Recursion

Another type of loop

Recursion ⟷ iteration

Factorial

$$x! = \begin{cases} 1 & x=0 \text{ or } x=1 \quad \text{// base case} \\ x \cdot (x-1)! & \text{otherwise} \quad \text{// recursive} \end{cases}$$

$$fact(x) = \begin{cases} 1 & x=0 \text{ or } x=1 \\ x \cdot fact(x-1) & \text{otherwise} \end{cases}$$

```
public static int fact (int num) {
        if (num <=1 ) {
                return 1;
        }
        return num * fact (num-1); //recursive case
}
```

base case

iteratively

```
public static int fact(int num) {
        int product = 1;
        for (int i=num; i>0; i--) {
                product *= i;
        }
        return product;
}
```

Fibonacci sequence

F_n	0	1	1	2	3	5	8	...
n	1	2	3	4	5	6	7	

$$F_4 = 2$$

$$F_n = \begin{cases} 0 & \text{if } n=1 \\ 1 & \text{if } n=2 \end{cases} \quad \text{base cases}$$

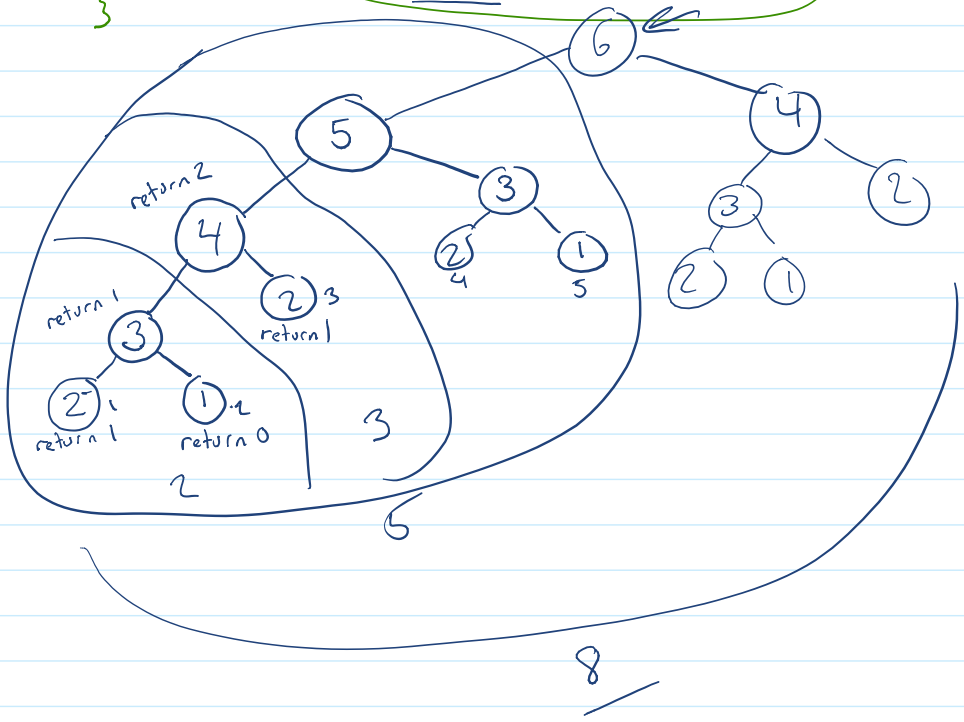
$$F_n = F_{n-1} + F_{n-2} \quad \text{// recursive case}$$

$$Fib(n) = \begin{cases} 0 & \text{if } n=1 \\ 1 & \text{if } n=2 \\ Fib(n-1) + Fib(n-2) & \text{otherwise} \end{cases}$$

```
public static int fib ( int index) {
        if (index <= 1) {
```

```
public static int fib(int index) {
    if (index <= 1) {
        return 0;
    }
    if (index == 2) {
        return 1;
    }
    return fib(index-1) + fib(index-2);
}
```

base cases



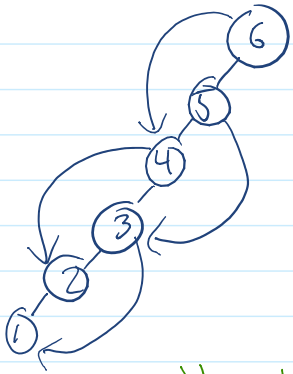
2, 3, 5, 8

$O(Fib(n))$

0, 1, 1, 2, 3, 5, 8, 13,

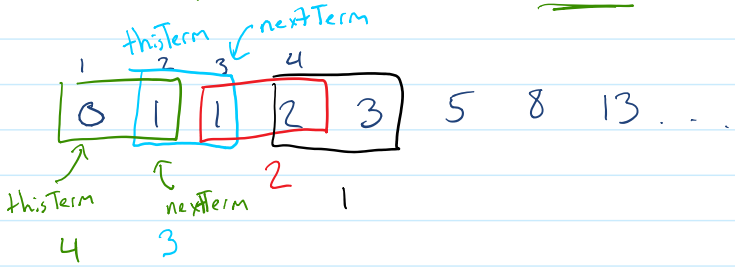
Time $O(n)$

Space



public static int fib(int index) {

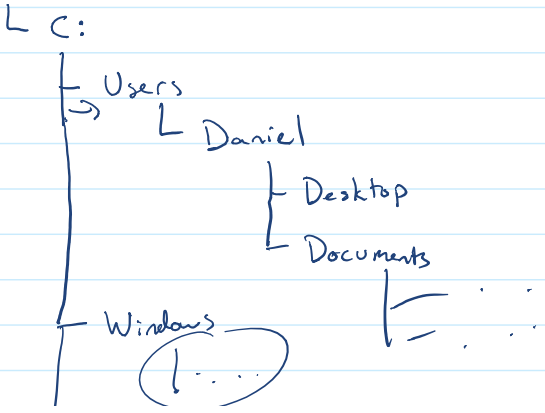
```
public static int fib(int index) {
    return fibHelper(index, 0, 1);
}

public static int fibHelper(int index, int thisTerm, int nextTerm) {
    if (index == 1) {
        return thisTerm
    }
    return fibHelper(index-1, nextTerm, thisTerm + nexTerm);
}
```

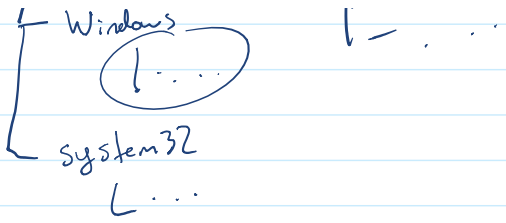



```
public static int fibHelper(int index, int thisTerm, int nextTerm) {
    for (int i=index; i > 0; i--) {
        int temp = thisTerm;
        thisTerm = nextTerm;
        nexTerm = temp + nextTerm;
    }
    return thisTerm;
}
```

First algorithm – $O(Fib(n)) \sim O(2^n)$
second algorithm – $O(n)$
better (exact) – $O(\log n)$ } go home and think about it.
better (inexact) – $O(1)$

Listing out directory contents






```
import java.io.File;
import java.util.Scanner;
public class Lister {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        println("Enter a directory name");
        String name = scan.nextLine();
        File directory = new File(name);
        listDirectory(directory);
    }
    public static void listDirectory(File infile) {
        if (!infile.isDirectory()) {
            println(infile.name());
            return;
        }
        File[] contents = infile.getContents();
        println(infile.name());
        for (File content : contents) {
            listDirectory(content);
        }
    }
```

base case

```
C:
Users
Daniel
Documents
Desktop
:
Windows
:
System32
```

```
public static void listContents(File infile, int depth) {
    for (int i=0; i<depth*4; i++) {
        print(" ");
    }
    :
    listContents(content, depth+1);
```

listContents(content, depth+1); ③

⋮

}

To make iterative → need a stack