ALGORITMOS E PROGRAMAÇÃO DE COMPUTADORES II

Recursão II

Recursão vs. Iteração

Problema: n-ésimo termo de Fibonacci

Quem é melhor?

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

```
def fib_it(n):
  res = n
  a, b = 0, 1
  for k in range(2, n+1):
    res = a + b
    a, b = b, res
  return res
```

Recursão vs. Iteração

Problema: n-ésimo termo de Fibonacci

Quem é melhor?

Estimativa de tempo para Fibonacci

n	*10*	*20*	*30*	*50*	*100*
Recursão	8 ms	1 s	2 min	21 dias	10^9 anos
Iteração	1/6 ms	1/3 ms	1/2 ms	3/4 ms	1,5 ms

Fonte: Brassard & Bradley, 1996

Memoização

Técnica de otimização que armazena os resultados de chamadas de funções custosas, e que retorna o valor armazenado quando a função é chamada novamente.

Memoização

Exemplo: fatorial:

```
m = dict()
def fat(n):
  if n == 0:
    return 1
  elif m.get(n) != None:
    return m[n]
  else:
    m[n] = n * fat(n-1)
    return m[n]
```

Memoização

Exemplo: n-ésimo termo da sequência de Fibonacci:

```
m = dict()
def fib_mem(n):
  if n < 2:
    return n
  elif m.get(n) != None:
    return m[n]
  else:
    m[n] = fib_mem(n-1) + fib_mem(n-2)
    return m[n]
```

Exercícios

Dada uma lista l de n números, implemente uma função recursiva que retorna o maior elemento do conjunto.

Dada uma lista l de n números, implemente uma função recursiva que retorna a soma de todos os elementos do conjunto.

ALGORITMOS E PROGRAMAÇÃO DE COMPUTADORES II

Recursão II