ATIVIDADE 007: Recursão em cauda

Paradigmas de Linguagens de Programação

Fabrizio Honda
1715280006
Lic. em Computação

UEA
UNIVERSIDADE DO ESTADO DO AMAZONAS

DESCRIÇÃO DA ATIVIDADE

Crie em Haskell:

- uma função fibonacci recursiva
- uma função fibonacci recursiva com cauda

Usando os comandos PS e TIME meça o tempo de execução e o uso de memória rodando cada programa para os valores:

- 10
- 20
- 30
- 100
- 1000

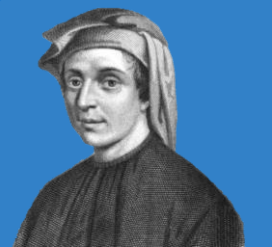
IMPLEMENTAÇÃO 1

fibonacci-normal.hs

```haskell
module Fib(
  fib
  ) where

fib :: Integer -> Integer
fib 0 = 0
fib 1 = 1
fib n = fib(n - 2) + fib(n - 1)
```

Fibonacci recursiva (sem cauda)



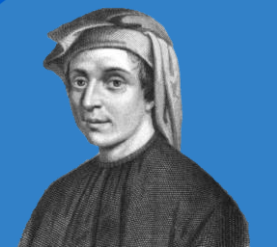
IMPLEMENTAÇÃO 2

fibonacci-cauda.hs ×

```haskell
module Fib(
    fibcauda
    ) where

fibcauda :: Integer -> Integer
fibcauda n = fibaux 0 1 n
        where
          fibaux a b n | n <= 1 = b
                       | otherwise = fibaux b (a+b) (n-1)
```

Fibonacci recursiva (com cauda)



TEMPO E MEMÓRIA

```
*Fib> fib 10
55
(0.02 secs, 104,168 bytes)
*Fib> fib 20
6765
(0.05 secs, 4,227,296 bytes)
*Fib> fib 30
832040
(8.83 secs, 511,353,848 bytes)
*Fib> fib 100
```

Fibonacci recursiva (sem cauda)

```
*Fib> fibcauda 10
55
(0.02 secs, 74,176 bytes)
*Fib> fibcauda 20
6765
(0.02 secs, 78,040 bytes)
*Fib> fibcauda 30
832040
(0.01 secs, 83,352 bytes)
*Fib> fibcauda 100
354224848179261915075
(0.01 secs, 122,448 bytes)
*Fib> fibcauda 1000
43466557686937456435688527675040625802564660517371780402481729089536555417949051890403879840079255169295922593080322634775209689623239873322471161642996440906533187938298969649928516003704476137795166849228875
(0.02 secs, 717,744 bytes)
```

Fibonacci recursiva (com cauda)

TEMPO E MEMÓRIA

Implementação	Fib 10	Fib 20	Fib 30	Fib 100	Fib 1000
Fibonacci com cauda	0.02s 74,176 bytes	0.02s 78,040 bytes	0.01s 83,352 bytes	0.01s 122,448 bytes	0.02s 717,744 bytes
Fibonacci sem cauda	0.02s 104,168 bytes	0.05s 4,227,296 bytes	8.83s 511,353,848 bytes	-	-

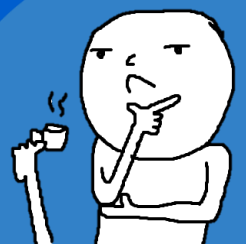
OBS: ao invés dos comandos PS e TIME, fez-se uso de ":set +s" para cálculo de tempo e memória.

INTERESSANTE

- O método da recursão com cauda segue princípios similares com os de Programação Dinâmica (PD), visando evitar que os cálculos sejam realizados só após o retorno das funções.

- A diferença é que, na Programação Dinâmica, os cálculos são minimizados caso já tenham sido calculados previamente.

- Utilizando o exemplo de Fibonacci – há um vetor que guarda os cálculos de fibonacci. Então, quando chamado fib(n), ele verifica se já foi calculado; se sim, retorna o valor respectivo; senão, faz o novo cálculo.



OBRIGADO PELA ATENÇÃO!

- Link do GitHub: https://github.com/fabhonda/estcmp005.git