ Harmonic Sum

n = 5

$ans = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} = 2.28333$

$ans = \frac{1}{1} + \frac{1}{2} + \cdots + \frac{1}{n}$

$ans = \frac{1}{5} + \frac{1}{4} + \frac{1}{3} + \frac{1}{2} + \frac{1}{1}$

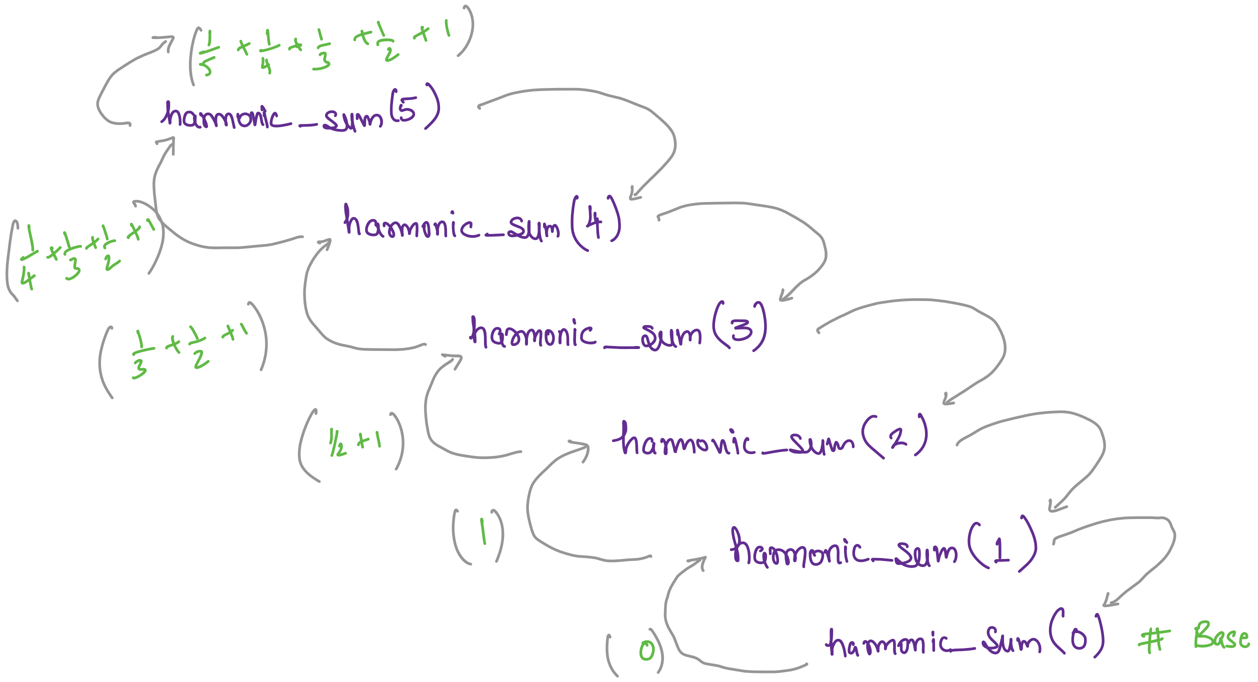
Base condition :

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

Recurrence relation = $\frac{1}{n}$ + harmonic (n-1)

★ Recursive tree

harmonic_sum(5)



TC = $O(n)$

★ **Space Complexity**

Whenever we create new objects some space is occupied in computer memory.

$\boxed{\text{space consumed} \propto \text{space complexity}}$

Q.1

```
def func():

    x = 5        -------- O(1)
    y = 10       ------- O(1)
    z = 15       ------- O(1)
```

Ans Total space complexity = $O(1) + O(1) + O(1)$

SC = $3 \times O(1)$

TC = $O(1)$

Q.2

```
def func():
    return "Rahul"
```

Ans

$SC = O(1)$

$TC = O(1)$

Q.3

```
def func(N) :

    x = N                 — O(1)
    y = $x^2$             — O(1)
    z = x + y             — O(1)
    lst => size (N)       — O(N)
```

Ans :

$SC = \cancel{3 \times O(1)} + O(N)$

$SC = O(N)$

$TC = O(N)$

Q.3

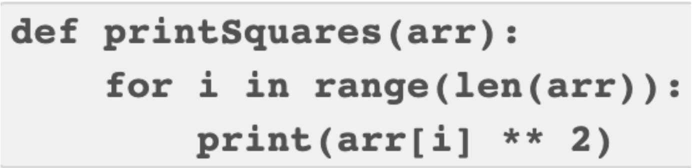
```
fun (int N) {

    int x = N                    — O(1)
    int y = $x^2$                — O(1)
    int z = x + y                — O(1)
    arr[] = int [N]              — O(N)
    l[][] = int [N] [N]          — $O(N^2)$

}
```

Ans : $SC = \cancel{O(1)} + \cancel{O(N)} + O(N^2)$

$SC = O(N^2)$

$TC = O(N^2)$

Quiz



```
def printSquares(arr):
    for i in range(len(arr)):
        print(arr[i] ** 2)
```

It is not consuming any new space, so SC is constant.

$SC = O(1)$

n → len(arr)

$TC = O(n)$

=> n = 128

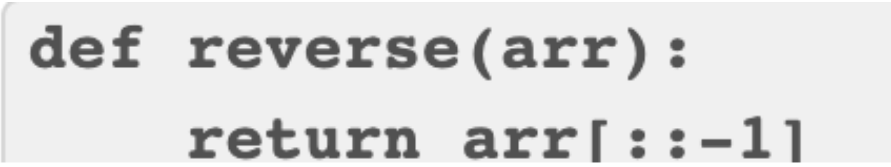
```
li = []
```

=>

```
while n > 1 :
    li.append(n)
    n = n // 2
```

Ans : SC = $O(\log_2 n)$

Quiz



```
def reverse(arr):
    return arr[::-1]
```

↳ list slicing

When we do list slicing a new list is created i.e of size corresponding to list slicing.

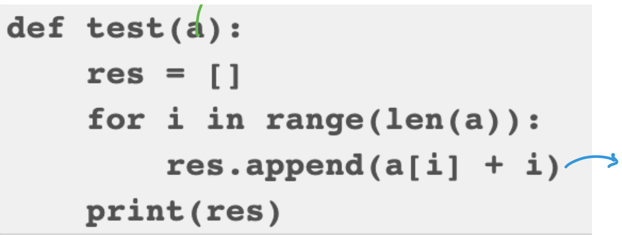
SC = $O(N)$

↳ size of list

TC = $O(N)$

Quiz

↗ list → size (N)



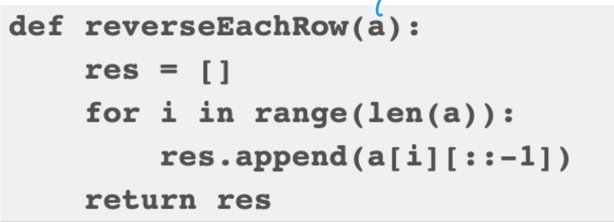
```
def test(a):
    res = []
    for i in range(len(a)):
        res.append(a[i] + i)
    print(res)
```

→ N

Ans SC = O(N)

Quiz

size → NxM



```
def reverseEachRow(a):
    res = []
    for i in range(len(a)):
        res.append(a[i][::-1])
    return res
```

Ans SC = O(N x M)

HW : find TC

Next Session : Work while stack is building v/s work while stack is falling.

Ex : print series in forward & reverse order