L2 Problems on Recursion

Saturday, December 10, 2022 11:39 AM

Now Here we deal with some basic Recursion problem and understand that How to work Recursion

- → Print Name 5 times
- → Print Linearly 1 to N
- → Print from N to 1

→ base discuss on recursion

- → Print Linearly from 1 to N (by backtracking)
- → Print from N to 1 (By Backtracking)

→ Backtracking

P1 Print Name 5 Time

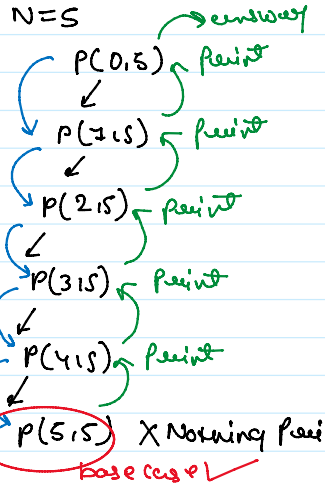
→ Print N times

```
def p1(count,N):
    if count >= N:
        return
    print("Prince Singh")
    count+=1
    p1(count,N)

p1(0,5)
```

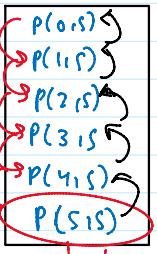
→ Base case

Recursion Tree

N=5



Recursive Stack =



→ base case

Time Complexity = O(N)

→ because Linear Operation

Space Complexity = O(N)

→ Recursion Stack Space

P2 Print 1 to N Times

Similar problem as we learned in Previously

→ N times print

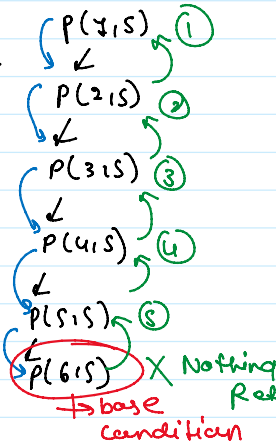
```
def p2(i,N):
    if i > N:
        return
    print(i)
    i+=1
    p2(i,N)

p2(1,10)
```

→ base condition

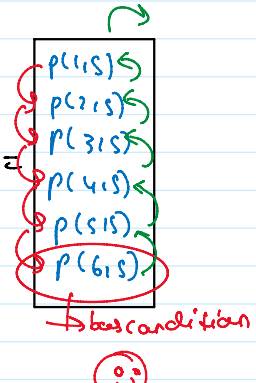
Recursion Tree → any

N=5

i=1



Recursion Stack =



Time Complexity = O(N)

→ because Linear Operation

Space Complexity = O(N)

→ because Linear Recursion

Space Complexity = O(N)
→ Recursion Stack Space

→ base condition Return

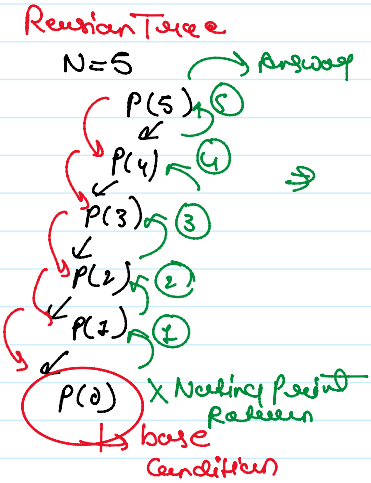
P3 Print N to 1 Time

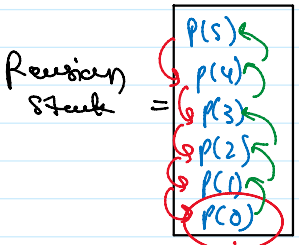
That is the Reverse problem of the (P2) that we learned previously and Hence we used the Reverse Concept of (P2)

```
def p3(N):
    if N == 0:
        return
    print(N)
    N-=1
    p3(N)

p3(10)
```

N to 1 print

base case





Time Complexity = O(N)
→ because Linear Recursion

Space Complexity = O(N)
→ Recursion Stack Space

P4 Print 1 to N time using Backtracking

Our task is print the 1 to N without increment the printing Variable

i+1 ✗ Not allowed
How we do that !!

```
def backTrackingP1(i,N):
    if i < 1:
        return
    backTrackingP1(i-1,N)
    print(i)

backTrackingP1(5,5)
```

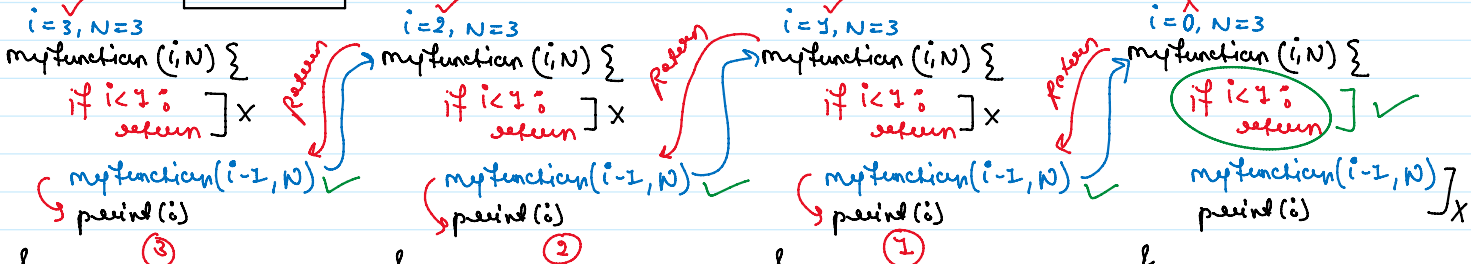
My target that I want to print

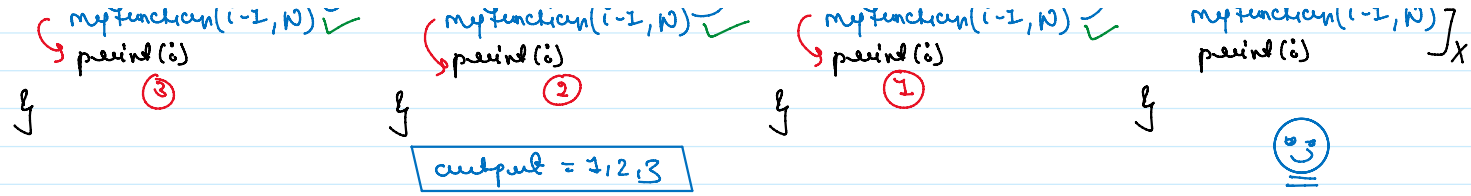
base case

Backtrack

⇒ Let's understand How to Code Work Internally and How Work backtracking there

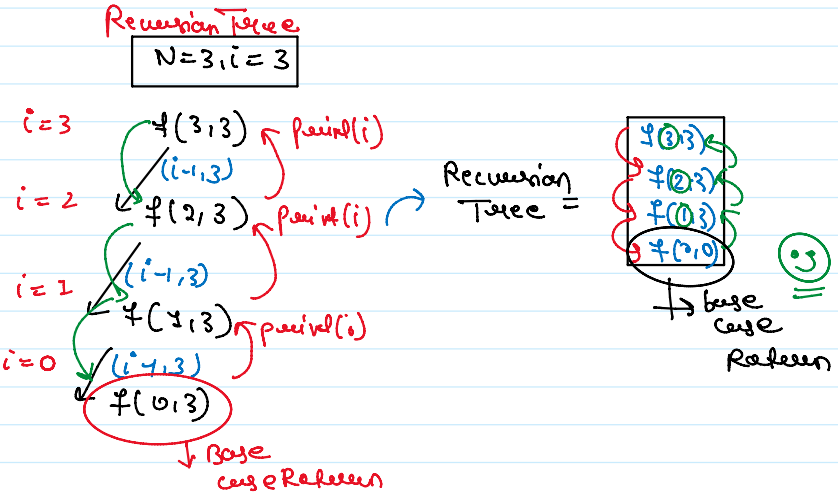
N=3, i=3





Time Complexity = O(N)
→ because Linear Operation

Space Complexity = O(N)
→ Recursion Stack Space



P5 Print N to 1 time using Backtracking

That is the Reverse Concept and Logic of previously problem

(i-1) ✗ Not Allowed

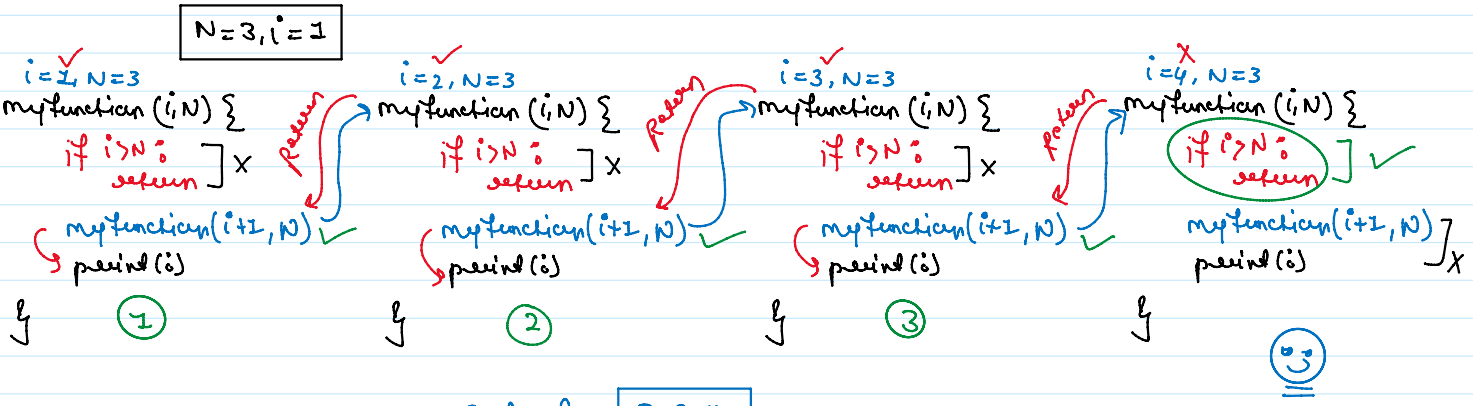
How to Solved !!

```
def backTrackingP2(index,N):
    if index > N:
        return
    backTrackingP2(index+1,N)
    print(index)

backTrackingP2(1,5)
```

→ base case

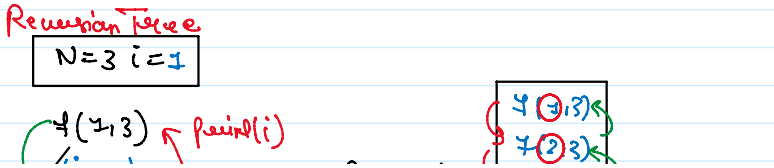
→ backtracking

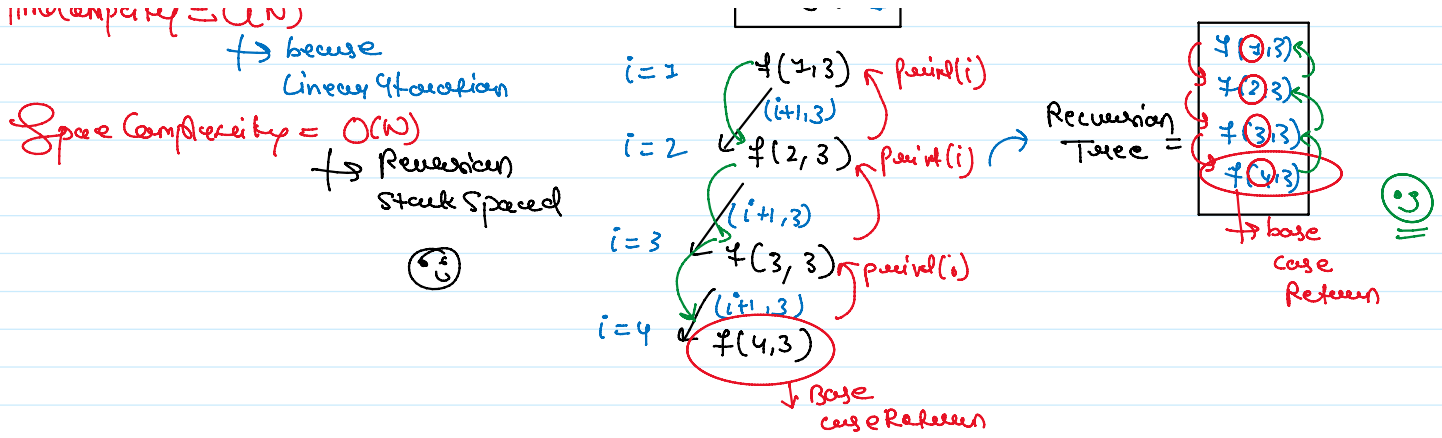
⇒ Let's understand How to Code Work Internally and How Work backtracking there



output = 3,2,1

Time Complexity = O(N)
→ because Linear Operation






```python
def p1(count,N):
    if count >= N:
        return
    print("Prince Singh")
    count+=1
    p1(count,N)

p1(0,5)

2 usages
def p2(i,N):

    if i > N:
        return
    print(i)
    i+=1
    p2(i,N)

p2(1,10)

2 usages
def p3(N):

    if N == 0:
        return
    print(N)
    N-=1
    p3(N)

p3(10)

2 usages
def backTrackingP1(i,N):
    if i < 1:
        return

    backTrackingP1(i-1,N)
    print(i)

backTrackingP1(5,5)

2 usages
def backTrackingP2(index,N):
    if index > N:
        return

    backTrackingP2(index+1,N)
    print(index)


backTrackingP2(1,5)
```

```java
public class L2_Problems_on_Recursion {

    public static void main(String[] args) {


        problem1( count: 0, N: 5);
        problem2( i: 1, N: 5);
        problem3( N: 5);
        BackTrackingproblem4( index: 5, N: 5);
        BackTrackingproblem5( index: 1, N: 5);
    }

    2 usages
    private static void BackTrackingproblem5(int index, int N) {

        if(index > N){
            return;
        }
        BackTrackingproblem5( index: index+1,N);
        System.out.println(index);
    }

    2 usages
    private static void BackTrackingproblem4(int index, int N) {

        if(index < 1){
            return;
        }
        BackTrackingproblem4( index: index-1,N);
        System.out.println(index);
    }

    2 usages
    private static void problem3(int N) {
        if(N == 0){
            return;
        }
        System.out.println(N);
        N-=1;
        problem3(N);
    }

    2 usages
    private static void problem2(int i, int N) {

        if(i > N){
            return;
        }
        System.out.println(i);
        i+=1;
        problem2(i,N);
    }

    2 usages
    private static void problem1(int count, int N) {

        if (count >= N){
            return;
        }
        System.out.println("Prince Singh");
        count+=1;
        problem1(count,N);

    }
}
```

→ All the code into the Single file