INSTITUTO POLITÉCNICO DE BEJA

Interacção Pessoa-Computador

Pedro Miguel Clemente Dias Moreira, 10015

Trabalho Individual 1

Engenharia Informática

20 de Março 2012

Conteúdo

Índice Geral i
Lista de Figuras iii
Lista de Tabelas v

1 Introdução **1**

2 Teoria **3**
2.1 Introdução 3
2.2 Complexidade Computacional 3
2.3 Conclusões 3

3 Resultados Experimentais **7**
3.1 Introdução 7
3.2 Protocolo Experimental 7
3.3 Análise dos Resultados Experimentais 7
3.4 Conclusões 7

4 Conclusões **11**

Bibliografia **13**

Lista de Figuras

2.1 Uma bela figura. 6

3.1 Os resultados experimentais para o algoritmo iterativo. 9

Lista de Tabelas

3.1 Uma tabela de tempos de execução. 8

1 Introdução

Objeto do Estudo

O documento apresenta um estudo sobre a usabilidade da aplicação Eugénio. São apresentadas DUAS FALHAS DE USABILIDADE que vão contra as *regras de blabla*

Trabalho do Autor

Estrutura do Documento

2 Teoria

A sequência dos números de Fibonacci pode ser representada por:

$$F(n) = \begin{cases} 0 & \text{se } n = 0; \\ 1 & \text{se } n = 1; \\ F(n-2) + F(n-1) & \text{outros} \end{cases} \tag{2.1}$$

2.1 Introdução

2.2 Complexidade Computacional

2.3 Conclusões

```
# -*- coding: utf-8 -*-

from fibo import fibonacci_recursivo

import time

N = 36

for n in range(N):
  t1 = time.clock()
  fibonacci_recursivo(n)
  t2 = time.clock()
  print n, t2-t1
  pass
```

testerec.py

As minhas conclusões.

```
# -*- coding: utf-8 -*-
'''
autor: Jose Jasnau Caeiro
data: 12 de marco de 2012
obs.: realizacoes das funcoes de calculo
      dos numeros de fibonacci
'''

def fibonacci_recursivo(n):
  '''
  realizacao recursiva do calculo dos numeros de
  fibonacci
  n - argumento inteiro positivo
  '''
  if n == 0:
    return 0
  elif n == 1:
    return 1
  else:
    return fibonacci_recursivo(n-2) +\
        fibonacci_recursivo(n-1)

def fibonacci_iterativo(n):
  '''
  realizacao iterativa
  '''
  i = 1
  j = 0
  for k in range(n):
    t = i + j
    i, j = j, t
    pass
```

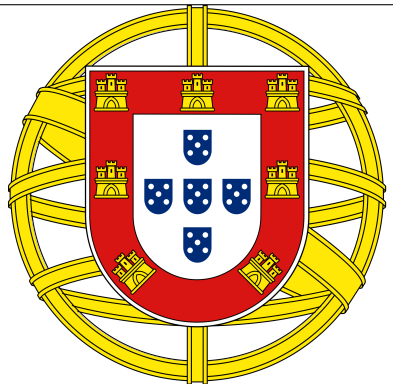

Figura 2.1: Uma bela figura.

```
    return j
```

fibo.py

$$\sqrt{\frac{1}{2}} \times f(x) = 12\beta\sqrt{3}. \tag{2.2}$$

3 Resultados Experimentais

3.1 Introdução

3.2 Protocolo Experimental

```c
#include <stdio.h>

int main(){
  int j = 0;
  for(j=0; j < 10; j++){
    printf("%d\n", j);
  }

  return(0);
}
```

teste.c

3.3 Análise dos Resultados Experimentais

3.4 Conclusões

Os resultados experimentais apontam para que a complexidade algorítimica seja do tipo $O(n)$. Na Tab. 3.1 apresentam-se os resultados experimentais para alguns valores de n.

Encontram-se em (Tiobe Index, 2012)[2] as estatísticas sobre a utilização das diversas linguagens de programação.

N	tempo de execução (seg)
0	0.0
1	0.0
2	0.0
3	0.0
4	0.0
5	0.0
6	0.0
7	0.0
8	0.0
9	0.0
10	0.0
11	0.0
12	0.0
13	0.008
14	0.012001
15	0.0
16	0.072004
17	0.0
18	0.004
19	0.004001
20	0.0
21	0.004
22	0.004
23	0.004
24	0.012001
25	0.016001
26	0.024002
27	0.044002
28	0.068005
29	0.112007
30	0.176011
31	0.292018
32	0.468029
33	0.760047
34	1.240078
35	1.972123

Tabela 3.1: Uma tabela de tempos de execução.

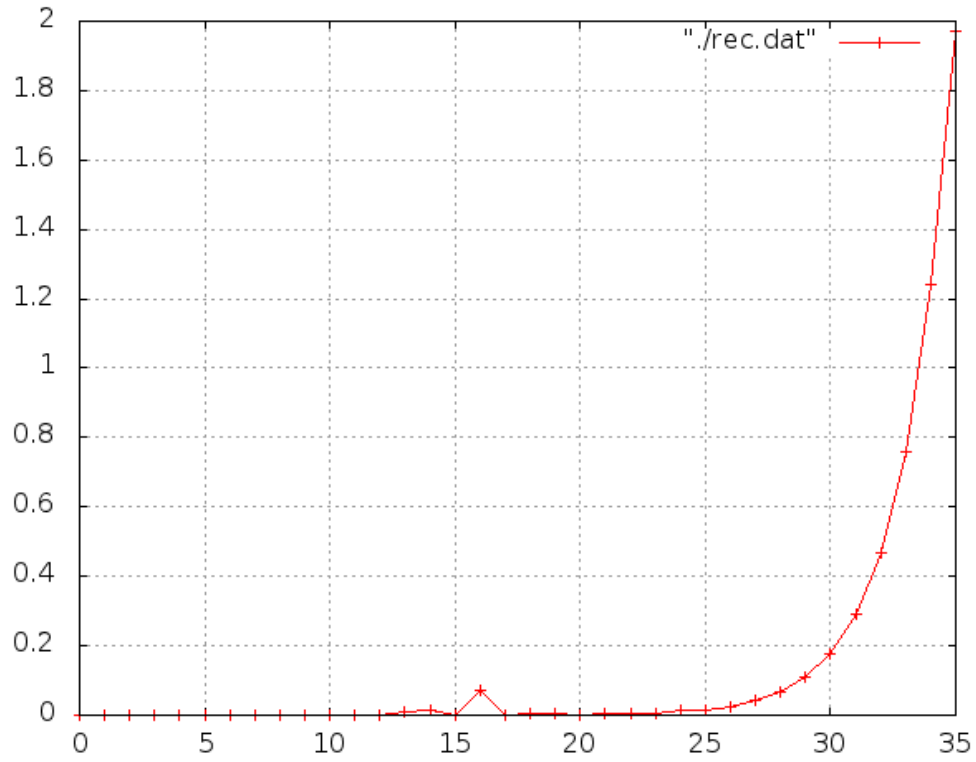

Figura 3.1: Os resultados experimentais para o algoritmo iterativo.

4 Conclusões

Complexidade

Este documento sucks. A programação em Python apresenta o potencial de ser utilizada em contextos perigosos (Seitz, 2009) [1].

Perspetivas Futuras

Desenvolvimento de software eficiente. Exemplos de programação com funções recursivas a evitar.

Bibliografia

[1] Justin Seitz. *Gray Hat Python: Python Programming for Hackers and Reverse Engineers*. No Starch Press, 2009.

[2] Tiobe programming community index. Internet, Setembro 2009. http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html.