Ruchika Shashidhara
NU 002245068

Date
Page 1

PSA Class - 5

Recursion

Need for recursion

Base condition: knows how to solve the problem

Other condition: Invokes other conditions & base condition i.e. Breaking down the problem

Ex: Factorial: (Iterative)

Factorial using iteration

```
int F(int n) {
    int s=1
    for(int i=2; i<=n; i++){
        s*=i
    }
    return s;
}
```

→ here it works for Python, but in C/Java int has limited space so we need to have long int to space to accomodate large n

n	s	i
5	1	
5	2	2
5	6	3
5	24	4
5	120	5

Time complexity = $\theta(n)$
Space complexity = $\theta(1)$

Scanned with OKEN Scanner

classmate
Date
Page 2

Factorial : (Recursive)

Factorial using Recursion

```
int F(int n) {
    if (n < 2) {
        return 1;
    }
    return n * F(n-1);
}
```

Stack

	return
n = 1	120 * 1
n = 2	120 * F(2-1)
n = 3	60 * F(3-1)
n = 4	20 * F(4-1)
n = 5	5 * F(5-1)

Time complexity = $\theta(n)$

Space complexity = $\theta(n)$

Stack overflow can occur

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

$T(1) = 1$, $T(0) = 0$

n → C

↓

n-1 → C

↓

n-2 → C

↓

⋮

↓

1 → n-(n-1) → C

Time complexity = nC

$= \theta(n)$

Scanned with OKEN Scanner

Print digits of a number in reverse order iteration

```
void P(int n){
    if n==0 : print(n)
    while(n){
        print(n%10)
        n = n//10
    }
```

n	print
n=1986	6
n=198	8
n=19	9
n=1	1

Time complexity = $\Theta(\log_{10} n)$

Space Complexity = $\Theta(1)$

Print digits of a number in reverse order recursion

```
void R(int n){
    if (n<10) {print(n); return}
    print(n%10)
    return R(n//10)
}
```

n	stdout	stack
1	1	return
19	9	R(1)
198	8	R(19)
n=1986	6	R(198

Time complexity = $\Theta(\log_{10} n)$

Space complexity = $\Theta(\log_{10} n)$

$T(n) = T(n/10) + C$

$T(0...9) = 1$

$n \to n/10 \to n/100 \to \dots \to n/10^k$

$\frac{n}{10^k} = 1$

$n = 10^k$

$\log_{10} n = k$

Scanned with OKEN Scanner

Print digits of a number recursion

```
void A (int n) {
    if (n < 10) {print(n); return}
    n = A(n//10)
    print(n%10)
    return;
}
```

n	Std out	Stack
1	1	9
19	9	A(1)
198	8	A(19)
1986	6	A(198)

Time complexity = $\Theta(\log_{10} n)$
Space complexity = $\Theta(\log_{10} n)$

Print digits of a number iteration

```
void A (int n) {
    a = []
    if n == 0 : print(n)
    while (n) {
        a.append(n%10)
        n = n//10
    }
    for ai in a[::-1] // a reversed
        print(ai)
    return;
}
```

n → [1986]
a → [6]

n → [198]
a → [6|8]

n → [19]
a → [6|8|9]

n → [1]
a → [6|8|9|1]

Time Complexity = $\Theta(\log_{10} n)$
Space Complexity = $\Theta(\log_{10} n)$

Scanned with OKEN Scanner

Print digits of a number iteration (constant space)

```
int R(int n){
    s = 0                          // memorized for entire procedure
    while (n) {
        s = s * 10 + (n % 10)
        n = n // 10
    }
    return s;
}
```

n → 1986, s → 6
n → 198, s → 68
n → 19, s → 689
n → 1, s → 6891

Time complexity = $\Theta(\log_{10} n)$
Space complexity = $\Theta(1)$

```
int R1 (int n) {
    int s = 0
    int x = R(n, s)
}
```

n	s
1986	0

Need for helper function

helper function:

```
int R(int n, int s) {
    s = s * 10 + (n % 10)
    if (n < 10) { return s; }
    int x = R(n // 10, s)
    return x;
}
```

n=1, s=689 → s = 6891
n=19, s=68 → s = 689, x = R(1, 689)
n=198, s=6 → s = 68, x = R(19, 68)
n=1986, s=0 → s = 6, x = R(198, 6)

6891

Time Complexity = $\Theta(\log_{10} n)$
Space Complexity = $\Theta(\log_{10} n)$

Scanned with OKEN Scanner

HOP : Return no. of hops

	0	1	2	3	4	5
a	5	1	0	4	2	3

Find no. of hops to get to
Ex: 3
len of list is unknown
there is no repetition of numbers

a[3] = 4
a[4] = 2
a[2] = 0
a[0] = 5
a[5] = 3

} 4

∴ No. of hops = 4

Hop.ipynb Iteration Solution

Stack Trace

```
def hop_easy(self, a: List[int], f: int) -> int:
    t = f
    h = 0 // number of hops
    while(True):
        if a[t] == f):
            return h
        else:
            t = a[t]
            h = h + 1
    return h
```

a -> 5 1 0 4 2 3

t -> 3
h -> 0

t -> 4
h -> 1

t -> 2
h -> 2

t -> 0
h -> 3

t -> 5
h -> 4

t -> 3
h -> 4

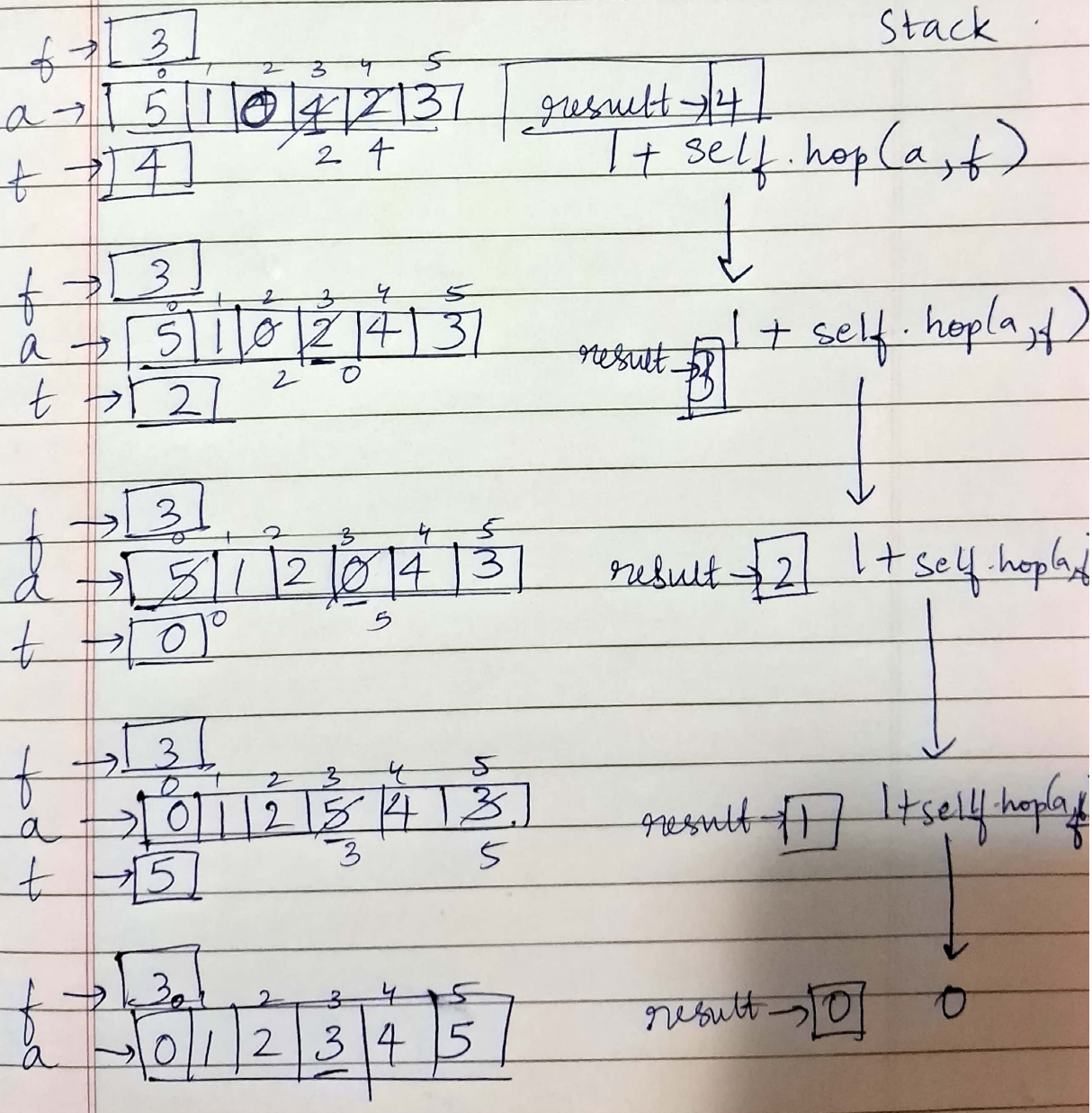
Time Complexity = O(n) Space Complexity = Θ(1)

Scanned with OKEN Scanner

Hop.ipynb Recursion Solution

```
def _hop(self, a: List[int], f: 'int') -> 'int':
    if (a[f] == f): return 0
    t = a[f]
    a[t], a[f] = a[f], a[t]
    result = 1 + self._hop(a,f)
    a[f], a[t] = a[t], a[f]
    return result
```

Stack Trace



Scanned with OKEN Scanner