

Recursão versus Não Recursão

Mário Leite

...

O termo “recursão” é empregado até na linguística para definir um idioma moderno, embora a aplicação prática, neste caso, seja bastante complexa quando aplicamos os princípios da “Máquina de Turing”.

Por outro lado, na Ciência da Computação a recursividade é uma técnica muito empregada nos programas como um recurso para economizar linhas de código. De uma maneira mais simples e objetiva pode-se dizer que “a recursividade é um processo no qual uma função se auto executa várias vezes”. Deste modo, na prática, a função chama a si mesma como se estivesse num *loop*. O esquema mostrado na **figura 1** ilustra o mecanismo de chamadas consecutivas da função *f*, executando ela mesma três vezes a partir da chamada inicial por outra rotina.

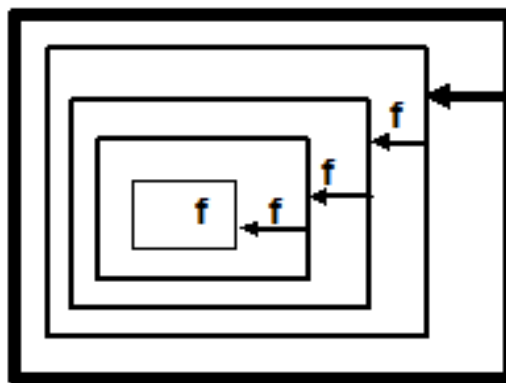


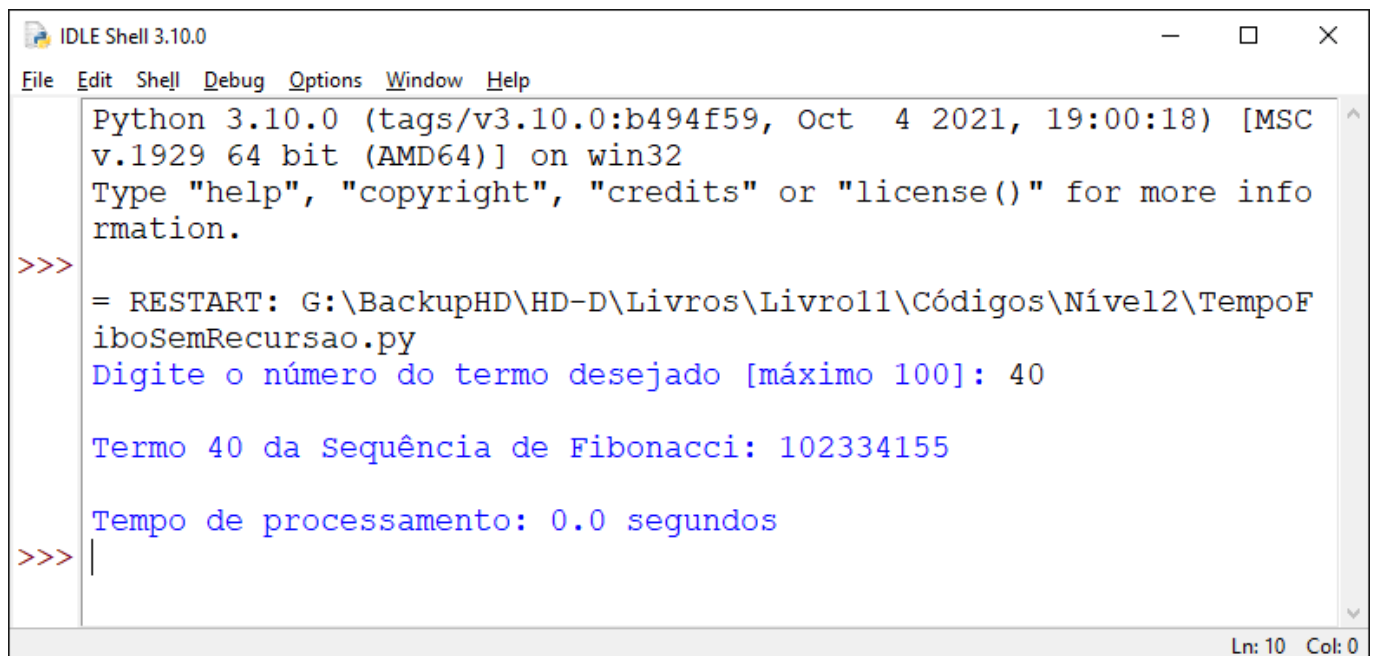
Figura 1 - Função Recursiva

Fonte: Livro “Curso Básico de Programação: Teoria e Prática”
Ed. Ciência Moderna/Amazon - 2017 - Mário Leite

Embora na academia alguns professores insistam em elogiar os “benefícios” desse tipo de função, a verdade é que embora uma função recursiva diminua o tamanho do código-fonte elas não são eficientes; pelo contrário: elas diminuem em muito a eficiência do programa quando comparadas com as funções não recursivas. E pode até ocorrer situações em que o programa nem termina normalmente, pois devido às inúmeras recorrências no código, pode chegar a um ponto em que o processamento é travado. Este é o caso do exemplo mostrado pelo programa abaixo, cujo objetivo é medir o tempo gasto para calcular e mostrar o valor do *n-ésimo* termo da “Sequência de Fibonacci. Dois códigos (em Python) são mostrados aqui: “TempoFiboSemRecursao” e “TempoFiboComRecursao”. Os resultados são apresentados nas **figuras 2a/2b** e **3a/3b** como exemplos que calculam os termos **40** e **4000**: sem recursão e com recursão, respectivamente.

Como pode ser notado nas **figuras 2a/2b**, o tempo de processamento com função “NÃO RECURSIVA” é infinitamente menor que o tempo gasto função “RECURSIVA” para resolver o mesmo problema. E para confirmar a fragilidade das funções recursivas, observe as **figuras 3a/3b** onde é solicitado calcular o **4000º** termo dessa sequência usando os dois tipos de função. A **figura 3b** mostra um *crash* no processamento devido às inúmeras chamadas recursivas da função.

Então, mesmo sendo “elegante” e “moderno” (como se apresenta no clássico exemplo do **Fatorial**) pode ser muito perigoso usar funções recursivas quando se trata de processamentos mais complexos e que demandam muitas chamadas à própria função.



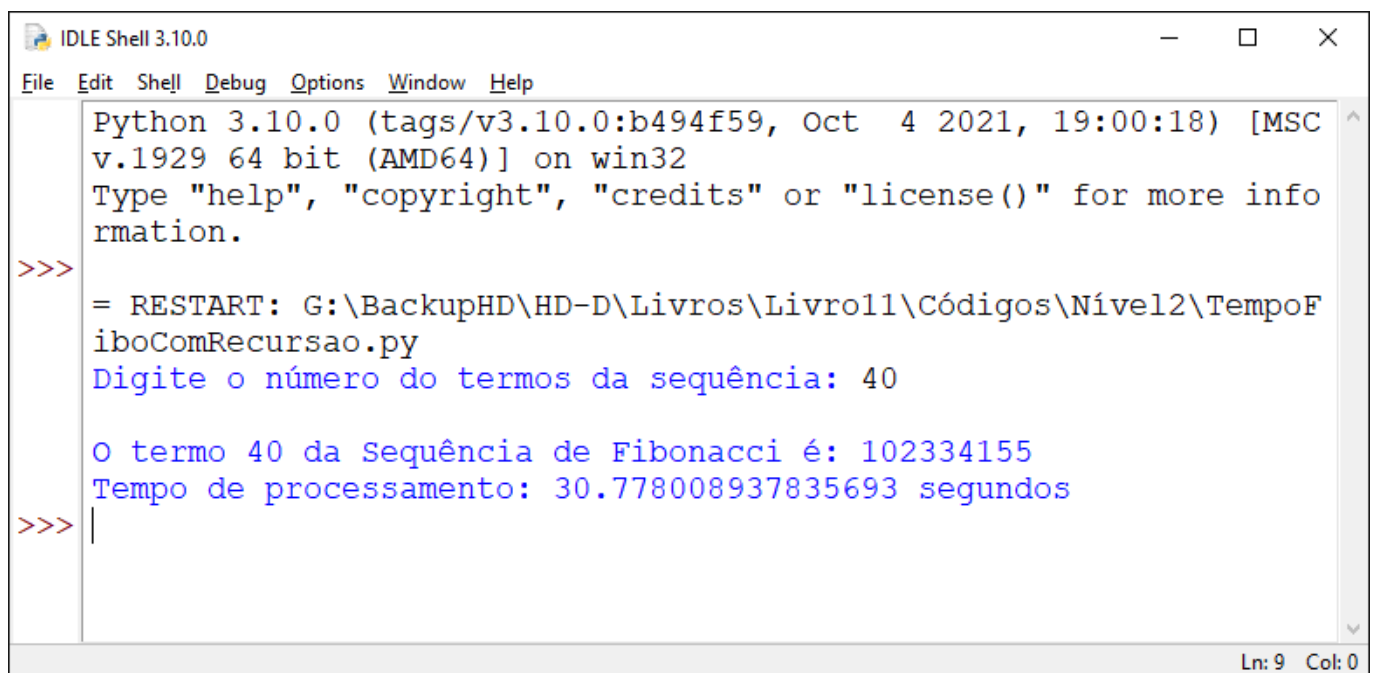
```
Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: G:\BackupHD\HD-D\Livros\Livro11\Códigos\Nível2\TempoFiboSemRecurcao.py
Digite o número do termo desejado [máximo 100]: 40

Termo 40 da Sequência de Fibonacci: 102334155

Tempo de processamento: 0.0 segundos
>>> |
```

Ln: 10 Col: 0

Figura 2a - Calculando o 40º termo da Sequência de Fibonacci “Sem Recursão”



```
Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: G:\BackupHD\HD-D\Livros\Livro11\Códigos\Nível2\TempoFiboComRecurcao.py
Digite o número do termos da sequência: 40

O termo 40 da Sequência de Fibonacci é: 102334155
Tempo de processamento: 30.778008937835693 segundos
>>> |
```

Ln: 9 Col: 0

Figura 2b - Calculando o 40º termo da Sequência de Fibonacci “Com Recursão”

```
IDLE Shell 3.10.0
File Edit Shell Debug Options Window Help
Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: G:\BackupHD\HD-D\Livros\Livro11\Códigos\Nível2\TempoFiboSemRecursao.py
Digite o número do termo desejado [máximo 100]: 4000

Termo 4000 da Sequência de Fibonacci: 39909473435004422792081248
0949609126007925709828202578526288763265230518186413734335491367
6942413244229396930653752011827387962802544323537036225095543565
4171592897966790864814458223141914272590897468472180370639695334
4496626503128747355609262982462494041683090642143510444590777494
2523677766080922609515185205278135297544948256583836980918377178
7439660825140502824343131911711296392457138867486593923544177893
7354286022382122491565646314525076586034000120036853229848384889
6235149263257775535445290404924129456566251941723502004987387387
8602731379207893212335423484873469083054556329894167262818692599
8152095825172779650590682355431394593750282768512214358159573742
7314382442290941639537517873926854436812689424097913532217608037
4780998010657710775625856041594078495411724236560242597759185543
824798332467919613598667003025993715274875

Tempo de processamento: 0.0 segundos
>>>
```

Figura 3a - Calculando o 4000º termo da Sequência de Fibonacci “Sem Recursão”

```
IDLE Shell 3.10.0
File Edit Shell Debug Options Window Help
Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: G:\BackupHD\HD-D\Livros\Livro11\Códigos\Nível2\TempoFiboComRecursao.py
Digite o número de termos da série [máximo 100]: 4000
Traceback (most recent call last):
  File "G:\BackupHD\HD-D\Livros\Livro11\Códigos\Nível2\TempoFiboComRecursao.py",
    line 24, in <module>
    F = fibonacci_recursivo(n) #Chama a função recursiva para calcular o termo
    desejado
  File "G:\BackupHD\HD-D\Livros\Livro11\Códigos\Nível2\TempoFiboComRecursao.py",
    line 15, in fibonacci_recursivo
    return fibonacci_recursivo(n - 1) + fibonacci_recursivo(n - 2)
  File "G:\BackupHD\HD-D\Livros\Livro11\Códigos\Nível2\TempoFiboComRecursao.py",
    line 15, in fibonacci_recursivo
    return fibonacci_recursivo(n - 1) + fibonacci_recursivo(n - 2)
  File "G:\BackupHD\HD-D\Livros\Livro11\Códigos\Nível2\TempoFiboComRecursao.py",
    line 15, in fibonacci_recursivo
    return fibonacci_recursivo(n - 1) + fibonacci_recursivo(n - 2)
  [Previous line repeated 1021 more times]
  File "G:\BackupHD\HD-D\Livros\Livro11\Códigos\Nível2\TempoFiboComRecursao.py",
    line 12, in fibonacci_recursivo
    if n <= 1:
RecursionError: maximum recursion depth exceeded in comparison
>>>
```

Figura 3b - Tentando calcular o 4000º termo da Sequência de Fibonacci “Com Recursão”

```

'''
TempoFiboSemRecurcao.py
-----
Calcula e mostra o n-ésimo termo da "Sequência de Fibonacci" sem usar recursão.
-----
Data: 01/10/2023
Autor: Mário Leite
-----
'''
import time

#Função para calcular o termo desejado da 'Sequência de Fibonacci' sem recursão
def FiboNaoRecurativa(n):
    if(n < 0):
        return "O número do termo deve ser não negativo"
    elif(n == 0):
        return 0
    elif(n == 1):
        return 1
    else:
        a, b = 0, 1
        for _ in range(2, n + 1):
            c = a + b
            a, b = b, c
        return b

#=====
#Programa principal
if __name__ == "__main__":
    n = int(input("Digite o número do termo desejado [máximo 100]: "))
    print()
    inicio = time.time() #Inicia o cronômetro

    resultado = FiboNaoRecurativa (n) #chama a função passando o parâmetro

    tempo = time.time() - inicio #Para o cronômetro e calcula o tempo

    if type(resultado) == int:
        print("Termo", n, "da Sequência de Fibonacci:", resultado)
    else:
        print(resultado)

    print("Tempo de processamento:", tempo, "segundos")
#Fim do programa "TempoFiboSemRecurcao" -----

```

```

'''
TempoFiboComRecursao.py
-----
Calcula e mostra o n-ésimo termo da "Sequência de Fibonacci" usando recursão.
-----
Data: 01/10/2023
Autor: Mário Leite
-----
'''
import time

#Função para calcular o termo desejado da 'Sequência de Fibonacci' com recursão
def FiboRecursiva(n):
    if(n <= 1):
        return n
    else:
        return FiboRecursiva(n-1)+ FiboRecursiva (n-2)  #chama recursiva
=====
#Programa principal
#Início do programa
if __name__ == "__main__":
    n = int(input("Digite o número do termos da sequência [máximo 100]: "))

    #Inicia o cronômetro
    inicio = time.time()

    F = FiboRecursiva (n) #Chama função recursiva para calcular o termo
    #Para o cronômetro e calcula o tempo
    tempo = time.time() - inicio

    print("\nO termo", n, "da Sequência de Fibonacci é:", F)
    print("Tempo de processamento:", tempo, "segundos")
#Fim do programa "TempoFiboComRecursao" -----

```