DSA through Java

Recursion

Saurabh Shukla (MySirG)

Agenda

1. What is a recursion?
2. Recursion Tree | Tracing code
3. How to approach recursive solution?
4. Few examples

What is a recursion?

- Function calling itself is called recursion
- A recursive method solves a problem by calling a copy of itself to work on a smaller problem.
- It is important to ensure that the recursion terminates.

```
void f1( )
{
    S.o.p ("Hello");
    f1 ();
    S.o.p ("Bye");
}
```

```
void f2()
{
    S.o.p ("A");
}
```

Never ending or infinite recursion

- Each time the function call itself with a slightly simpler version of the original problem.

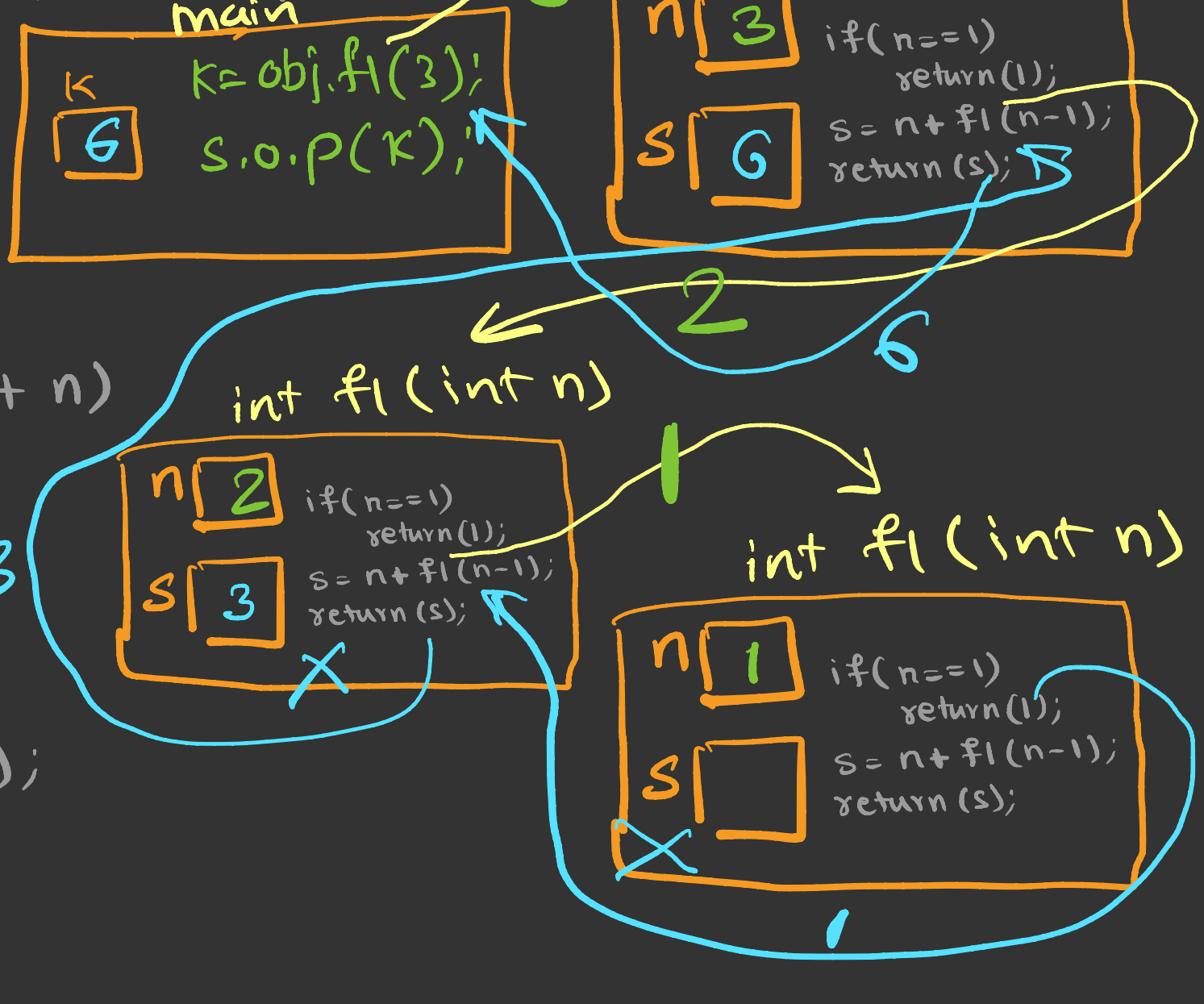
- Recursive code is generally shorter and easier to write than iterative code.

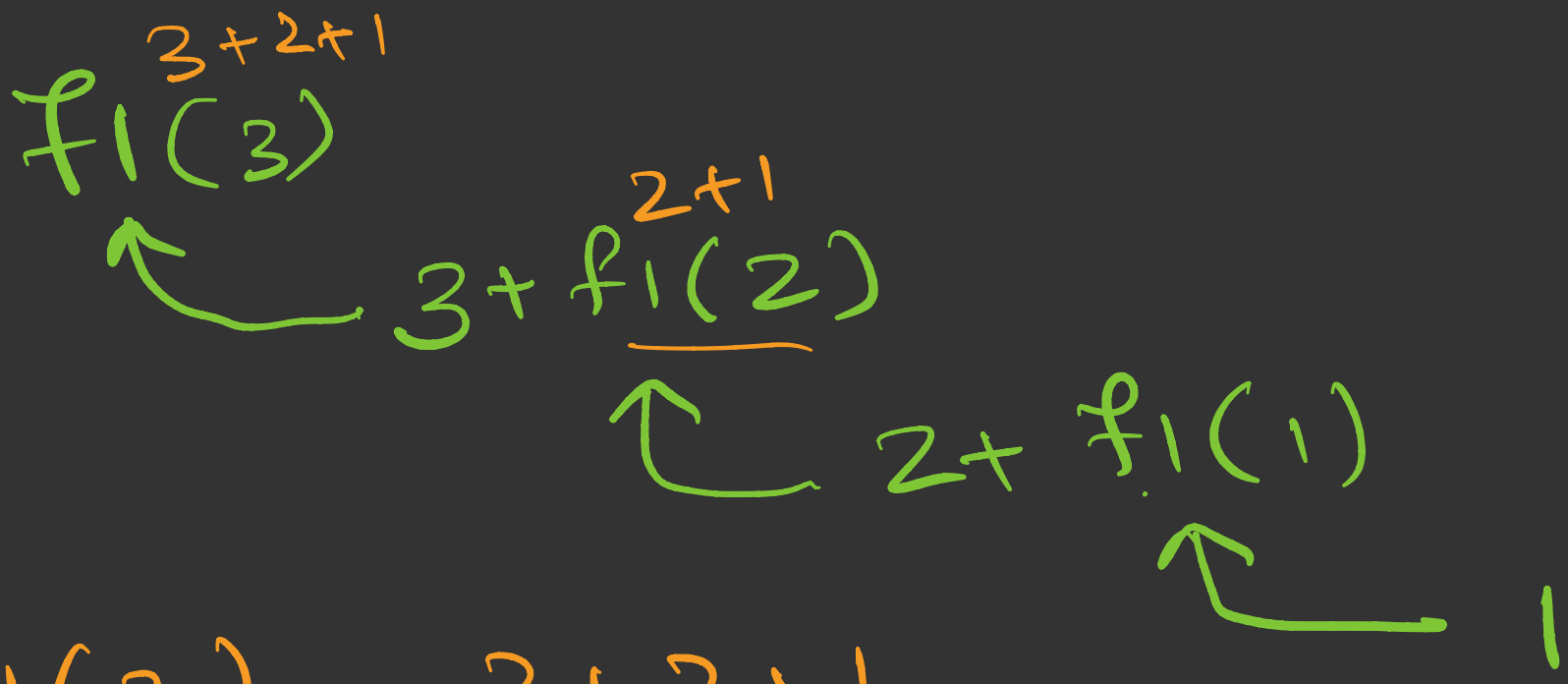
- Solution to some problems are easier to formulate recursively.

Recursion Tree

```
public static void main(String []args) {
    Example obj = new Example();
    int k;
    k = obj.f1(3);
    S.o.p(k);
}

public int f1(int n)
{
    int s;
    if(n==1)
        return(1);
    s = n + f1(n-1);
    return(s);
}
```





$f_1(3) = 3+2+1$

$f_1(5) = 5+4+3+2+1$ $\quad 2+1$

How to approach a Recursive Problem?

Write a recursive function to calculate sum of first n natural numbers.

① sum(n) $1+2+3+4+\cdots+n$

RC ② $n+$ sum(n-1) $1+2+3+\cdots+(n-1)$

BC ③ $n==1$ 1

```
int sum(int n)
{
    if(n==1)
        return(1);
    return n+sum(n-1);
}
```

Write a recursive function to print first n natural numbers.

```
void printN(int n)
{
    if (n>0){
        printN(n-1);
        s.o.p(n);
    }
}
```

① printN(n) 1 2 3 4 ... n

RC ② printN(n-1) 1 2 3 4 ... n-1
s.o.p(n) n

B ③ n == 0