Análise experimental de algoritmos usando Python

Patrícia Mariana Ramos Marcolino
pmrmarcolino@hotmail.com

Eduardo Pinheiro Barbosa
eduardptu@hotmail.com

Faculdade de Computação
Universidade Federal de Uberlândia

1 de julho de 2016

Lista de Figuras

Lista de Tabelas

Lista de Listagens

Sumário

Capítulo 1

Análise

O algoritmo resolve o problema da ordenação usando um max-heap. A rotina Constrói-Max-Heap transforma $1..n$ em um max-heap e a rotina Corrige-Descendo transforma o vetor $1..m1$ em max-heap supondo que as subárvores com raizes 2 e 3 são max-heaps.

- o vetor $A[1..m]$ é um max-heap, $A[1..m] \leq A[m+1..n]$ (ou seja, $A[1] \leq A[m+1]$, $A[2] \leq A[m+2]$, etc.) e - o vetor $A[m+1..n]$ está em ordem crescente. - É claro que $A[1..n]$ estará em ordem crescente quando m for igual a 1.

No pior caso, Constrói-Max-Heap consome (n) unidades de tempo e cada invocação de Corrige-Descendo consome $(\log(n)$ unidades de tempo. Logo, Heapsort consome $(n \log(n))$

unidades de tempo no pior caso. Assim, o algoritmo é linearítmico.

Capítulo 2

Resultados

2.1 Tabelas

n	comparações	tempo(s)
32	31	0.000510
64	63	0.001352
128	127	0.002918
256	255	0.006309
512	511	0.014427
1024	1023	0.033338
2048	2047	0.072172
4096	4095	0.156134
8192	8191	0.373871

Tabela 2.1: *Tabela com vetor teste aleatório: A linha te interesse analisada para este caso é a 16.*

n	comparações	tempo(s)
32	31	0.000549
64	63	0.001314
128	127	0.003028
256	255	0.007136
512	511	0.016743
1024	1023	0.034056
2048	2047	0.081393
4096	4095	0.180865
8192	8191	0.367149

Tabela 2.2: *Tabela com vetor teste crescente: A linha te interesse analisada para este caso é a 16.*

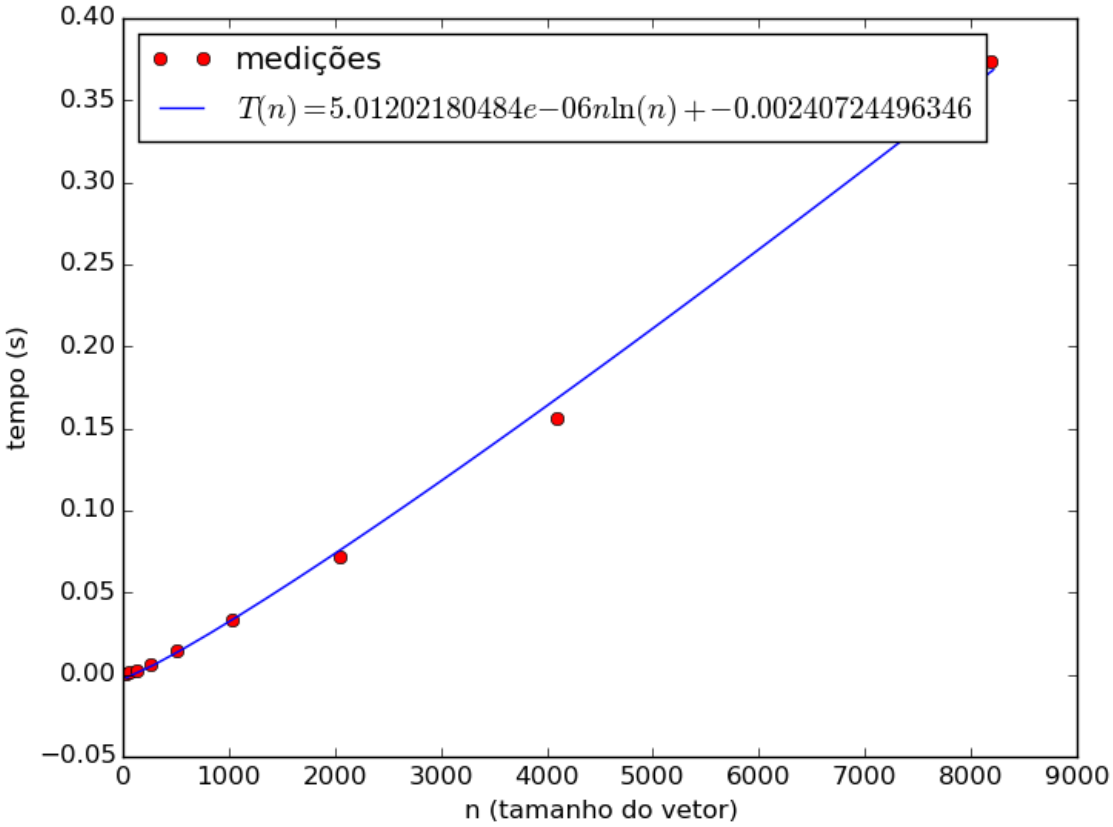

Figura 2.1: *A análise do grafico para 2^{32} segue abaixo para heapsort*
Tendo a função $T(n) = 5.012021\text{e} - 6 * n \log_2(n) - 0.0024072$ e para o $n = 2^{32}$, $T(2^{32}) = 477472.3$

n	comparações	tempo(s)
32	31	0.000451
64	63	0.001122
128	127	0.002635
256	255	0.005843
512	511	0.013617
1024	1023	0.030917
2048	2047	0.067609
4096	4095	0.162030
8192	8191	0.315894

Tabela 2.3: *Tabela com vetor teste decrescente: A linha te interesse analisada para este caso é a 16.*

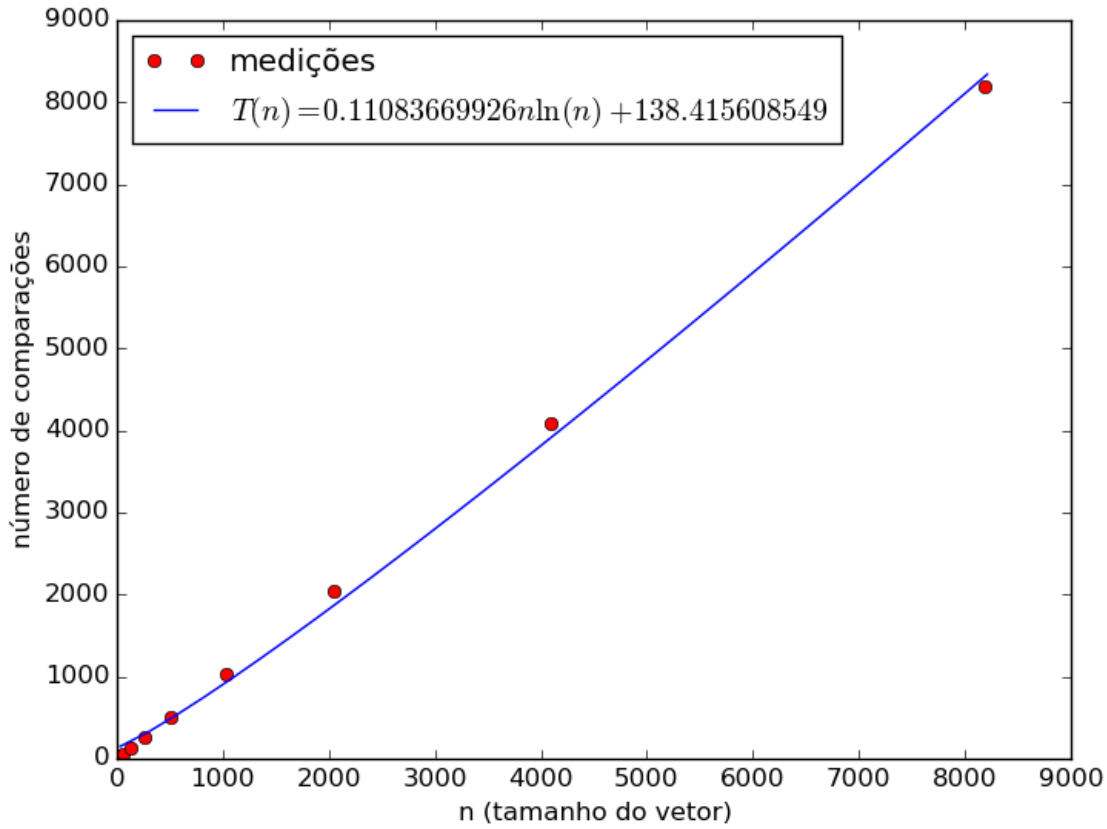

Figura 2.2: *A análise do grafico para* 2^{32} *segue abaixo para heapsort*
Tendo a função $T(n) = 0.11083\text{e} - 6 * n \log_2(n) + 138.4156$ e para o $n = 2^{32}$,
$T(2^{32}) = 10696.7$

n	comparações	tempo(s)
32	31	0.000541
64	63	0.001332
128	127	0.003056
256	255	0.007292
512	511	0.015537
1024	1023	0.034572
2048	2047	0.082278
4096	4095	0.165015
8192	8191	0.359047

Tabela 2.4: *Tabela com vetor teste quase crescente 10%: A linha te interesse analisada para este caso é a 16.*

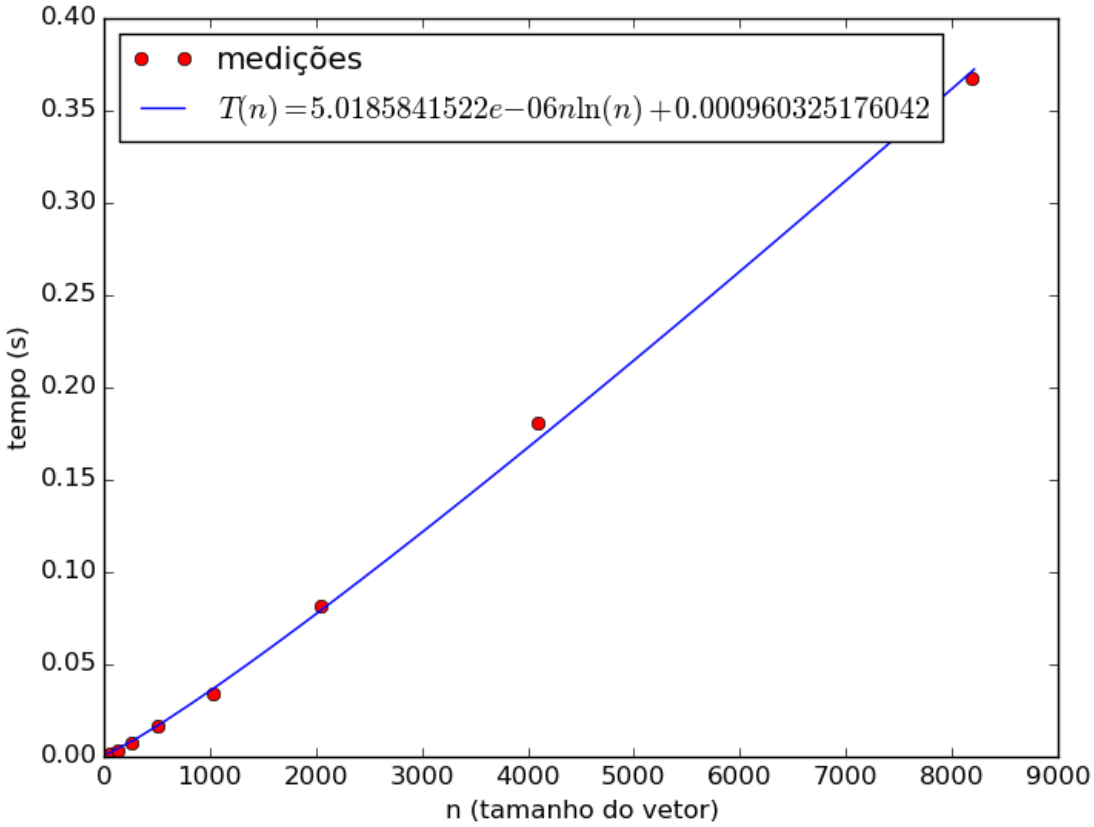

Figura 2.3: *A análise do grafico para 2^{32} segue abaixo para heapsort*
Tendo a função $T(n) = 5.01858\text{e} - 6 * n\log_2(n) - 0.0009603$ e para o $n = 2^{32}$, $T(2^{32}) = 4.78097 * 10^5$

n	comparações	tempo(s)
32	31	0.000605
64	63	0.001323
128	127	0.003082
256	255	0.006918
512	511	0.015085
1024	1023	0.034702
2048	2047	0.075582
4096	4095	0.164716
8192	8191	0.366439

Tabela 2.5: *Tabela com vetor teste quase crescente 20%: A linha te interesse analisada para este caso é a 16.*

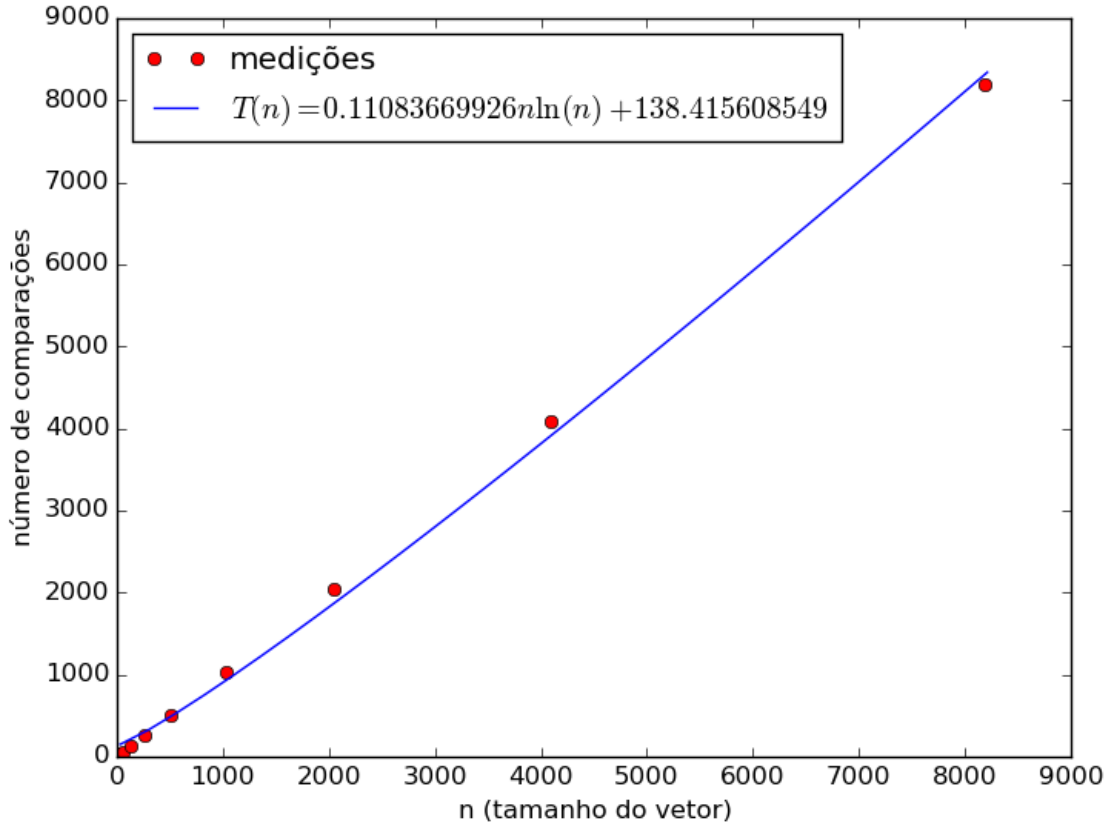

Figura 2.4: *A análise do grafico para* 2^{32} *segue abaixo para heapsort*
Tendo a função $T(n) = 0.11083\text{e} - 6 * n \log_2(n) + 138.4156$ e para o $n = 2^{32}$, $T(2^{32}) = 10696.7$

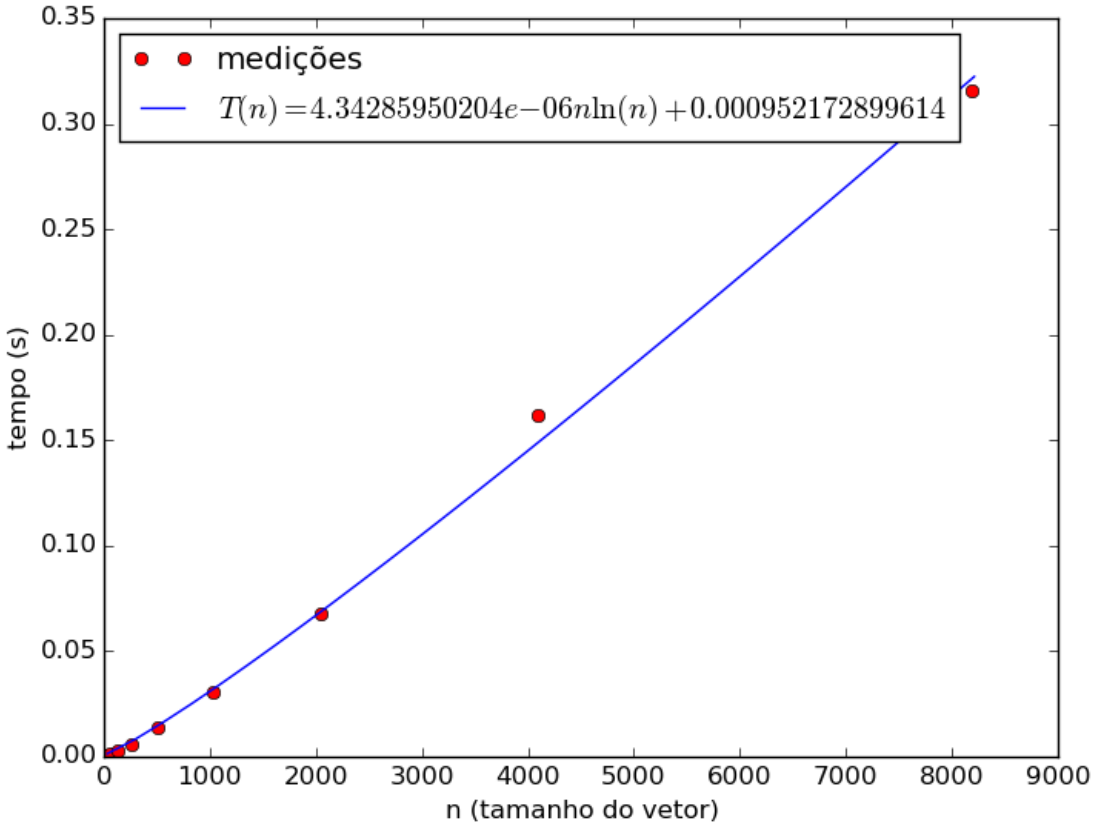

Figura 2.5: *A análise do grafico para* 2^{32} *segue abaixo para heapsort*
Tendo a função $T(n) = 4.34285\text{e} - 6 * n\log_2(n) + 0.0009521$ e para o $n = 2^{32}$, $T(2^{32}) = 4.13723 * 10^5$

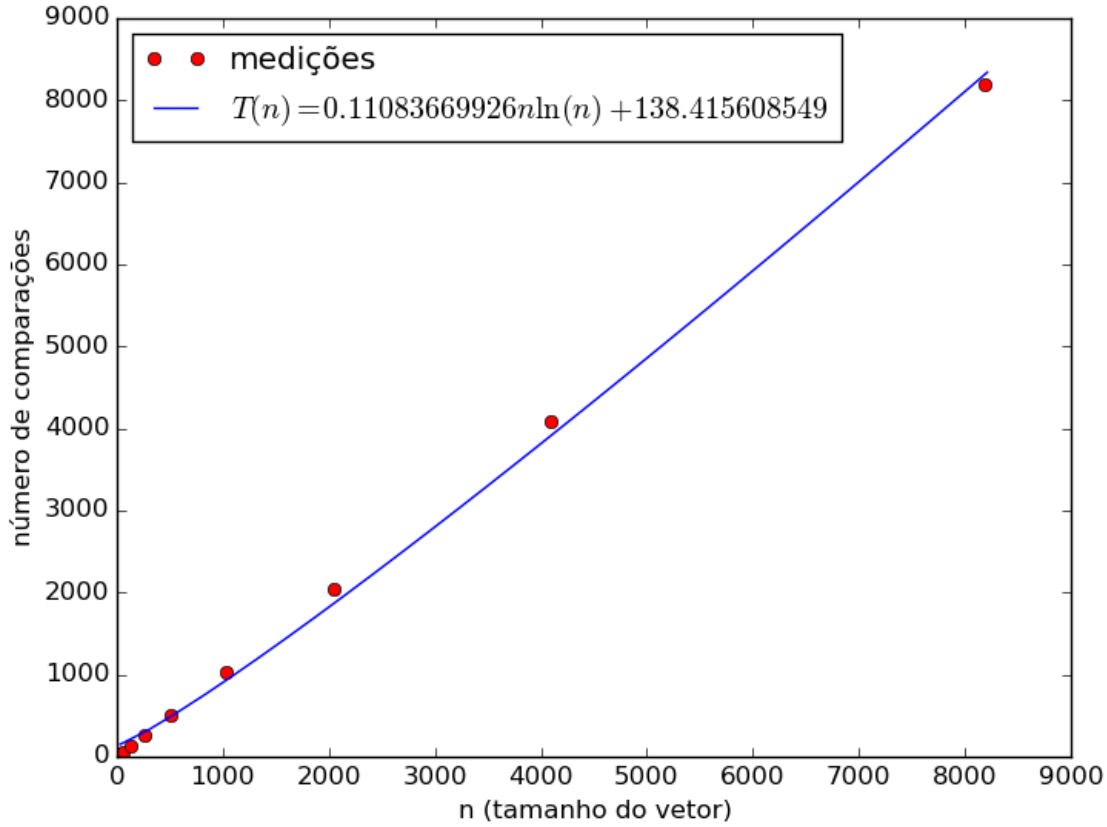

Figura 2.6: *A análise do grafico para* 2^{32} *segue abaixo para heapsort*
Tendo a função $T(n) = 0.11083\text{e} - 6 * n \log_2(n) + 138.4156$ e para o $n = 2^{32}$, $T(2^{32}) = 10696.7$

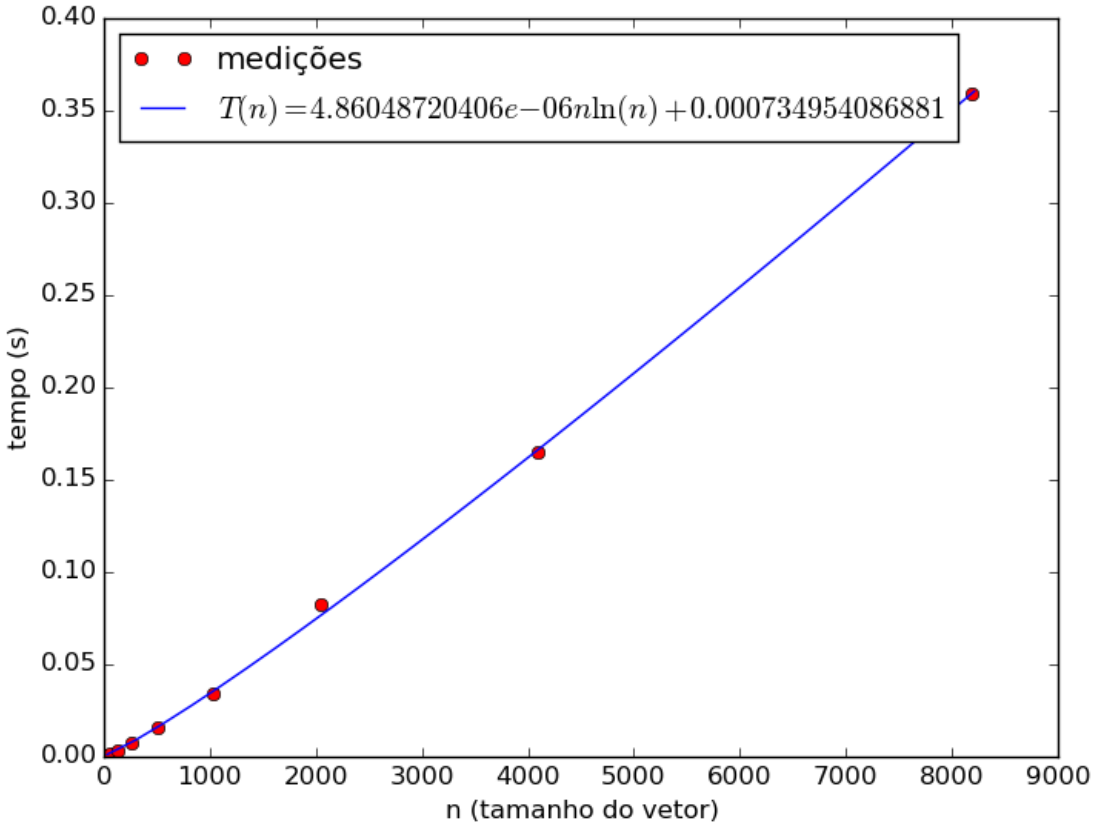

Figura 2.7: *A análise do grafico para* 2^{32} *segue abaixo para heapsort*
Tendo a função $T(n) = 4.86048\text{e} - 6 * n\log_2(n) + 0.0007349$ e para o $n = 2^{32}$, $T(2^{32}) = 4.63036 * 10^5$

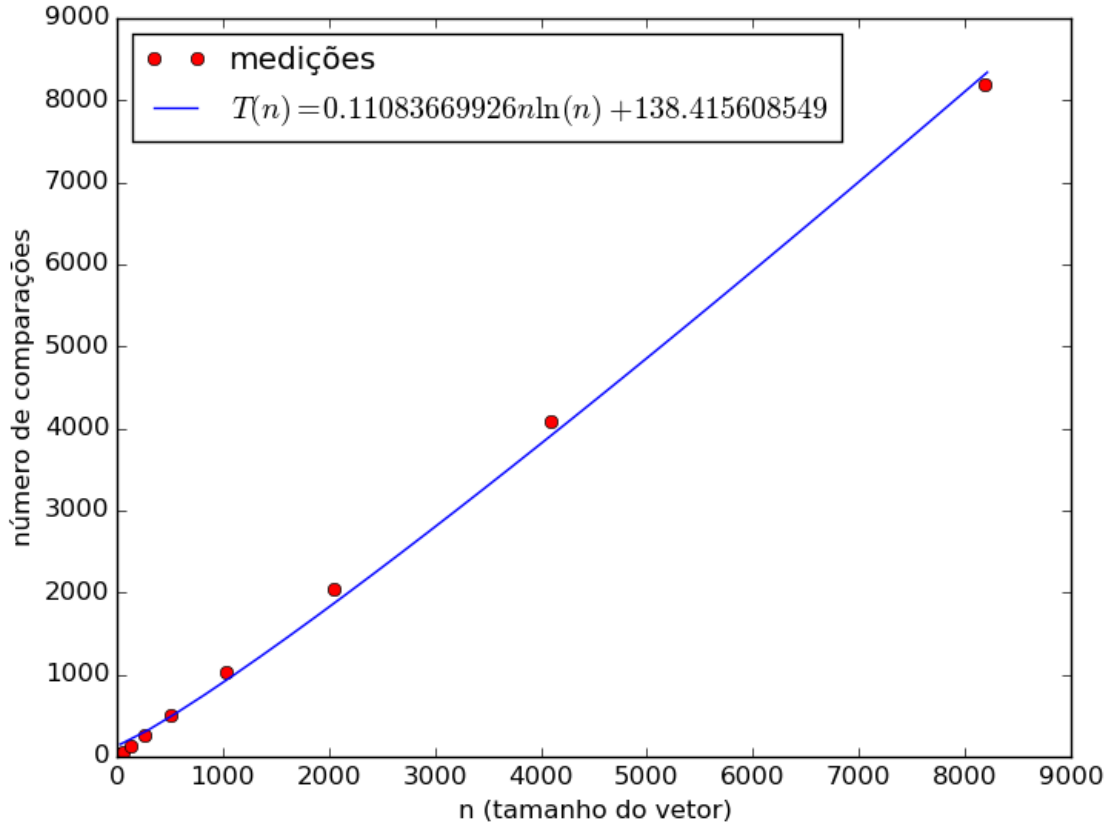

Figura 2.8: *A análise do grafico para* 2^{32} *segue abaixo para heapsort*
Tendo a função $T(n) = 0.11083\text{e} - 6 * n \log_2(n) + 138.4156$ e para o $n = 2^{32}$, $T(2^{32}) = 10696.7$

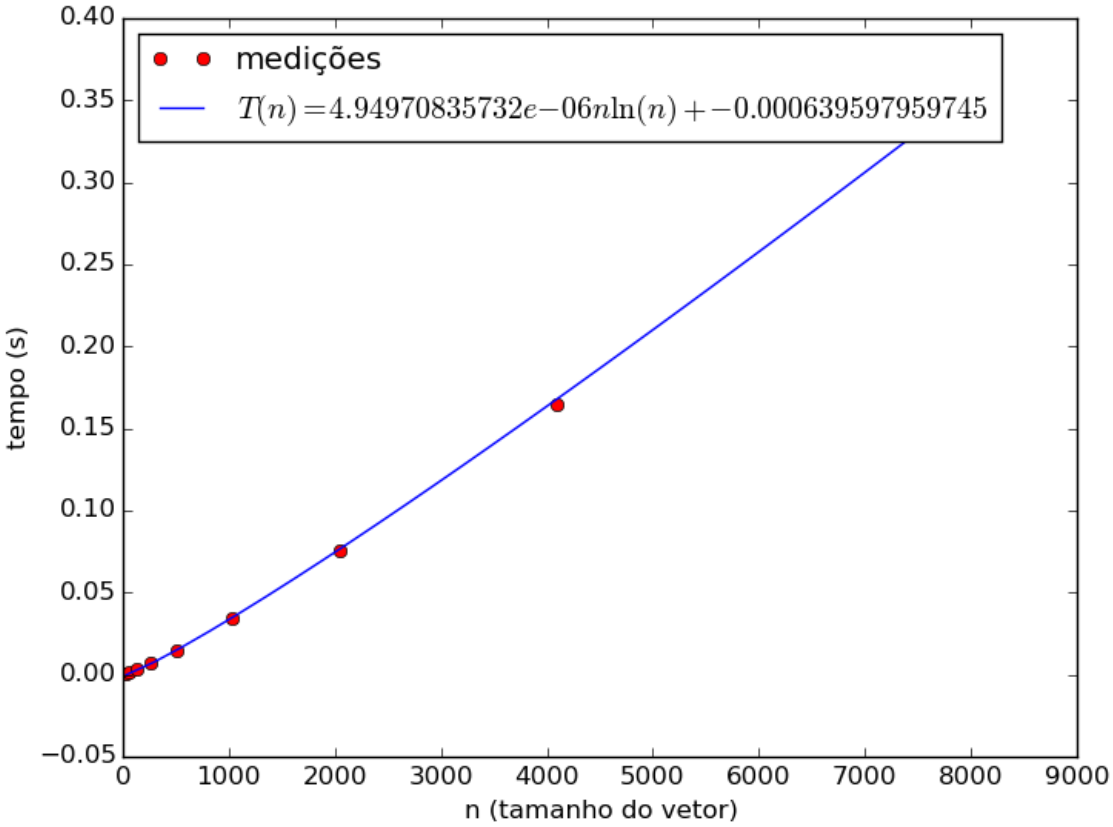

Figura 2.9: *A análise do grafico para* 2^{32} *segue abaixo para heapsort*
Tendo a função $T(n) = 4.949708\text{e} - 6 * n \log_2(n) - 0.0006395$ e para o $n = 2^{32}$, $T(2^{32}) = 471536.0$

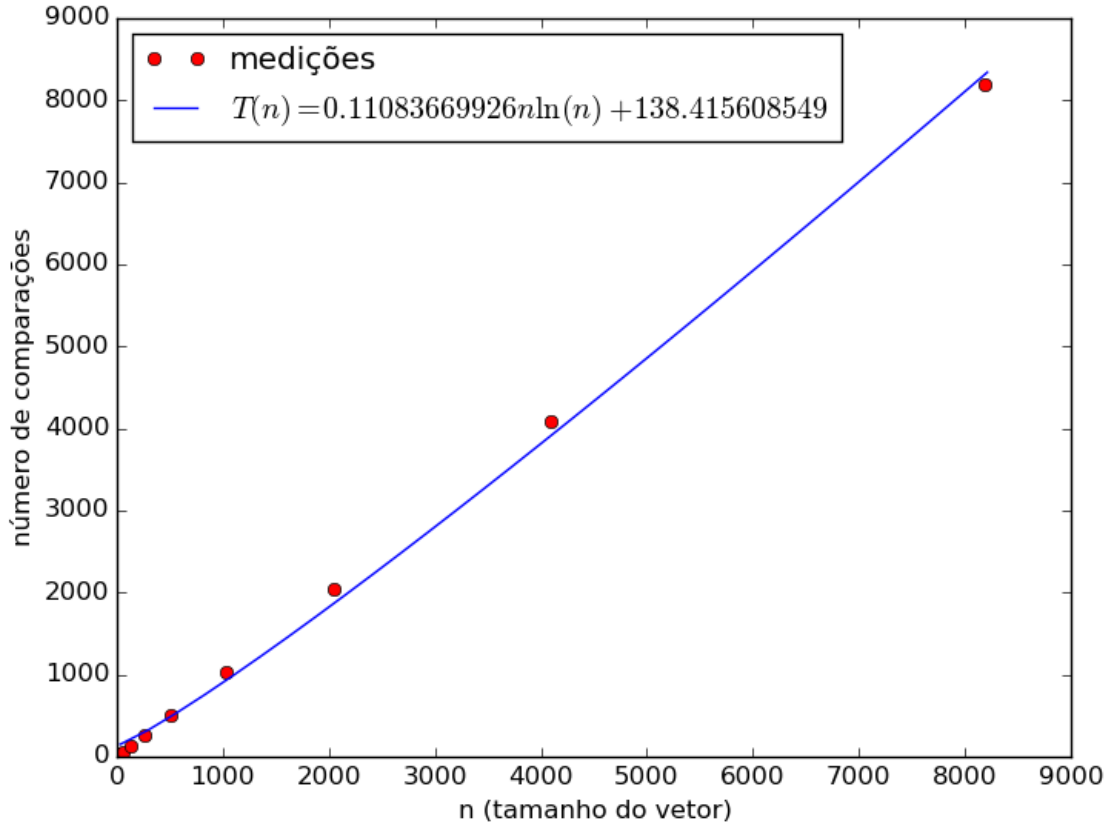

Figura 2.10: *A análise do grafico para 2^{32} segue abaixo para heapsort*
Tendo a função $T(n) = 0.11083\text{e} - 6 * n \log_2(n) + 138.4156$ e para o $n = 2^{32}$, $T(2^{32}) = 10696.7$

n	comparações	tempo(s)
32	31	0.000541
64	63	0.001350
128	127	0.002993
256	255	0.007470
512	511	0.015455
1024	1023	0.034731
2048	2047	0.074103
4096	4095	0.160569
8192	8191	0.362721

Tabela 2.6: *Tabela com vetor teste quase crescente 30%: A linha te interesse analisada para este caso é a 16.*

n	comparações	tempo(s)
32	31	0.000563
64	63	0.001342
128	127	0.003059
256	255	0.006743
512	511	0.015309
1024	1023	0.034003
2048	2047	0.073092
4096	4095	0.165887
8192	8191	0.382764

Tabela 2.7: *Tabela com vetor teste quase crescente 40%: A linha te interesse analisada para este caso é a 16.*

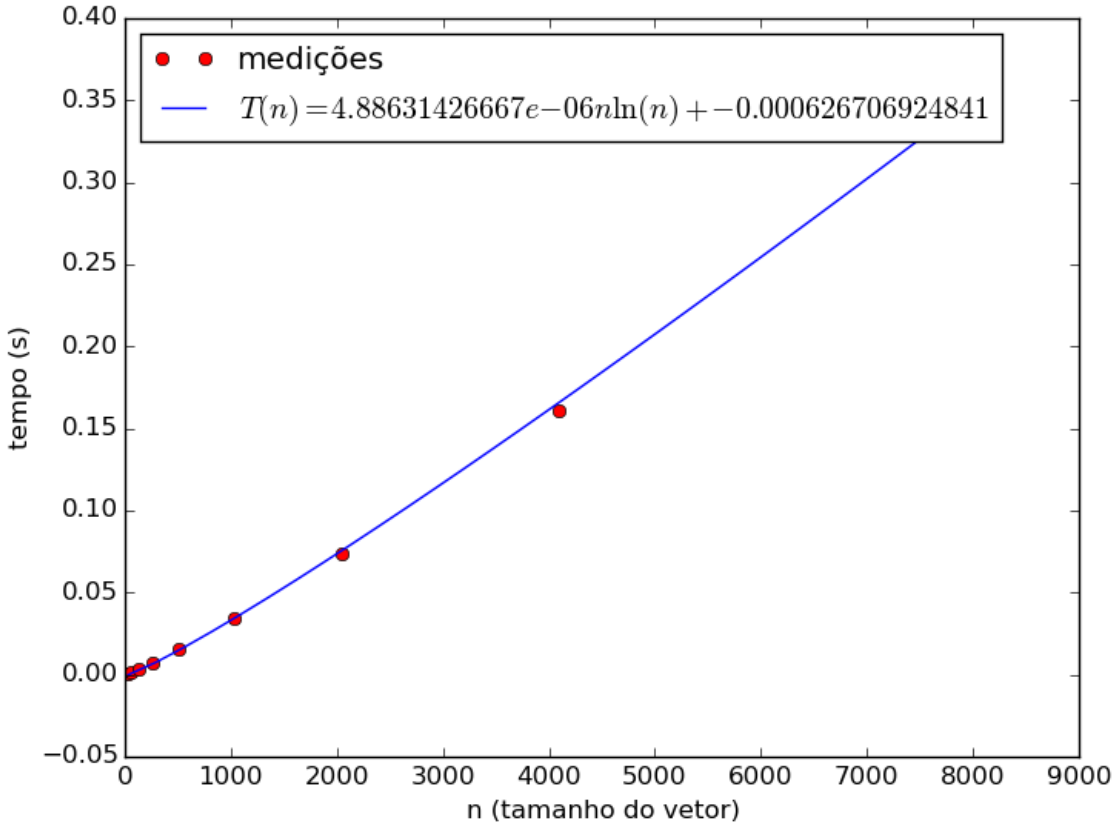

Figura 2.11: *A análise do grafico para 2^{32} segue abaixo para heapsort*
Tendo a função $T(n) = 4.88631\text{e} - 6 * n\log_2(n) + 0.0006267$ e para o $n = 2^{32}$, $T(2^{32}) = 0.169809$

n	comparações	tempo(s)
32	31	0.000535
64	63	0.001315
128	127	0.003006
256	255	0.006813
512	511	0.015541
1024	1023	0.034505
2048	2047	0.083698
4096	4095	0.175428
8192	8191	0.346865

Tabela 2.8: *Tabela com vetor teste quase crescente 50%: A linha te interesse analisada para este caso é a 16.*

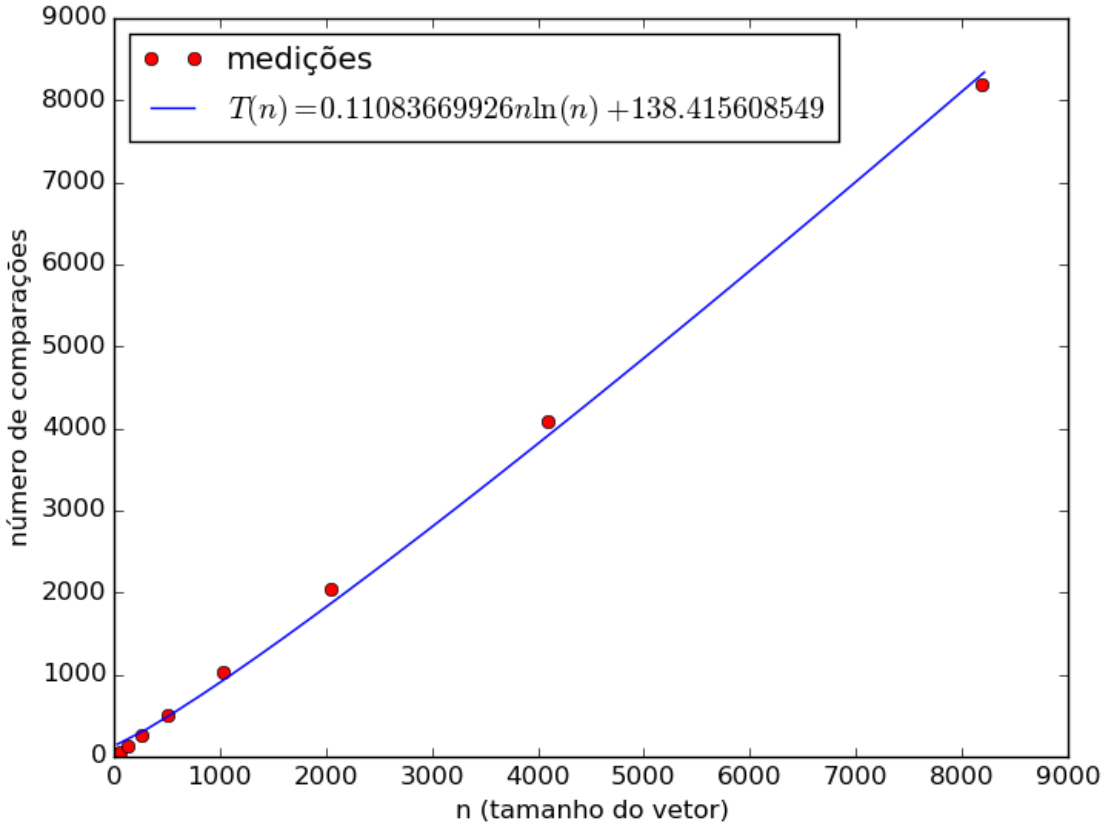

Figura 2.12: *A análise do grafico para 2^{32} segue abaixo para heapsort*
Tendo a função $T(n) = 0.11083\text{e} - 6 * n \log_2(n) + 138.4156$ e para o $n = 2^{32}$, $T(2^{32}) = 4.65496 * 10^5$

n	comparações	tempo(s)
32	31	0.000462
64	63	0.001099
128	127	0.002711
256	255	0.006049
512	511	0.013882
1024	1023	0.030022
2048	2047	0.067893
4096	4095	0.150838
8192	8191	0.327281

Tabela 2.9: *Tabela com vetor teste quase decrescente 10%: A linha te interesse analisada para este caso é a 16.*

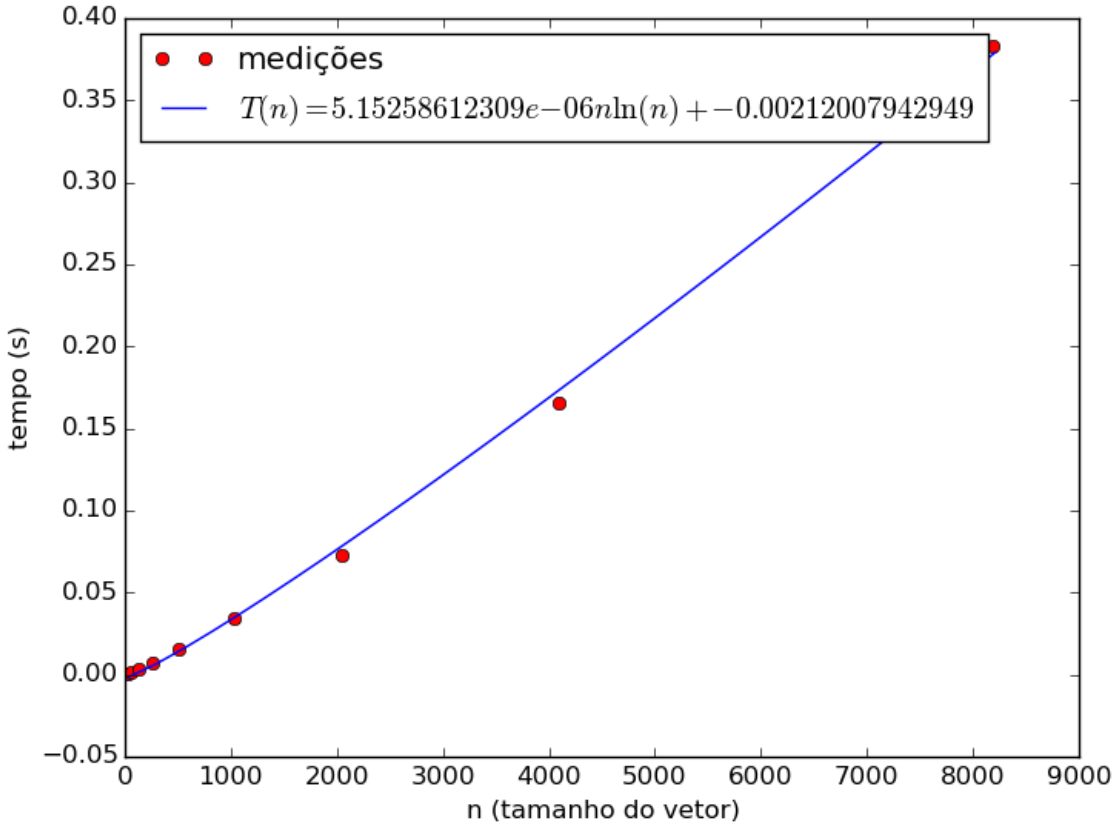

Figura 2.13: *A análise do grafico para 2^{32} segue abaixo para heapsort*
Tendo a função $T(n) = 5.1525\text{e} - 6 * n\log_2(n) - 0.002120$ e para o $n = 2^{32}$, $T(2^{32}) = 4.90855 * 10^5$

n	comparações	tempo(s)
32	31	0.000512
64	63	0.001142
128	127	0.002644
256	255	0.006195
512	511	0.013508
1024	1023	0.030987
2048	2047	0.065723
4096	4095	0.151637
8192	8191	0.327121

Tabela 2.10: *Tabela com vetor teste quase decrescente 20%: A linha te interesse analisada para este caso é a 16.*

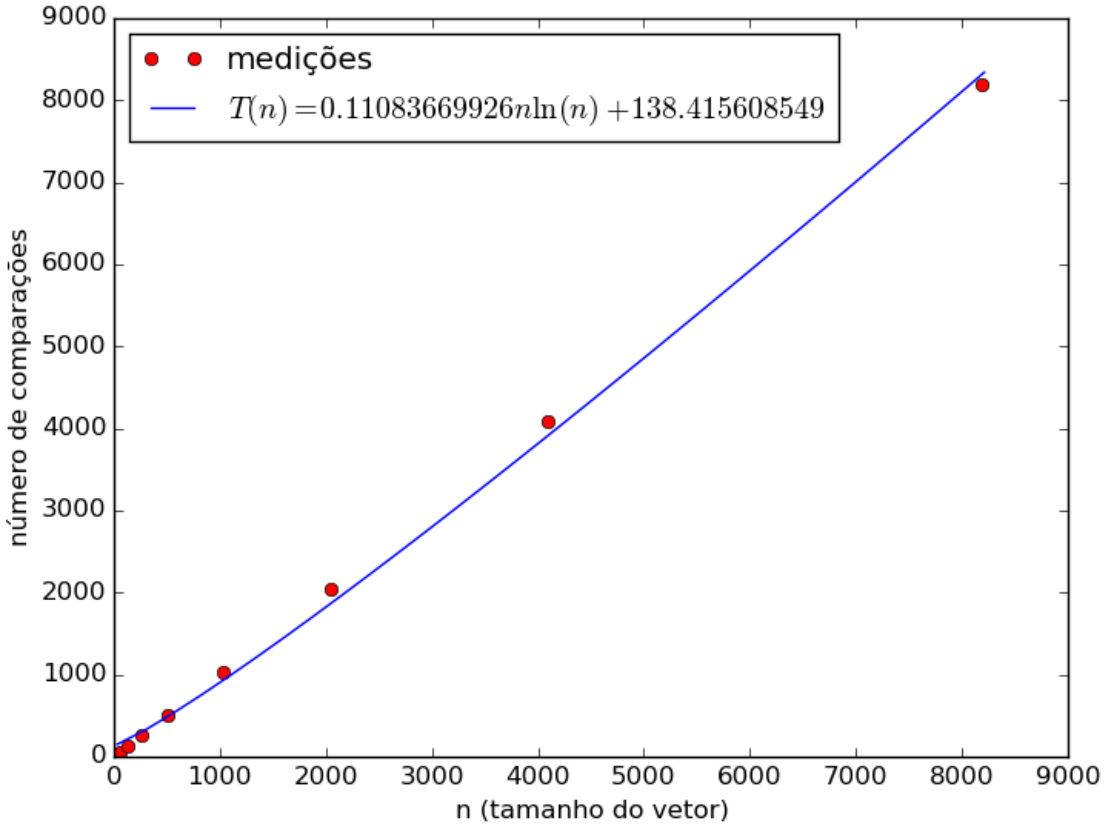

Figura 2.14: *A análise do grafico para* 2^{32} *segue abaixo para heapsort*
Tendo a função $T(n) = 0.11083\text{e} - 6 * n \log_2(n) + 138.4156$ e para o $n = 2^{32}$, $T(2^{32}) = 10696.7$

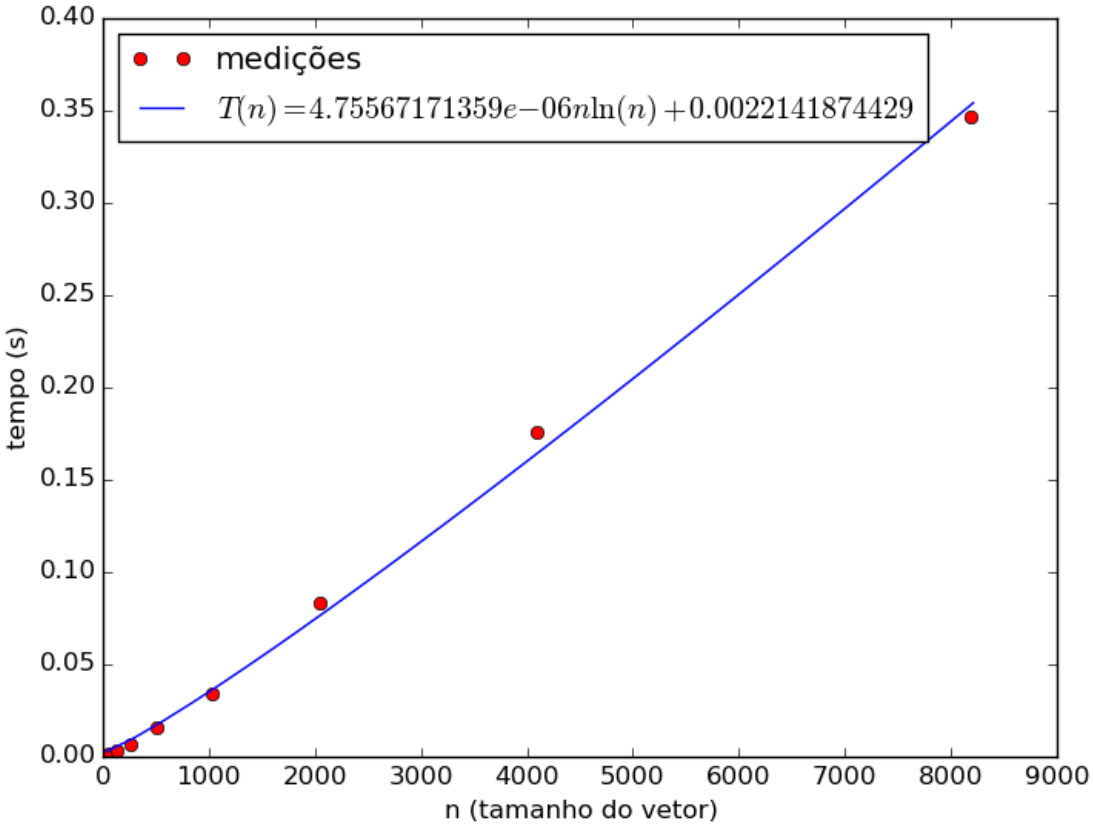

Figura 2.15: *A análise do grafico para 2^{32} segue abaixo para heapsort*
Tendo a função $T(n) = 4.7556\text{e} - 6 * n \log_2(n) + 0.00221$ e para o $n = 2^{32}$, $T(2^{32}) = 4.53044 * 10^5$

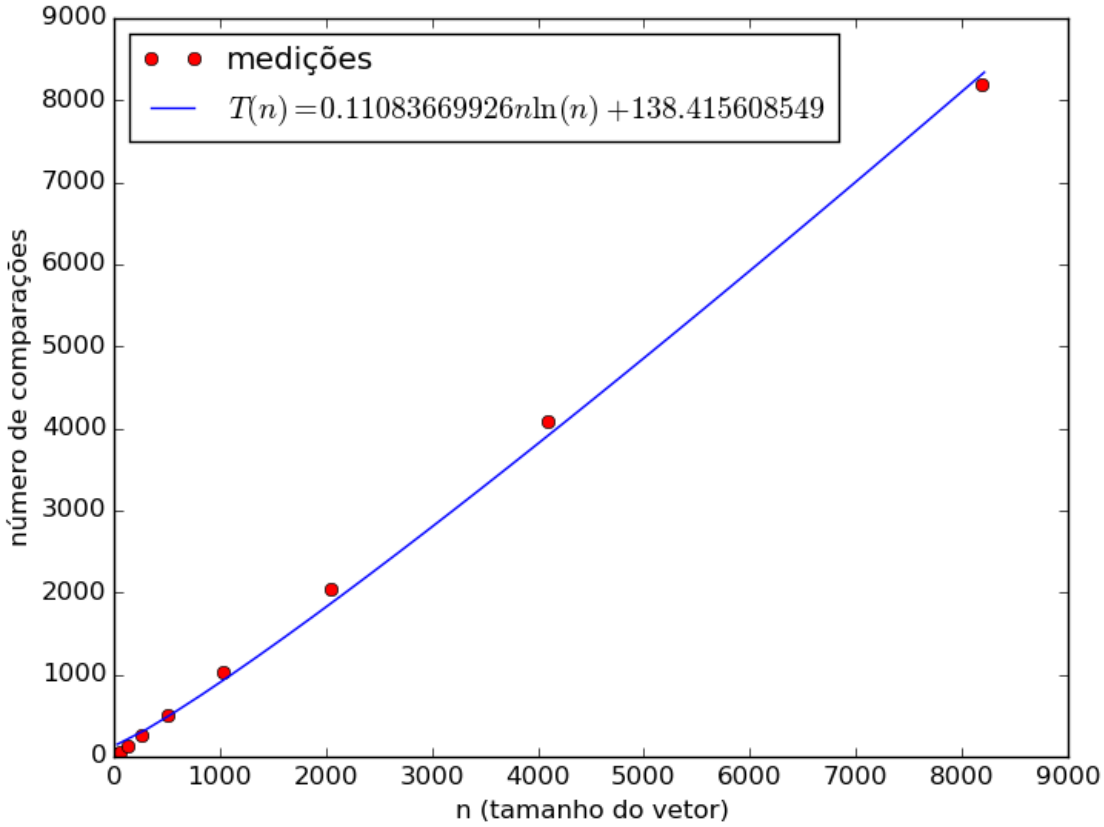

Figura 2.16: *A análise do grafico para* 2^{32} *segue abaixo para heapsort*
Tendo a função $T(n) = 0.11083\text{e} - 6 * n \log_2(n) + 138.4156$ e para o $n = 2^{32}$, $T(2^{32}) = 10696.7$

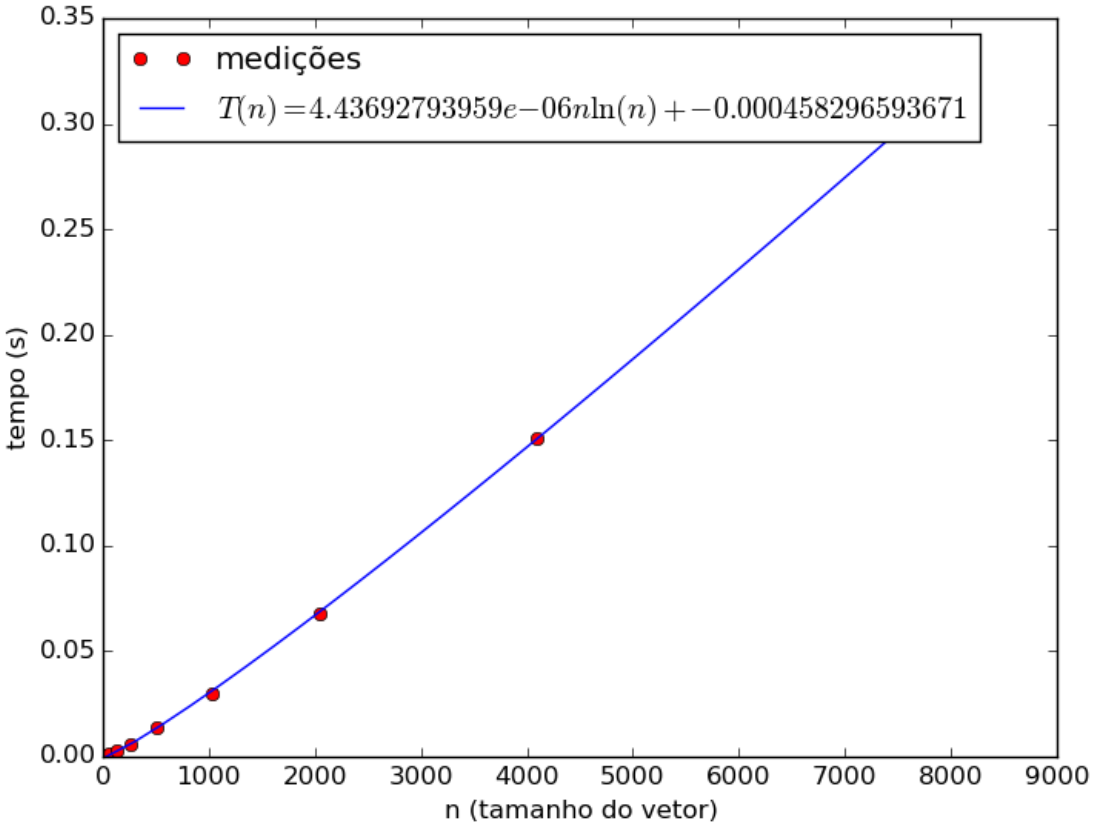

Figura 2.17: *A análise do grafico para 2^{32} segue abaixo para heapsort*
Tendo a função $T(n) = 4.43692\text{e} - 6 * n \log_2(n) - 0.0004582$ e para o $n = 2^{32}$, $T(2^{32}) = 4.22685 * 10^5$

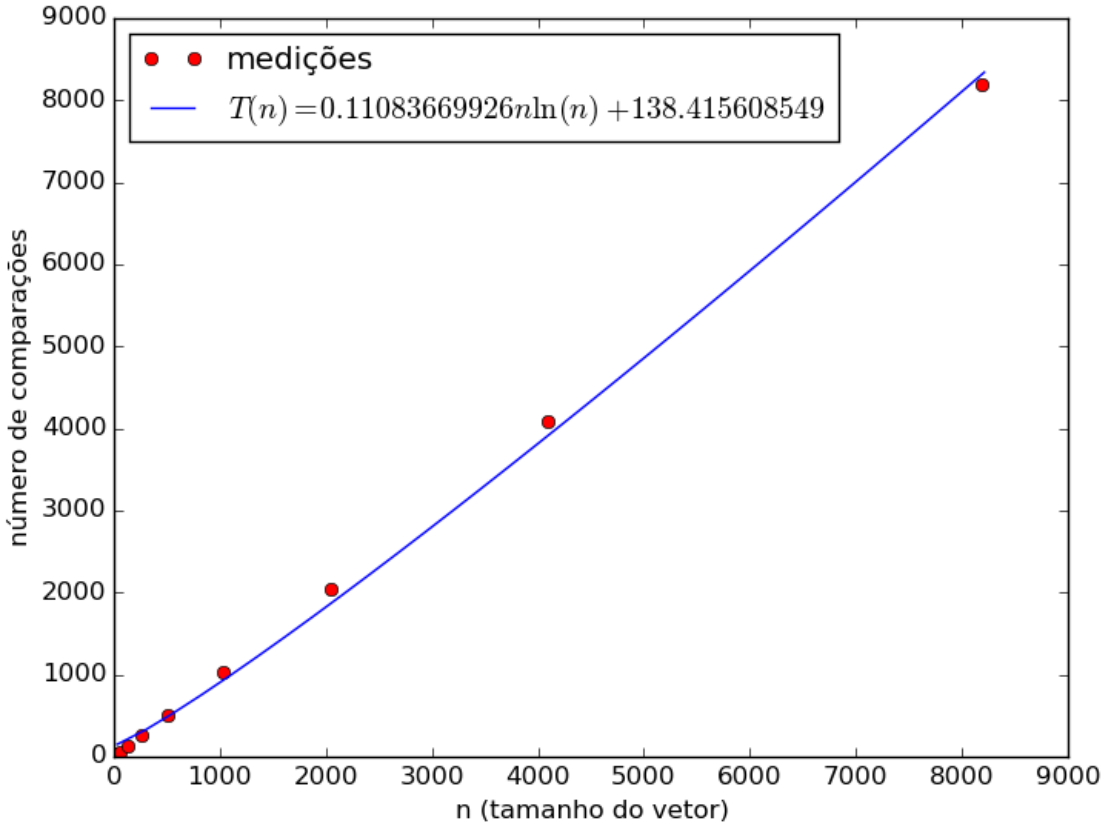

Figura 2.18: *A análise do grafico para* 2^{32} *segue abaixo para heapsort*
Tendo a função $T(n) = 0.110836\text{e} - 6 * n \log_2(n) + 138.4156$ e para o $n = 2^{32}$, $T(2^{32}) = 10697.3$

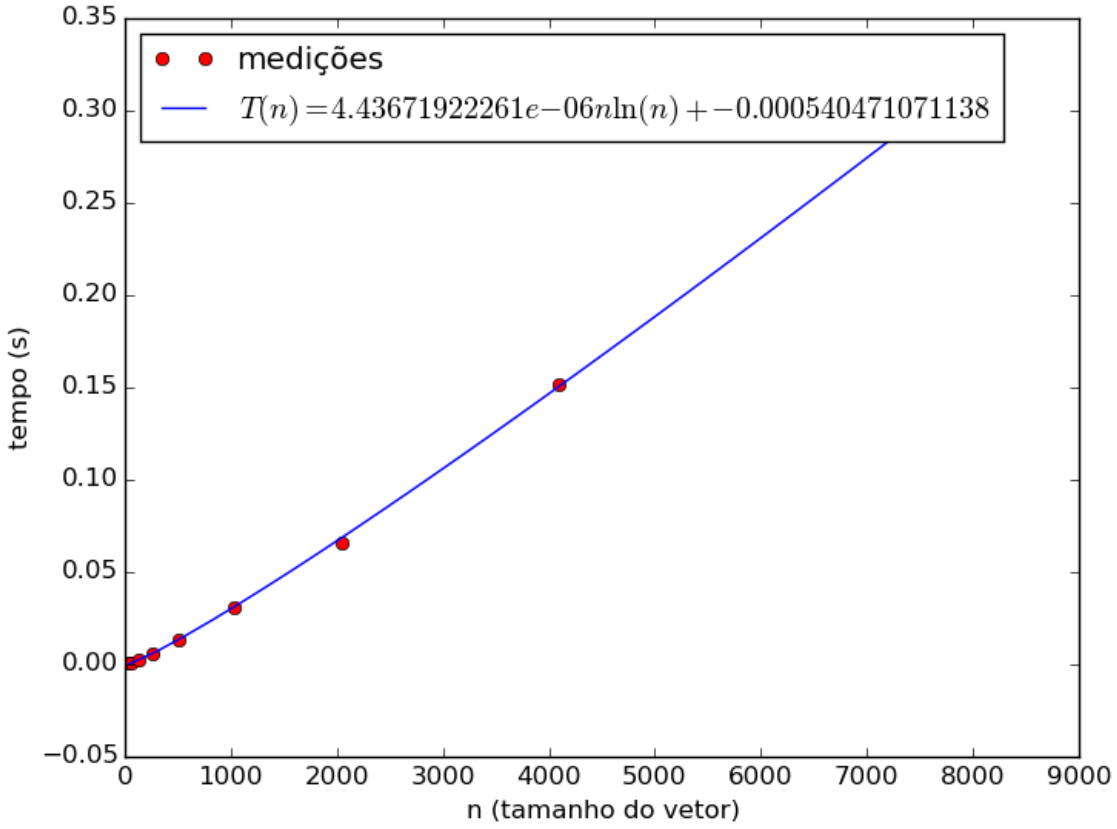

Figura 2.19: *A análise do grafico para 2^{32} segue abaixo para heapsort*
Tendo a função $T(n) = 4.43671\text{e} - 6 * n \log_2(n) - 0.00054047$ e para o $n = 2^{32}$, $T(2^{32}) = 4.22665 * 10^5$

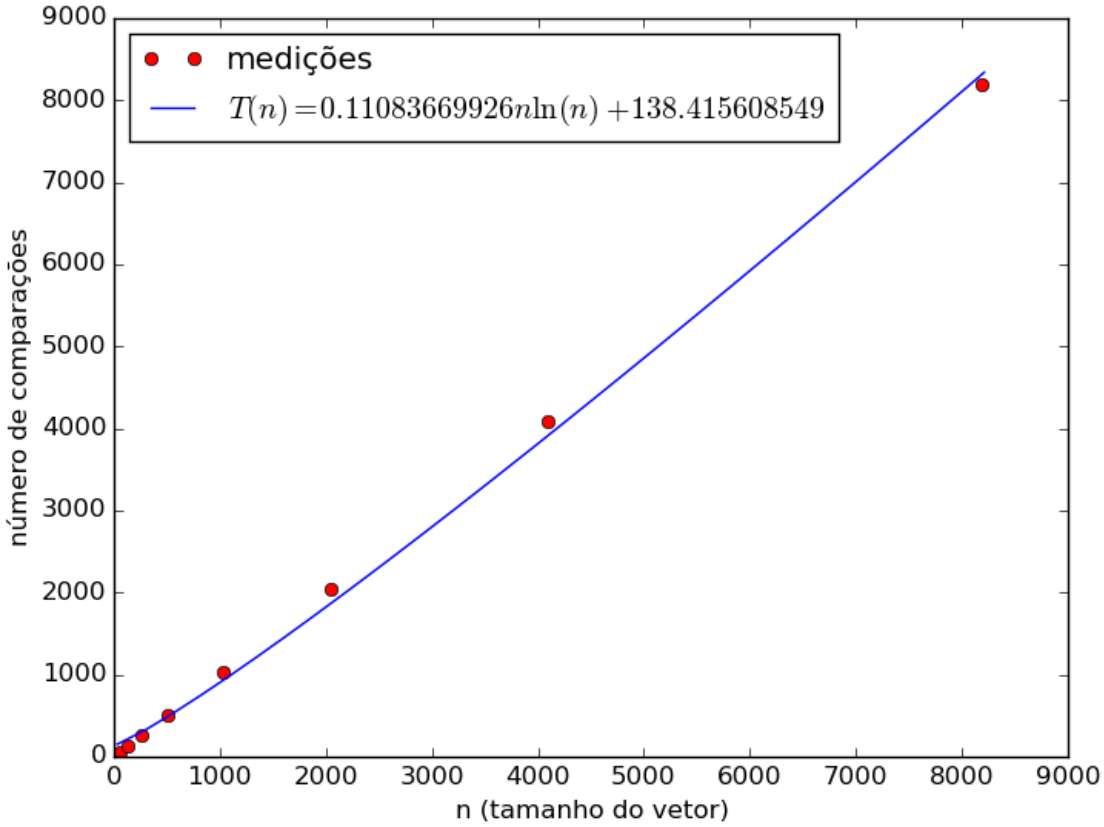

Figura 2.20: *A análise do grafico para* 2^{32} *segue abaixo para heapsort*
Tendo a função $T(n) = 0.110836\text{e} - 6 * n \log_2(n) + 138.4156$ e para o $n = 2^{32}$, $T(2^{32}) = 10697.3$

n	comparações	tempo(s)
32	31	0.000478
64	63	0.001113
128	127	0.002631
256	255	0.006168
512	511	0.015042
1024	1023	0.030395
2048	2047	0.072558
4096	4095	0.148326
8192	8191	0.324818

Tabela 2.11: *Tabela com vetor teste quase decrescente 30%: A linha te interesse analisada para este caso é a 16.*

n	comparações	tempo(s)
32	31	0.000472
64	63	0.001054
128	127	0.002621
256	255	0.006083
512	511	0.013541
1024	1023	0.031835
2048	2047	0.066935
4096	4095	0.148213
8192	8191	0.344303

Tabela 2.12: *Tabela com vetor teste quase decrescente 40%: A linha te interesse analisada para este caso é a 16.*

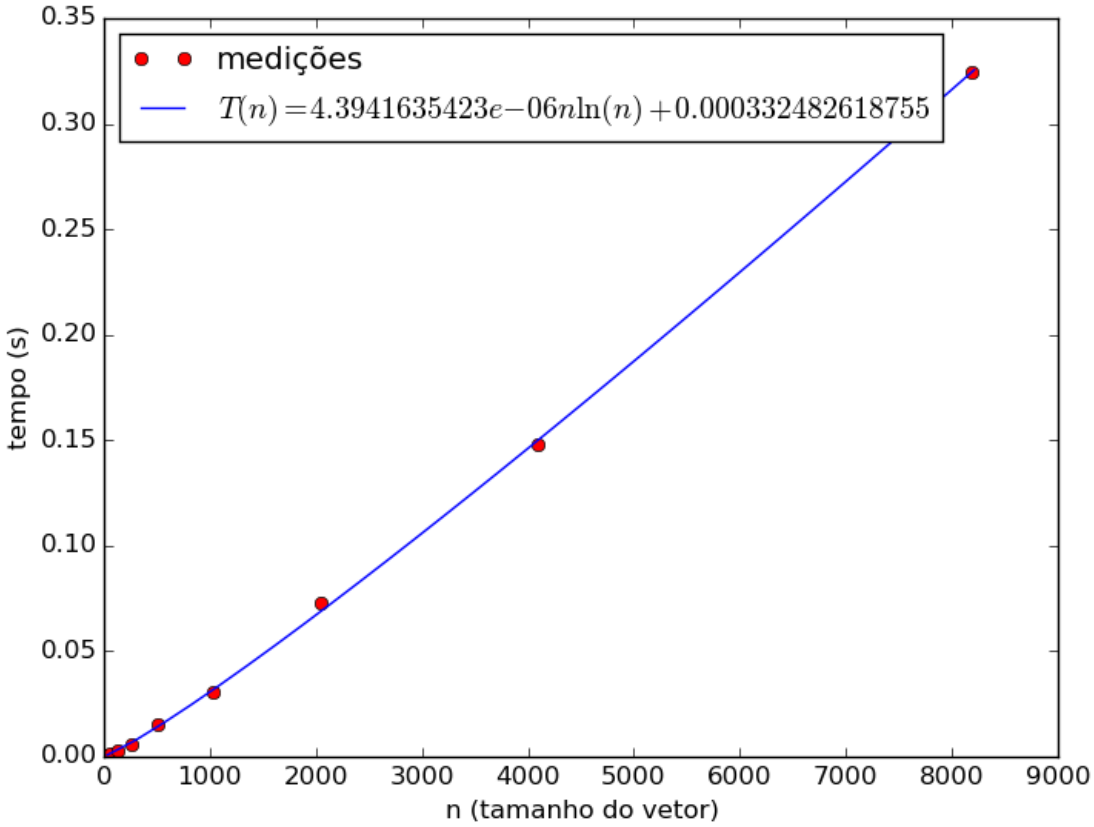

Figura 2.21: *A análise do grafico para* 2^{32} *segue abaixo para heapsort*
Tendo a função $T(n) = 4.43941\text{e} - 6 * n \log_2(n) + 0.0003324$ e para o $n = 2^{32}$, $T(2^{32}) = 4.22922 * 10^5$

n	comparações	tempo(s)
32	31	0.000469
64	63	0.001122
128	127	0.002575
256	255	0.006166
512	511	0.013633
1024	1023	0.030865
2048	2047	0.069581
4096	4095	0.147714
8192	8191	0.340304

Tabela 2.13: *Tabela com vetor teste quase decrescente 50%: A linha te interesse analisada para este caso é a 16.*

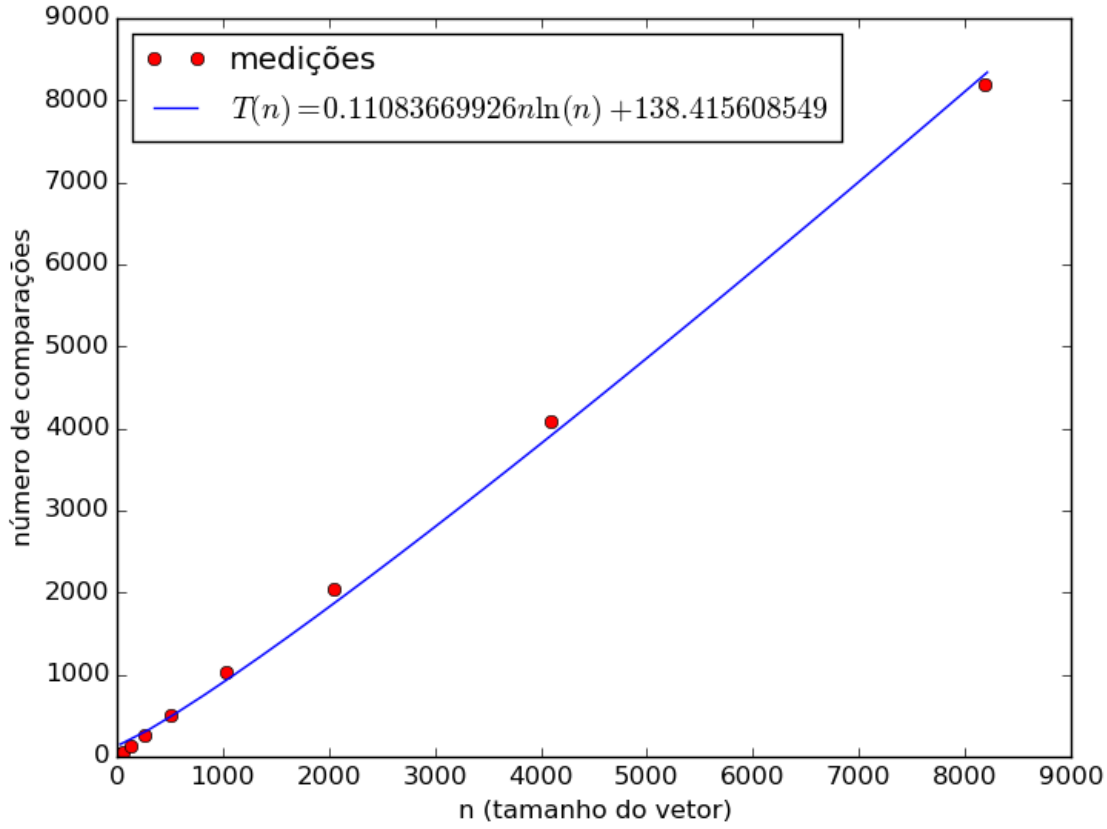

Figura 2.22: *A análise do grafico para 2^{32} segue abaixo para heapsort*
Tendo a função $T(n) = 0.11083\text{e} - 6 * n \log_2(n) + 138.4156$ e para o $n = 2^{32}$, $T(2^{32}) = 10696.7$

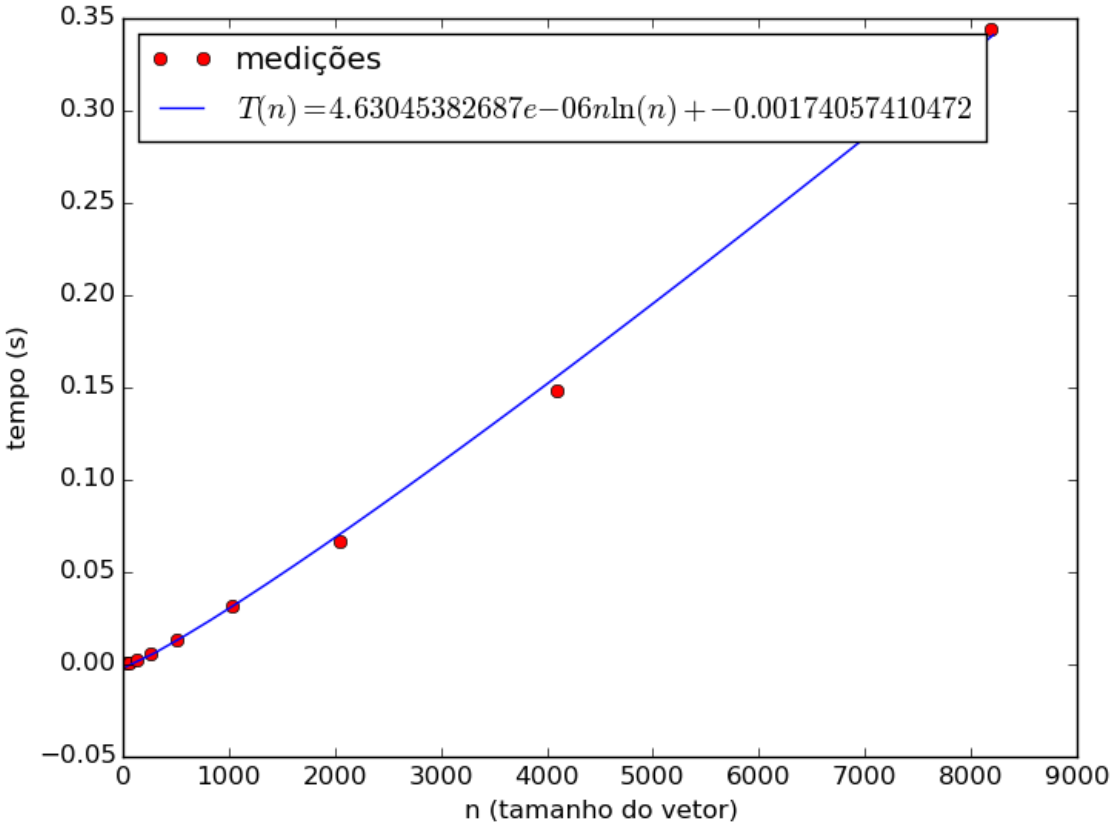

Figura 2.23: *A análise do grafico para* 2^{32} *segue abaixo para heapsort*
Tendo a função $T(n) = 4.63045\text{e} - 6 * n \log_2(n) - 0.0017405$ e para o $n = 2^{32}$, $T(2^{32}) = 4.22922 * 10^5$

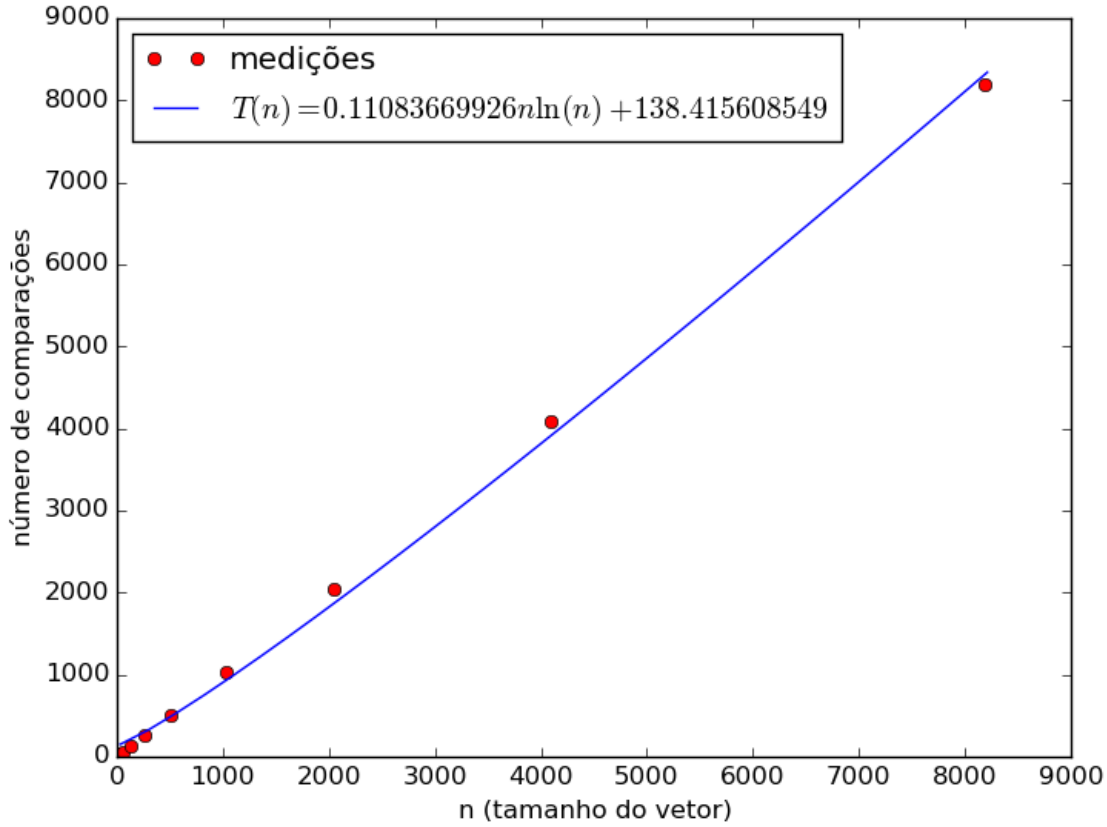

Figura 2.24: *A análise do grafico para 2^{32} segue abaixo para heapsort*
Tendo a função $T(n) = 0.11083\text{e} - 6 * n \log_2(n) + 138.41560$ e para o $n = 2^{32}$, $T(2^{32}) = 10696.7$

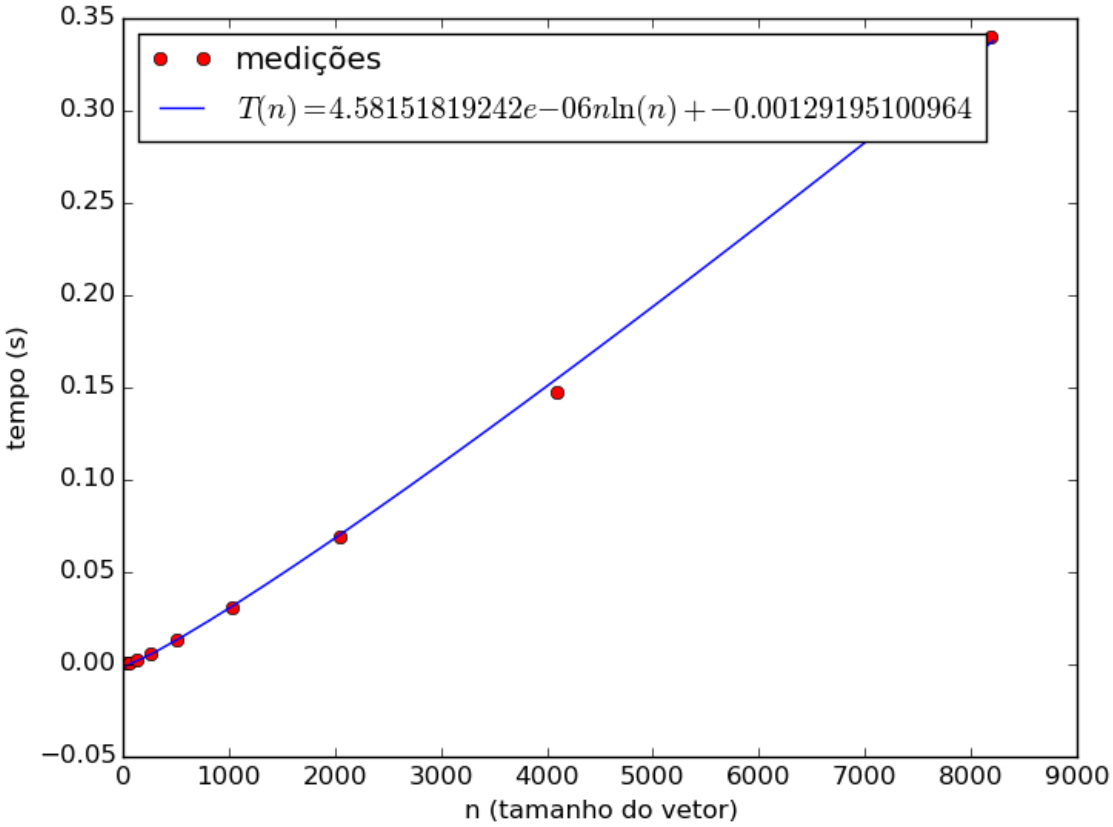

Figura 2.25: *A análise do grafico para* 2^{32} *segue abaixo para heapsort*
Tendo a função $T(n) = 4.58151\text{e} - 6 * n \log_2(n) - 0.001291$ e para o $n = 2^{32}$, $T(2^{32}) = 4.36459 * 10^5$

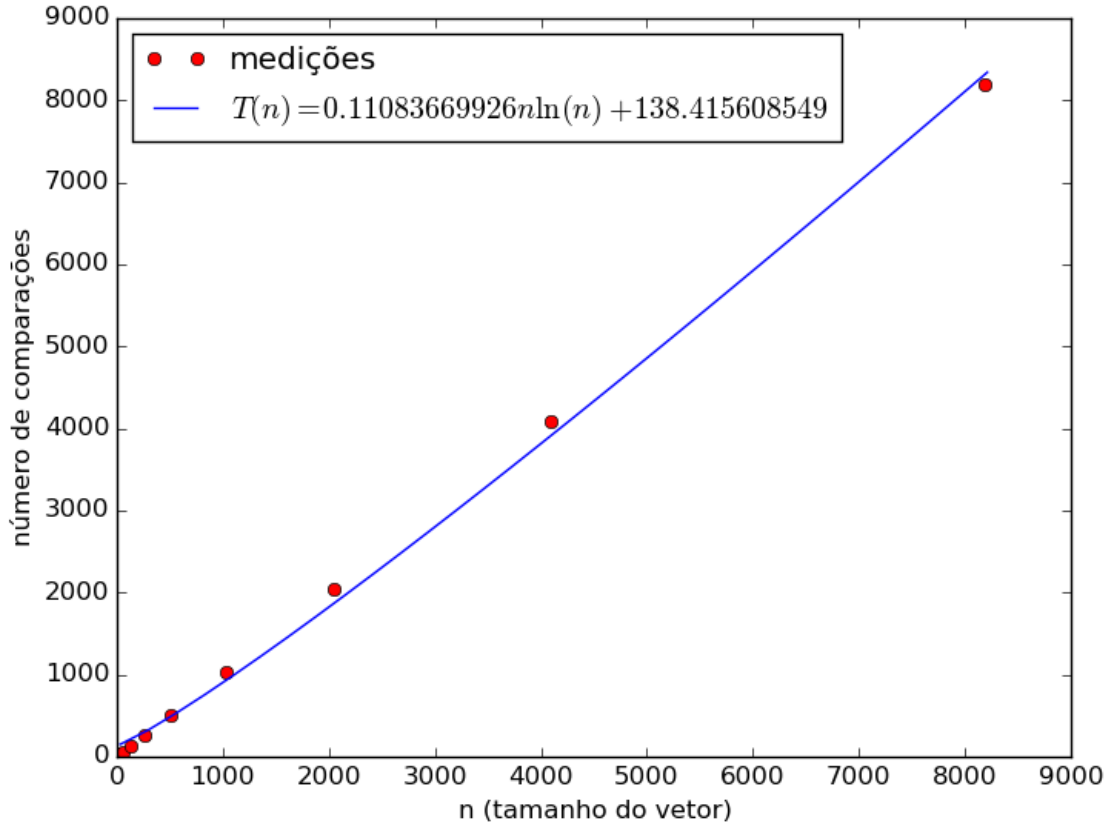

Figura 2.26: *A análise do grafico para 2^{32} segue abaixo para heapsort*
Tendo a função $T(n) = 0.11083\text{e} - 6 * n \log_2(n) + 138.4156$ e para o $n = 2^{32}$, $T(2^{32}) = 10696.7$

Apêndice A

Arquivo ../heapsort/heapsort.py

Listagem A.1: ../heapsort/heapsort.py

```
1  def trocaElementos(A,x,y):
2      aux = A[y]
3      A[y] = A[x]
4      A[x] = aux
5
6  def maxHeapify(A,n,i):
7      esquerda = 2*i + 1
8      direita = 2*i +2
9
10     if esquerda < n and A[esquerda] > A[i]:
11         maior = esquerda
12     else:
13         maior = i
14     if direita < n and A[direita] > A[maior]:
15         maior = direita
16     if maior!=i:
17         trocaElementos(A,i,maior)
18         maxHeapify(A,n,maior)
19
20 def constroiMaxHeap(A,n):
21     for i in range(n // 2, -1, -1):
22         maxHeapify(A,n,i)
23
24
25 @profile
26 def heapsort(A):
27     n = len(A)
28     constroiMaxHeap(A,n)
29     m = n
30     for i in range((n-1), 0, -1):
31         trocaElementos(A,0,i)
32         m = m - 1
33         maxHeapify(A,m,0)
```

Apêndice B

Arquivo ../heapsort/ensaio.py

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

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