Análise de Complexidade de Tempo do Método Insertion Sort

Eduardo Costa de Paiva
eduardocspv@gmail.com
Frederico Franco Calhau
fredericoffc@gmail.com
Gabriel Augusto Marson
gabrielmarson@live.com

Faculdade de Computação
Universidade Federal de Uberlândia

18 de dezembro de 2015

Lista de Figuras

Lista de Tabelas

Lista de Listagens

Sumário

Capítulo 1

Introdução

Este documento foi feito com o intuito de exibir uma análise do algoritmo Insertion Sort com relação a tempo. Além disso, será feita uma comparação da curva de tempo do que se espera do algoritmo, ou seja, $\theta(n^2)$ com o caso prático.

1.1 Diretório

Dada a seguinte organização das pastas, utilizamos o arquivo testdriver.py, executando, uma função conveniente por vez. Para mais informações vá até ao apêndice.

OBS.: É necessário instalar o programa tree pelo terminal. Isso pode ser feito da seguinte maneira.

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

A seguir é mostrada a organização das pastas sendo que os diretórios significativas para o projeto são Codigos e Relatorio além do raíz:

```
tree --charset=ASCII -c
.
|-- testdriver.py
|-- testeGeneric.py.lprof
|-- __pycache__
|   |-- monitor.cpython-34.pyc
|   |-- testeGeneric.cpython-34.pyc
|   |-- memoria.cpython-34.pyc
|   `-- tempo.cpython-34.pyc
|-- monitor.py
|-- testeGeneric.py
|-- testeGeneric2.py
|-- relatorio
|   |-- Relatorio_Selection
|   |   |-- RelatorioSelection.tex
|   |   |-- RelatorioSelection.aux
|   |   |-- RelatorioSelection.lof
|   |   |-- RelatorioSelection.log
|   |   |-- RelatorioSelection.lol
```

```
|   |   |-- RelatorioSelection.lot
|   |   |-- RelatorioSelection.out
|   |   |-- RelatorioSelection.pdf
|   |   |-- RelatorioSelection.synctex.gz
|   |   |-- RelatorioSelection.toc
|   |   |-- RelatorioSelection.idx
|   |   |-- testdriver.py
|   |   `-- testeGeneric.py
|   |-- Relatorio_Insertion
|   |   |-- testdriver.py
|   |   `-- testeGeneric.py
|   |-- Relatorio_Bubble
|   |   |-- testdriver.py
|   |   |-- testeGeneric.py
|   |   |-- RelatorioBubble.pdf
|   |   |-- RelatorioBubble.idx
|   |   `-- RelatorioBubble.tex
|   |-- imagens
|   |   |-- Selection
|   |   |   |-- selection_plot_3_ordenado_descresente.png
|   |   |   |-- selection_plot_2_ordenado_descresente.png
|   |   |   |-- selection_plot_1_ordenado_descresente.png
|   |   |   |-- selection_plot_3_aleatorio.png
|   |   |   |-- selection_plot_2_aleatorio.png
|   |   |   |-- selection_plot_1_aleatorio.png
|   |   |   |-- selection_plot_3_ordenado_crescente.png
|   |   |   |-- selection_plot_2_ordenado_crescente.png
|   |   |   |-- selection_plot_1_ordenado_crescente.png
|   |   |   |-- selection_plot_3_ordenado_decrescente.png
|   |   |   |-- selection_plot_2_ordenado_decrescente.png
|   |   |   |-- selection_plot_1_ordenado_decrescente.png
|   |   |   |-- selection_plot_3_parcialmente_ordenado_crescente.png
|   |   |   |-- selection_plot_2_parcialmente_ordenado_crescente.png
|   |   |   |-- selection_plot_1_parcialmente_ordenado_crescente.png
|   |   |   |-- selection_plot_3_parcialmente_ordenado_decrescente.png
|   |   |   |-- selection_plot_2_parcialmente_ordenado_decrescente.png
|   |   |   `-- selection_plot_1_parcialmente_ordenado_decrescente.png
|   |   |-- Insertion
|   |   |   |-- insertion_plot_3_parcialmente_ordenado_decrescente.png
|   |   |   |-- insertion_plot_2_parcialmente_ordenado_decrescente.png
|   |   |   |-- insertion_plot_1_parcialmente_ordenado_decrescente.png
|   |   |   |-- insertion_plot_3_parcialmente_ordenado_crescente.png
|   |   |   |-- insertion_plot_2_parcialmente_ordenado_crescente.png
|   |   |   |-- insertion_plot_1_parcialmente_ordenado_crescente.png
|   |   |   |-- insertion_plot_3_ordenado_decrescente.png
|   |   |   |-- insertion_plot_2_ordenado_decrescente.png
|   |   |   |-- insertion_plot_1_ordenado_decrescente.png
|   |   |   |-- insertion_plot_3_ordenado_crescente.png
|   |   |   |-- insertion_plot_2_ordenado_crescente.png
|   |   |   |-- insertion_plot_1_ordenado_crescente.png
|   |   |   |-- insertion_plot_3_aleatorio.png
|   |   |   |-- insertion_plot_2_aleatorio.png
|   |   |   `-- insertion_plot_1_aleatorio.png
|   |   |-- Bubble
|   |   |   |-- bubble_plot_3_parcialmente_ordenado_decrescente.png
|   |   |   |-- bubble_plot_2_parcialmente_ordenado_decrescente.png
|   |   |   |-- bubble_plot_1_parcialmente_ordenado_decrescente.png
|   |   |   |-- bubble_plot_3_parcialmente_ordenado_crescente.png
|   |   |   |-- bubble_plot_2_parcialmente_ordenado_crescente.png
```

```
|   |   |   |-- bubble_plot_1_parcialmente_ordenado_crescente.png
|   |   |   |-- bubble_plot_3_ordenado_decrescente.png
|   |   |   |-- bubble_plot_2_ordenado_decrescente.png
|   |   |   |-- bubble_plot_1_ordenado_decrescente.png
|   |   |   |-- bubble_plot_3_ordenado_crescente.png
|   |   |   |-- bubble_plot_2_ordenado_crescente.png
|   |   |   |-- bubble_plot_1_ordenado_crescente.png
|   |   |   |-- bubble_plot_3_aleatorio.png
|   |   |   |-- bubble_plot_2_aleatorio.png
|   |   |   `-- bubble_plot_1_aleatorio.png
|   |   |-- README.md
|   |   `-- Merge
|   |-- Resultados
|   |   |-- Selection
|   |   |   |-- tSelection_vetor_ordenado_descresente.dat
|   |   |   |-- tSelection_vetor_aleatorio.dat
|   |   |   |-- tSelection_vetor_ordenado_crescente.dat
|   |   |   |-- tSelection_vetor_parcialmente_ordenado_crescente.dat
|   |   |   `-- tSelection_vetor_parcialmente_ordenado_decrescente.dat
|   |   |-- Insertion
|   |   |   |-- tInsertion_vetor_parcialmente_ordenado_decrescente.dat
|   |   |   |-- tInsertion_vetor_parcialmente_ordenado_crescente.dat
|   |   |   |-- tInsertion_vetor_ordenado_decrescente.dat
|   |   |   |-- tInsertion_vetor_ordenado_crescente.dat
|   |   |   `-- tInsertion_vetor_aleatorio.dat
|   |   |-- Bubble
|   |   |   |-- tBolha_vetor_parcialmente_ordenado_decrescente.dat
|   |   |   |-- tBolha_vetor_parcialmente_ordenado_crescente.dat
|   |   |   |-- tBolha_vetor_ordenado_decrescente.dat
|   |   |   |-- tBolha_vetor_ordenado_crescente.dat
|   |   |   `-- tBolha_vetor_aleatorio.dat
|   |   `-- Merge
|   `-- Relatorio_Merge
|-- Codigos
|   |-- Selection
|   |   |-- __pycache__
|   |   |   `-- SelectionSort.cpython-34.pyc
|   |   `-- SelectionSort.py
|   |-- Insertion
|   |   |-- __pycache__
|   |   |   `-- InsertionSort.cpython-34.pyc
|   |   `-- InsertionSort.py
|   |-- Merge
|   |   `-- mergeSort.py
|   |-- Quick
|   |   `-- quickSort.py
|   |-- Bubble
|   |   |-- BubbleSort.py
|   |   `-- __pycache__
|   |       |-- BubbleSort.cpython-34.pyc
|   |       `-- testeBubble.cpython-34.pyc
|   `-- README.md
|-- Referências.txt
|-- Referências.txt~
|-- Other
|   |-- expfit0.py
|   |-- expfit.py
|   |-- leialprof.py
|   |-- leitura1.py
```

```
|   |-- leitura2.py
|   |-- leitura.py
|   `-- logfit.py
|-- Plot
|   |-- plot_tempo.py
|   |-- plot1.py
|   |-- plot2.py
|   `-- plot3.py
|-- memoria.py
`-- tempo.py
```

1.2 Códigos de programas

Seguem os códigos utilizados na análise de tempo do algoritmo Insertion Sort.

1. InsertionSort.py: Disponível na Listagem 1.1.

Listagem 1.1: InsertionSort.py

```
1
2 @profile
3 def insertionSort(lista):
4   for j in range(1,len(lista)):
5     chave = lista[j]
6     i = j
7     while (i>0 and lista[i-1]>chave):
8       lista[i] = lista[i-1]
9       i = i-1
10    lista[i] = chave
11
12 #lista = [60,20,30,12,1,2,3,39,45,10]
13 #insertionSort(lista)
14 #print(lista)
```

2. testeGeneric.py Disponível na Listagem 1.2

Listagem 1.2: testeGeneric.py

```
1 ##adicionei - Serve para importar arquivos em outro diretório
2 ###  A CADA NOVO MÉTODO MUDAR O IMPORT,  A CHAMADA DA FUNÇÃO E O SYS.
      PATH
3
4 import sys
5 sys.path.append('/home/gmarson/Git/AnaliseDeAlgoritmos/Trabalho_Final
      /Codigos/Insertion')
6
7 from monitor import *
8
9
10 from InsertionSort import *
11 import argparse
12
13 parser = argparse.ArgumentParser()
14 parser.add_argument("n", type=int, help="número de elementos no vetor
       de teste")
```

```
15 args = parser.parse_args()
16
17 v = criavet(args.n)
18 insertionSort(v)
19
20
21
22 ## A EXECUÇÃO DESSE ARQUIVO EH ASSIM
23 ## NA LINHA DE COMANDO VC MANDA O NOME DO ARQUIVO E O TAMANHO DO
       ELEMNTO DO vetor
24 ##EXEMPLO testeBubble.py 10
25 ##ele gera um vetor aleatório (criavet) e manda pro bubble_sort
```

3. monitor.py Disponível na Listagem 1.3

Listagem 1.3: monitor.py

```
1  # Para instalar o Python 3 no Ubuntu 14 ou 15
2  #
3  # sudo apt-get install python3 python3-numpy python3-matplotlib
       ipython3 python3-psutil
4  #
5
6  from math import *
7  import gc
8  import random
9  import numpy as np
10
11
12 from tempo import *
13
14 # Vetores de teste
15 def troca(m,v,n): ## seleciona o nível de embaralhamento do vetor
16     m = trunc(m)
17     mi = (n-m)//2
18     mf = (n+m)//2
19     for num in range(mi,mf):
20         i = np.random.randint(mi,mf)
21         j = np.random.randint(mi,mf)
22         #print("i= ", i, " j= ", j)
23         t = v[i]
24         v[i] = v[j]
25         v[j] = t
26     return v
27
28
29 def criavet(n, grau=0, inf=0, sup=0.9999999999):
30     passo = (sup - inf)/n
31     if grau < 0.0:
32         v = np.arange(sup, inf, -passo)
33         if grau <= -1.0:
34             return v
35         else:
36             return troca(-grau*n, v, n)
37     elif grau > 0.0:
38         v = np.arange(inf, sup, passo)
39         if grau >= 1.0:
40             return v
41         else:
```

```
42              return troca(grau*n, v, n)
43      else:
44          #return np.random.randint(inf, sup, size=n)
45          return [random.random() for i in range(n)] # for bucket sort
46
47
48
49  #print(criavet(20))
50
51  #Tipo                                    grau
52  #aleatorio                               0
53  #ordenado_crescente                      1
54  #ordenado_decrescente                    -1
55  #parcialmente_ordenado_crescente         0.5
56  #parcialmente_ordenado_decrescente      -0.5
57
58
59  def executa(fn, v):
60      gc.disable()
61      with Tempo(True) as tempo:
62          fn(v)
63      gc.enable()
```

4. testdriver.py Referenciado no apêndice A.

Capítulo 2

Gráficos

Seguem os Gráficos utilizadas no processo de análise do método Insertion Sort:

1. Para um vetor aleatório
   (a) Complexidade de custo do método da inserção disponível na lista de imagens 2.1.

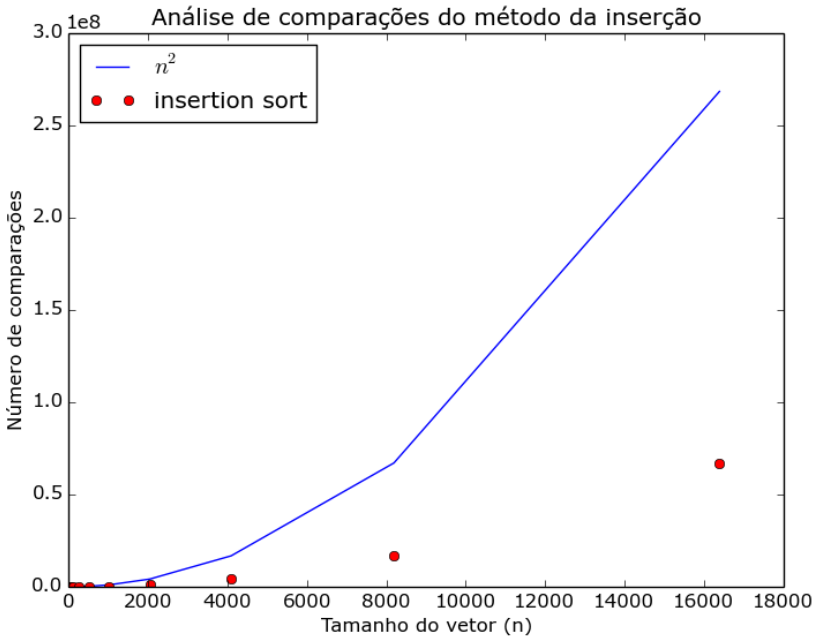

Figura 2.1: *Complexidade de custo do método da inserção (Vetor Aleatório)*

   (b) Complexidade de tempo do método da inserção disponível na lista de imagens 2.2.
   (c) Complexidade de tempo do método da inserção com mínimos quadrados disponível na lista de imagens 2.3.

2. Para um vetor ordenado crescente

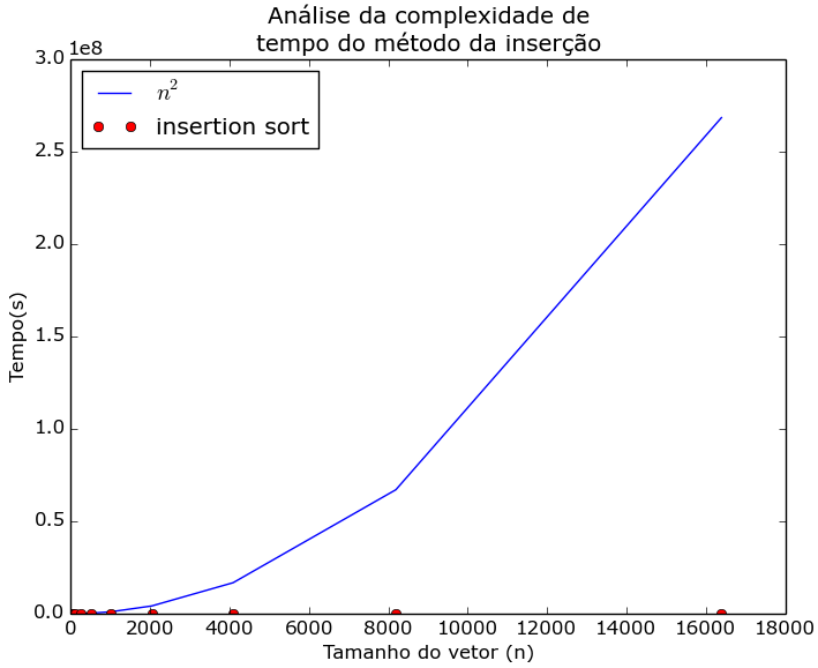

Figura 2.2: *Complexidade de tempo do método da inserção (Vetor Aleatório)*

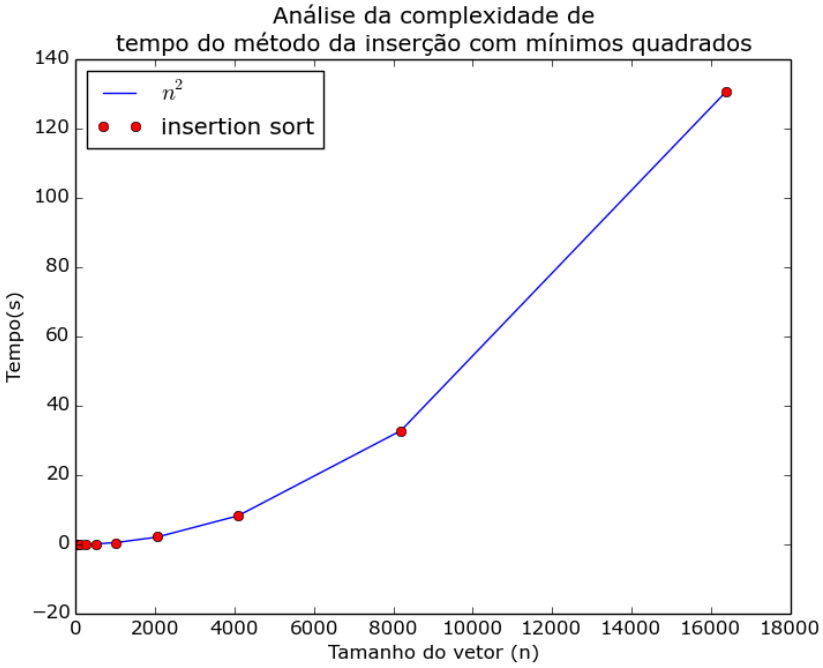

Figura 2.3: *Complexidade de tempo do método da inserção com mínimos quadrados (Vetor Aleatório)*

(a) Complexidade de custo do método da inserção disponível na lista de imagens 2.4.

(b) Complexidade de tempo do método da inserção disponível na lista de imagens 2.5.

(c) Complexidade de tempo do método da inserção com mínimos quadrados disponível na lista de imagens 2.6.

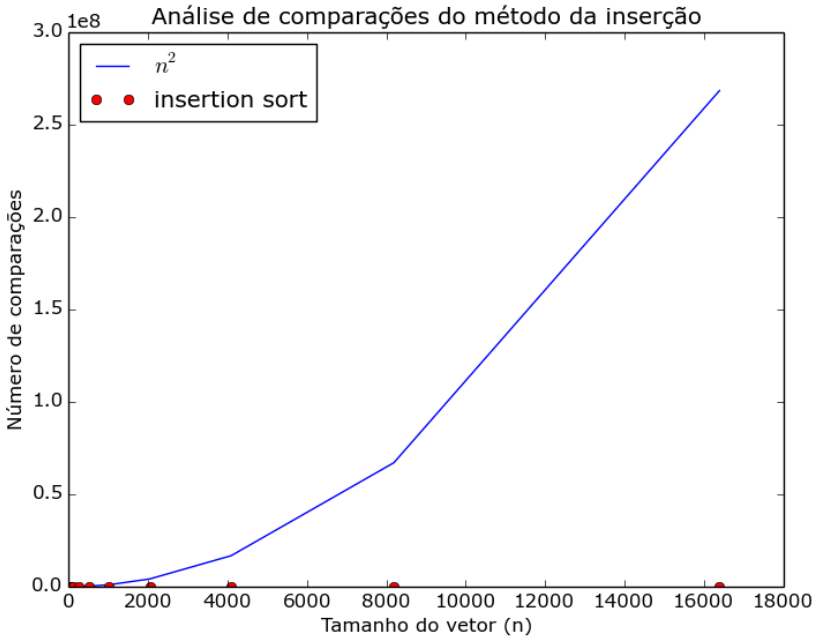

Figura 2.4: *Complexidade de custo do método da inserção (Vetor Ordenado Crescente)*

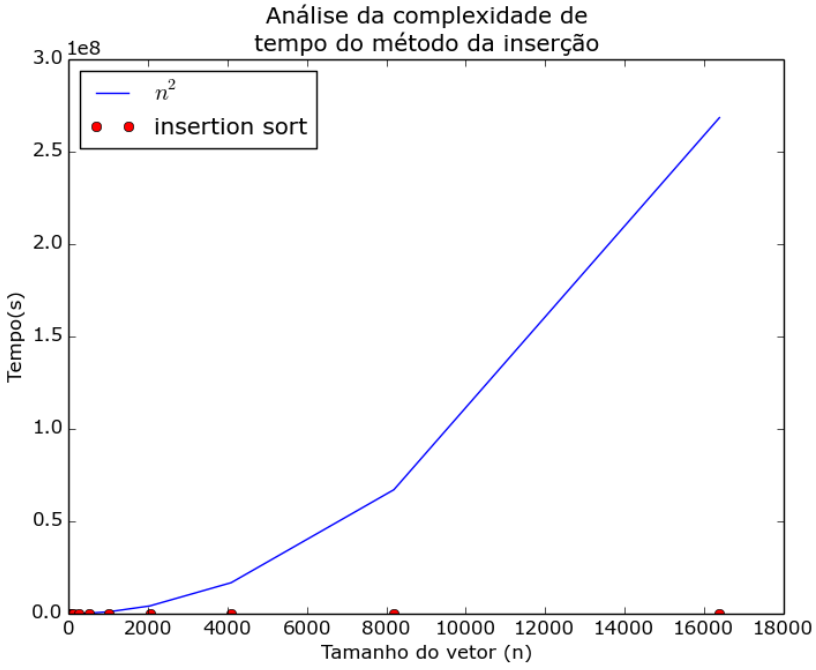

Figura 2.5: *Complexidade de tempo do método da inserção (Vetor Ordenado Crescente)*

3. Para um vetor ordenado decrescente

   (a) Complexidade de custo do método da inserção disponível na lista de imagens 2.7.

   (b) Complexidade de tempo do método da inserção disponível na lista de imagens 2.8.

   (c) Complexidade de tempo do método da inserção com mínimos quadrados disponível na lista de imagens 2.9.

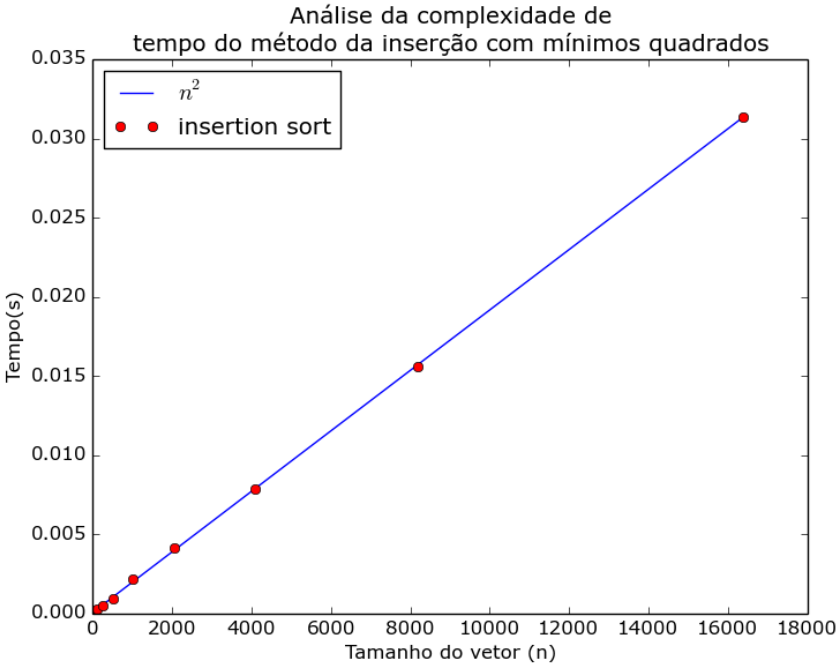

Figura 2.6: *Complexidade de tempo do método da inserção com mínimos quadrados (Vetor Ordenado Crescente)*

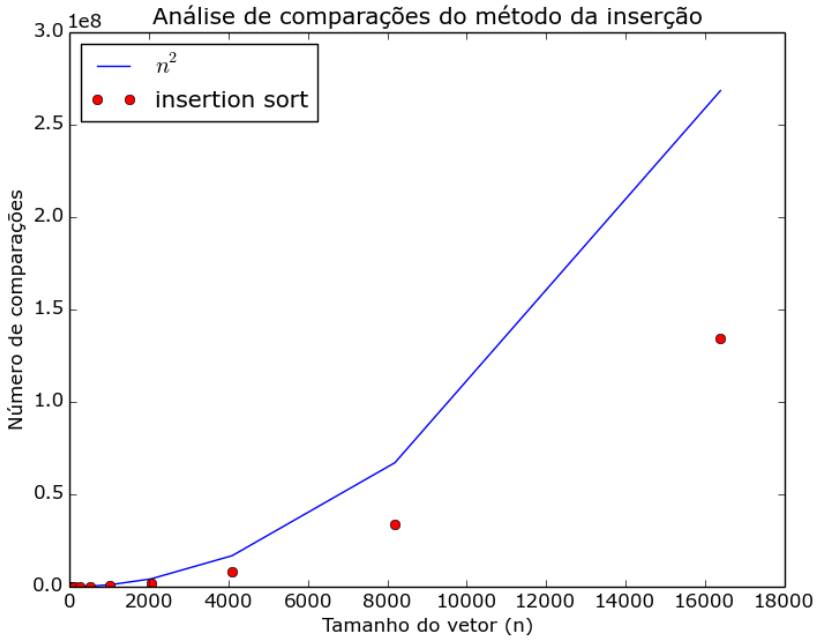

Figura 2.7: *Complexidade de custo do método da inserção (Vetor Ordenado Decrescente)*

4. Para um vetor parcialmente ordenado crescente

   (a) Complexidade de custo do método da inserção disponível na lista de imagens 2.10.

   (b) Complexidade de tempo do método da inserção disponível na lista de imagens 2.11.

   (c) Complexidade de tempo do método da inserção com mínimos quadrados dispo-

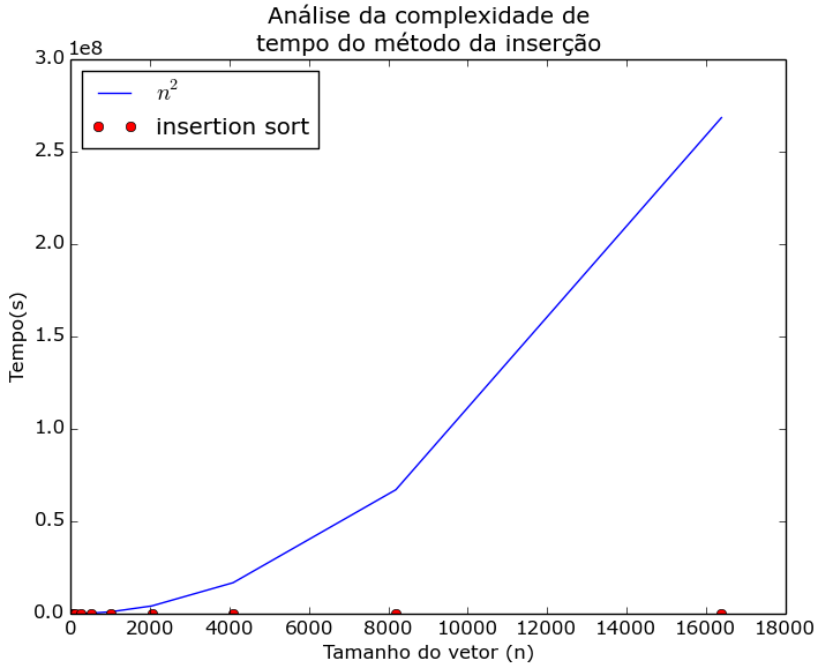

Figura 2.8: *Complexidade de tempo do método da inserção (Vetor Ordenado Decrescente)*

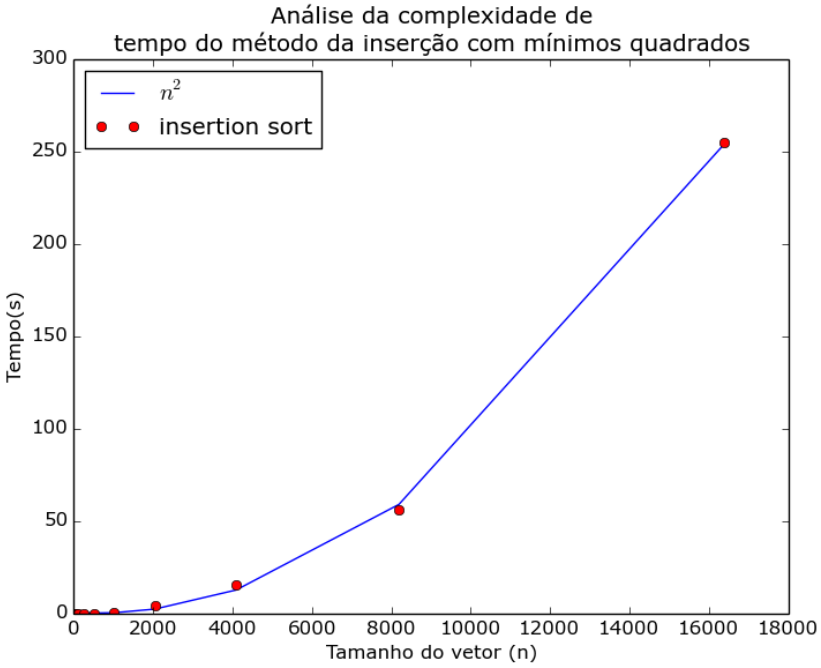

Figura 2.9: *Complexidade de tempo do método da inserção com mínimos quadrados (Vetor Ordenado Decrescente)*

nível na lista de imagens 2.12.

5. Para um vetor parcialmente ordenado decrescente

   (a) Complexidade de custo do método da inserção disponível na lista de imagens 2.13.

   (b) Complexidade de tempo do método da inserção disponível na lista de imagens

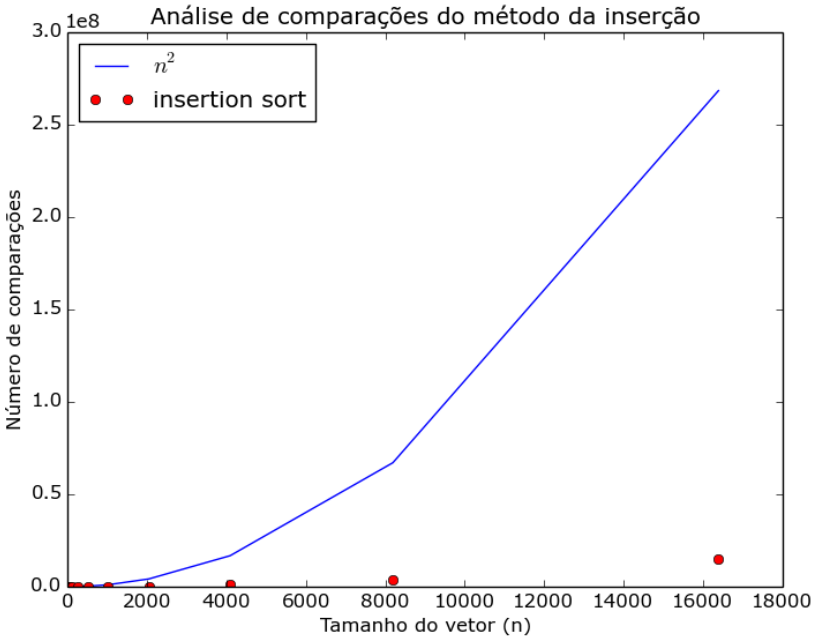

Figura 2.10: *Complexidade de custo do método da inserção (Vetor Parcialmente Ordenado Crescente)*

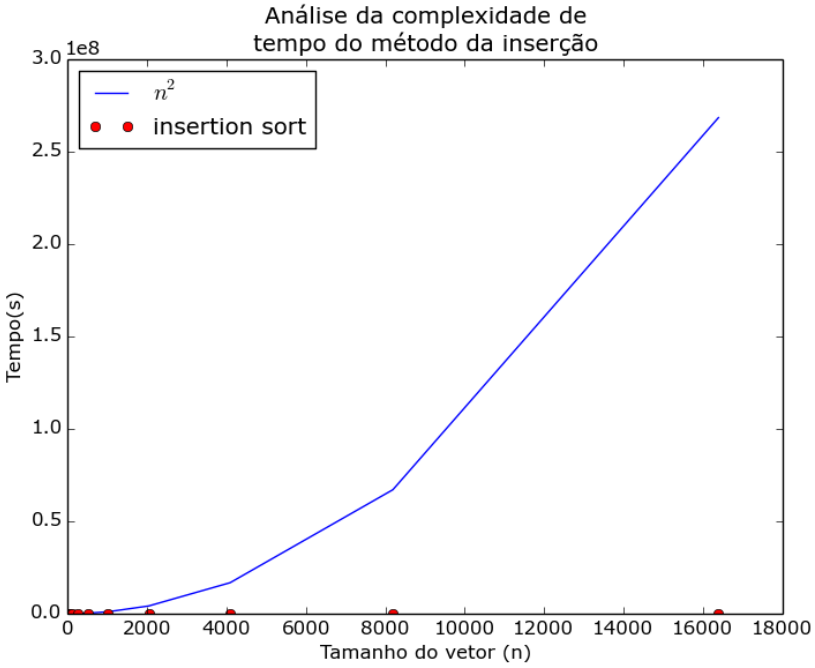

Figura 2.11: *Complexidade de tempo do método da inserção (Vetor Parcialmente Ordenado Crescente)*

2.14.

(c) Complexidade de tempo do método da inserção com mínimos quadrados disponível na lista de imagens 2.15.

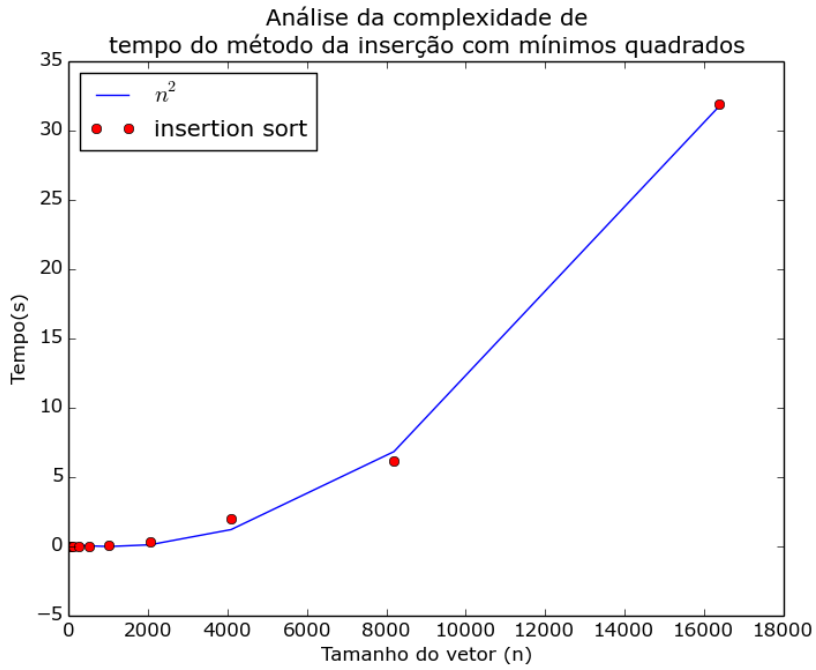

Figura 2.12: *Complexidade de tempo do método da inserção com mínimos quadrados (Vetor Parcialmente Ordenado Crescente)*

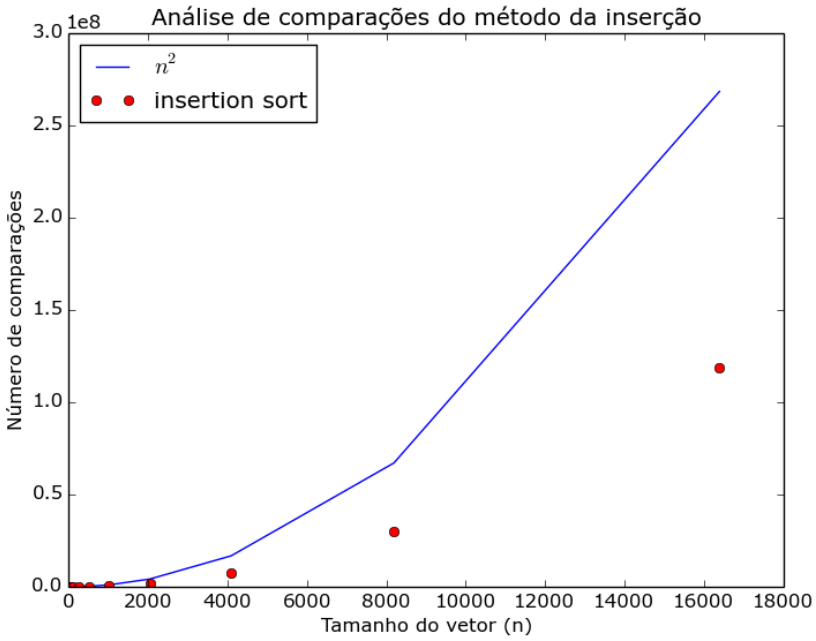

Figura 2.13: *Complexidade de custo do método da inserção (Vetor Parcialmente Ordenado Decrescente)*

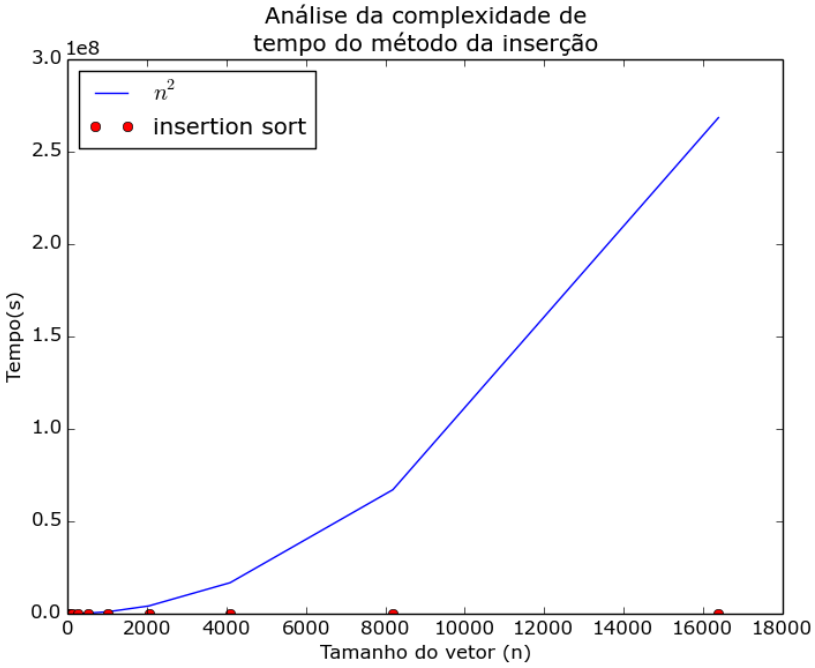

Figura 2.14: *Complexidade de tempo do método da inserção (Vetor Parcialmente Ordenado Decrescente)*

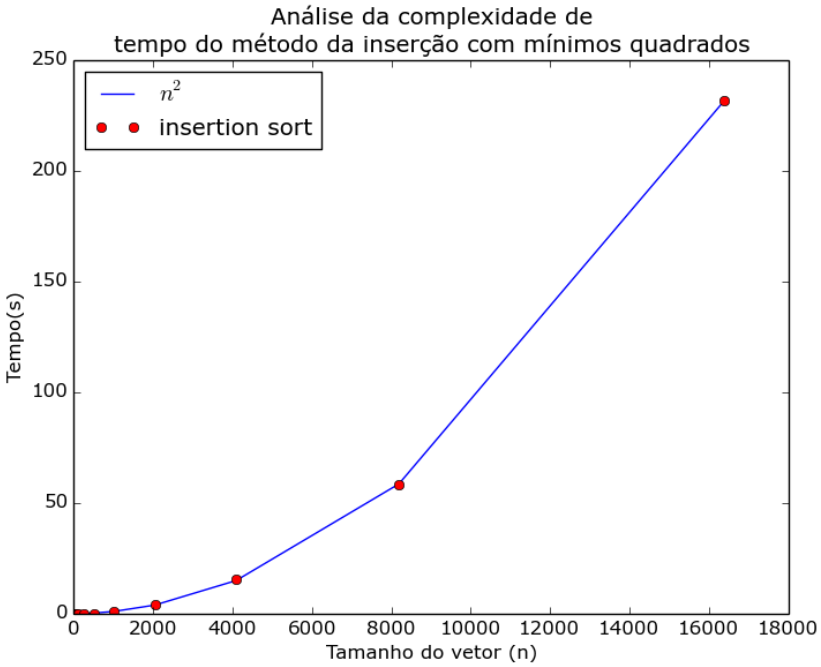

Figura 2.15: *Complexidade de tempo do método da inserção com mínimos quadrados (Vetor Parcialmente Ordenado Decrescente)*

Capítulo 3

Tabelas

Seguem as tabelas utilizadas para a análise do método Insertion Sort.

Tabela 3.1: *Vetor Aleatorio*

Tamanho do Vetor	Comparações	Tempo(s)
32	496	0.000437
64	2016	0.001770
128	8128	0.006380
256	32640	0.021552
512	130816	0.082465
1024	523776	0.355748
2048	2096128	2.281690
4096	8386560	8.310780
8192	33550336	32.757100
16384	134209536	130.790000

Tabela 3.2: *Vetor Ordenado Crescente*

Tamanho do Vetor	Comparações	Tempo(s)
32	496	0.000068
64	2016	0.000143
128	8128	0.000260
256	32640	0.000497
512	130816	0.000959
1024	523776	0.002222
2048	2096128	0.004169
4096	8386560	0.007900
8192	33550336	0.015616
16384	134209536	0.031387

Tabela 3.3: *Vetor Ordenado Decrescente*

Tamanho do Vetor	Comparações	Tempo(s)
32	496	0.000730
64	2016	0.002855
128	8128	0.011152
256	32640	0.047155
512	130816	0.171817
1024	523776	0.738637
2048	2096128	4.272890
4096	8386560	15.397500
8192	33550336	56.451300
16384	134209536	254.726000

Tabela 3.4: *Vetor Parcialmente Ordenado Crescente*

Tamanho do Vetor	Comparações	Tempo(s)
32	496	0.000163
64	2016	0.000443
128	8128	0.001684
256	32640	0.005593
512	130816	0.019021
1024	523776	0.087877
2048	2096128	0.310158
4096	8386560	1.982500
8192	33550336	6.169420
16384	134209536	31.893300

Tabela 3.5: *Vetor Parcialmente Ordenado Decrescente*

Tamanho do Vetor	Comparações	Tempo(s)
32	496	0.000613
64	2016	0.002507
128	8128	0.010110
256	32640	0.040448
512	130816	0.154370
1024	523776	0.905664
2048	2096128	4.333680
4096	8386560	15.375300
8192	33550336	58.285700
16384	134209536	231.875000

Tabela 3.6: *Dados para Análise de Memória*

Tamanho do Vetor	**Memória (MiB)**
32	24.125000
64	24.078000
128	23.270000
256	24.074000
512	24.098000

Capítulo 4

Análise

O Insertion Sort é comumente comparado à forma com que as pessoas ordenam cartas de baralho em suas mãos.

A ordem de complexidade esperada para o Insertion Sort para o pior caso é de $\theta(n^2)$, e ela ocorre quando o arranjo está na ordem inversa. O melhor caso ocorre quando o arranjo já está praticamente ordenado, conseguindo um tempo de ordem $\theta(n)$. No caso médio, pode-se dizer que ele está na ordem $O(n)$.

O Insertion Sort possui a seguintes características: estável e in-place. Além disso, ele possui a característica de – para vetores extremamente pequenos – conseguir ordernar em tempo linear. Por isso, outros algoritmos utilizam o Insertion Sort em passos bases de suas implementações, como por exemplo o Bucket Sort e implementações eficientes do Quick Sort.

Podemos observar que todas as curvas de todos os gráficos, exceto os de complexidade de tempo sem a interpolação dos mínimos quadrados(Gráficos 2.2,2.5,2.8,2.11,2.14), apresentaram uma correspondência forte com a curva da função $F(x) = x^2$, o que nos permite concluir que, dada a complexidade de tempo do algoritmo Insertion Sort por $G(x)$ então $F(x) = c * G(x)$ sendo que c é uma constante maior que zero e $x > x_0$. Portanto, o Insertion Sort é $O(n^2)$.

Capítulo 5

Citações e referências bibliográficas

[1] Algoritmos: Teoria e Prática. Thomas H. Cormen Today

Apêndice A

Códigos extensos

A.1 testdriver.py

Listagem A.1: testdriver.py

```
1  import subprocess
2  import numpy as np
3  import matplotlib.pyplot as plt
4  import sys , shutil
5
6
7  ##PRA CADA NOVO METODO TEM QUE MUDAR
8  #Sys.path()
9
10 ## PARA CADA VETOR NOVO OU NOVO MÉTODO TEM QUE MUDAR
11 #Para o executa_teste a chamada das funções e o shutil.move()
12 #para os plots        a chamada das funções e o savefig
13
14 sys.path.append('/home/gmarson/Git/AnaliseDeAlgoritmos/Trabalho_Final/
       Codigos/Insertion')  ## adicionei o código de ordenação
15 sys.path.append('/home/gmarson/Git/AnaliseDeAlgoritmos/Trabalho_Final/
       relatorio/Resultados/Insertion') ## adicionei o resultado do
       executa_teste
16
17
18 def executa_teste(arqteste, arqsaida, nlin, intervalo):
19     """Executa uma sequência de testes contidos em arqteste, com:
20        arqsaida: nome do arquivo de saída, ex: tBolha.dat
21        nlin: número da linha no arquivo gerado pelo line_profiler contendo
22              os dados de interesse. Ex: 14
23        intervalo: tamanhos dos vetores: Ex: 2 ** np.arange(5,10)
24     """
25     f = open(arqsaida,mode='w', encoding='utf-8')
26     f.write('#      n   comparações      tempo(s)\n')
27
28     for n in intervalo:
29         cmd = ' '.join(["kernprof -l -v", "testeGeneric.py", str(n)])
30         str_saida = subprocess.check_output(cmd, shell=True).decode('utf-8
               ')
31         linhas = str_saida.split('\n')
32         unidade_tempo = float(linhas[1].split()[2])
```

```
33          #print("CMD:", cmd, "\nSTR_SAIDA: ",str_saida,"\nLINHAS: ",linhas
                ,"\nUNIDADE_TEMPO: ",unidade_tempo)
34          #print("Linhas4:",linhas[4]," ----->  Linhas 4 float: ",linhas[4].
                split()[2])
35          tempo_total = float(linhas[3].split()[2])
36          lcomp = linhas[nlin].split()
37          num_comps = int(lcomp[1])
38          str_res = '{:>8} {:>13} {:13.6f}'.format(n, num_comps, tempo_total
                )
39          print(str_res)
40          f.write(str_res + '\n')
41      f.close()
42      shutil.move("tInsertion_vetor_aleatorio.dat", "/home/gmarson/Git/
            AnaliseDeAlgoritmos/Trabalho_Final/relatorio/Resultados/Insertion/
            tInsertion_vetor_aleatorio.dat")
43
44  #executa_teste("testeGeneric.py", "tInsertion_vetor_aleatorio.dat", 14, 2
        ** np.arange(5,15))
45
46  def plota_teste1(arqsaida):
47      n, c, t = np.loadtxt(arqsaida, unpack=True)
48      #print("n: ",n,"\nc: ",c,"\nt: ",t)
49      #n eh o tamanho da entrada , c eh o tanto de comparações e t eh o
            tempo gasto
50      plt.plot(n, n ** 2, label='$n^2$')  ## custo esperado bubble Sort
51      plt.plot(n, c, 'ro', label='insertion sort')
52
53      # Posiciona a legenda
54      plt.legend(loc='upper left')
55
56      # Posiciona o título
57      plt.title('Análise de comparações do método da inserção')
58
59      # Rotula os eixos
60      plt.xlabel('Tamanho do vetor (n)')
61      plt.ylabel('Número de comparações')
62
63      plt.savefig('relatorio/imagens/Insertion/insertion_plot_1_aleatorio.
            png')
64      plt.show()
65
66
67
68  def plota_teste2(arqsaida):
69      n, c, t = np.loadtxt(arqsaida, unpack=True)
70      plt.plot(n, n ** 2, label='$n^2$')
71      plt.plot(n, t, 'ro', label='insertion sort')
72
73      # Posiciona a legenda
74      plt.legend(loc='upper left')
75
76      # Posiciona o título
77      plt.title('Análise da complexidade de \ntempo do método da inserção')
78
79      # Rotula os eixos
80      plt.xlabel('Tamanho do vetor (n)')
81      plt.ylabel('Tempo(s)')
82
```

```
83      plt.savefig('relatorio/imagens/Insertion/insertion_plot_2_aleatorio.
            png')
84      plt.show()
85
86
87
88
89  def plota_teste3(arqsaida):
90      n, c, t = np.loadtxt(arqsaida, unpack=True)
91
92      # Calcula os coeficientes de um ajuste a um polinômio de grau 2 usando
93      # o método dos mínimos quadrados
94      coefs = np.polyfit(n, t, 2)
95      p = np.poly1d(coefs)
96
97      plt.plot(n, p(n), label='$n^2$')
98      plt.plot(n, t, 'ro', label='insertion sort')
99
100     # Posiciona a legenda
101     plt.legend(loc='upper left')
102
103     # Posiciona o título
104     plt.title('Análise da complexidade de \ntempo do método da inserção
            com mínimos quadrados')
105
106     # Rotula os eixos
107     plt.xlabel('Tamanho do vetor (n)')
108     plt.ylabel('Tempo(s)')
109
110     plt.savefig('relatorio/imagens/Insertion/insertion_plot_3_aleatorio.
            png')
111     plt.show()
112
113 plota_teste1("/home/gmarson/Git/AnaliseDeAlgoritmos/Trabalho_Final/
        relatorio/Resultados/Insertion/tInsertion_vetor_aleatorio.dat")
114 plota_teste2("/home/gmarson/Git/AnaliseDeAlgoritmos/Trabalho_Final/
        relatorio/Resultados/Insertion/tInsertion_vetor_aleatorio.dat")
115 plota_teste3("/home/gmarson/Git/AnaliseDeAlgoritmos/Trabalho_Final/
        relatorio/Resultados/Insertion/tInsertion_vetor_aleatorio.dat")
116
117
118 def plota_teste4(arqsaida):
119     n, c, t = np.loadtxt(arqsaida, unpack=True)
120
121     # Calcula os coeficientes de um ajuste a um polinômio de grau 2 usando
122     # o método dos mínimos quadrados
123     coefs = np.polyfit(n, c, 2)
124     p = np.poly1d(coefs)
125
126     plt.plot(n, p(n), label='$n^2$')
127     plt.plot(n, c, 'ro', label='bubble sort')
128
129     # Posiciona a legenda
130     plt.legend(loc='upper left')
131
132     # Posiciona o título
133     plt.title('Análise da complexidade de \ntempo do método da bolha')
134
135     # Rotula os eixos
```

```
136     plt.xlabel('Tamanho do vetor (n)')
137     plt.ylabel('Número de comparações')
138 
139     plt.savefig('bubble4.png')
140     plt.show()
141 
142 def plota_teste5(arqsaida):
143     n, c, t = np.loadtxt(arqsaida, unpack=True)
144 
145     # Calcula os coeficientes de um ajuste a um polinômio de grau 2 usando
146     # o método dos mínimos quadrados
147     coefs = np.polyfit(n, c, 2)
148     p = np.poly1d(coefs)
149 
150     # set_yscale('log')
151     # set_yscale('log')
152     plt.semilogy(n, p(n), label='$n^2$')
153     plt.semilogy(n, c, 'ro', label='bubble sort')
154 
155     # Posiciona a legenda
156     plt.legend(loc='upper left')
157 
158     # Posiciona o título
159     plt.title('Análise da complexidade de \ntempo do método da bolha')
160 
161     # Rotula os eixos
162     plt.xlabel('Tamanho do vetor (n)')
163     plt.ylabel('Número de comparações')
164 
165     plt.savefig('bubble5.png')
166     plt.show()
```