

Fibonacci



Fonte: <http://www.geek.com/geek-cetera/>

PF 2.3 S 5.2

<http://www.ime.usp.br/~pf/algoritmos/aulas/recu.html>

Números de Fibonacci

$$F_0 = 0 \quad F_1 = 1 \quad F_n = F_{n-1} + F_{n-2}$$

n	0	1	2	3	4	5	6	7	8	9
F_n	0	1	1	2	3	5	8	13	21	34

Algoritmo recursivo para F_n :

```
def fibonacciR(n)
    if n == 0: return 0
    if n == 1: return 1
    return fibonacciR(n-1) +
           fibonacciR(n-2)
```

fibonacciR(4)

```
fibonacciR(4)
  fibonacciR(3)
    fibonacciR(2)
      fibonacciR(1)
        fibonacciR(0)
      fibonacciR(1)
    fibonacciR(2)
      fibonacciR(1)
        fibonacciR(0)
  fibonacci(4) = 3.
```

Fibonacci iterativo

```
def fibonacciI(n)
    if n == 0: return 0
    if n == 1: return 1
    anterior = 0
    atual = 1
    for i in range(1,n,1):
        proximo = atual + anterior
        anterior = atual
        atual = proximo
    return atual
```

Qual é mais eficiente?

```
meu_prompt> time ./fibonacciI.py 10
fibonacci(10)=55
real                0m0.028s
user                0m0.024s
sys                 0m0.000s
```

```
meu_prompt> time ./fibonacciR.py 10
fibonacci(10)=55
real                0m0.028s
user                0m0.024s
sys                 0m0.000s
```

Qual é mais eficiente?

```
meu_prompt> time ./fibonacciI.py 20
fibonacci(20) = 6765
real                0m0.028s
user                0m0.024s
sys                 0m0.000s
```

```
meu_prompt> time ./fibonacciR.py 20
fibonacci(20) = 6765
real                0m0.030s
user                0m0.024s
sys                 0m0.000s
```

Qual é mais eficiente?

```
meu_prompt> time ./fibonacciI.py 30
fibonacci(30) = 832040
real                0m0.028s
user                0m0.024s
sys                 0m0.000s
```

```
meu_prompt> time ./fibonacciR.py 30
fibonacci(30) = 832040
real                0m0.584s
user                0m0.580s
sys                 0m0.000s
```

Qual é mais eficiente?

```
meu_prompt> time ./fibonacciI.py 40
fibonacci(40) = 102334155
real                0m0.026s
user                0m0.024s
sys                 0m0.000s
```

```
meu_prompt> time ./fibonacciR.py 40
fibonacci(40) = 102334155
real                1m8.530s
user                1m8.508s
sys                 0m0.004s
```


Qual é mais eficiente?

```
meu_prompt> time ./fibonacciI.py 45
fibonacci(45) = 1134903170
real                0m0.032s
user                0m0.028s
sys                 0m0.000s
```

```
meu_prompt> time ./fibonacciR.py 45
fibonacci(45) = 1134903170
real                12m47.577s
user                12m47.248s
sys                 0m0.080s
```

fibonacciR(5)

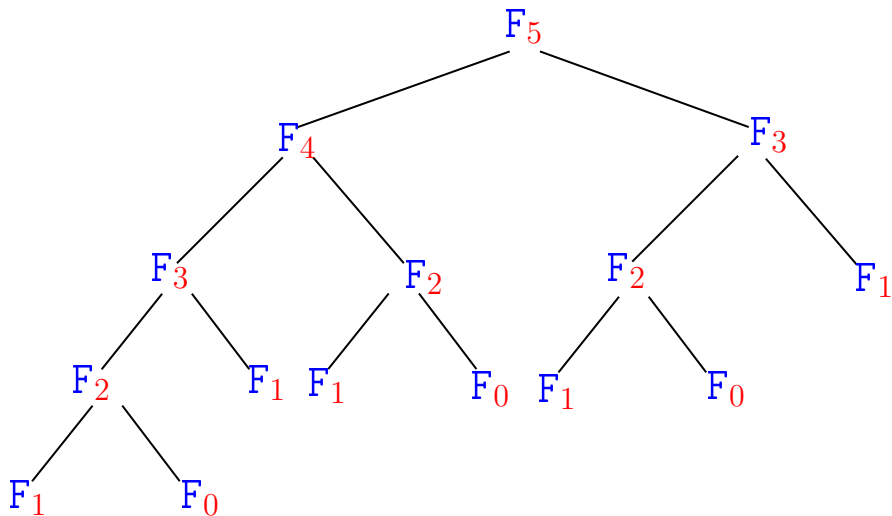
fibonacciR resolve subproblemas muitas vezes.

fibonacciR(5)	fibonacciR(1)
fibonacciR(4)	fibonacciR(0)
fibonacciR(3)	fibonacciR(3)
fibonacciR(2)	fibonacciR(2)
fibonacciR(1)	fibonacciR(1)
fibonacciR(0)	fibonacciR(0)
fibonacciR(1)	fibonacciR(1)
fibonacciR(2)	fibonacci(5) = 5.

Árvore da recursão

Consumo de tempo é **exponencial**.

`fibonacciR` resolve subproblemas muitas vezes.



Consumo de tempo

$T(n)$:= número de somas feitas por `fibonacciR(n)`

```
def fibonacciR(n)
1     if n == 0: return 0
2     if n == 1: return 1
3     return fibonacciR(n-1) +
4           fibonacciR(n-2)
```

Consumo de tempo

linha	número de somas
-------	-----------------

$$1 = 0$$

$$2 = 0$$

$$3 = T(n - 1)$$

$$4 = T(n - 2) + 1$$

$$T(n) = T(n - 1) + T(n - 2) + 1$$

Recorrência

$$T(0) = 0$$

$$T(1) = 0$$

$$T(n) = T(n - 1) + T(n - 2) + 1 \quad \text{para } n = 2, 3, \dots$$

Uma estimativa para $T(n)$?

Recorrência

$$T(0) = 0$$

$$T(1) = 0$$

$$T(n) = T(n-1) + T(n-2) + 1 \quad \text{para } n = 2, 3, \dots$$

Uma estimativa para $T(n)$?

Solução: $T(n) > (3/2)^n$ para $n \geq 6$.

n	0	1	2	3	4	5	6	7	8	9
T_n	0	0	1	2	4	7	12	20	33	54
$(3/2)^n$	1	1.5	2.25	3.38	5.06	7.59	11.39	17.09	25.63	38.44

Recorrência

Prova: $T(6) = 12 > 11.40 > (3/2)^6$ e

$T(7) = 20 > 18 > (3/2)^7$.

Se $n \geq 8$, então

$$\begin{aligned}T(n) &= T(n-1) + T(n-2) + 1 \\&\stackrel{hi}{>} (3/2)^{n-1} + (3/2)^{n-2} + 1 \\&= (3/2 + 1)(3/2)^{n-2} + 1 \\&> (5/2)(3/2)^{n-2} \\&> (9/4)(3/2)^{n-2} \\&= (3/2)^2(3/2)^{n-2} \\&= (3/2)^n.\end{aligned}$$

Logo, $T(n) \geq (3/2)^n$.

Consumo de tempo é **exponencial**.

Conclusão

O consumo de tempo é da função `fibonacciI(n)` é proporcional a `n`.

O consumo de tempo da função `fibonacciR` é **exponencial**.

Exercícios

Prove que

$$T(n) = \frac{\phi^{n+1} - \hat{\phi}^{n+1}}{\sqrt{5}} - 1 \quad \text{para } n = 0, 1, 2, \dots$$

onde

$$\phi = \frac{1 + \sqrt{5}}{2} \approx 1,61803 \quad \text{e} \quad \hat{\phi} = \frac{1 - \sqrt{5}}{2} \approx -0,61803.$$

Prove que $1 + \phi = \phi^2$.

Prove que $1 + \hat{\phi} = \hat{\phi}^2$.