Instituto Federal de Educação Ciência e Tecnologia
do Norte de Minas Gerais - IFNMG
Bacharelado em Ciência da Computação

Disciplina de Laboratório de PAA

Lab 2: Stooge Sort.

Thiago Emanuel Silva Antunes Lopes

Sumário

Capítulo 1

Introdução

Este relatório tem o objetivo de detalhar o funcionamento do algoritmo de ordenação Stooge Sort, analisando também sua complexidades e tempo de execução com gráficos, e por fim comparar com os resultados do Lab 1. A atividade foi proposta pelo Professor Alberto Miranda na disciplina Laboratório de Projeto de analise de algoritmos e realizado pelo discente Thiago Emanuel Silva Antunes Lopes.

Capítulo 2

Algoritmo de ordenação

Na atividade foi pedido a implementação do algoritmo Stooge Sort. A partir da implementação é feita a analise do algoritmo.
Primeiramente verifica se o primeiro e ultimo elemento estão na ordem, se não eles são trocados. Depois disso é realizada as chamadas recursivas.

2.1 Implementação

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4  #include <sys/time.h>
5  // LAB 2 PAA - THIAGO EMANUEL SILVA ANTUNES LOPES
6  // STOOGE SORT - 3 PATETAS
7
8  void stoogesort(int *vetor, int primeiro, int ultimo)
9  {
10     if(primeiro >= ultimo)
11         return;
12
13     //verifica se primeiro do vetor eh maior, se sim faz a troca
14     if(vetor[primeiro] > vetor[ultimo])
15     {
16         int auxiliar;
17         auxiliar = vetor[primeiro];
```

```
18          vetor[primeiro] = vetor[ultimo];
19          vetor[ultimo] = auxiliar;
20      }
21      //verifica se o vetor nao tem tamanho 1
22      if(ultimo - primeiro + 1 > 2)
23      {
24          //calcular tamanho de cada terco do vetor
25          int terco = (ultimo - primeiro + 1) / 3;
26          //chamar recursivamente para a primeira e segunda parte
27          stoogesort(vetor, primeiro, ultimo - terco);
28          //chamar recursivamente para a segunda e terceira parte
29          stoogesort(vetor, primeiro + terco, ultimo);
30          //chamar recursivamente novamente para a primeira e segunda
                parte
31          stoogesort(vetor, primeiro, ultimo - terco);
32      }
33  }
34
35  // FUNCAO TEMPO //
36  long double getNow()
37  {
38      struct timeval now;
39      long double valor = 1000000;
40
41      gettimeofday(&now,NULL);
42
43      return ((long double)(now.tv_sec*valor)+(long double)(now.tv_usec))
            ;
44  }
45
46  void temp(int *vetor)
47  {
48      long double t=0, ti, tf;
49      int i=10;
50      while(t<60)
51      {
```

```
52          vetor=(int *) malloc(i * sizeof (int));
53          for(int j=0;j<i;j++)
54          {
55              vetor[j]=rand()%100;
56          }
57          //for(int k=0;k<i; k++)
58              //printf("%d ", vetor[k]);
59
60          ti = getNow();
61
62          stoogesort(vetor,0,i-1);
63
64          tf = getNow();
65          t=(tf-ti)/1000000;
66              printf("\n %d elementos = %Lf segundos\n", i, t);
67          i=i*2;
68          free(vetor);
69      }
70      //for(int k=0;k<i; k++)
71          //printf("\n%d", vetor[k]);
72
73  }
74
75  int main()
76  {
77      srand(time(NULL));
78      int* vetor;
79      temp(vetor);
80      return 0;
81  }
```

2.2 Gráfico da execução do algoritmo

2.2.1 Stooge Sort

Tabela Stooge Sort:

Quantidade de elementos	Tempo (s)
10	0.000004
20	0.000015
40	0.000041
80	0.000403
160	0.002676
320	0.024167
640	0.056221
1280	0.515037
2560	4.501718
3060	4.517003
4060	14.719734
4560	15.409518
5060	14.946096
5560	44.471745
6060	43.930346
6560	44.286025
7060	44.185314
7560	45.201612
10240	132.068649

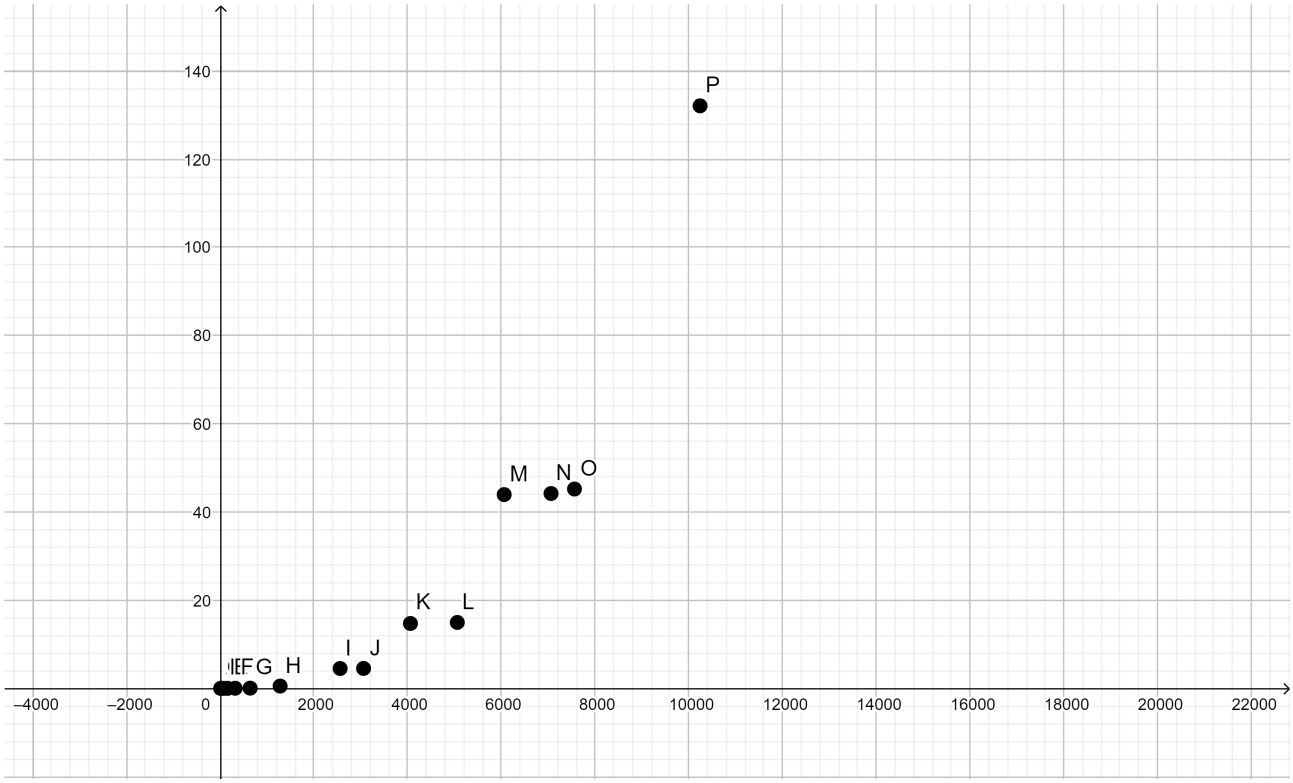

Figura 2.1: (Tamanho, Tempo(s))

2.3 Análise de complexidade

2.3.1 Stooge Sort

O Stooge Sort é o algoritmo conhecido como o algoritmo dos três patetas, pois para ordenar o vetor ele divide em três partes, chama recursivamente as duas primeras partes e ordena ela, depois chama recursivamente as duas ultimas partes, e por fim chama recursivamente as duas primeiras partes novamente, assim o vetor fica totalmente ordenado.

Provando (teorema master):

$$T(n) = 3 * T((2/3)n) + \Theta(1)$$

$$T(n) = a * T(n/b) + f(n)$$

$$a = 3$$

$$b = 3/2$$

$$f(n) = 1$$

Como o vetor só é ordenado quando ele tiver tamanho maior que 1, vamos então considerar n > 2 e vamos considerar e como euler, sendo maior que 0.
Assim temos:

$$n^{\log_{3/2} 3-e}$$

$$e = 1$$

$$n^{\log_{3/2} 2}$$

$$= n^{1.7}$$

Como $n^{1.7}$ é maior que 1 sempre, entramos no 1º caso do teorema master. Em que T(n) = O($n^{\log_b a}$), substituindo temos:
T(n) = O($n^{\log_{3/2} 3}$)
=O($n^{2.7}$).

2.4 Estimativa de tamanho do vetor para ordenação em 24 horas

Para fazer a estimativa de tempo de execução do algoritmo para qualquer tamanho é necessário se calcular uma constante que é multiplicada pela função do algoritmo. O exercício proposto foi para fazer a estimativa do tamanho do vetor que sua ordenação seja realizada em 24 horas para o algoritmo de ordenação.

2.4.1 Stooge Sort

A partir dos dados tabelados do Stooge, vamos calcular sua constante. Para isso, vamos utilizar o tempo para ordenar 3 vetores, e fazer a média das constantes.
Para vetor de tamanho 5060:

$$Const * 5060^{2,7} = 14.946096$$

$$Const = 1,490464 * 10^{-9}$$

Para vetor de tamanho 6060:

$$Const * 6060^{2,7} = 43.930346$$
$$Const = 2,692084 * 10^{-9}$$

Para vetor de tamanho 7060:

$$Const * 7060^{2,7} = 44.185314$$
$$Const = 1,79269 * 10^{-9}$$

Fazendo a média:

$$Const = \frac{1,490464 * 10^{-9} + 2,692084 * 10^{-9} + 1,79269 * 10^{-9}}{3}$$
$$Const = 1,991746 * 10^{-9}$$

Com a constante calculada, agora vamos fazer a estimativa do tamanho do vetor que sua ordenação leve 24 horas.
Como foi utilizado o tempo em segundos, vamos transformar as 24 horas em segundos. Isso dá um total de 86400 segundos.
A equação fica:

$$1,991746 * 10^{-9} * n^{2,7} = 86400$$
$$n^{2,7} = 42475295544713,03$$
$$n = 111547,00892702943$$

Em 24 horas esse algoritimo consegue ordenar um vetor de tamanho até 111546.
Comparando esse resultado se pode concluir que esse algoritimo é muito mais lento que os dois outros do Lab 1, que são o merge e heap, que em 24 horas ordenam vetores de tamanho $5,293996 * 10^{11}$ e $2,073633279 * 10^{11}$ respectivamente.

2.4.2 Gráfico das funções

Gráfico das funções multiplicadas pelas constantes. Stooge em preto, heap em vermelho e merge em azul.

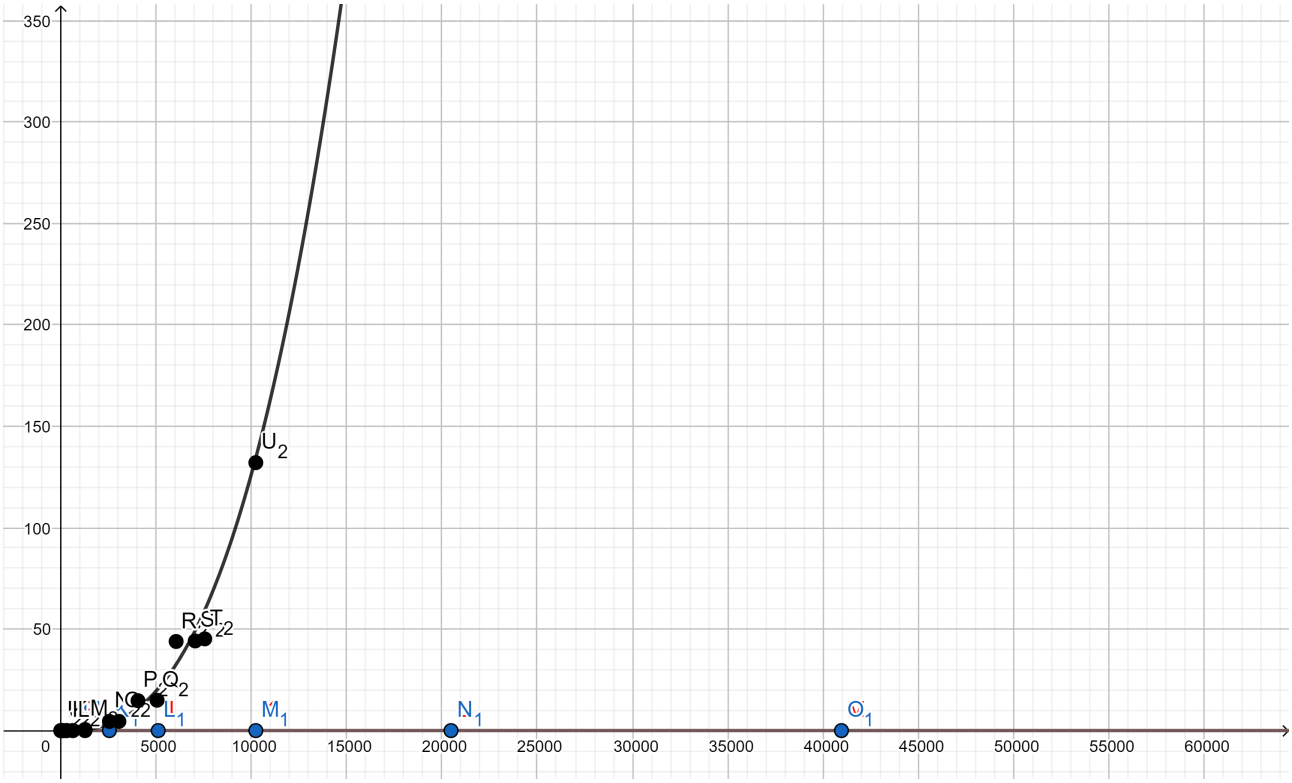

Figura 2.2: (Tamanho, Tempo(s)

2.4.3 Gráfico para estimativas

Gráfico de execução dos algorítimos em 24 horas (Stooge)

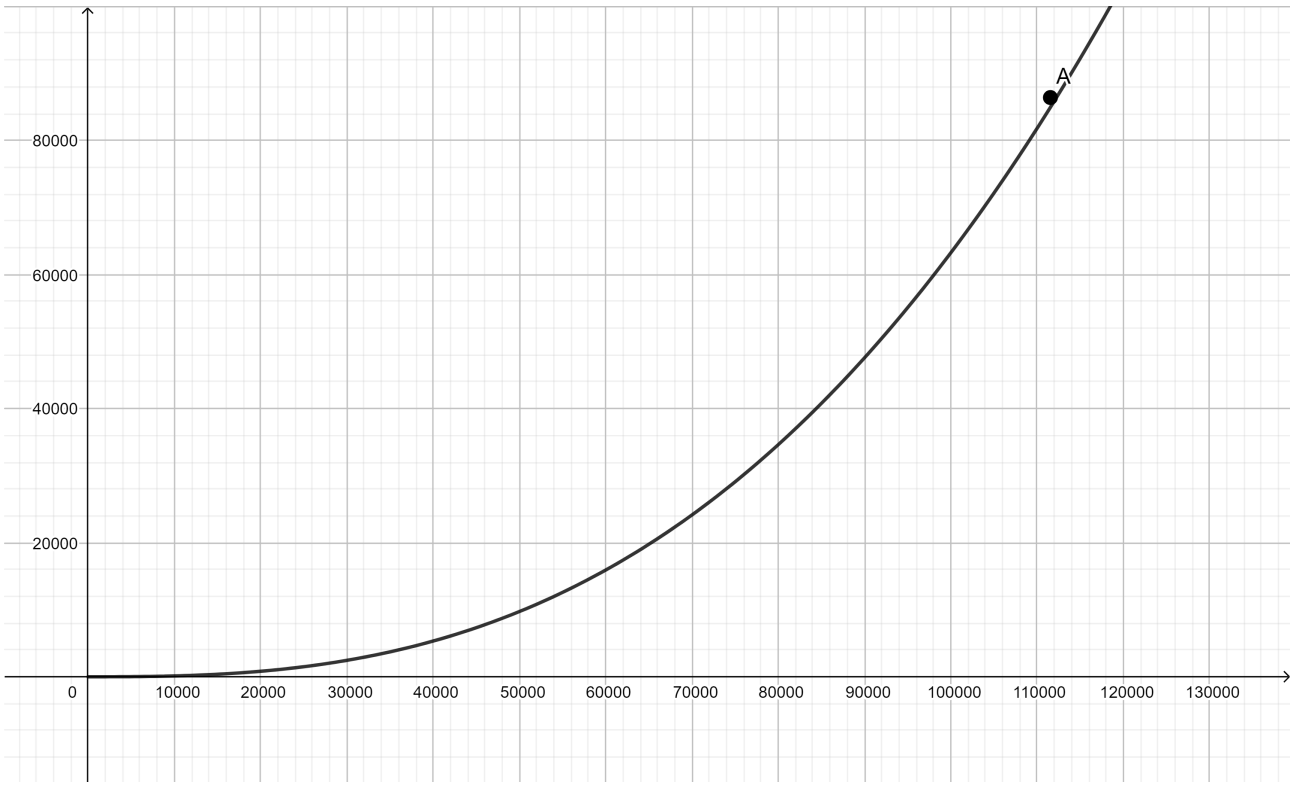

Figura 2.3: (Tamanho, Tempo(s)

Gráfico de execução dos algorítimos em 24 horas (Stooge - preto, Heap - vermelho, Merge - azul)

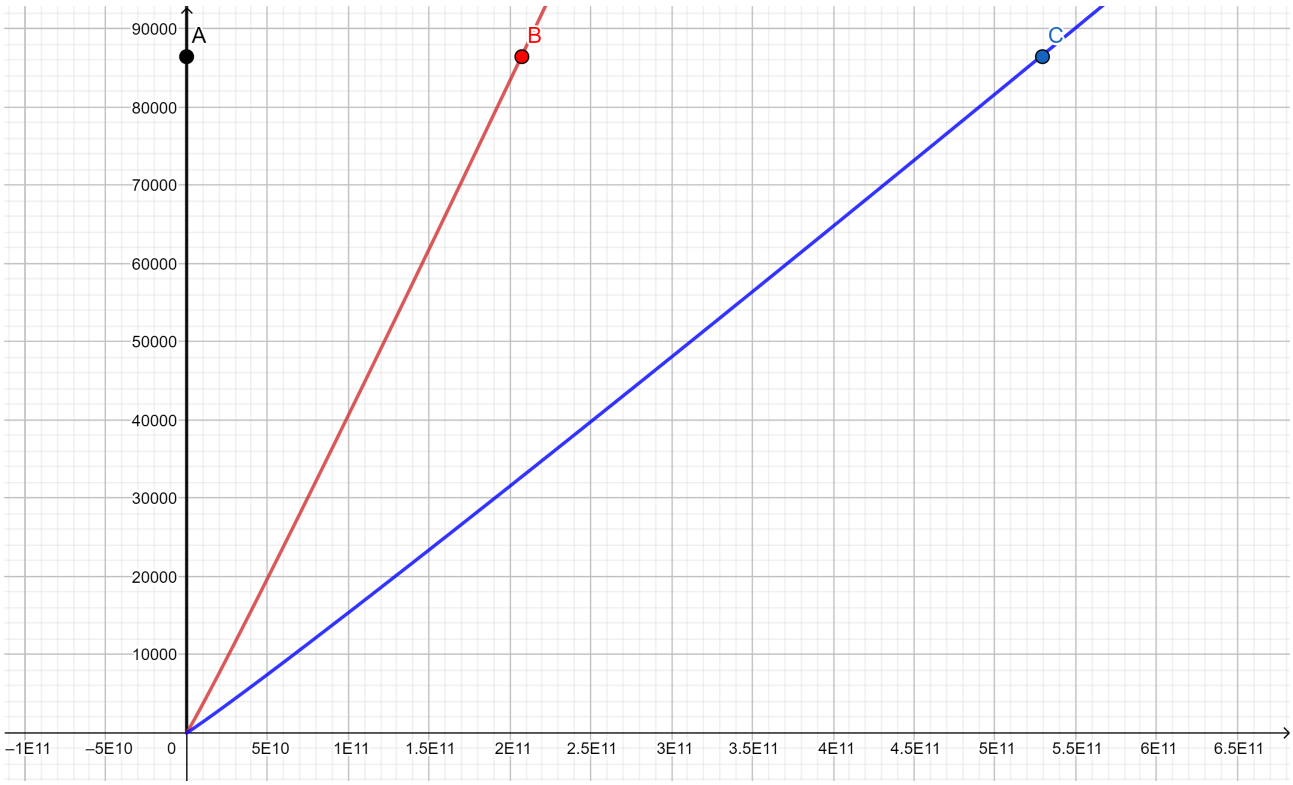

Figura 2.4: (Tamanho, Tempo(s)