

# Análise de Complexidade de Tempo do Método Bubble Sort

Eduardo Costa de Paiva

[eduardocspv@gmail.com](mailto:eduardocspv@gmail.com)

Frederico Franco Calhau

[fredericoffc@gmail.com](mailto:fredericoffc@gmail.com)

Gabriel Augusto Marson

[gabrielmarson@live.com](mailto:gabrielmarson@live.com)

Faculdade de Computação  
Universidade Federal de Uberlândia

5 de dezembro de 2015

# Lista de Figuras

2.1	Complexidade de tempo do método da bolha . . . . .	8
-----	--	---

# Lista de Tabelas

3.1	Alguns opcodes do Dalvik . . . . .	9
3.2	Alguns opcodes do Dalvik . . . . .	10

# Lista de Listagens

1.1	BubbleSort.py . . . . .	7
A.1	testdriver.py . . . . .	12

# Sumário

<b>Lista de Figuras</b>	<b>2</b>
<b>Lista de Tabelas</b>	<b>3</b>
<b>1 Introdução</b>	<b>6</b>
1.1 Linha de comando . . . . .	6
1.2 Códigos de programas . . . . .	7
<b>2 Imagens</b>	<b>8</b>
<b>3 Tabelas</b>	<b>9</b>
<b>4 Citações e referências bibliográficas</b>	<b>11</b>
<b>Apêndice</b>	<b>12</b>
<b>A Exemplos de programas escritos em Python</b>	<b>12</b>
A.1 testdriver.py . . . . .	12

# Capítulo 1

## Introdução

Este documento foi escrito para auxiliar na confecção do relatório da disciplina. É necessário olhar os fontes deste documento em L<sup>A</sup>T<sub>E</sub>X para compreender algumas coisas.

### 1.1 Linha de comando

Para dar instruções sobre linha de comando use um ambiente que preparei (veja o preâmbulo desse documento para aprender como criar seu próprio ambiente):

```
\begin{terminal}  
> sudo apt-get install tree  
\end{terminal}
```

Um programa bastante útil é o `tree` que lista o conteúdo de um diretório e de seus subdiretórios em forma de árvore. Para instalá-lo no Ubuntu, use:

```
> sudo apt-get install tree
```

A seguir é mostrado o uso:

```
> tree --charset=ASCII  
.  
|-- codigos  
|   |-- bolha.py  
|   `-- testdriver.py  
|-- imagens  
|   `-- bolha1.png  
|-- relatorio1.pdf  
`-- relatorio1.tex  
  
2 directories, 14 files
```

## 1.2 Códigos de programas

Para introduzir a listagem do código no documento existem pelo menos duas formas básicas, ambas usando o pacote `listings`:

### 1. Diretamente do documento L<sup>A</sup>T<sub>E</sub>X usando por exemplo

```
\begin{python}
import psutil
import resource

def memory_usage_resource():
    # return the memory usage in MB
    rusage_denom = 1024.
    mem = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss / rusage_denom
    return mem

\end{python}
```

cujo resultado é

```
import psutil
import resource

def memory_usage_resource():
    # return the memory usage in MB
    rusage_denom = 1024.
    mem = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss / rusage_denom
    return mem
```

### 2. Lendo o código diretamente do arquivo:

```
\lstinputlisting[caption={bolha.py}]{Codigos/Bubble/BubbleSort.py}
```

cujo resultado é mostrado na listagem 1.1.

Listagem 1.1: BubbleSort.py

```
1 import numpy as np
2
3 def bubble_sort(a):
4     """ Implementação do método da bolha """
5     for i in range(len(a)):
6         for j in range(len(a)-1-i):
7             if a[j] > a[j+1]:
8                 t = a[j]
9                 a[j] = a[j+1]
10                a[j+1] = t
11
12     print(a)
```

Códigos muito grandes podem ser colocados nos apêndices e referenciados em seu texto. Um exemplo é o arquivo no apêndice [A](#).

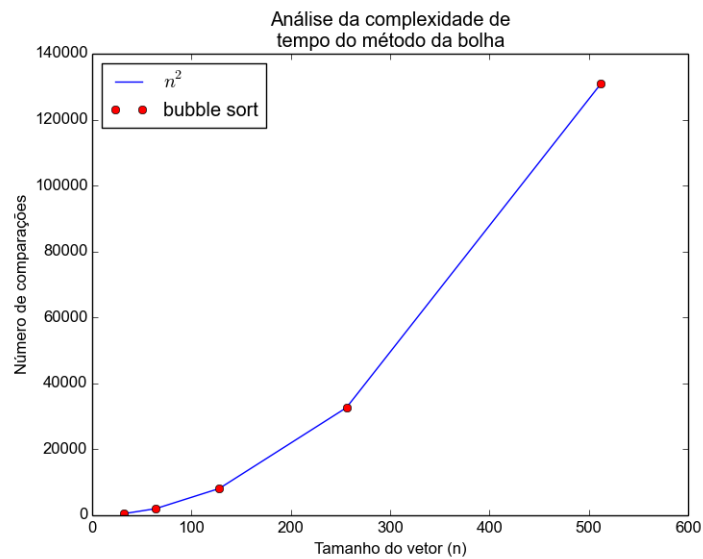
# Capítulo 2

## Imagens

Se desejar incluir imagens você poderá usar o comando a seguir:

```
\begin{figure}[!ht]
\centering
\includegraphics[scale=0.5]{imagens/bolha1.png}
\caption{Complexidade de tempo do método da bolha \label{fig:1}}
\end{figure}
```

O comando anterior gerará a imagem a seguir, dado que o arquivo `fig1.png` esteja no subdiretório `imagens`.



**Figura 2.1:** *Complexidade de tempo do método da bolha*

Note que a Figura 2.1 pode ser referenciada em qualquer parte do documento. Você também pode incluir diretamente outros formatos de imagens tais como o `jpg`.



# Capítulo 3

## Tabelas

Aqui vão alguns exemplos de criação de tabelas.

Vamos criar uma tabela simples com o código a seguir. O resultado é a Tabela 3.1

```
\begin{table}[h]
\centering
\caption{Alguns opcodes do Dalvik \label{tab:opcode1}}
\begin{tabular}{lll} \hline
{\bf Opcode (hex)} & {\bf Nome do opcode} & {\bf Explicação} \\ \hline
00 & nop & Somente gasta alguns ciclos do processador \\ \hline
01 & move vx, vy & Move o conteúdo de vy em vx. Ambos os registradores
devem estar no intervalo de 0 a 255 \\ \hline
\end{tabular}
\end{table}
```

**Tabela 3.1:** *Alguns opcodes do Dalvik*

Opcode (hex)	Nome do opcode	Explicação
00	nop	Somente gasta alguns ciclos do processador
01	move vx, vy	Move o conteúdo de vy em vx. Ambos os registradores devem es

Note que o texto sob a coluna Explicação é muito longo e a tabela 3.1 ficou mal formatada. Podemos resolver esse problema usando colunas de largura fixa, conforme mostrado no código a seguir e cujo resultado é a tabela 3.2.

```
\begin{table}[h]
\centering
\caption{Alguns opcodes do Dalvik \label{tab:opcode2}}
\begin{tabular}{p{1.5cm}lp{9cm}} \hline
{\bf Opcode (hex)} & {\bf Nome do opcode} & {\bf Explicação} \\ \hline
00 & nop & Somente gasta alguns ciclos do processador \\ \hline
01 & move vx, vy & Move o conteúdo de vy em vx. Ambos os registradores
devem estar no intervalo de 0 a 255 \\ \hline
\end{tabular}
\end{table}
```

**Tabela 3.2:** *Alguns opcodes do Dalvik*

Opcode (hex)	Nome do opcode	Explicação
00	nop	Somente gasta alguns ciclos do processador
01	move vx, vy	Move o conteúdo de vy em vx. Ambos os registradores devem estar no intervalo de 0 a 255

## Capítulo 4

# Citações e referências bibliográficas

Todas as fontes usadas para a confecção de seu relatório devem ser citadas. Isso inclui livros e documentos da internet, tais como tutoriais e páginas.

# Apêndice A

## Exemplos de programas escritos em Python

### A.1 testdriver.py

Listagem A.1: testdriver.py

```
1 import subprocess
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5
6 def executa_teste(arqteste, arqsaida, nlin, intervalo):
7     """Executa uma sequência de testes contidos em arqteste, com:
8         arqsaida: nome do arquivo de saída, ex: tBolha.dat
9         nlin: número da linha no arquivo gerado pelo line_profiler contendo
10             os dados de interesse. Ex: 14
11         intervalo: tamanhos dos vetores: Ex: 2 ** np.arange(5,10)
12     """
13     f = open(arqsaida, mode='w', encoding='utf-8')
14     f.write('#          n    comparações          tempo(s)\n')
15
16     for n in intervalo:
17         cmd = ' '.join(["kernprof -l -v", "testeBubble.py", str(n)])
18         str_saida = subprocess.check_output(cmd, shell=True).decode('utf-8')
19         linhas = str_saida.split('\n')
20         unidade_tempo = float(linhas[1].split()[2])
21         tempo_total = float(linhas[3].split()[2])
22         lcomp = linhas[nlin].split()
23         num_comps = int(lcomp[1])
24         str_res = '{:>8} {:>13} {:13.6f}'.format(n, num_comps, tempo_total /
25             unidade_tempo)
26         print(str_res)
27         f.write(str_res + '\n')
28     f.close()
29
30 # executa_teste("testeBubble.py", "tBolha.dat", 14, 2 ** np.arange(5,10))
31
32 def plota_teste1(arqsaida):
33     n, c, t = np.loadtxt(arqsaida, unpack=True)
```

## A.1

```
33 plt.plot(n, n ** 2, label='$n^2$')
34 plt.plot(n, c, 'ro', label='bubble sort')
35
36 # Posiciona a legenda
37 plt.legend(loc='upper left')
38
39 # Posiciona o título
40 plt.title('Análise da complexidade de \ntempo do método da bolha')
41
42 # Rotula os eixos
43 plt.xlabel('Tamanho do vetor (n)')
44 plt.ylabel('Número de comparações')
45
46 plt.savefig('bubble1.png')
47 plt.show()
48
49 def plota_teste2(arqsaida):
50     n, c, t = np.loadtxt(arqsaida, unpack=True)
51     plt.plot(n, n ** 2, label='$n^2$')
52     plt.plot(n, t, 'ro', label='bubble sort')
53
54     # Posiciona a legenda
55     plt.legend(loc='upper left')
56
57     # Posiciona o título
58     plt.title('Análise da complexidade de \ntempo do método da bolha')
59
60     # Rotula os eixos
61     plt.xlabel('Tamanho do vetor (n)')
62     plt.ylabel('Tempo(s)')
63
64     plt.savefig('bubble2.png')
65     plt.show()
66
67 def plota_teste3(arqsaida):
68     n, c, t = np.loadtxt(arqsaida, unpack=True)
69
70     # Calcula os coeficientes de um ajuste a um polinômio de grau 2 usando
71     # o método dos mínimos quadrados
72     coefs = np.polyfit(n, t, 2)
73     p = np.polyld(coefs)
74
75     plt.plot(n, p(n), label='$n^2$')
76     plt.plot(n, t, 'ro', label='bubble sort')
77
78     # Posiciona a legenda
79     plt.legend(loc='upper left')
80
81     # Posiciona o título
82     plt.title('Análise da complexidade de \ntempo do método da bolha')
83
84     # Rotula os eixos
85     plt.xlabel('Tamanho do vetor (n)')
86     plt.ylabel('Tempo(s)')
87
88     plt.savefig('bubble3.png')
89     plt.show()
90
91 def plota_teste4(arqsaida):
```

```

92     n, c, t = np.loadtxt(arqsaida, unpack=True)
93
94     # Calcula os coeficientes de um ajuste a um polinômio de grau 2 usando
95     # o método dos mínimos quadrados
96     coefs = np.polyfit(n, c, 2)
97     p = np.polyld(coefs)
98
99     plt.plot(n, p(n), label='$n^2$')
100    plt.plot(n, c, 'ro', label='bubble sort')
101
102    # Posiciona a legenda
103    plt.legend(loc='upper left')
104
105    # Posiciona o título
106    plt.title('Análise da complexidade de \ntempo do método da bolha')
107
108    # Rotula os eixos
109    plt.xlabel('Tamanho do vetor (n)')
110    plt.ylabel('Número de comparações')
111
112    plt.savefig('bubble4.png')
113    plt.show()
114
115    def plota_teste5(arqsaida):
116        n, c, t = np.loadtxt(arqsaida, unpack=True)
117
118        # Calcula os coeficientes de um ajuste a um polinômio de grau 2 usando
119        # o método dos mínimos quadrados
120        coefs = np.polyfit(n, c, 2)
121        p = np.polyld(coefs)
122
123        # set_yscale('log')
124        # set_yscale('log')
125        plt.semilogy(n, p(n), label='$n^2$')
126        plt.semilogy(n, c, 'ro', label='bubble sort')
127
128        # Posiciona a legenda
129        plt.legend(loc='upper left')
130
131        # Posiciona o título
132        plt.title('Análise da complexidade de \ntempo do método da bolha')
133
134        # Rotula os eixos
135        plt.xlabel('Tamanho do vetor (n)')
136        plt.ylabel('Número de comparações')
137
138        plt.savefig('bubble5.png')
139        plt.show()

```

---