# 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 Merge Sort baseia-se no paradigma de divisão e conquista, pois divide o vetor em dois subvetores de tamanho igual, metade do vetor passado, ou com diferença de um entre eles até que alcance o caso trivial, vetor com apenas um elemento. Depois ordena os dois vetores recursivamente utilizando o Merge Sort. E, por fim, utiliza a subrotina Intercala de modo a combinar os dois subvetores gerados e obter um vetor ordenado.

Dessa forma, temos que o algoritmo realiz a divisão da metade dos elementos do vetor passado até que alcance o caso trivial,vetor com apenas um elemento. tem custo $teta(1)$, a conquista tem custo $piso(n/2) + teto(n/2)$ e a combinação tem custo linear, $teta(n)$. Assim, temos a seguinte recorrência representada por $T(n)$:

$T(n) = T(piso(n/2)) + T(teto(n/2)) + teta(n) + teta(1)$

Temos que resolvendo a recorrência, a complexidade de tempo do Merge Sort em todos os casos é $teta(n * lgn)$, pois independente do vetor de entrada, o algoritmo realiza a divisão do vetor até alcançar o caso base, para depois combinar os elemento a partir da subrotina Intercala.

Em termos de memória, o Merge Sort necessita de espaço linear devido ao fato da subrotina Intercala demandar a criação de um vetor axiliar do tamanho do vetor original a fim de combinar os elementos dos subvetores gerados. Assim, a complexidade de espaço do algoritmo é $teta(n)$.

# Capítulo 2

# Resultados

## 2.1 Tabelas

| n | comparações | tempo(s) |
|---|---|---|
| 32 | 31 | 0.000564 |
| 64 | 63 | 0.001187 |
| 128 | 127 | 0.002625 |
| 256 | 255 | 0.005897 |
| 512 | 511 | 0.012599 |
| 1024 | 1023 | 0.027598 |
| 2048 | 2047 | 0.063681 |
| 4096 | 4095 | 0.161993 |
| 8192 | 8191 | 0.414297 |

**Tabela 2.1:** *MergeSort com Vetores Aleatorio*

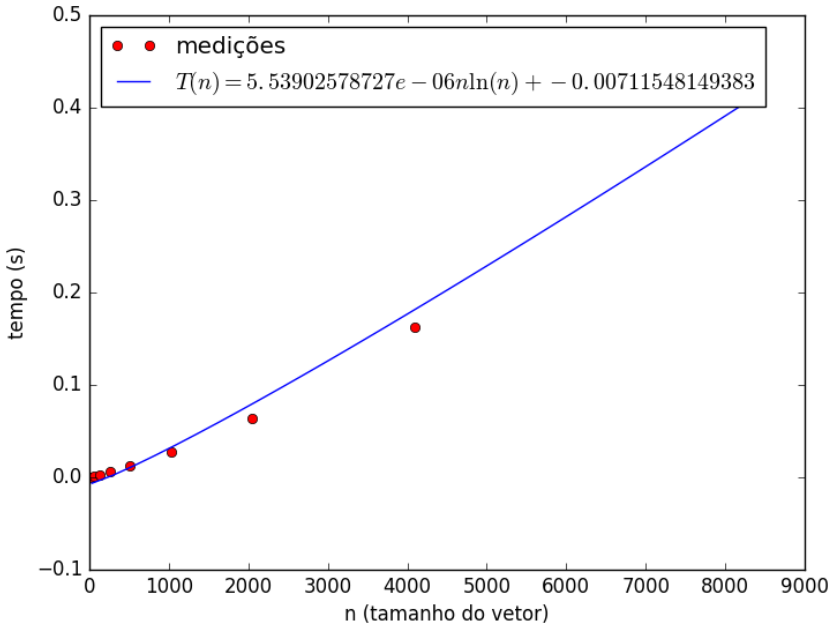


**Figura 2.1:** *Mergesort com relação ao tempo com vetor aleatório.*

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

$5.53902578727 * 10^{-6} * n * ln(n) - 0.00711548149383 = 551281.28171$

2.1

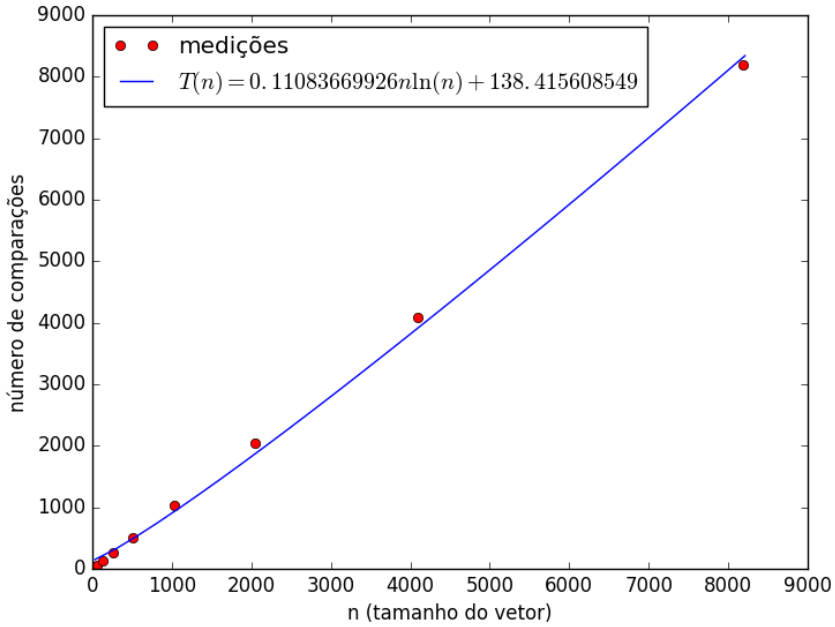


**Figura 2.2:** *Mergesort com relação ao número de comparações com vetor aleatório.*

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

$0.1108367 * n * ln(n) + 138.415608549 = 10558905163$

| n | comparações | tempo(s) |
|---|---|---|
| 32 | 31 | 0.000599 |
| 64 | 63 | 0.001224 |
| 128 | 127 | 0.002711 |
| 256 | 255 | 0.005734 |
| 512 | 511 | 0.012676 |
| 1024 | 1023 | 0.027952 |
| 2048 | 2047 | 0.063704 |
| 4096 | 4095 | 0.157437 |
| 8192 | 8191 | 0.411806 |

**Tabela 2.2:** *MergeSort com Vetores Crescentes*

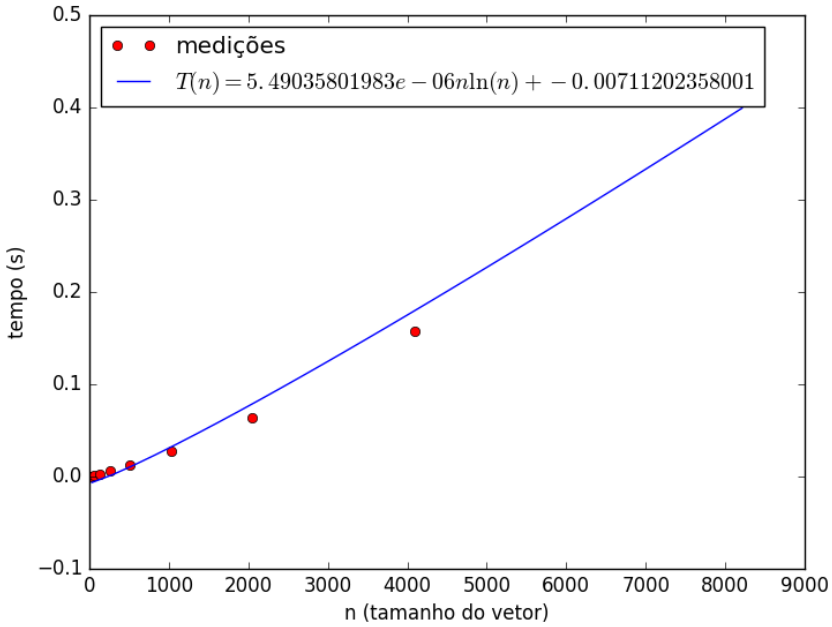


**Figura 2.3:** *Mergesort com relação ao tempo com vetor crescente.*

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

$5.49035801983 * 10^{-6} * n * ln(n) - 0.00711202358001 = 548274.78192$

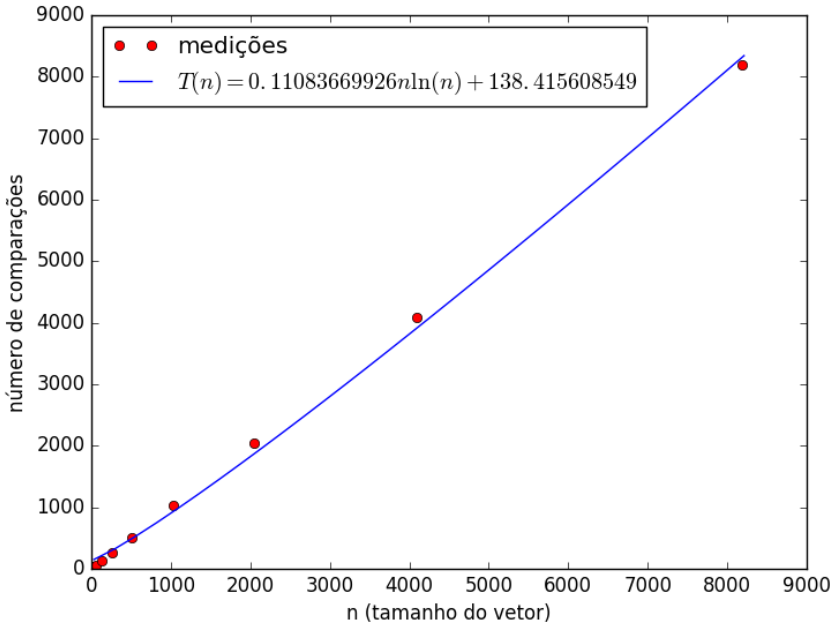


**Figura 2.4:** *Mergesort com relação ao número de comparações com vetor crescente.*

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

$0.1108367 * n * ln(n) + 138.415608549 = 10558905163$

| n | comparações | tempo(s) |
|---|---|---|
| 32 | 31 | 0.000609 |
| 64 | 63 | 0.001249 |
| 128 | 127 | 0.002686 |
| 256 | 255 | 0.005552 |
| 512 | 511 | 0.012529 |
| 1024 | 1023 | 0.029065 |
| 2048 | 2047 | 0.062935 |
| 4096 | 4095 | 0.153579 |
| 8192 | 8191 | 0.414567 |

**Tabela 2.3:** *MergeSort com Vetores Decrescentes*

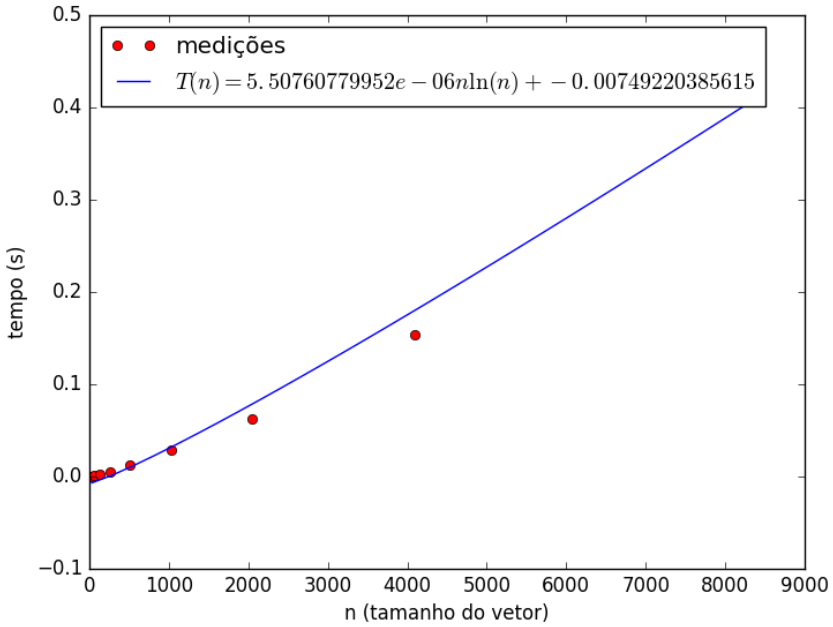


**Figura 2.5:** *Mergesort com relação ao tempo com vetor decrescente.*

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

$5.50760779952 * 10^{-6} * n * ln(n) - 0.00749220385615 = 549551.32148$

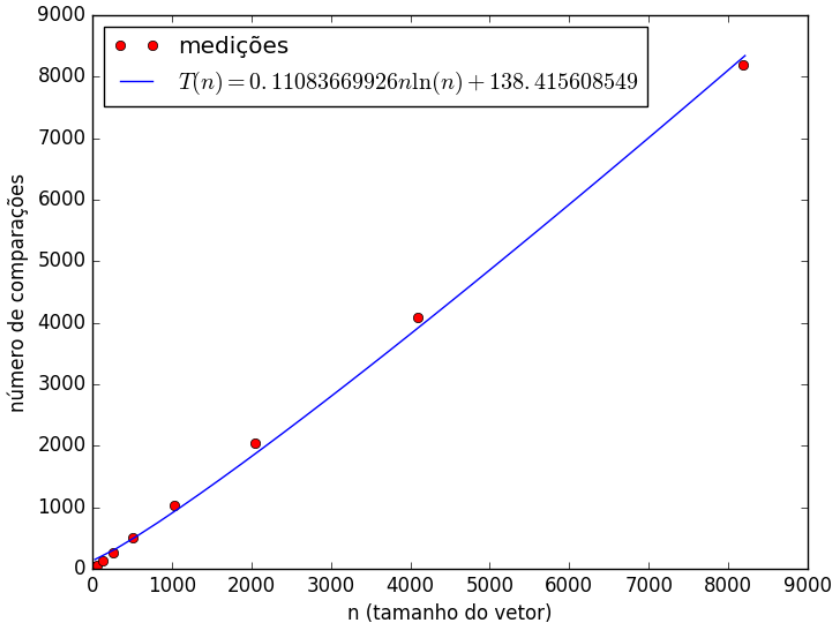


**Figura 2.6:** *Mergesort com relação ao número de comparações com vetor decrescente.*

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

$0.1108367 * n * ln(n) + 138.415608549 = 10558905163$

| n | comparações | tempo(s) |
|---|---|---|
| 32 | 31 | 0.000558 |
| 64 | 63 | 0.001275 |
| 128 | 127 | 0.002606 |

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

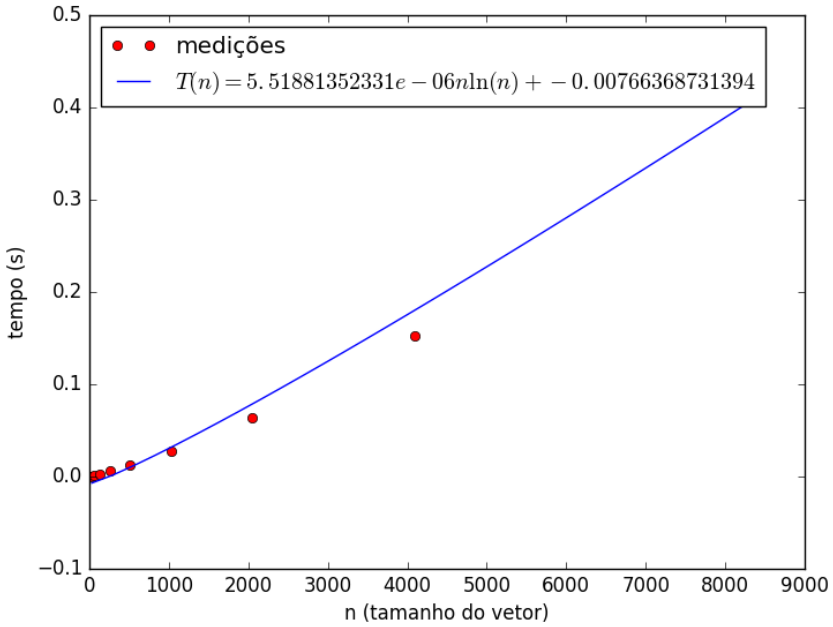


**Figura 2.7:** *mergesort com relação ao tempo com vetor quase crescente* 10%.

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

$5.51881352331 * 10^{-6} * n * ln(n) - 0.00766368731394 = 55122.29347$

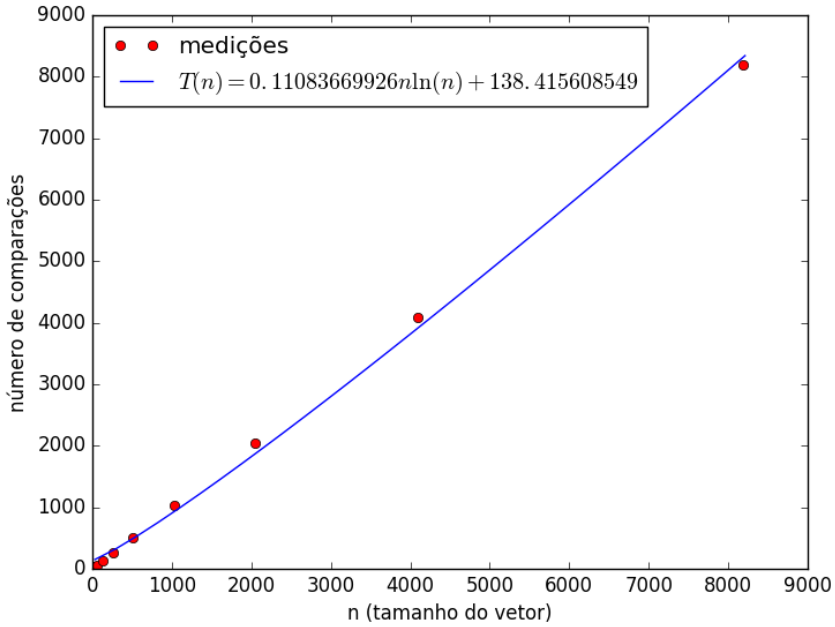


**Figura 2.8:** *Mergesort com relação ao número de comparações com vetor quase crescente* 10%.

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

$0.1108367 * n * ln(n) + 138.415608549 = 10558905163$

| n | comparações | tempo(s) |
|---|---|---|
| 32 | 31 | 0.000575 |
| 64 | 63 | 0.001226 |
| 128 | 127 | 0.002614 |
| 256 | 255 | 0.005683 |
| 512 | 511 | 0.012552 |
| 1024 | 1023 | 0.027633 |
| 2048 | 2047 | 0.063362 |
| 4096 | 4095 | 0.161215 |
| 8192 | 8191 | 0.407783 |

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

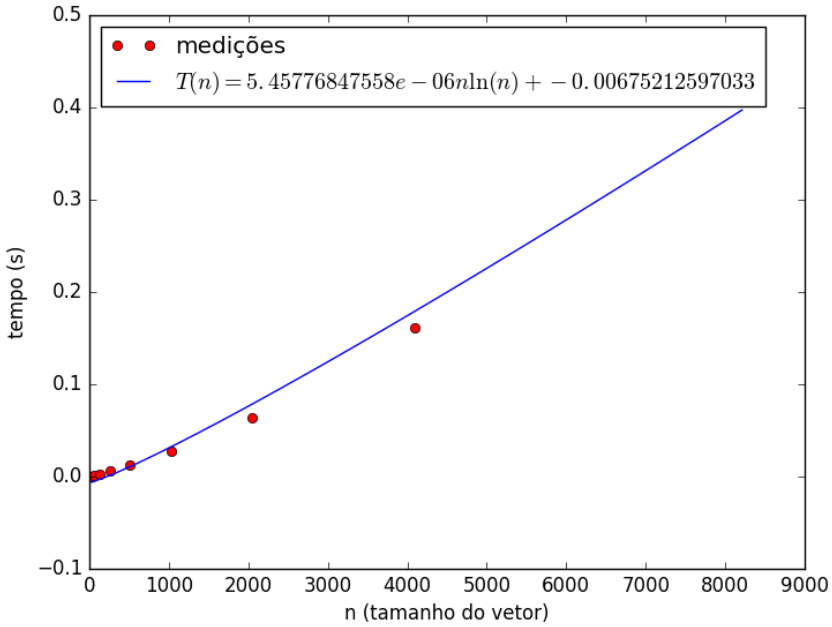


**Figura 2.9:** *Mergesort com relação ao tempo com vetor quase crescente* 20%.

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

$5.45776847558 * 10^{-6} * n * ln(n) - 0.00675212597033 = 54298.27160$

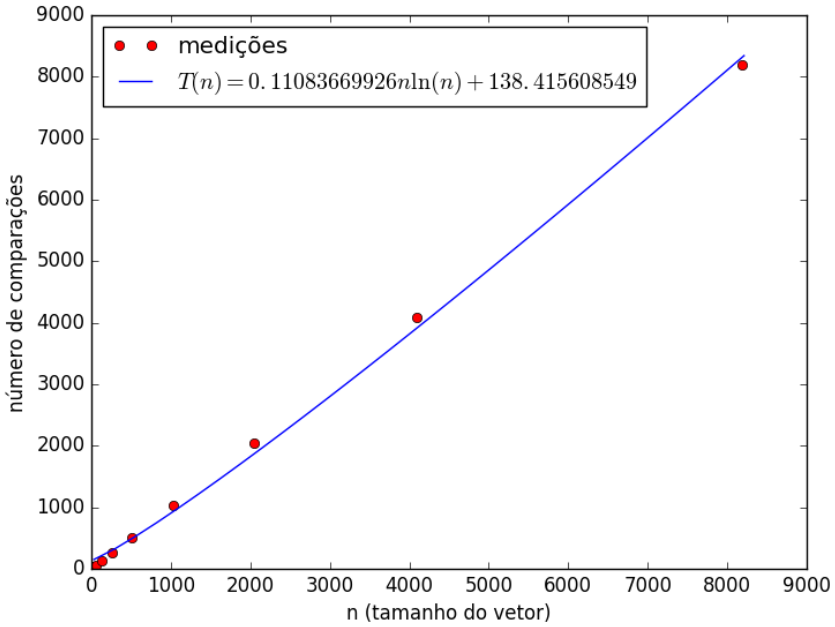


**Figura 2.10:** *Mergesort com relação ao número de comparações com vetor quase crescente* 20%.

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

$0.1108367 * n * ln(n) + 138.415608549 = 10558905163$

| n | comparações | tempo(s) |
|---|---|---|
| 32 | 31 | 0.000543 |
| 64 | 63 | 0.001269 |
| 128 | 127 | 0.002676 |

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

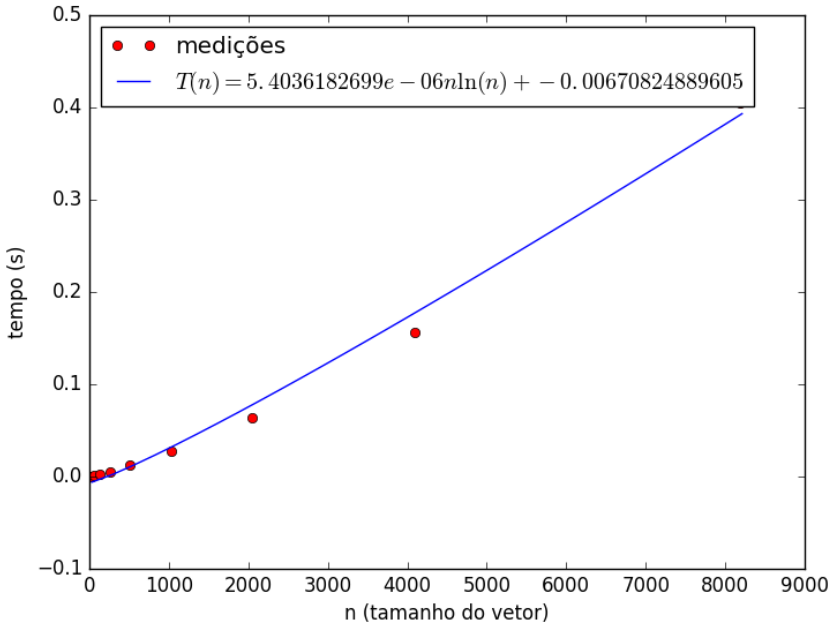


**Figura 2.11:** *Mergesort com relação ao tempo com vetor quase crescente* 30%.

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

$5.4036182699 * 10^{-6} * n * ln(n) - 0.00670824889605 = 53918.38192$

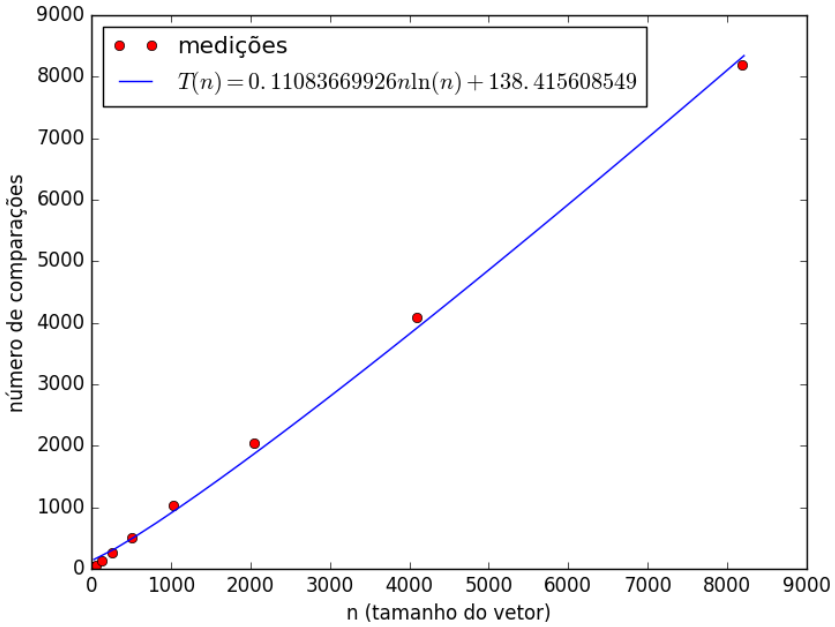


**Figura 2.12:** *Mergesort com relação ao número de comparações com vetor quase crescente* 30%.

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

$0.1108367 * n * ln(n) + 138.415608549 = 10558905163$

| n | comparações | tempo(s) |
|---|---|---|
| 32 | 31 | 0.000582 |
| 64 | 63 | 0.001248 |
| 128 | 127 | 0.002682 |

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

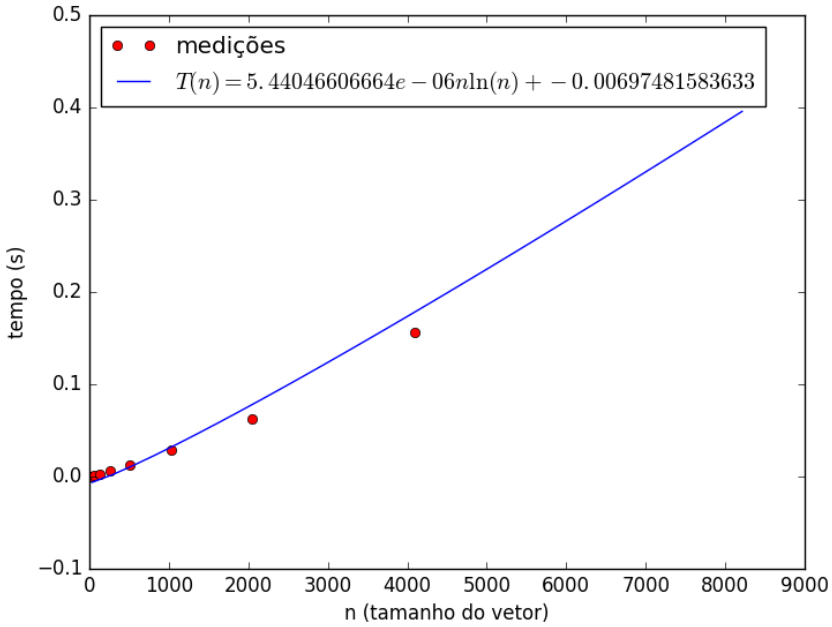


**Figura 2.13:** *Mergesort com relação ao tempo com vetor quase crescente* 40%.

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

$5.44046606664 * 10^{-6} * n * ln(n) - 0.00697481583633 = 54102.77152$

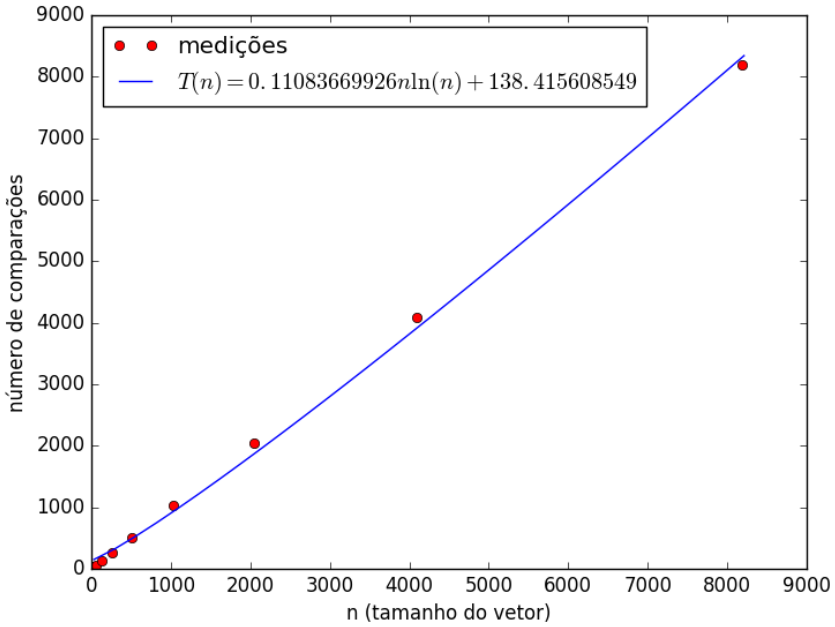


**Figura 2.14:** *Mergesort com relação ao número de comparações com vetor quase crescente* 40%.

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

$0.1108367 * n * ln(n) + 138.415608549 = 10558905163$

| n | comparações | tempo(s) |
|---|---|---|
| 32 | 31 | 0.000567 |
| 64 | 63 | 0.001204 |
| 128 | 127 | 0.002598 |

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

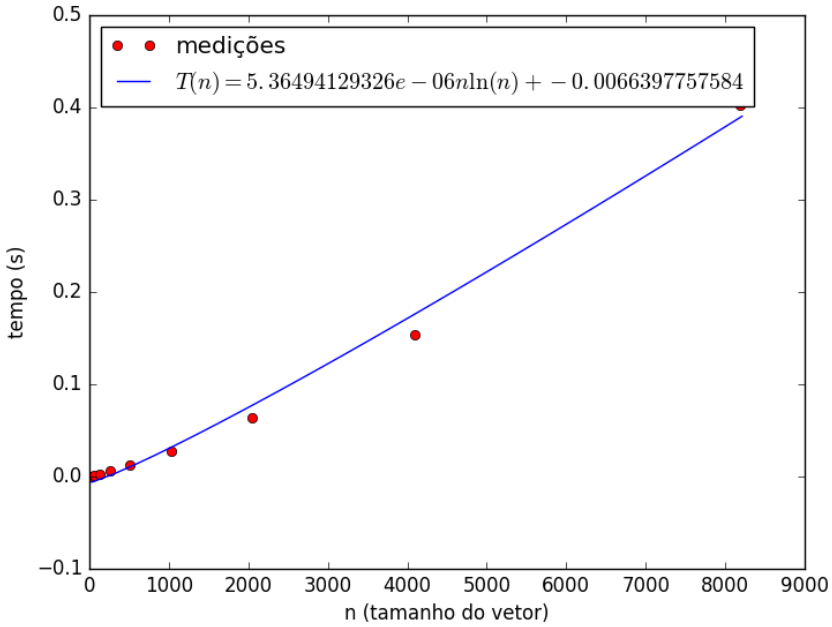


**Figura 2.15:** *Mergesort com relação ao tempo com vetor quase crescente* 50%.

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

$5.36494129326 * 10^{-6} * n * ln(n) - 0.0066397757584 = 53771.31092$

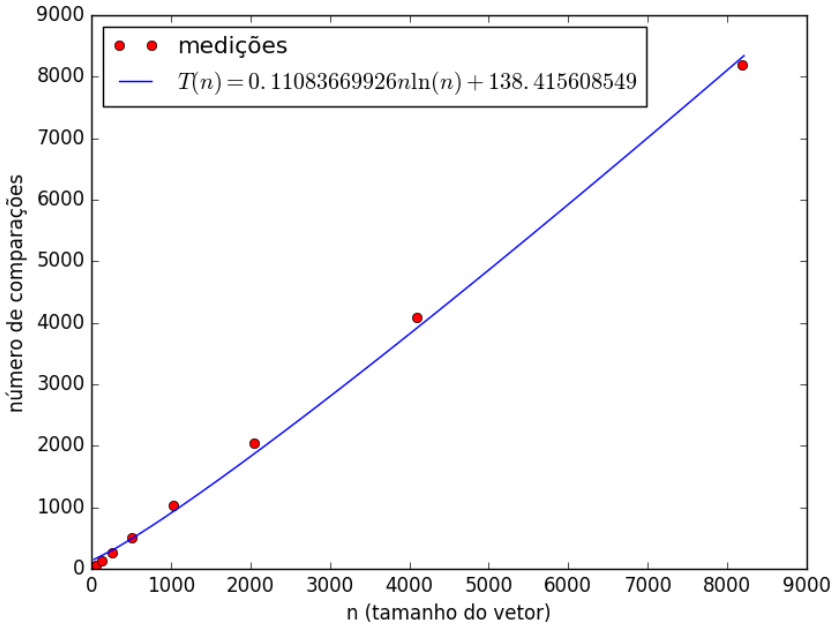


**Figura 2.16:** *Mergesort com relação ao número de comparações com vetor quase crescente* 50%.

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

$0.1108367 * n * ln(n) + 138.415608549 = 10558905163$

| n | comparações | tempo(s) |
|---|---|---|
| 32 | 31 | 0.000569 |
| 64 | 63 | 0.001198 |
| 128 | 127 | 0.002699 |

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

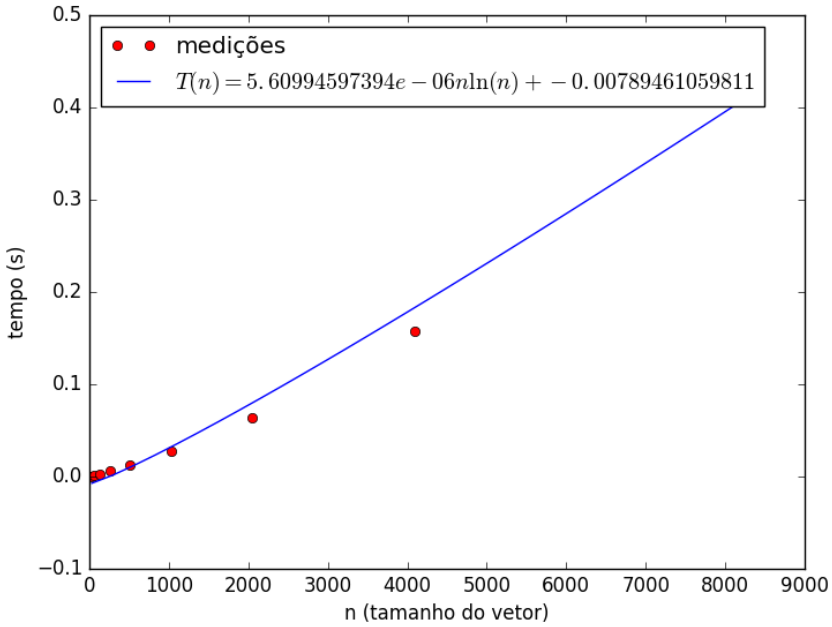


**Figura 2.17:** *Mergesort com relação ao tempo com vetor quase decrescente* 10%.

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

$5.60994597394 * 10^{-6} * n * ln(n) - 0.00789461059811 = 54328.48169$

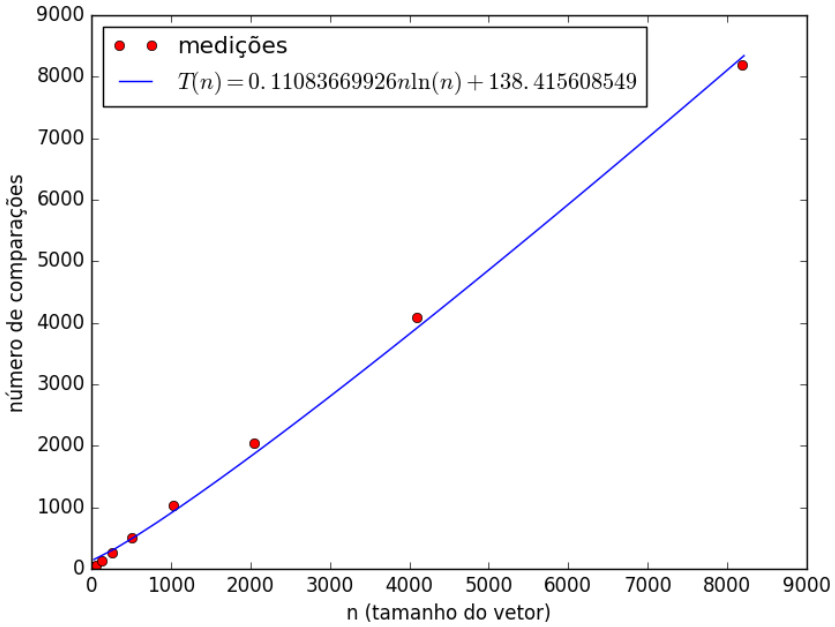


**Figura 2.18:** *Mergesort com relação ao número de comparações com vetor quase decrescente* 10%.

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

$0.1108367 * n * ln(n) + 138.415608549 = 10558905163$

| n | comparações | tempo(s) |
|---|---|---|
| 32 | 31 | 0.000594 |
| 64 | 63 | 0.001271 |
| 128 | 127 | 0.002697 |

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

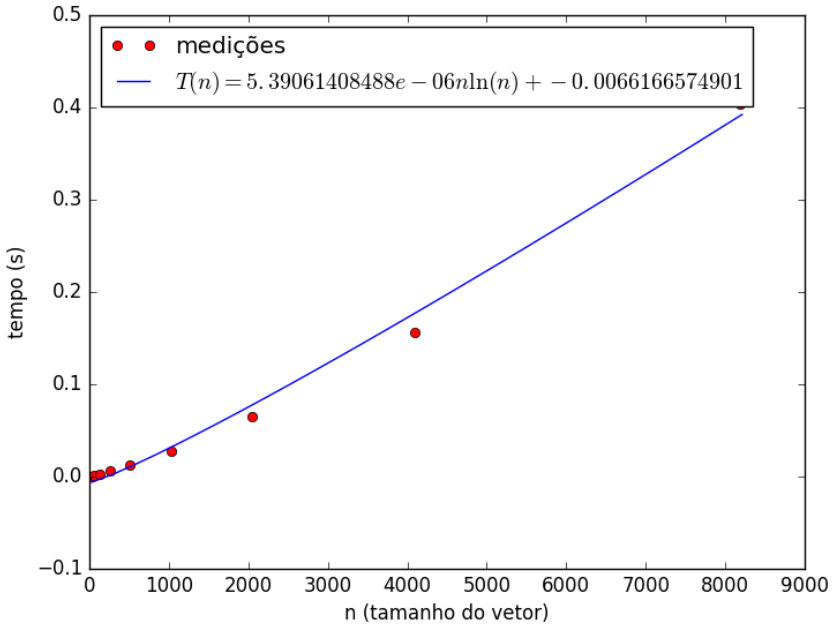


**Figura 2.19:** *Mergesort com relação ao tempo com vetor quase decrescente* 20%.

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

$5.39061408488 * 10^{-6} * n * ln(n) - 0.0066166574901 = 53891.67193$

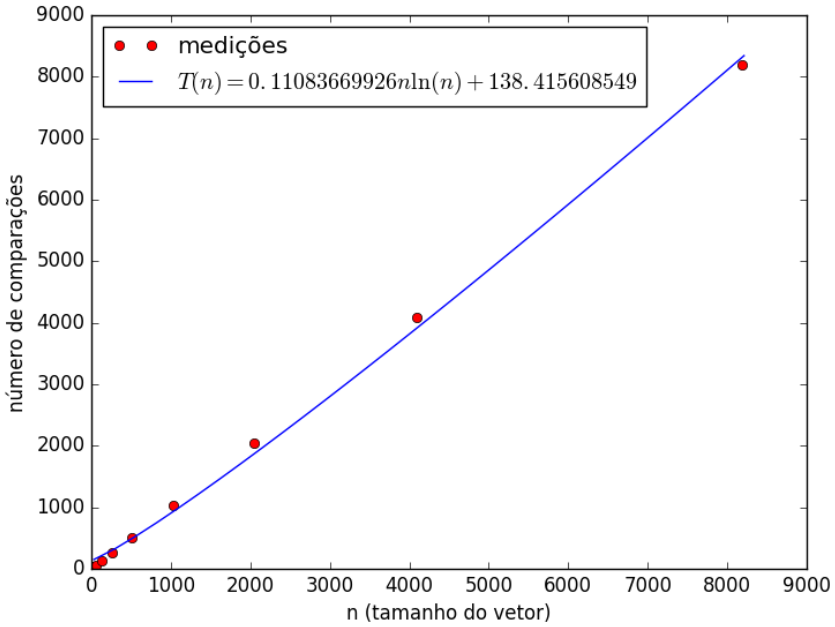


**Figura 2.20:** *Mergesort com relação ao número de comparações com vetor quase decrescente* 20%.

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

$0.1108367 * n * ln(n) + 138.415608549 = 10558905163$

| n | comparações | tempo(s) |
|---|---|---|
| 32 | 31 | 0.000592 |
| 64 | 63 | 0.001200 |
| 128 | 127 | 0.002546 |

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

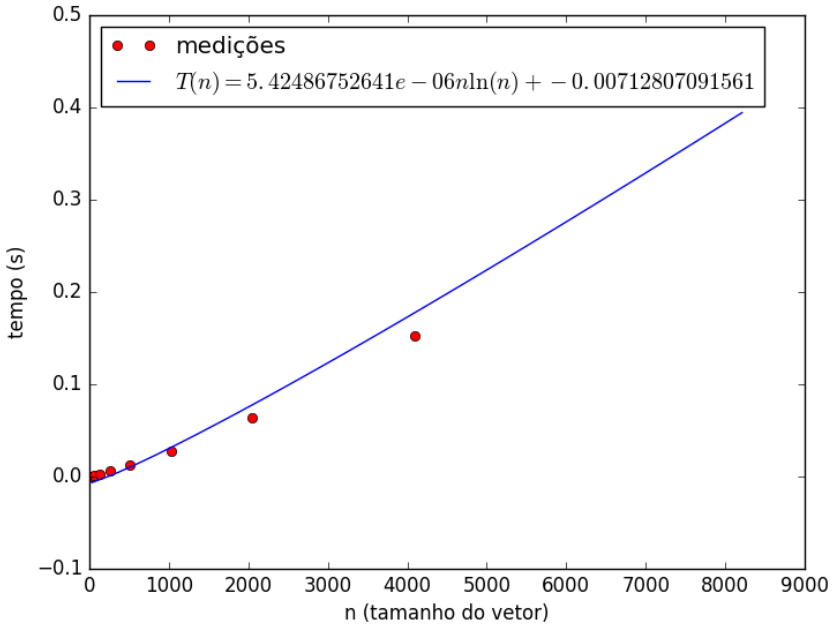


**Figura 2.21:** *Mergesort com relação ao tempo com vetor quase decrescente* 30%.

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

$5.4248675261 * 10^{-6} * n * ln(n) - 0.00712807091561 = 54010.28164$

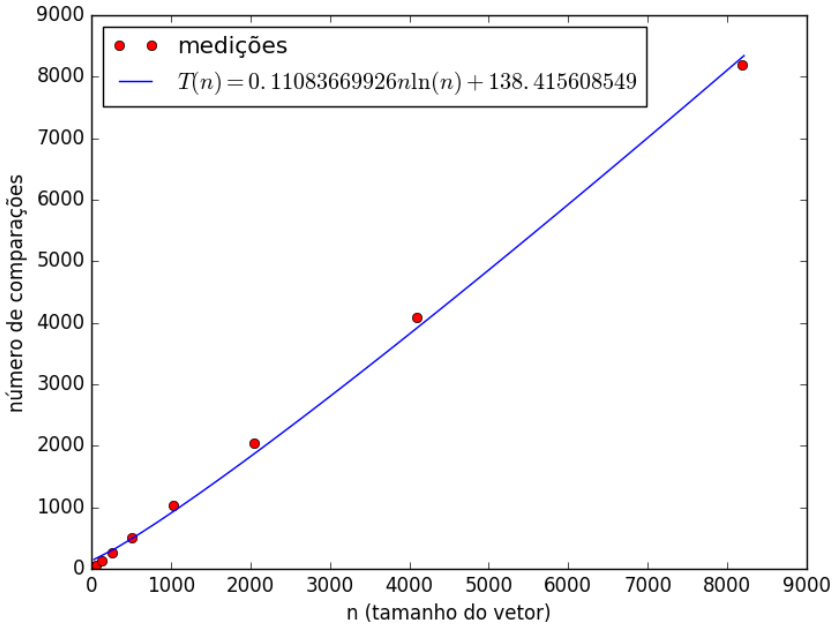


**Figura 2.22:** *Mergesort com relação ao número de comparações com vetor quase decrescente* 30%.

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

$0.1108367 * n * ln(n) + 138.415608549 = 10558905163$

| n | comparações | tempo(s) |
|---|---|---|
| 32 | 31 | 0.000569 |
| 64 | 63 | 0.001289 |
| 128 | 127 | 0.002667 |
| 256 | 255 | 0.005804 |
| 512 | 511 | 0.012282 |
| 1024 | 1023 | 0.027300 |
| 2048 | 2047 | 0.063969 |
| 4096 | 4095 | 0.158139 |
| 8192 | 8191 | 0.401932 |

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

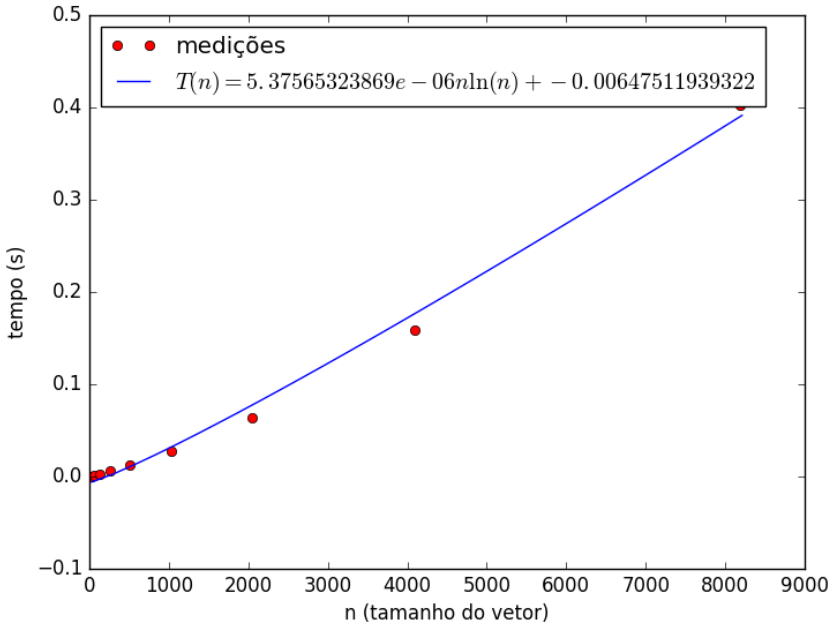


**Figura 2.23:** *Mergesort com relação ao tempo com vetor quase decrescente* 40%.

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

$5.37565323869 * 10^{-6} * n * ln(n) - 0.00647511939322 = 53793.19462$

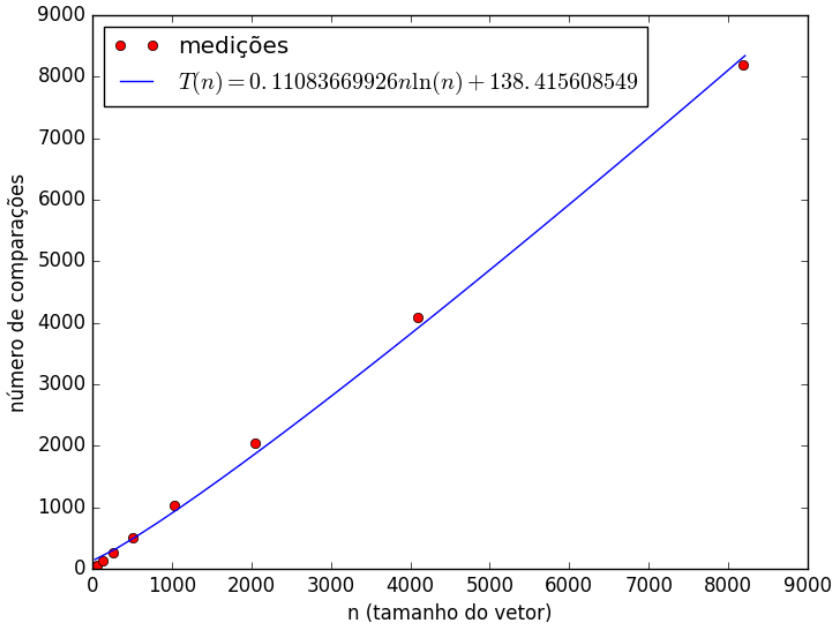


**Figura 2.24:** *Mergesort com relação ao número de comparações com vetor quase decrescente* 40%.

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

$0.1108367 * n * ln(n) + 138.415608549 = 10558905163$

| n | comparações | tempo(s) |
|---|---|---|
| 32 | 31 | 0.000571 |
| 64 | 63 | 0.001224 |
| 128 | 127 | 0.002673 |
| 256 | 255 | 0.005522 |
| 512 | 511 | 0.012766 |
| 1024 | 1023 | 0.027884 |
| 2048 | 2047 | 0.063459 |
| 4096 | 4095 | 0.158878 |
| 8192 | 8191 | 0.416121 |

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

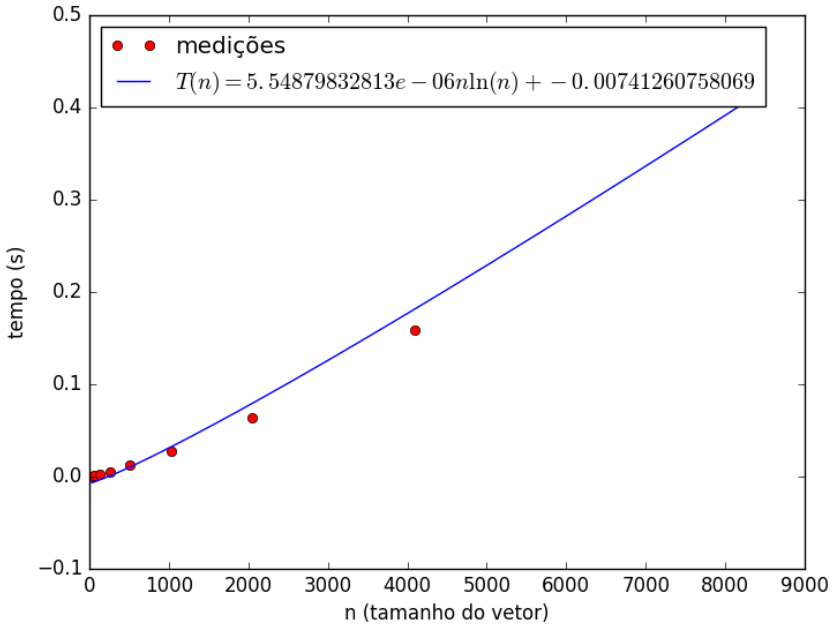


**Figura 2.25:** *Mergesort com relação ao tempo com vetor quase decrescente* 50%.

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

$5.54879832813 * 10^{-6} * n * ln(n) - 0.00741260758069 = 54872.76514$

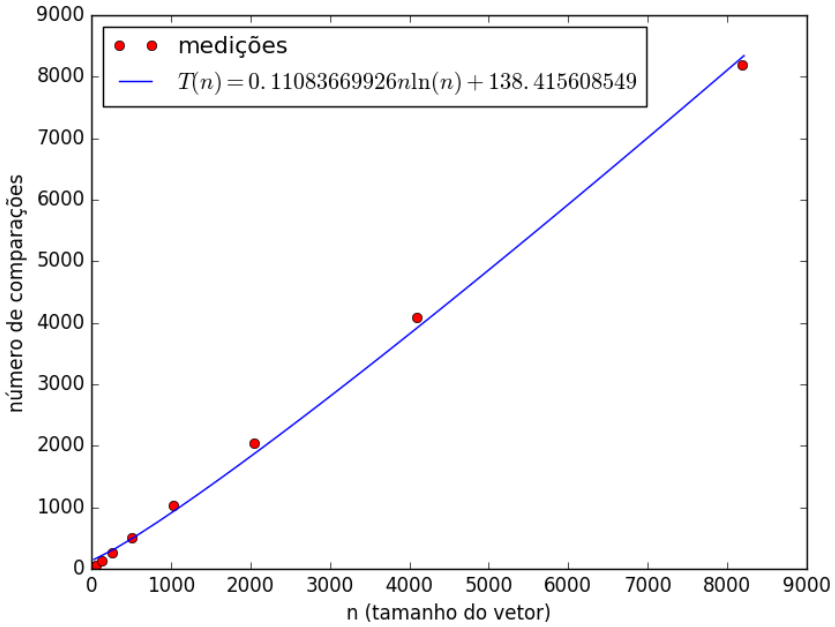


**Figura 2.26:** *Mergesort com relação ao número de comparações com vetor quase decrescente* 50%.

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

$0.1108367 * n * ln(n) + 138.415608549 = 10558905163$

# Apêndice A

# Arquivo ../mergesort/mergesort.py

Listagem A.1: ../mergesort/mergesort.py

```
1  import math
2
3  #@profile
4  def intercala(A,p,q,r):
5    #print("antes: ",  memory_usage_resource())
6    B = [0] *len(A)
7    for i in range(p,(q+1)):
8      B[i] = A[i]
9    for j in range(q+1,(r+1)):
10     B[r+q+1-j] = A[j]
11
12   i = p
13   j = r
14   #print("depois", memory_usage_resource())
15
16   for k in range (p,(r+1)):
17     if(B[i] <= B[j]):
18       A[k] = B[i]
19       i = i+1
20     else:
21       A[k] = B[j]
22       j = j-1
23
24 #@profile
25 def merge(A):
26   mergesort(A,0,len(A)-1)
27
28   #return A
29
30
31 @profile
32 def mergesort(A,esquerda,direita):
33   if(esquerda<direita):
34     meio = math.floor((esquerda+direita)/2)
35     mergesort(A,esquerda,meio)
36     mergesort(A,meio+1,direita)
37     intercala(A,esquerda,meio,direita)
```

# Apêndice B

# Arquivo ../mergesort/ensaio.py

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

```
1  import numpy as np
2  import argparse
3
4  from mergesort 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 merge(vet)
```