Análise experimental de algoritmos usando Python

Karen Catiguá Junqueira
karen@ufu.com

Matheus Prado Prandini Faria
matheus_prandini@ufu.com

Pedro Augusto Correa Braz
pedro_acbraz@hotmail.com

Faculdade de Computação
Universidade Federal de Uberlândia

16 de dezembro de 2016

Lista de Figuras

Lista de Tabelas

Lista de Listagens

Sumário

Capítulo 1

Análise

O algoritmo insertion sort é considerado análogo ao processo de ordenar uma pilha de cartas usando as mãos. Nele, temos dois loops, de forma que o primeiro seja executado n vezes e o segundo $n-1$ vezes, resultando num total de $n*(n-1)$ repetições. Assim, podemos afirmar que o insertion sort é um algoritmo de tempo quadrático.

Como veremos com os resultados obtidos, o pior caso é representado por vetores ordenados de forma decrescente, resultando em um tempo $teta(n^2)$, além de $teta(n^2)$ comparações e um número de trocas linear, $teta(n)$. Já o melhor caso é representado quando o vetor já está ordenado em ordem crescente, resultando em um tempo linear, $teta(n)$, além de um número linear de comparações, $teta(n)$, e nenhuma troca, pois os elementos já estão em seus devidos lugares. Além disso, são obtidos bons resultados quando os vetores a serem ordenados são pequenos ou já estão quase ordenados.

Capítulo 2

Resultados

2.1 Tabelas

n	comparações	tempo(s)
32	312	0.000668
64	1157	0.002129
128	4265	0.007484
256	16706	0.030135
512	65277	0.111077
1024	260380	0.524285
2048	1022864	1.814850
4096	4286253	7.827180
8192	16866997	29.855800

Tabela 2.1: *InsertionSort com Vetores Aleatorio*

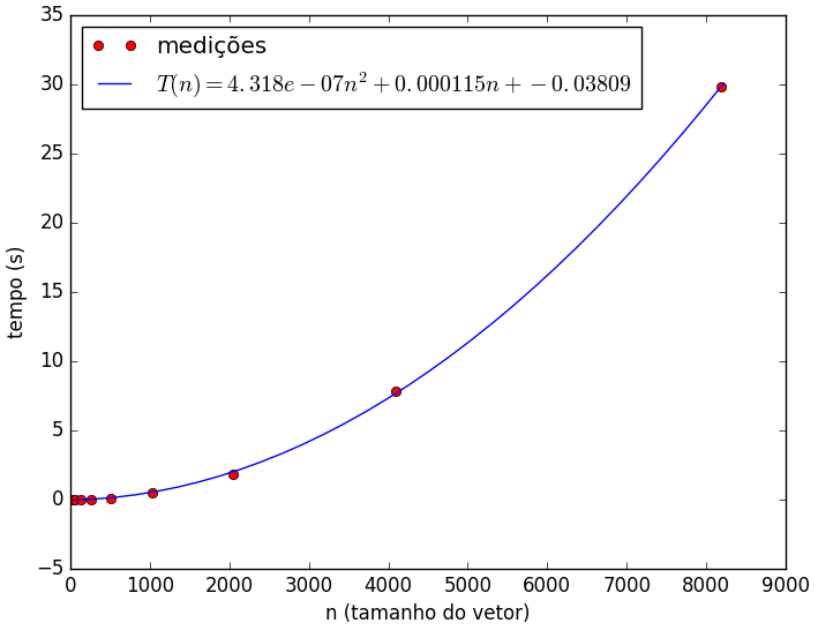

Figura 2.1: *Insertion com relação ao tempo com vetor aleatório.*

Análise com relação ao tamanho do vetor 2^{32}:

$4.318 * 10^{-7} * n^2 + 0.000115 * n - 0.03809 = 7.965304e + 12$

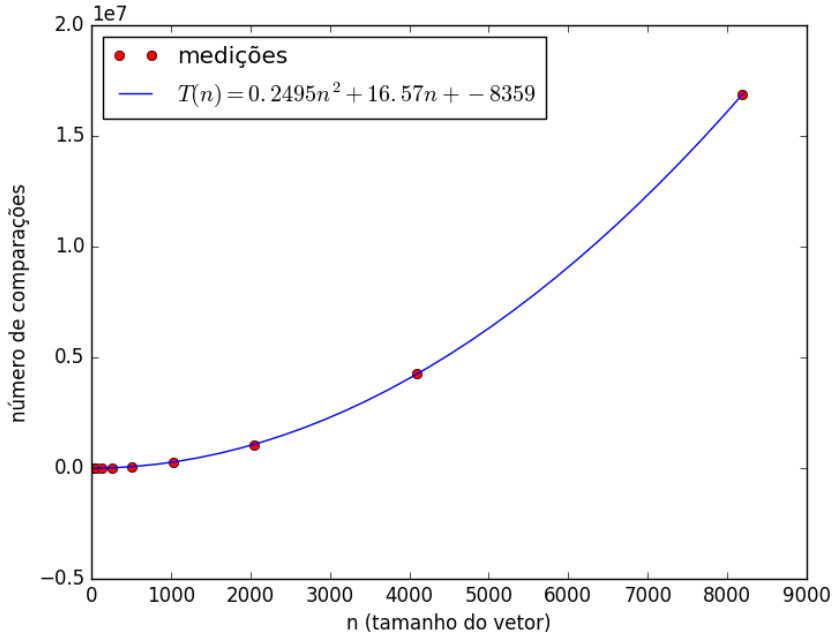

Figura 2.2: *Insertion com relação ao número de comparações com vetor aleatório.*

Análise com relação ao tamanho do vetor 2^{32}:

$0.2495 * n^2 + 16.57 * n - 8359 = 4.6024627e + 18$

n	comparações	tempo(s)
32	31	0.000092
64	63	0.000184
128	127	0.000366
256	255	0.000720
512	511	0.001510
1024	1023	0.002773
2048	2047	0.005270
4096	4095	0.010834
8192	8191	0.024804

Tabela 2.2: *InsertionSort com Vetores Cescentes*

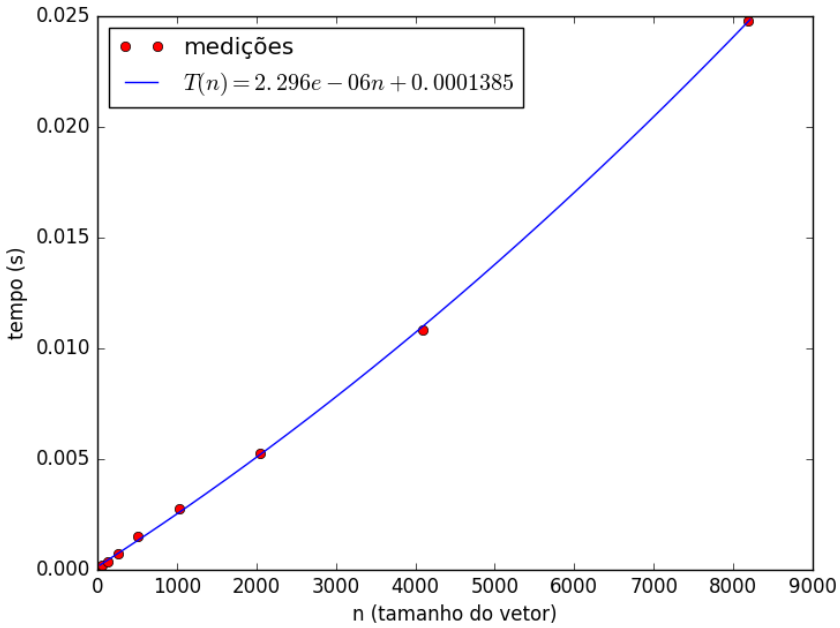

Figura 2.3: *Insertion com relação ao tempo com vetor crescente.*

Análise com relação ao tamanho do vetor 2^{32}:

$2.296 * 10^{-6} * n - 0.0001385 = 9861.244912$

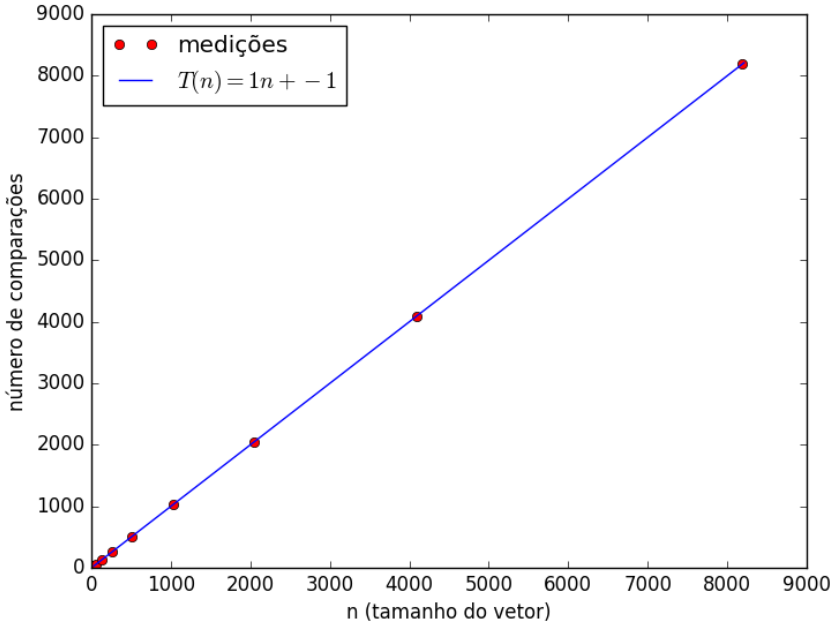

Figura 2.4: *Insertion com relação ao número de comparações com vetor crescente.*

Análise com relação ao tamanho do vetor 2^{32}:

1

n	comparações	tempo(s)
32	527	0.001065
64	2079	0.003999
128	8255	0.014620
256	32895	0.057867
512	131327	0.234727
1024	524798	0.829646
2048	2098128	3.714570
4096	8388463	15.547900
8192	33545751	57.902200

Tabela 2.3: *InsertionSort com Vetores Decrescentes*

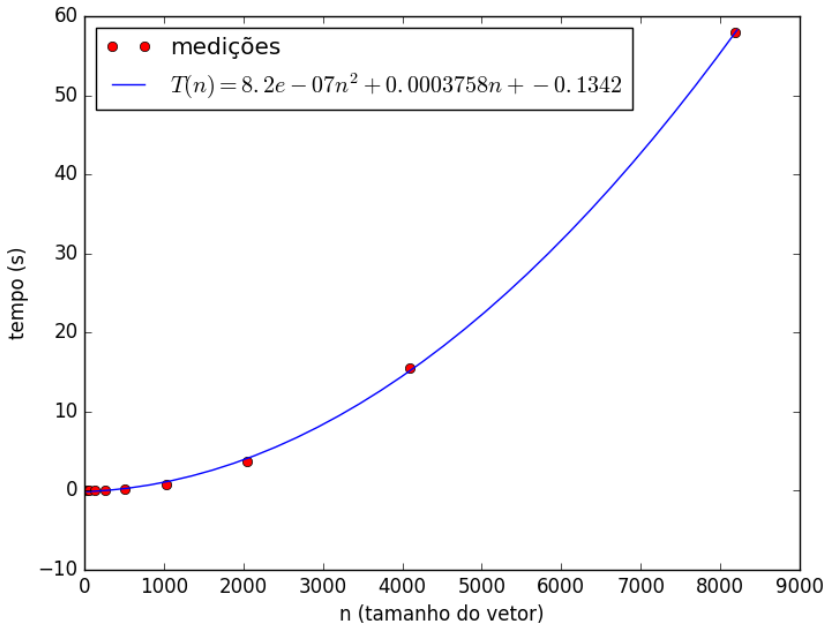

Figura 2.5: *Insertion com relação ao tempo com vetor decrescente.*

Análise com relação ao tamanho do vetor 2^{32}:

$8.2 * 10^{-7} * n^2 + 0.0003758 * n - 0.1342 = 1.512633e + 13$

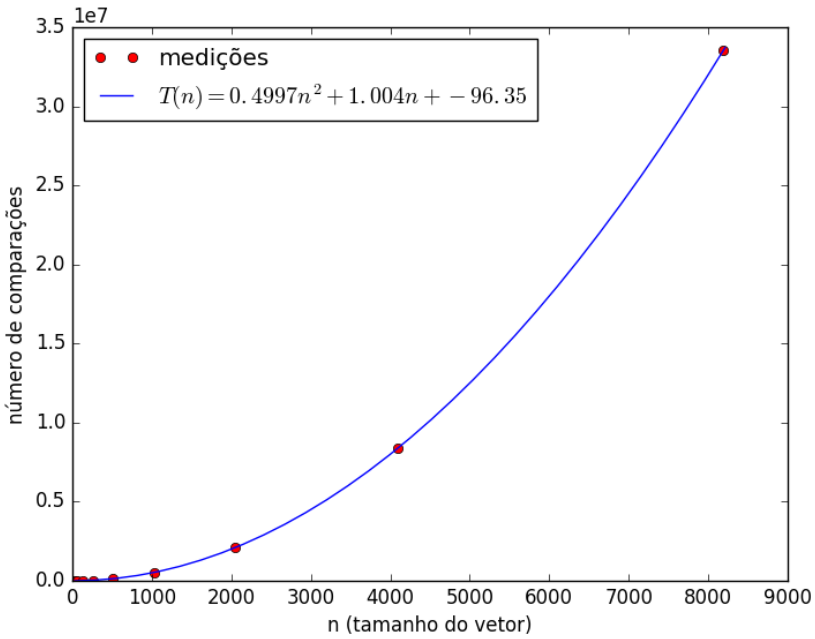

Figura 2.6: *Insertion com relação ao número de comparações com vetor decrescente.*

Análise com relação ao tamanho do vetor 2^{32}:

$0.4997 * n^2 + 1.004 * n - 96.35 = 9.2178380e + 18$

n	comparações	tempo(s)
32	32	0.000104
64	66	0.000188
128	162	0.000429
256	374	0.000863
512	1093	0.002411
1024	2983	0.006062
2048	11449	0.023471
4096	40845	0.076961
8192	164101	0.311340

Tabela 2.4: *InsertionSort com Vetores Quase Crescentes* 10%

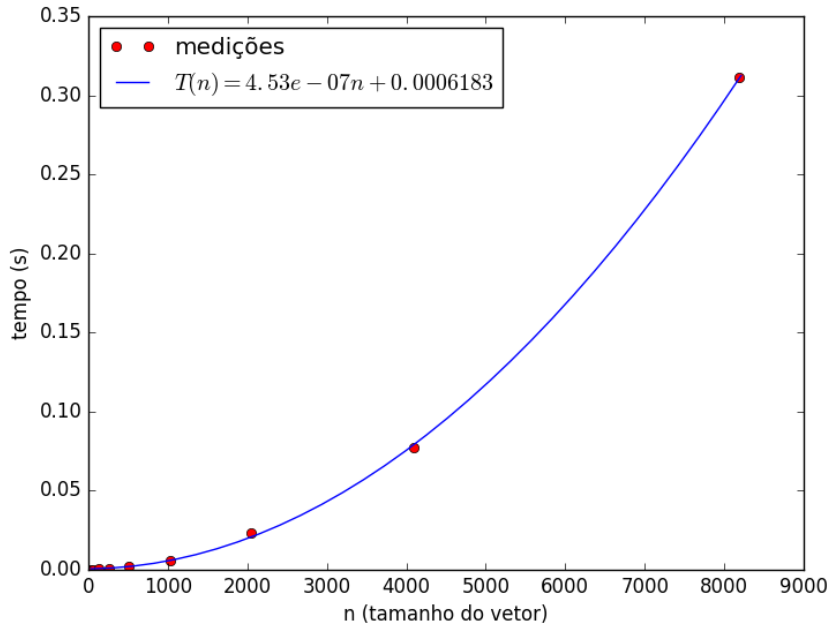

Figura 2.7: *insertion com relação ao tempo com vetor quase crescente* 10%.

Análise com relação ao tamanho do vetor 2^{32}:

$4.53 * 10^{-7} * n + 0.0006183 = 1945.620185$

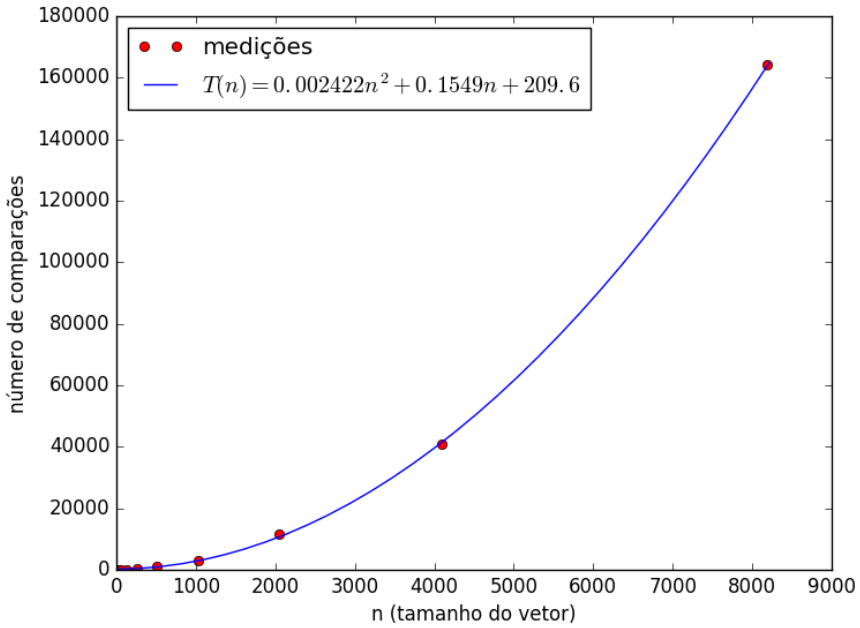

Figura 2.8: *Insertion com relação ao número de comparações com vetor quase crescente* 10%.

Análise com relação ao tamanho do vetor 2^{32}:

$0.002422 * n^2 + 0.1549 * n + 209.6 = 4.467801e + 16$

n	comparações	tempo(s)
32	36	0.000113
64	94	0.000247
128	261	0.000597

Tabela 2.5: *InsertionSort com Vetores Quase Crescentes* 20%

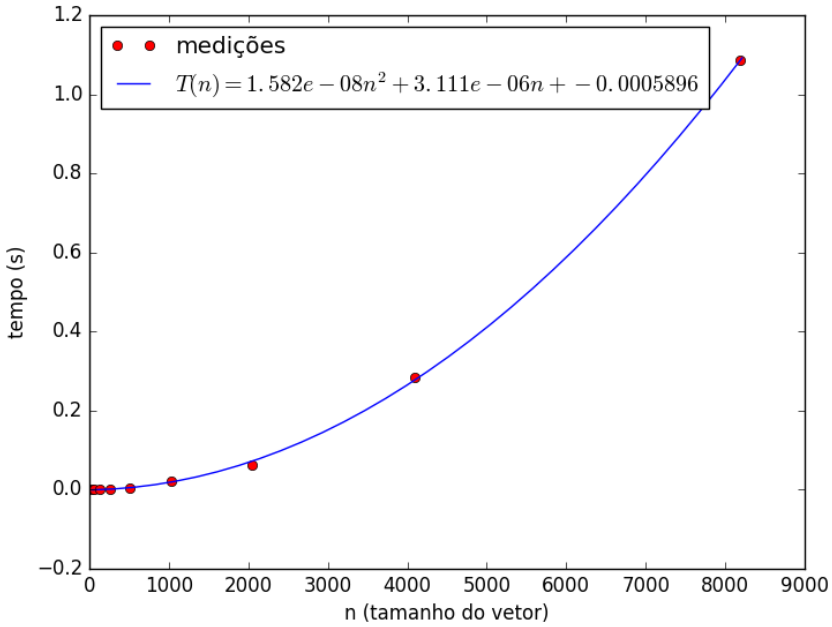

Figura 2.9: *Insertion com relação ao tempo com vetor quase crescente* 20%.

Análise com relação ao tamanho do vetor 2^{32}:

$1.582 * 10^{-8} * n^2 + 3.111 * 10^{-6} * n - 0.0005896 = 2.9182750e + 11$

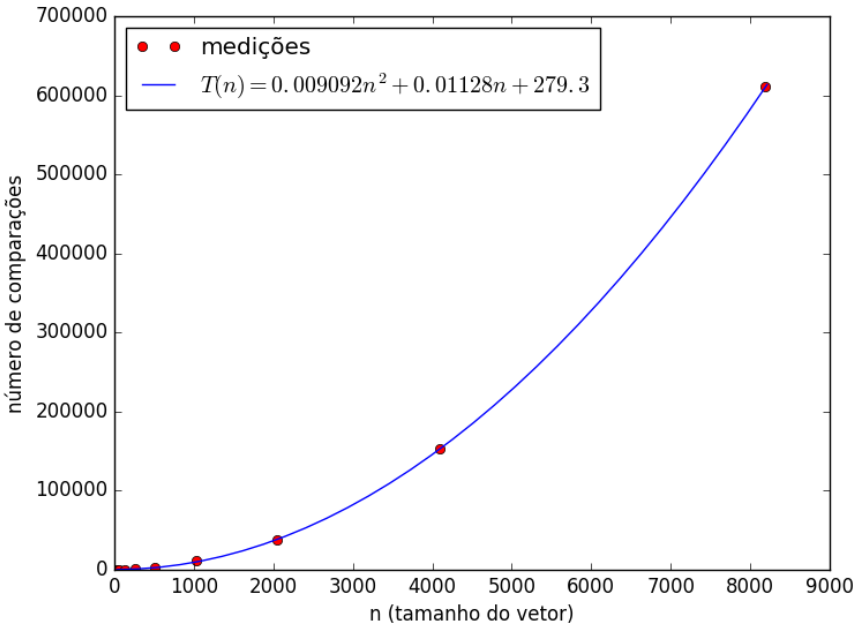

Figura 2.10: *Insertion com relação ao número de comparações com vetor quase crescente* 20%.

Análise com relação ao tamanho do vetor 2^{32}:

$0.009092 * n^2 + 0.01128 * n + 279.3 = 1.6771779e + 17$

n	comparações	tempo(s)
32	45	0.000139
64	147	0.000308
128	417	0.000900

Tabela 2.6: *InsertionSort com Vetores Quase Crescentes* 30%

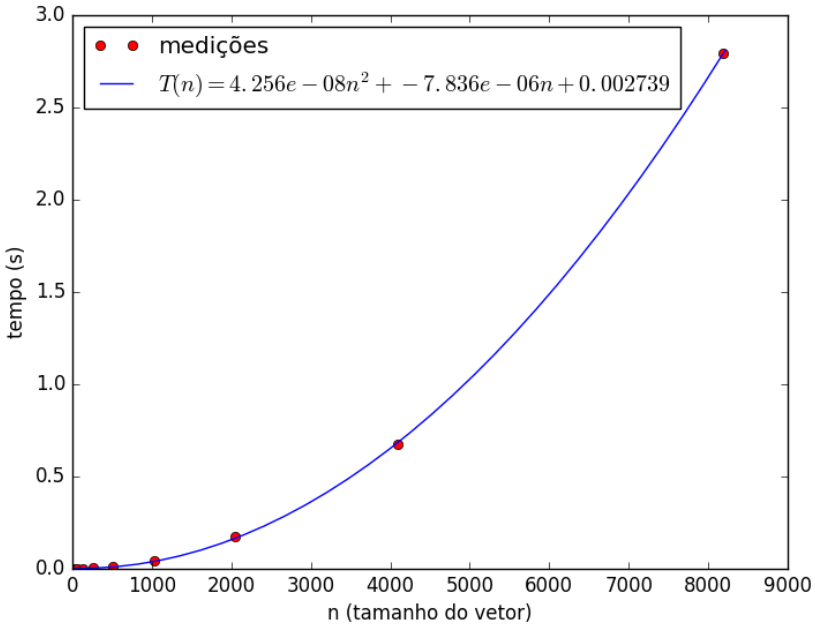

Figura 2.11: *Insertion com relação ao tempo com vetor quase crescente* 30%.

Análise com relação ao tamanho do vetor 2^{32}:

$4.2456 * 10^{-8} * n^2 - 7.836 * 10^{-6} * n + 0.002739 = 7.831749e + 11$

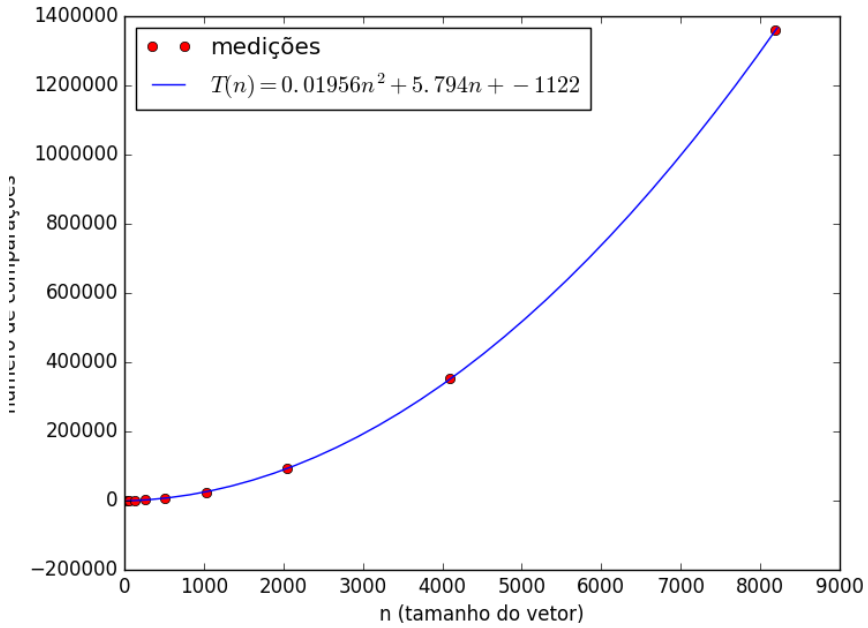

Figura 2.12: *Insertion com relação ao número de comparações com vetor quase crescente* 30%.

Análise com relação ao tamanho do vetor 2^{32}:

$0.01956 * n^2 + 5.794 * n - 1122 = 3.60818339e + 17$

n	comparações	tempo(s)
32	72	0.000199
64	200	0.000460
128	741	0.001348

Tabela 2.7: *InsertionSort com Vetores Quase Crescentes* 40%

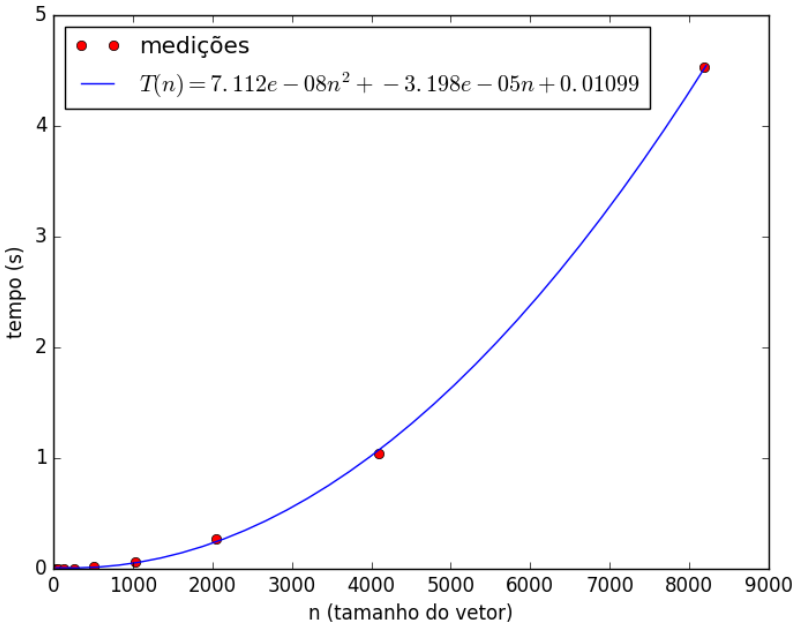

Figura 2.13: *Insertion com relação ao tempo com vetor quase crescente* 40%.

Análise com relação ao tamanho do vetor 2^{32}:

$7.112 * 10^{-8} * n^2 - 3.198 * 10^{-5} * n + 0.01099 = 1.3119323e + 12$

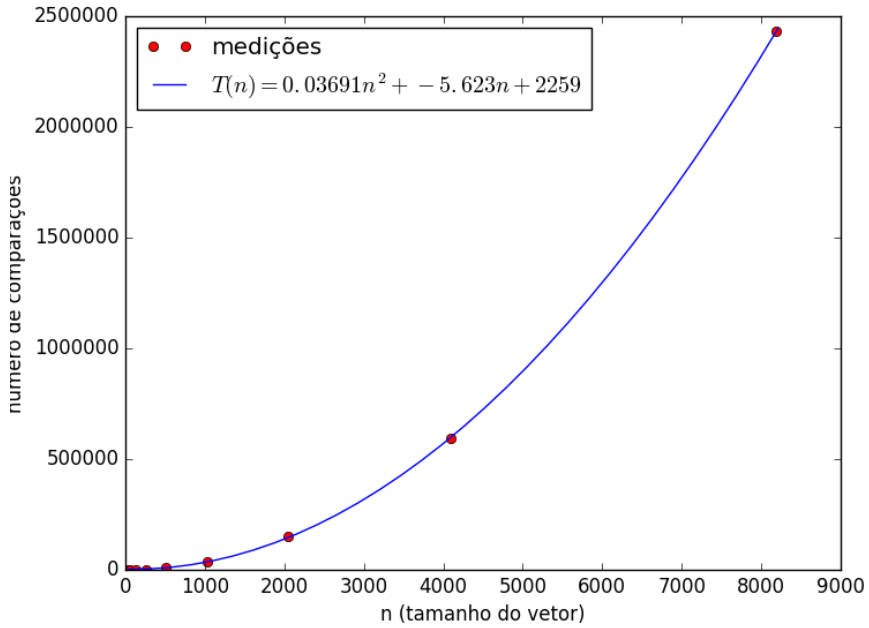

Figura 2.14: *Insertion com relação ao número de comparações com vetor quase crescente* 40%.

Análise com relação ao tamanho do vetor 2^{32}:

$0.03691 * n^2 - 5.623 * n - 2259 = 6.808692e + 17$

n	comparações	tempo(s)
32	89	0.000214
64	343	0.000689
128	1154	0.002201

Tabela 2.8: *InsertionSort com Vetores Quase Crescentes* 50%

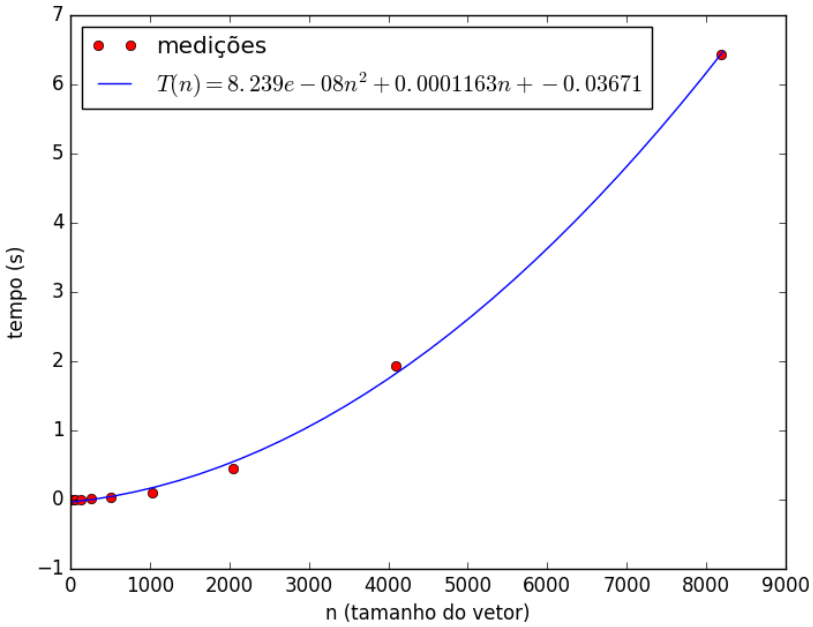

Figura 2.15: *Insertion com relação ao tempo com vetor quase crescente* 50%.

Análise com relação ao tamanho do vetor 2^{32}:

$8.239 * 10^{-8} * n^2 - 0.0001163 * n - 0.03671 = 1.5198267e + 12$

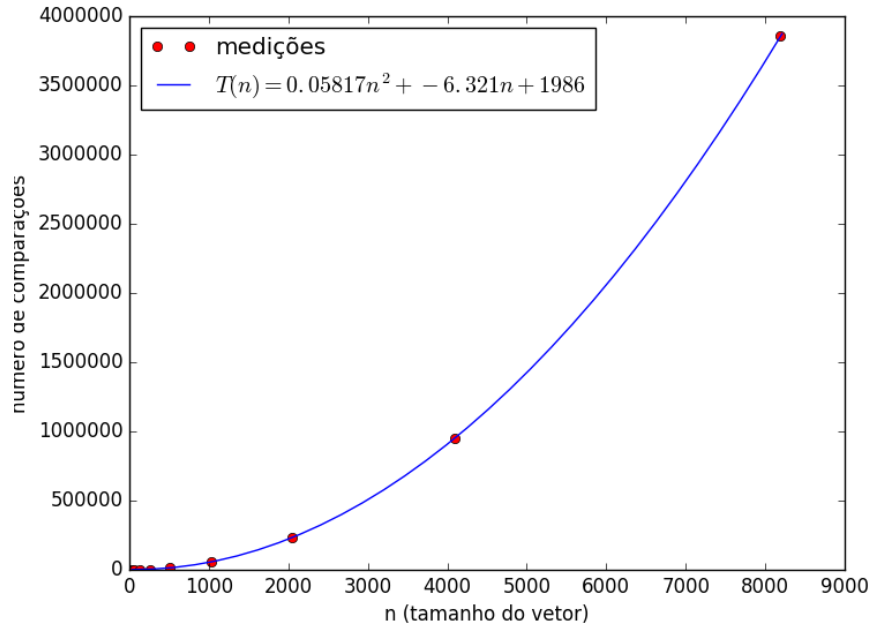

Figura 2.16: *Insertion com relação ao número de comparações com vetor quase crescente* 50%.

Análise com relação ao tamanho do vetor 2^{32}:

$0.05817 * n^2 - 6.321 * n + 1986 = 1.073047076e + 18$

n	comparações	tempo(s)
32	525	0.001095
64	2066	0.003612
128	8218	0.015600

Tabela 2.9: *InsertionSort com Vetores Quase Decrescentes* 10%

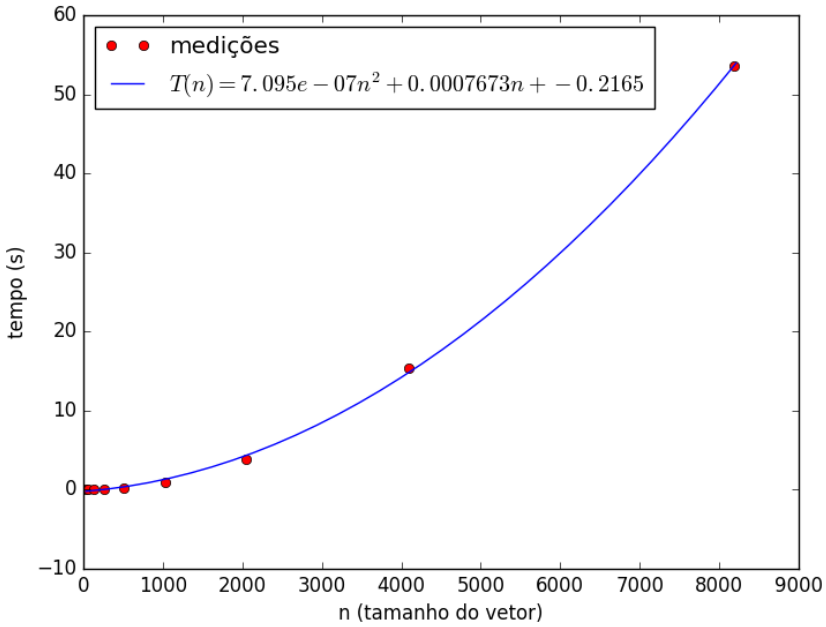

Figura 2.17: *Insertion com relação ao tempo com vetor quase decrescente* 10%.

Análise com relação ao tamanho do vetor 2^{32}:

$7.095 * 10^{-7} * n^2 - 0.0007673 * n - 0.2165 = 1.3087961e + 13$

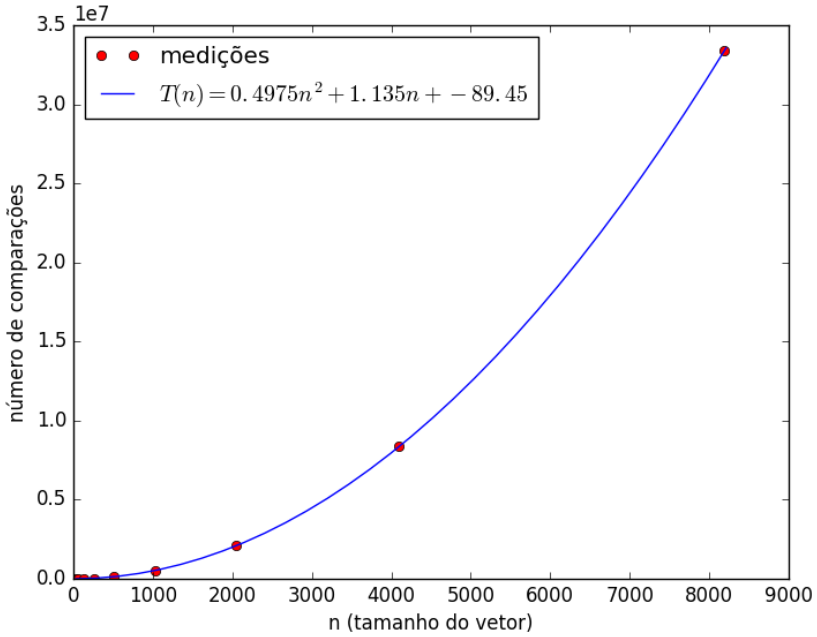

Figura 2.18: *Insertion com relação ao número de comparações com vetor quase decrescente* 10%.

Análise com relação ao tamanho do vetor 2^{32}:

$0.04975 * n^2 - 1.135 * n - 89.45 = 9.1772551288e + 17$

n	comparações	tempo(s)
32	517	0.001060
64	2040	0.003877
128	8135	0.015449

Tabela 2.10: *InsertionSort com Vetores Quase Decrescentes* 20%

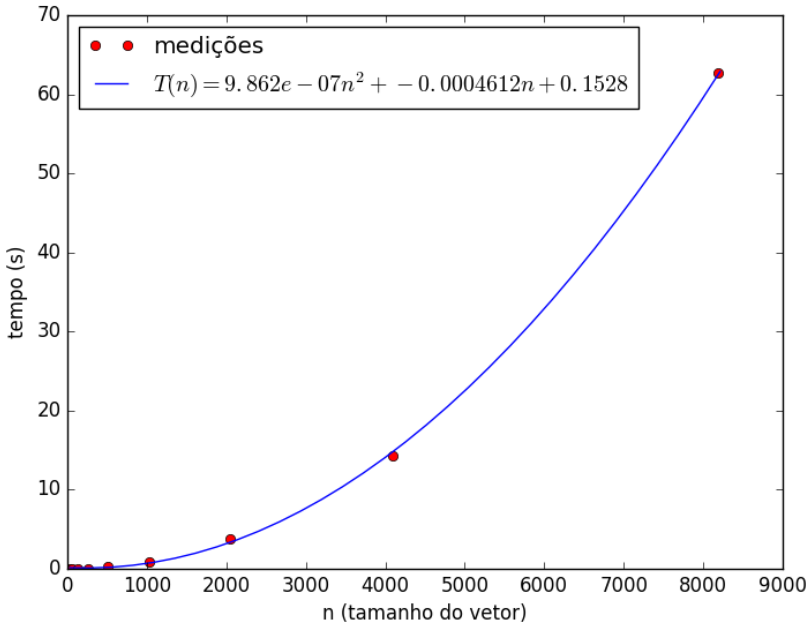

Figura 2.19: *Insertion com relação ao tempo com vetor quase decrescente* 20%.

Análise com relação ao tamanho do vetor 2^{32}:

$9.862 * 10^{-7} * n^2 - 0.0004612 * n + 0.1528 = 1.8192177e + 13$

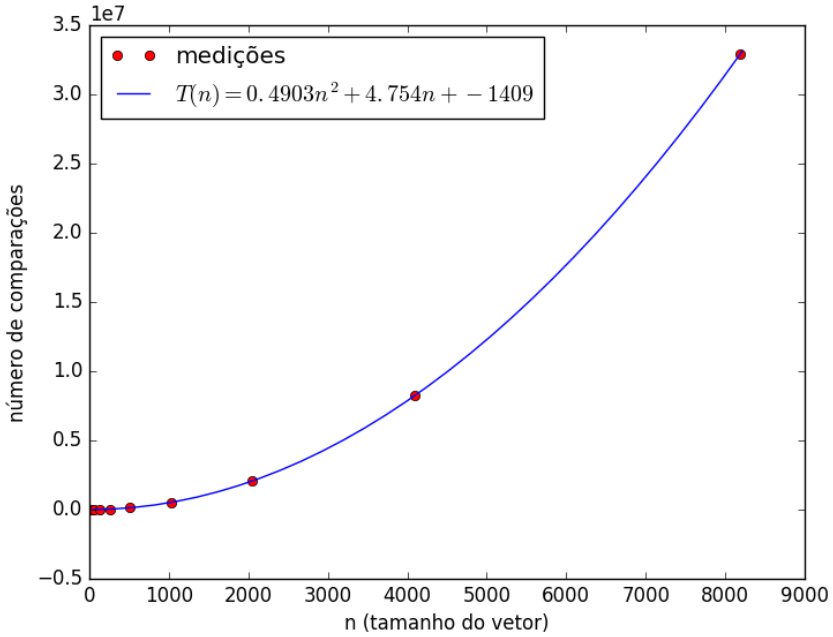

Figura 2.20: *Insertion com relação ao número de comparações com vetor quase decrescente* 20%.

Análise com relação ao tamanho do vetor 2^{32}:

$0.04903 * n^2 - 4.754 * n - 1409 = 9.044438e + 17$

n	comparações	tempo(s)
32	503	0.000977
64	1967	0.003335
128	8008	0.014160
256	31412	0.059297
512	126024	0.215075
1024	501143	0.873110
2048	2011066	3.613190
4096	8039701	14.242200
8192	32184774	60.094500

Tabela 2.11: *InsertionSort com Vetores Quase Decrescentes* 30%

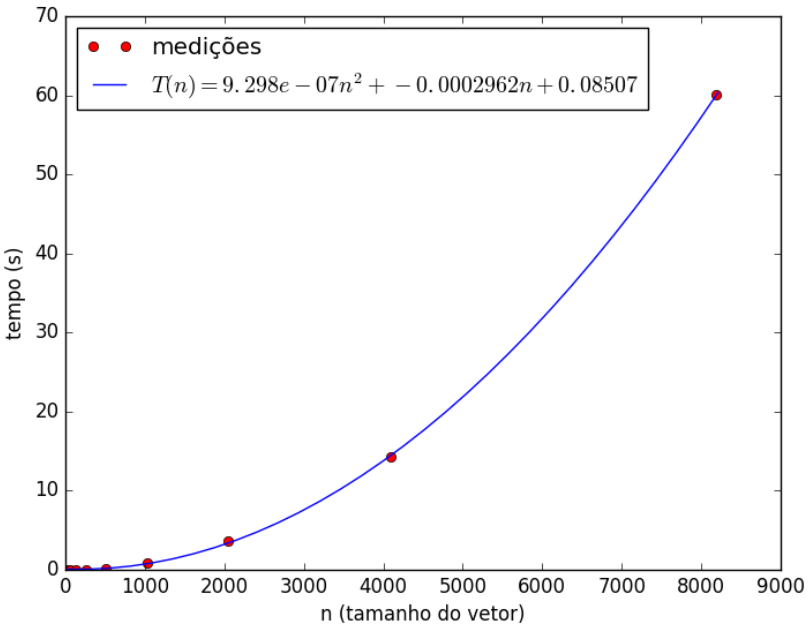

Figura 2.21: *Insertion com relação ao tempo com vetor quase decrescente* 30%.

Análise com relação ao tamanho do vetor 2^{32}:

$9.298 * 10^{-7} * n^2 - 0.0002962 * n + 0.08507 = 1.715178137e + 13$

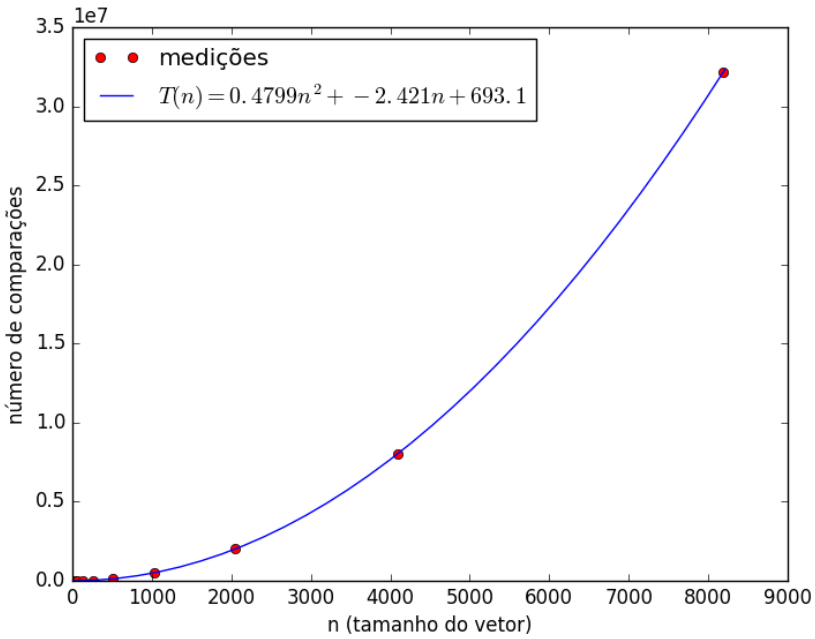

Figura 2.22: *Insertion com relação ao número de comparações com vetor quase decrescente* 30%.

Análise com relação ao tamanho do vetor 2^{32}:

$0.04799 * n^2 - 2.421 * n + 693.1 = 8.852492385e + 17$

n	comparações	tempo(s)
32	502	0.000957
64	1929	0.003514
128	7514	0.014810

Tabela 2.12: *InsertionSort com Vetores Quase Decrescentes* 40%

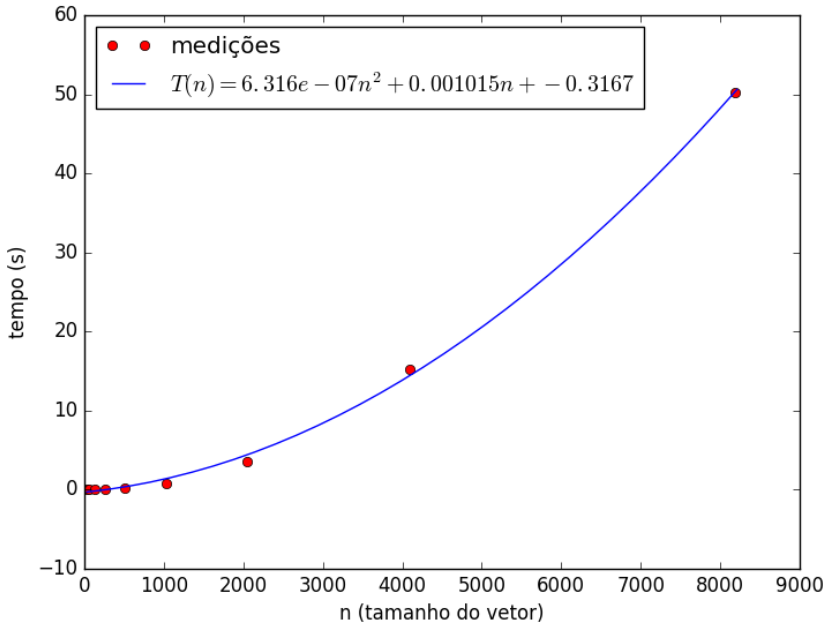

Figura 2.23: *Insertion com relação ao tempo com vetor quase decrescente* 40%.

Análise com relação ao tamanho do vetor 2^{32}:

$6.316 * 10^{-7} * n^2 + 0.001015 * n - 0.3167 = 1.1650967e + 13$

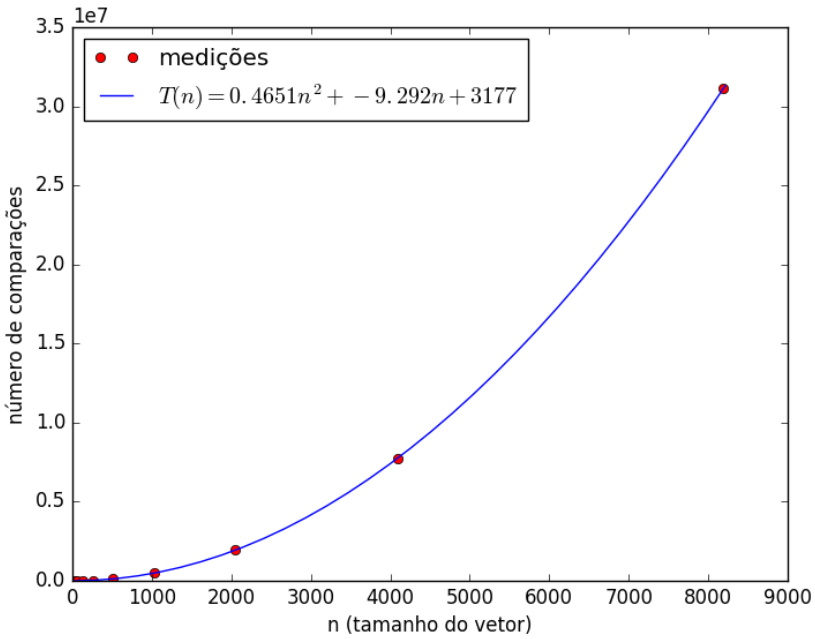

Figura 2.24: *Insertion com relação ao número de comparações com vetor quase decrescente* 40%.

Análise com relação ao tamanho do vetor 2^{32}:

$0.04651 * n^2 - 9.292 * n + 3177 = 8.579580274e + 17$

n	comparações	tempo(s)
32	481	0.000875
64	1819	0.003601
128	7261	0.013067
256	29328	0.052265
512	116412	0.199282
1024	462305	0.932767
2048	1872133	3.247850
4096	7463655	13.643700
8192	29858237	54.944500

Tabela 2.13: *InsertionSort com Vetores Quase Decrescentes* 50%

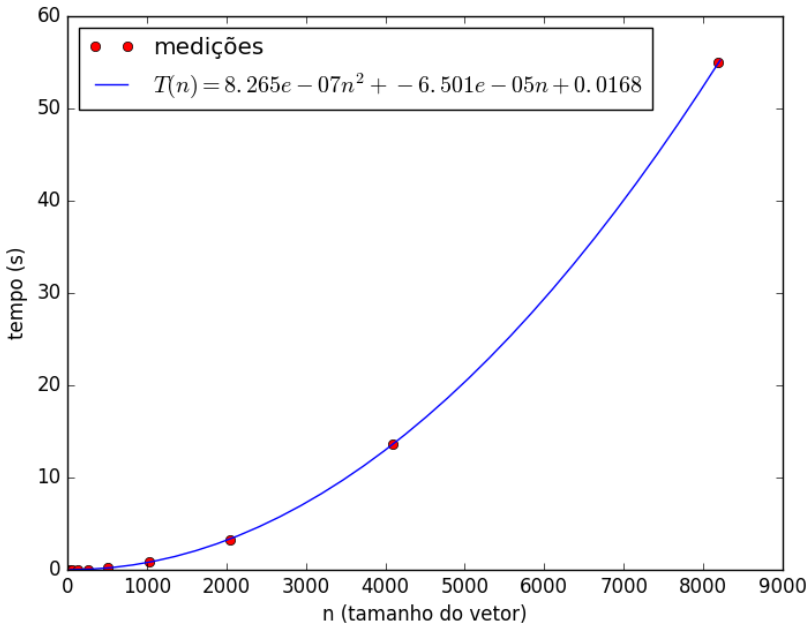

Figura 2.25: *Insertion com relação ao tempo com vetor quase decrescente* 50%.

Análise com relação ao tamanho do vetor 2^{32}:

$8.265 * 10^{-7} * n^2 - 6.501 * 10^{-5} * n + 0.0168 = 1.52462337e + 13$

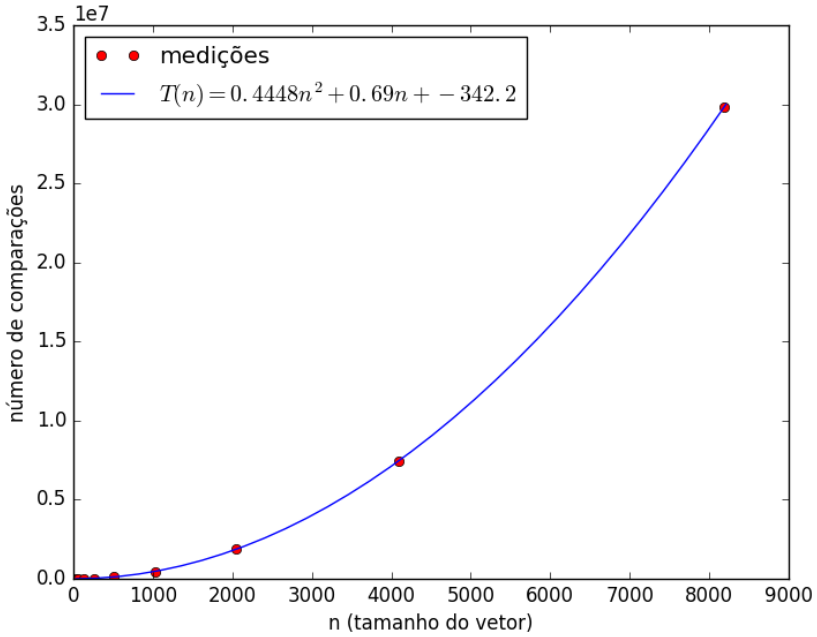

Figura 2.26: *Insertion com relação ao número de comparações com vetor quase decrescente* 50%.

Análise com relação ao tamanho do vetor 2^{32}:

$0.04448 * n^2 + 0.69 * n - 342.2 = 8.205111794e + 17$

Apêndice A

Arquivo ../insertion/insertion.py

Listagem A.1: ../insertion/insertion.py

```
1 import numpy as np
2
3
4 @profile
5 def insertionSort(lista):
6   for j in range(1,len(lista)):
7     chave = lista[j]
8     i = j
9     while (i>0 and lista[i-1]>chave):
10      lista[i] = lista[i-1]
11      i = i-1
12    lista[i] = chave
```

Apêndice B

Arquivo ../insertion/ensaio.py

Listagem B.1: ../insertion/ensaio.py

```
1  import numpy as np
2  import argparse
3
4  from insertion import *
5
6  parser = argparse.ArgumentParser()
7  parser.add_argument("arq_vetor",
8                      help="nome do arquivo contendo o vetor de teste")
9  args = parser.parse_args()
10
11 # Lê o arquivo contendo o vetor e passado na linha de comando como um
12 # vetor do Numpy.
13
14 vet = np.loadtxt(args.arq_vetor)
15 insertionSort(vet)
```