

- Sequência de Fibonacci
↳ 3 versões (1 T.P.C.)

1.

- Cálculo de Potência
↳ 2 versões

- Problema "Maximum - susanag"
↳ 2 versões (1 T.P.C.)

Sequência de Fibonacci

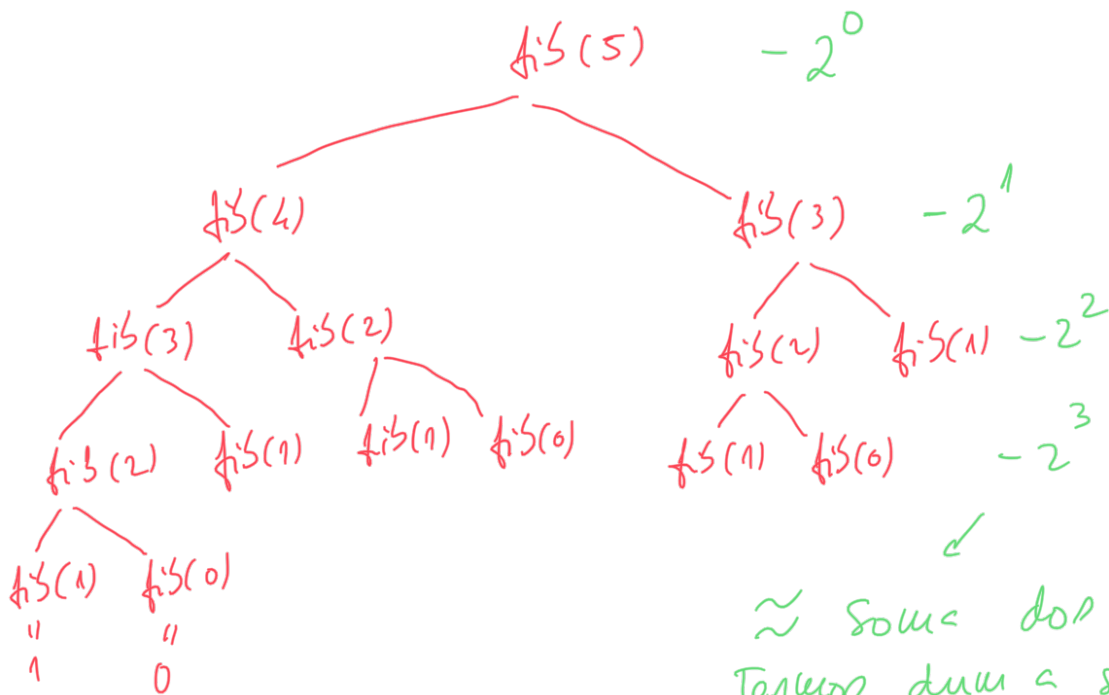
2.

$$\begin{cases} F(0) = 0 \\ F(1) = 1 \\ F(n) = F(n-1) + F(n-2), \quad n > 1 \end{cases}$$

Método 1

```
fun fib_1 (n: Int): Int =  
  if (n <= 1) n  
  else fib_1(n-1) + fib_1(n-2)
```

3.



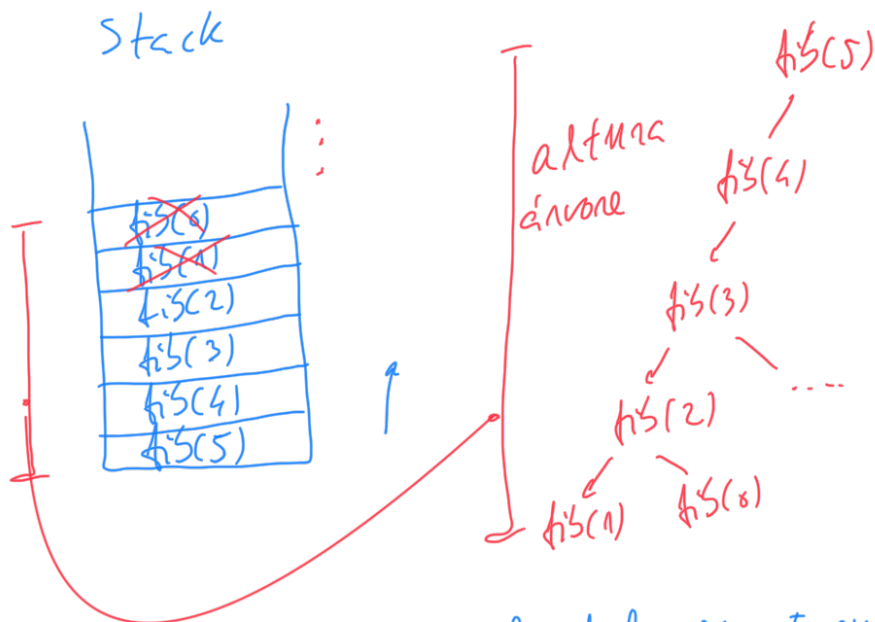
\approx soma dos n primeiros
termos duma sêrie geométrica

$$= \sum_{i=0}^{n-1} 2^i = 2^0 \times \frac{2^n}{2-1} = 2^n$$

Complexidade em termos de
tempo $O(2^n)$

Notação O -grande ou
Big-Oh

4.



Complexidade em termos de
 espaço: $O(n)$ → tudo em conta
 o stack size
 nº de chamadas
 recursivas

Método 2

5.

↳ Programação dinâmica: técnica de memorização
Soluções: utilização de um array para
armazenar os $n-1$ calculados até agora

```
fun fib-2 (n: Int): Int {  
  val f = IntArray (n+2) // +1 para guardar o passo  
  var i: Int                extra no caso do 0.
```

```
  f[0] = 0
```

```
  f[1] = 1
```

```
  i = 2
```

```
  while (i <= n) {
```

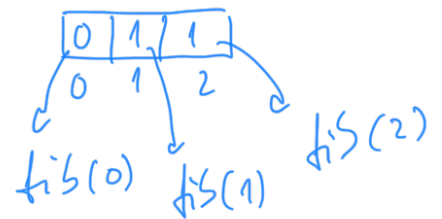
```
    f[i] = f[i-1] + f[i-2]
```

```
    ++i
```

```
  }  
  return f[n]
```

```
}
```

$\text{fib}(2)$



Complexidade em termos de
tempo: $O(n)$

complexidade em termos de
espaço: $O(n)$

T.P.C.

Uma aplicação a
memorização dos
últimos 2 termos

\Rightarrow compl. espacial reduz para $O(1)$

Cálculo de potência

6.

a^n , com $a > 0$ e $n \geq 0$

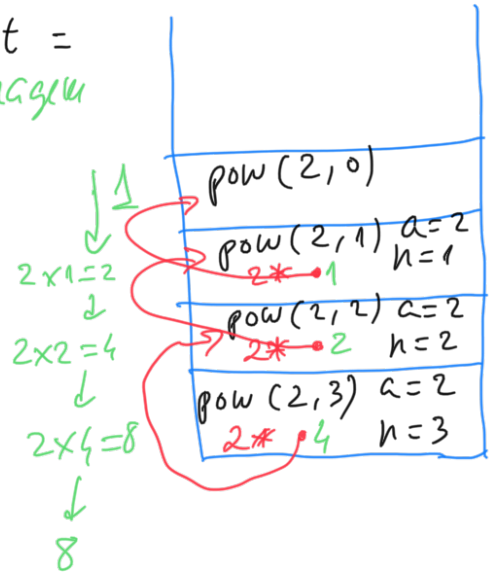
Método 1

```
fun pow-1 (a: Int, n: Int) : Int =  
  if (n == 0) 1 // caso de paragem  
  else a * pow-1(a, n-1)
```

Complexidade Temporal: $O(n)$
" Espaço: $O(n)$

Linear

nº de
funções
expandidas
" stack size



Metodo 2

$$a^n = \begin{cases} 1, & n=0 \\ a^{n/2} * a^{n/2}, & \text{se } n \text{ par} \\ a^{n/2} * a^{n/2} * a, & \text{se } n \text{ impar} \end{cases}$$

```

fun pow-2 (a: Int, n: Int) {
  if (n == 0) return 1
  val z = pow-2(a, n/2)
  return if (n % 2 == 0) z * z else z * z * a
}

```

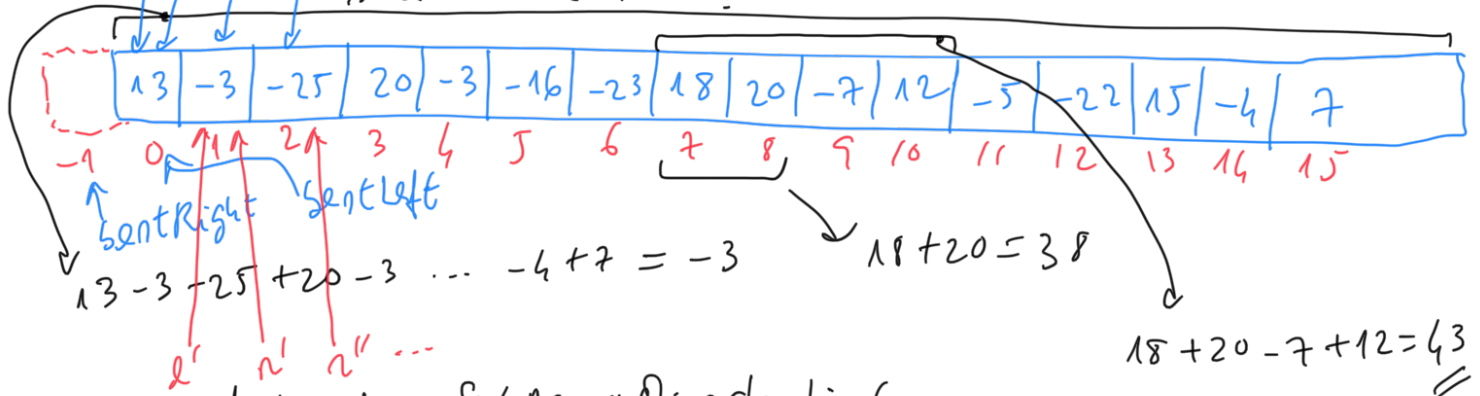
4

Complexidade Temporal: $O(\log_2 N) \rightarrow$ Logarítmico
Espacial: $O(\log_2 N)$

7.

Maximum Subarray problem

some recursive?



fun maxSubArrayRecursive (array: DoubleArray,
left: Int, right: Int):
Triple {

T.P.C. 1 for (...)
for (...)

T.P.C. 2 \rightarrow Linear \rightarrow método de Kadane