Algorítimica y Programación

Enero - Mayo 2020

Dr. Iván S. Razo Zapata
(ivan.razo@itam.mx)

ITAM

Contenido

- Recursividad
- Búsqueda binaria

Recursividad o recursión

- Cuando una función se llama/invoca a si misma
- Usualmente se aplica una técnica de divide y vencerás
- Dividir el problema en subproblemas
- Recombinar la solución de dichos subproblemas, i.e. vencer

Factorial

- Factorial de N
- N! = 1 * 2 * 3 * … *(N-1) * N
- N! = N * (N-1) * (N-2) * (N-3) * … * 2 * 1

Factorial

5! = 5 * 4 * 3 * 2 * 1 = 120

Divide

5! = 5 * factorial(4)

4 * factorial(3)

3 * factorial(2)

2 factorial(1)

1 factorial (0)

1

Factorial

- Caso base (trivial)
  - Factorial de cero = **1**
- Formula genereal
  - Factorial N = **n * factorial (n-1)**

Factorial

- Ejemplo

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial (n-1)
```

Caso base a.k.a. kick

Formula general

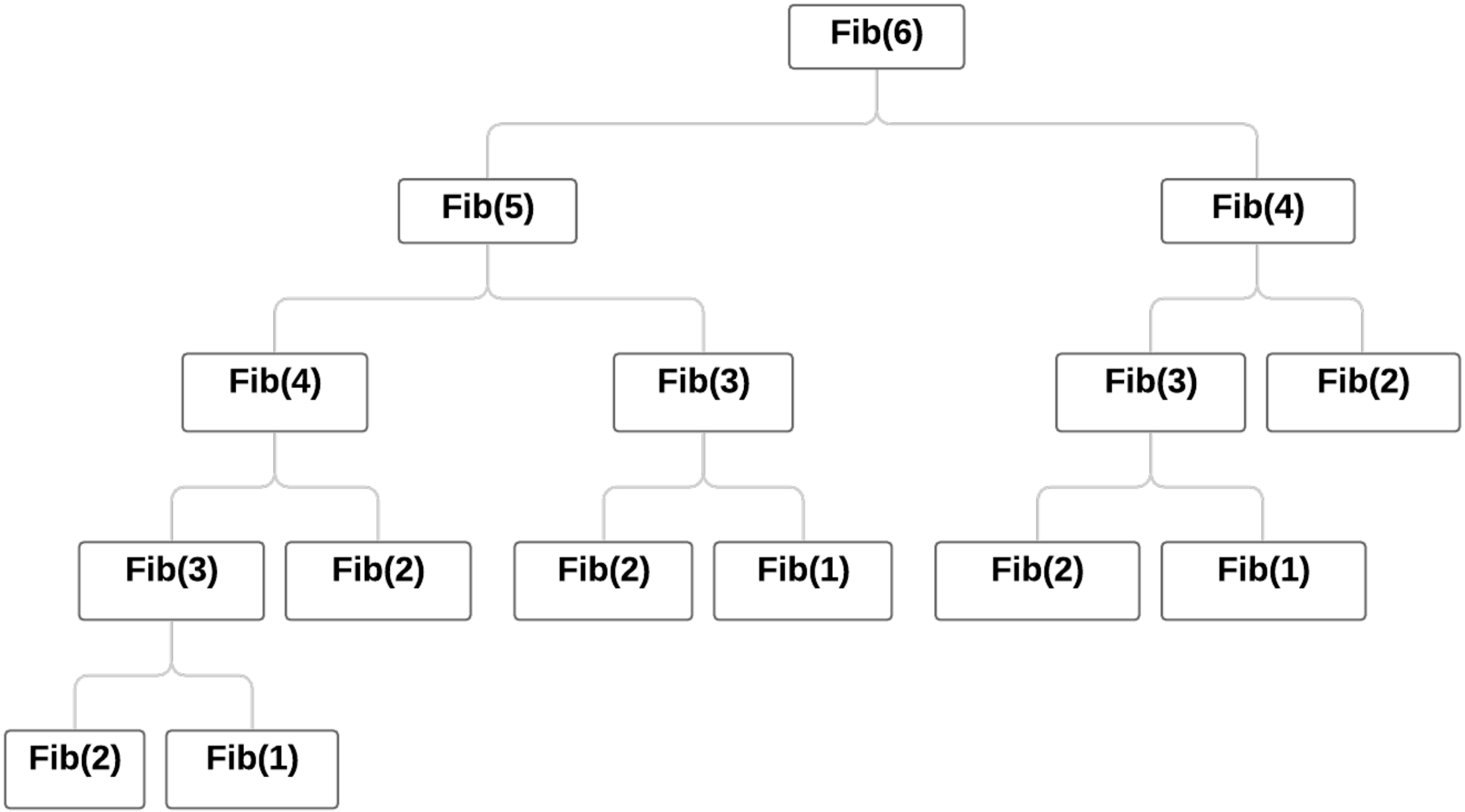
Sin caso base → Limbo

Fibbonacci

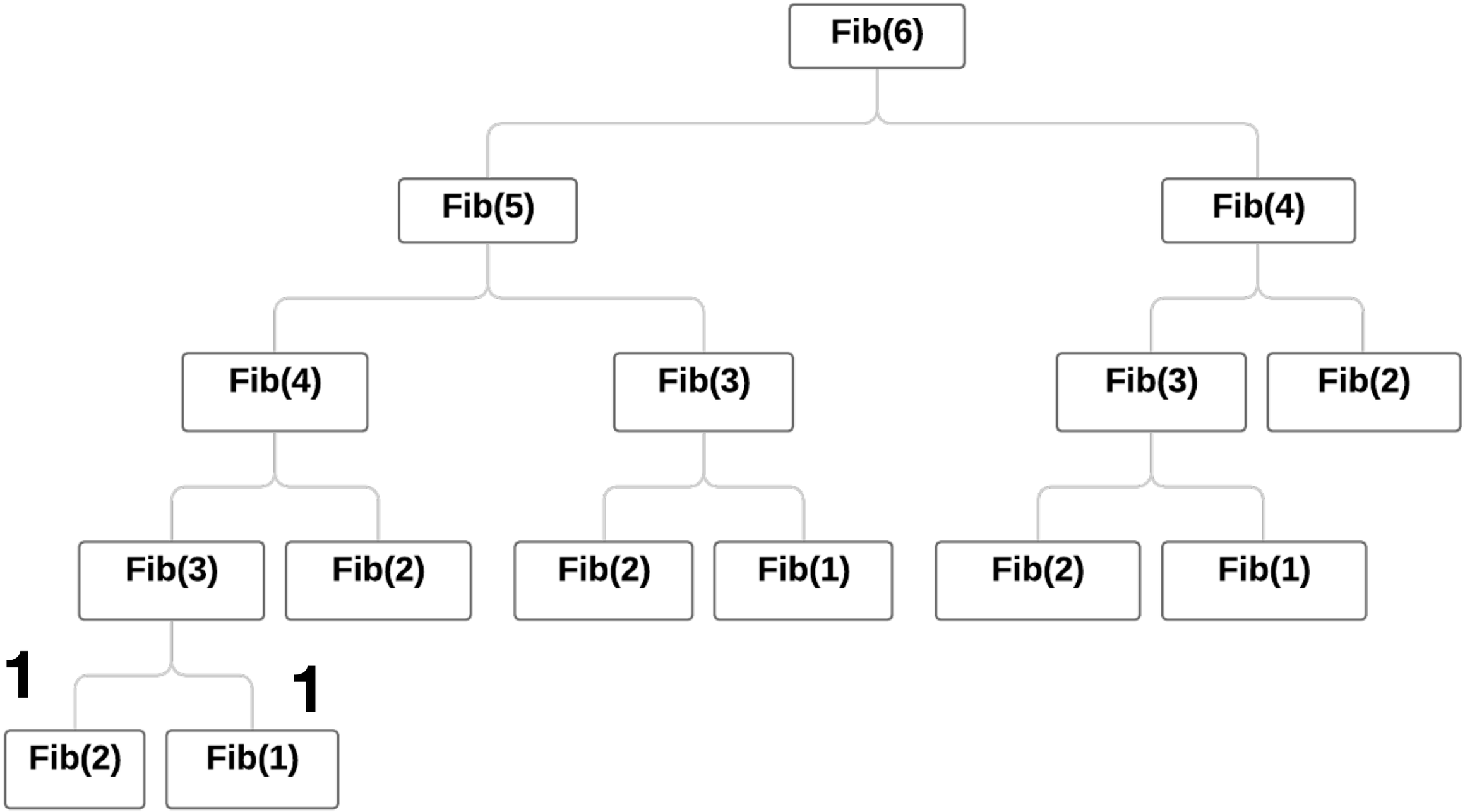
- Fibonacci
- 1, 1, 2, 3, 5, 8, 13, 21, …
- Fib(6) = 8
- Casos especiales 1 y 2
  - Fib(1) = 1
  - Fib(2) = 1
- Fib(3) = 2 <- Fib(2) + Fib(1)
- Fib(4) = 3 <- Fib(3) + Fib(2)

ITAM

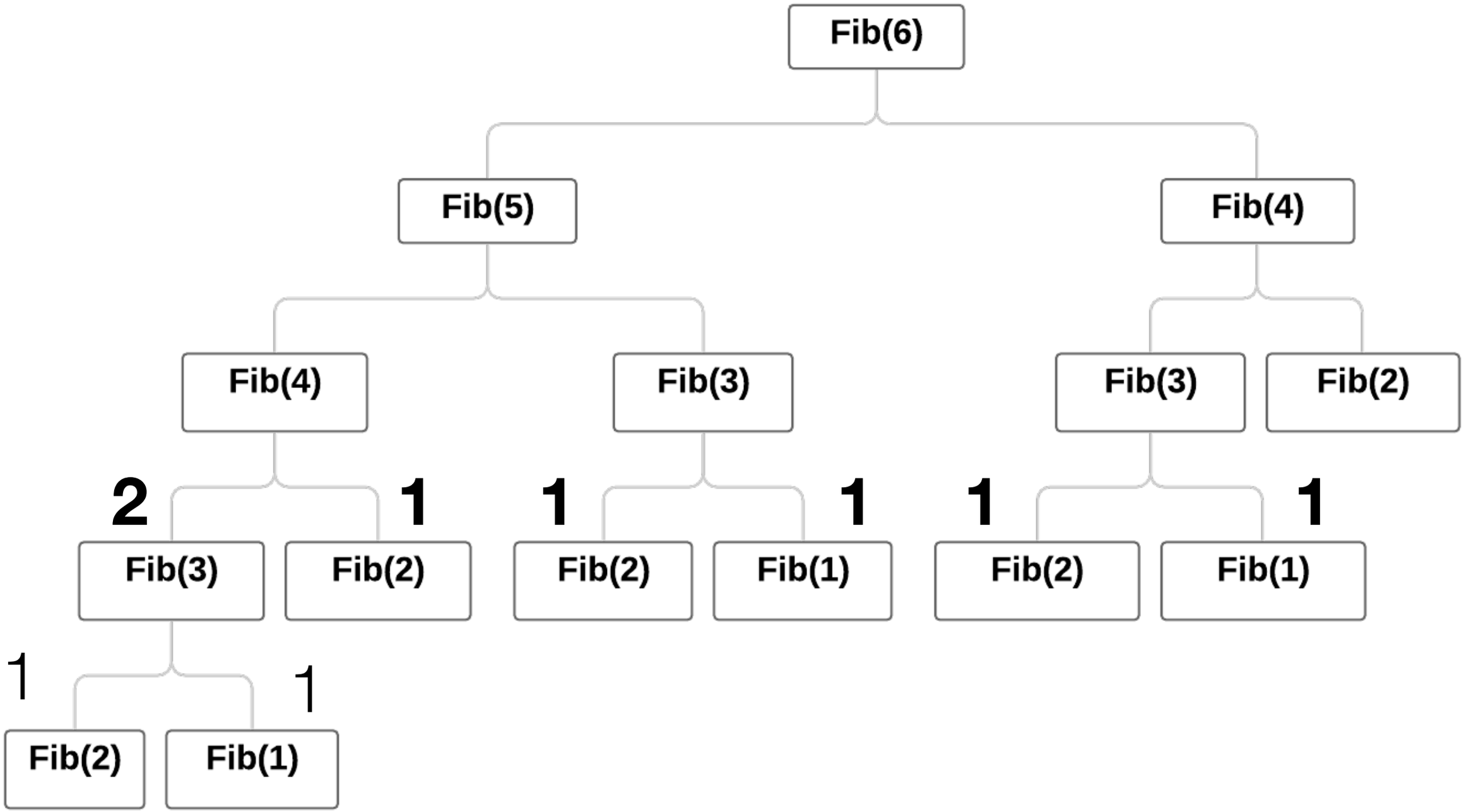
Fibbonacci


Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(2)
Fib(1)

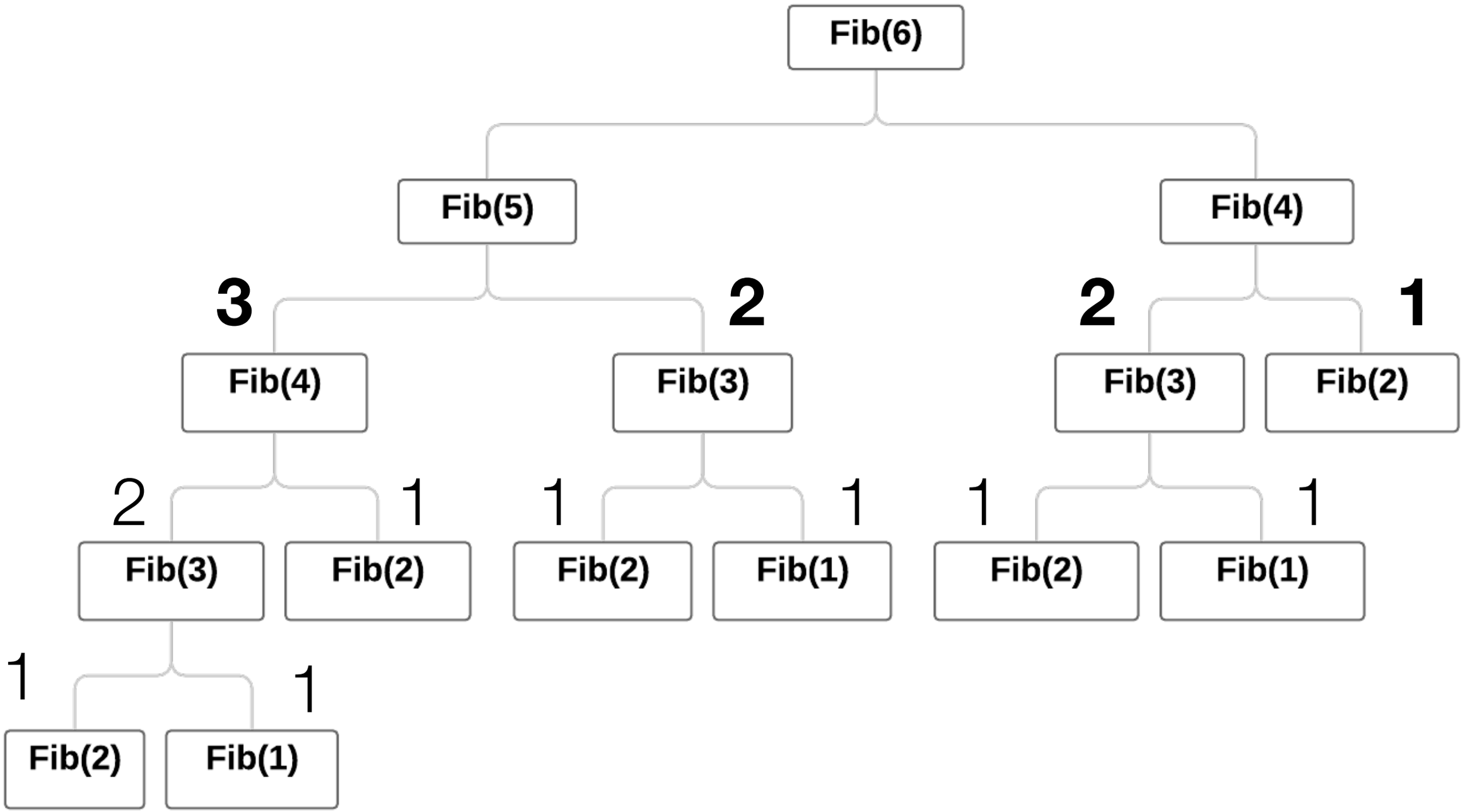
Fibbonacci


Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
1
1
Fib(2)
Fib(1)

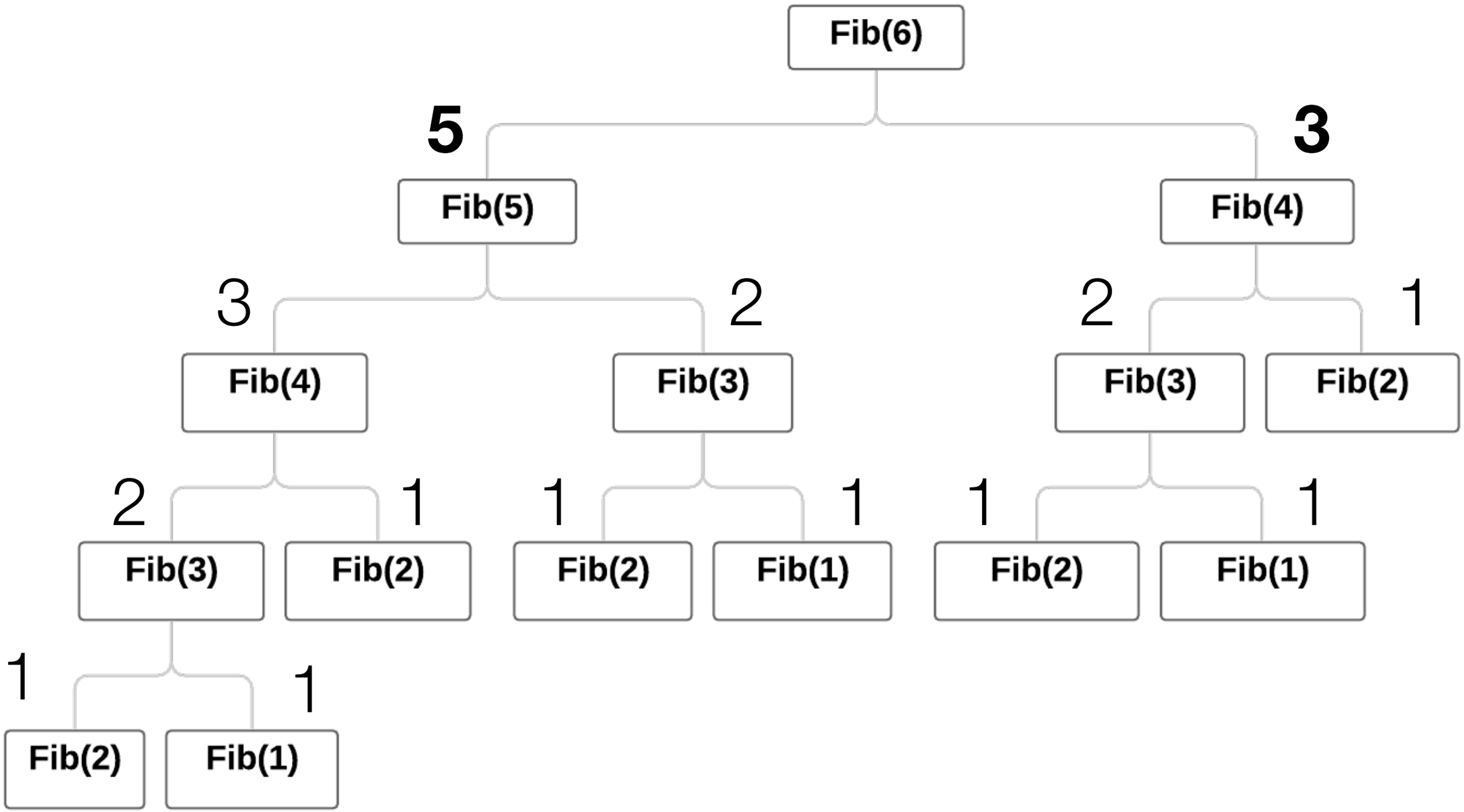
Fibbonacci


Fib(6)
Fib(5)
Fib(4)
2
Fib(3)
Fib(2)
1
1
Fib(2)
Fib(1)
1
Fib(2)
Fib(1)
Fib(4)
Fib(3)
1
Fib(2)
1
Fib(1)
Fib(3)
Fib(2)
1
Fib(2)
1
Fib(1)

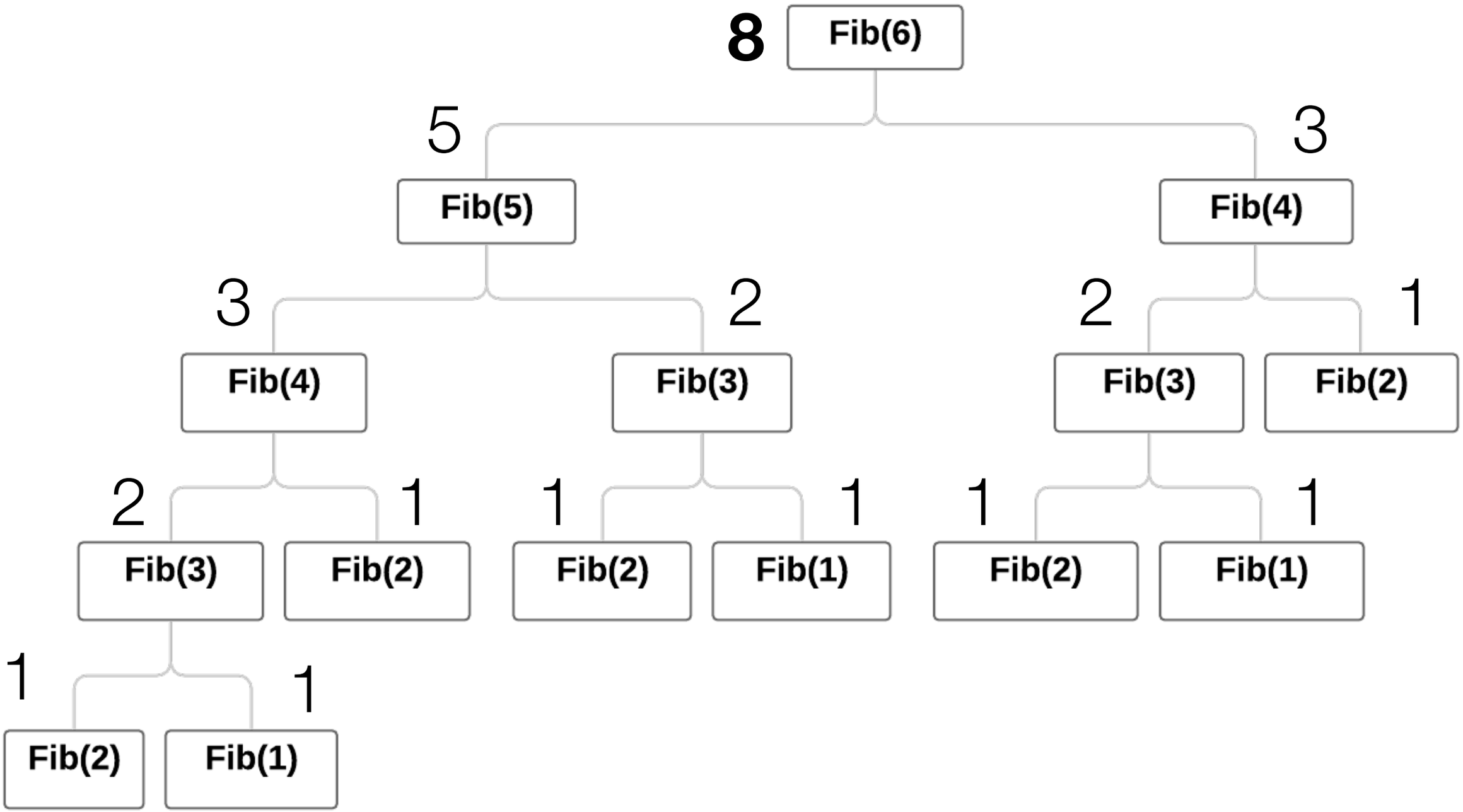
Fibbonacci


Fib(6)
Fib(5)
Fib(4)
3
2
2
1
Fib(4)
Fib(3)
Fib(3)
Fib(2)
2
1
1
1
1
1
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
1
1
Fib(2)
Fib(1)

Fibbonacci


Fib(6)
5
3
Fib(5)
Fib(4)
3
2
2
1
Fib(4)
Fib(3)
Fib(3)
Fib(2)
2
1
1
1
1
1
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
1
1
Fib(2)
Fib(1)

Fibbonacci


8
Fib(6)
5
Fib(5)
3
Fib(4)
3
Fib(4)
2
Fib(3)
2
Fib(3)
1
Fib(2)
2
Fib(3)
1
Fib(2)
1
Fib(2)
1
Fib(2)
1
Fib(1)
1
Fib(2)
1
Fib(1)
1
Fib(2)
1
Fib(1)

Fibbonacci

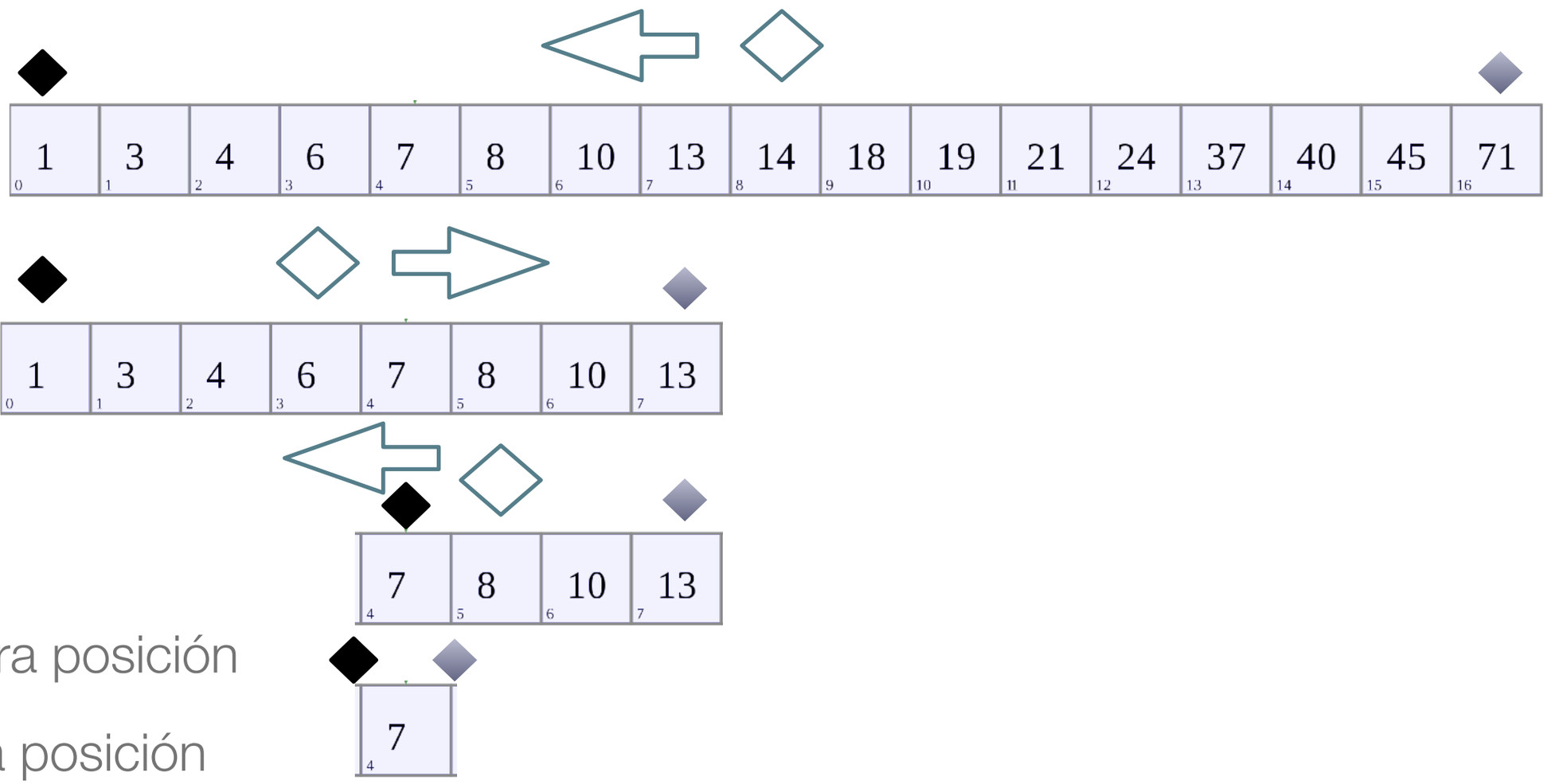
- Caso base (trivial)
  - Fibonnaci 1 y 2 = **1**
- Formula genereal
  - Fib (N) = **Fib(N-1) + Fib(N-2)**

Fibbonacci

```
def fibbonacci(n):
    if (n < 2):
        return n
    else:
        return fibbonacci(n-1) + fibbonacci(n-2)
```

Búsqueda binaria - con **recursividad**

Queremos saber si 7 está en el arreglo



- Primera posición
- Última posición

Búsqueda binaria

```
def busquedaBinaria(arreglo, elemento):
    encontrado = False
    primero = 0
    ultimo = arreglo.size - 1
    while (primero <= ultimo) and (encontrado == False):
        puntoMedio = (primero + ultimo) // 2
        if arreglo[puntoMedio] == elemento:
            encontrado = True
        else:
            if elemento < arreglo[puntoMedio]:
                ultimo = puntoMedio - 1
            else:
                primero = puntoMedio + 1

return encontrado, puntoMedio
```

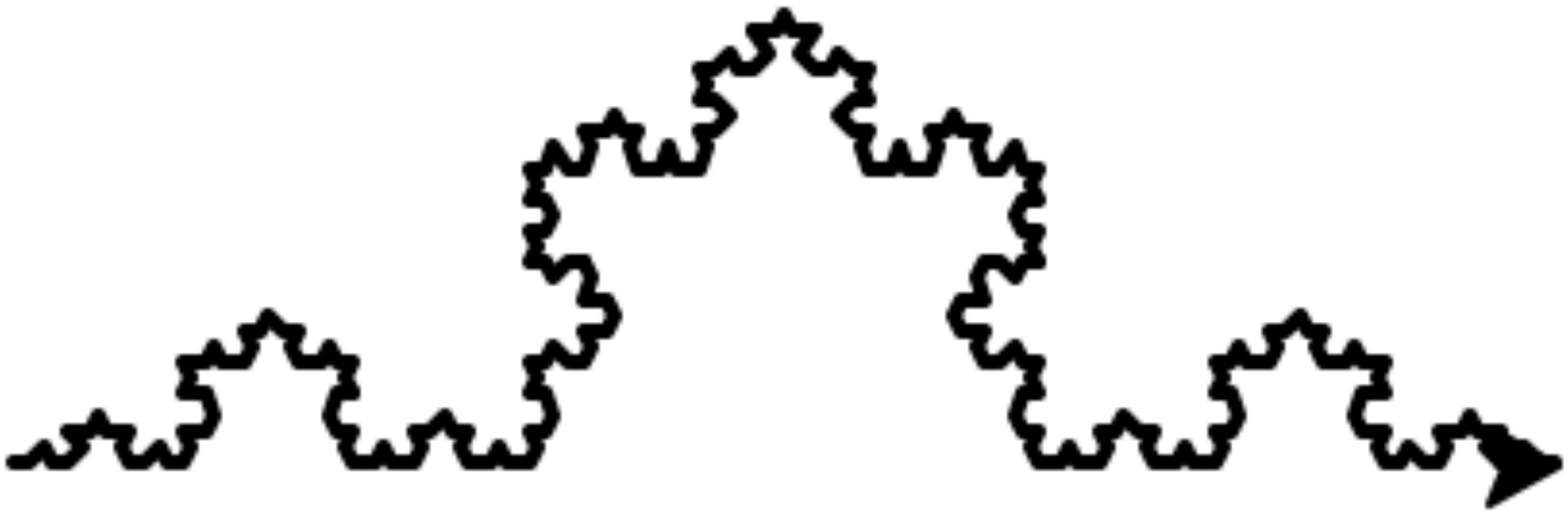
Búsqueda binaria - con **recursividad**

```
def busquedaBinariaRecursiva(arreglo, primero, ultimo, elemento):
    if (ultimo < primero):
        posicion = -1
```

- 10 minutos
- Caso base
- Formula general

Recursividad

Recursividad



Recursividad

```
import turtle

def koch(a, order):
    if order > 0:
        for t in [60, -120, 60, 0]:
            koch(a/3, order-1)
            turtle.left(t)
    else:
        turtle.forward(a)

turtle.speed(10)
turtle.pensize(2)
koch(200, 4)

turtle.done()
try:
    turtle.bye()
except turtle.Terminator:
    pass
```

ITAM