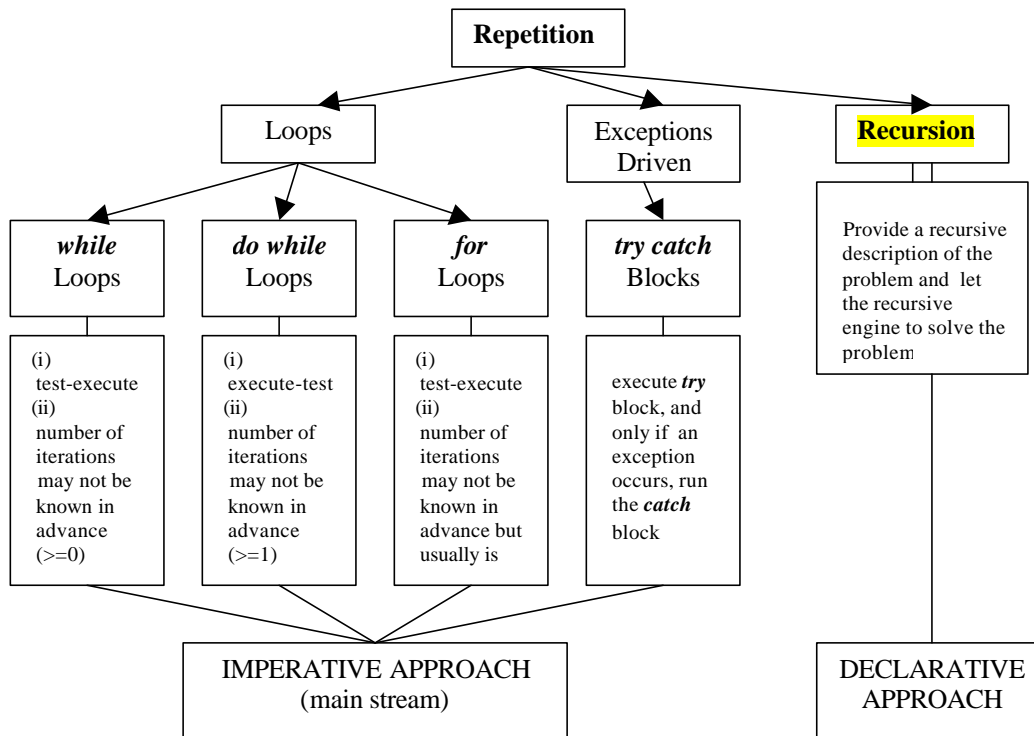


Implementing Repetition : Recursion



Example 1 – Computing factorial function :

```

public int fact(int n) {
    if (n<=0)
        // BASE CASES
        // NON-RECURSIVE BRANCH
        return 1;
    else
        // RECURSIVE CASES
        // CONVERGENCE OBVIOUS:
        // n is here positive
        // Recursive call made with argument (n-1)
        // i.e. fact(3) calls fact(2) in 3*fact(2)
        // fact(2) calls fact(1) in 2*fact(1)
        // fact(1) calls fact(0) in 1*fact(0)
        // fact(0) returns 1 into fact(1) in 1*fact(0) = 1*1
        // fact(1) computes 1*1 and returns 1 into fact(2) in 2*fact(1) = 2*1
        // fact(2) computes 2*1 and returns 2 into fact(3) in 3*fact(2) = 3*2=6
        return (n*fact(n-1));
}
    
```

}

RECURSIVE DEFINITION RULES

(i) BASE CASE(S) RULE :

The algorithm must have ways of resolving some of its problems non-recursively (i.e. the method implementing the algorithm must have at least one branch in which it doesn't call itself).

(ii) CONVERGENCE RULE :

When applied recursively, the algorithm must converge to the BASE CASE(S) (i.e. every recursive call in the method has to bring the method closer to the calls which will result in non-recursive executions).

The above rules are necessary (but not sufficient) conditions for a proper (no infinite loops) recursion.

Example 2: Fibonacci's Sequence

Fibonacci's sequence is a (infinite) sequence of following numbers :

1,	1,	2,	3,	5,	8,	13,	21,	34,	55,	89,	144 ...
0	1	2	3	4	5	6	7	8	9	10	11 ...

Note : Obviously, as a function of n, Fib(n) grows faster than linear and quadratic function.

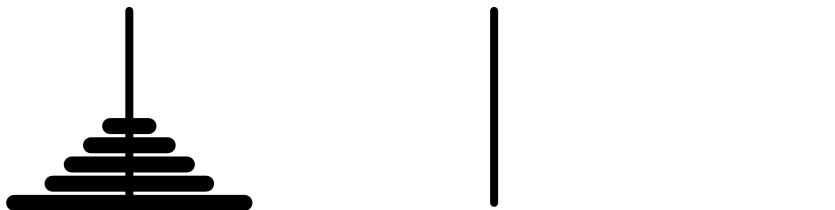
Here is the mathematical definition for Fibonacci's numbers :

- (i) $F_1 = 1,$ if $I \leq 1$
- (ii) $F_I = F_{I-1} + F_{I-2},$ if $I > 1.$

The following is a recursive method that computes Fibonacci's number for its given index :

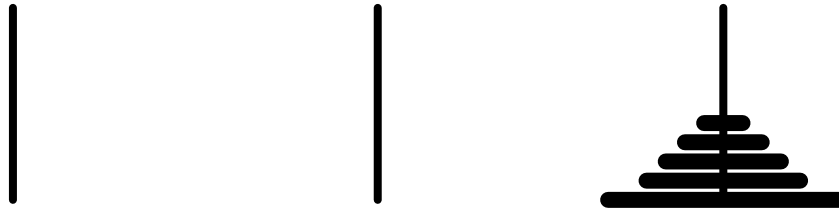
```
public int Fib (int i) {  
    if ((i==0) || (i==1))  
        return 1;  
    else if (i<0) {  
        // Error message  
        return 0;  
    } else  
        return (Fib(i-1) + Fib(i-2));  
}
```

Example 3: Towers of Hanoi

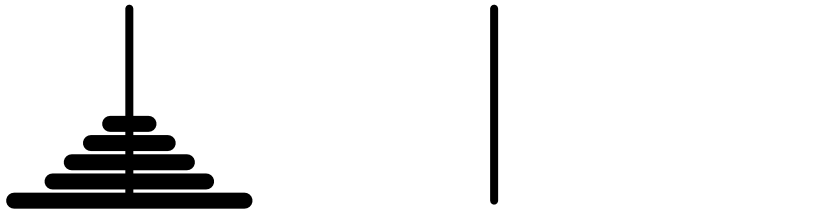


Move n disks from the left pole to the right one

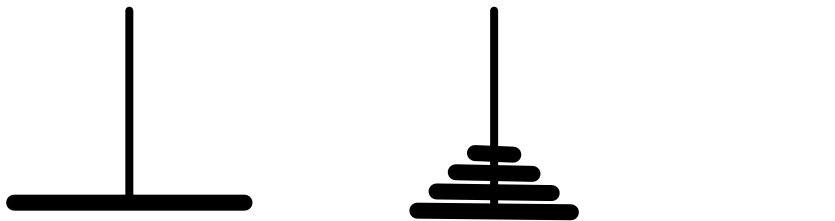
- (i) moving one disk at a time,
- (ii) never having a larger disk on top of a smaller one, and
- (iii) never putting a disk aside.



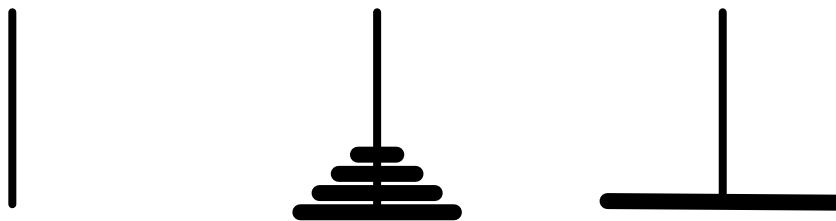
Recursive idea :
Start from :



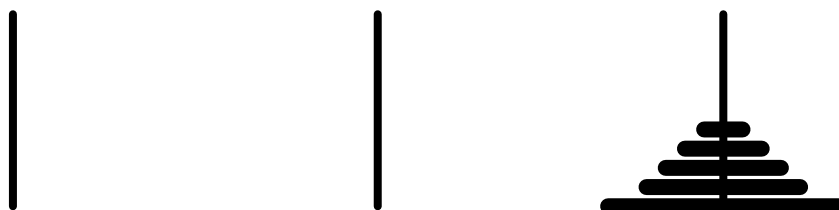
- (I) Do $n-1$ disks recursively into the middle :



- (II) Move the largest disk to its destination :



- (III) Move $n-1$ disks to their destination recursively :



Pseudo-code :

```
public void move (int num, Pole orig, Pole aux, Pole dest) {  
    if (num==1)  
        System.out.println ("Move a disk from " + orig + " to " + dest + " pole.");  
    else {  
        move (num-1, orig, dest, aux);  
        System.out.println ("Move a disk from " + orig + " to " + dest + " pole.");  
        move (num-1, aux, orig, dest);  
    }  
}
```