UNIVERSIDADE FEDERAL DO TOCANTINS
CAMPUS UNIVERSITÁRIO DE PALMAS
CURSO DE CIÊNCIA DA COMPUTAÇÃO
RELATÓRIO DE PROJETO ANÁLISE DE ALGORITMO

METODO DE ORDENAÇÃO

Náyron dos Anjos Seilert
John Lennon Soares de Souza

Orientador: M.Sc. Warley Gramacho

Palmas
Abril de 2017

Resumo

O meta desse artigo é descrever e analisar o desempenho dos algoritmos de ordenação: SelectionSort, InsertionSort, BubbleSort, ShellSort, QuickSort e MergeSort. A análise consiste em comparar os algoritmos considerando três métricas de desempenho: número de comparações de chaves, número de trocas realizados e o tempo total gasto para ordenação.

Palavra-chave: Bubble. UFTEX. Métodos. Ordenação. Merge.

Abstract

The goal of this article is to describe and analyze the performance of sorting algorithms: SelectionSort, InsertionSort, BubbleSort, ShellSort, QuickSort, and MergeSort. The analysis consists of comparing the algorithms considering three performance metrics: number of key comparisons, number of exchanges performed and total time spent for sorting.

Keywords: Bubble. UFTEX. methods. sorting. Merge.

Sumário

1 Introdução

A ordenação consiste em arranjar os elementos de um conjunto de modo a facilitar a posterior recuperação ou análise dos dados. Por ser tão importante a ordenação é uma das atividades mais utilizadas na computação. Existem diversos métodos para dispor os dados da melhor forma, para uma subsequente consulta, análise ou remoção de algum item, se for conveniente. Neste trabalho iremos abordar alguns métodos de ordenação interna, são eles: Bubble Sort, Insetion Sort, SelectionSort, Shell Sort, Quick Sort e o Merge Sort. Serão apresentados testes com vetores de palavras, de diferentes tamanhos (100, 1000, 10000,50000 e 100000) e tipos (ordenados em ordem crescente e decrescente e randômico).

1.1 Métodos de Ordenação

Durante a escolha de um algoritmo de ordenação, deve-se observar uns aspectos importantes: o tempo gasto durante a sua execução. Para algoritmos de ordenação interna, as medidas de complexidade relevantes contam o número de comparações entre as chaves e o número de movimentações de itens do arquivo.

Podemos separar os métodos de ordenação em dois grupos: Métodos simples - Bubble Sort, Insertion Sort, Selection Sort. Outro grupo e os métodos eficientes - Shell Sort, Quick Sort, Merge Sort.

1.1.1 Bubble Sort

É o método mais simples em termos de implementação, porém é o menos eficiente. A idéia principal do algoritmo é percorrer o vetor n-1 vezes, a cada passagem fazendo flutuar para o inicio o menor elemento da sequencia. Essa movimentação lembra a forma coma as bolhas procuram seu próprio nível, por isso o nome do algoritmo. Seu uso não é recomendado para vetores com muitos elementos.

No melhor caso, o algoritmo executa n operações relevantes, onde n representa o número de elementos do vetor. No pior caso, são feitas $O(n^2)$ operações. A complexidade desse algoritmo e de ordem quadrática. Por isso, ele não é recomendado para programas que precisem de velocidade e operem com quantidade elevada de dados.

Vantagens: é um algoritmo de fácil implementação, algoritmo estável. Desvantagens: O fato de o arquivo estar ordenado não ajuda em nada, ordem de complexidade quadrática. Logo abaixo temos um gráfico comparando o tempo de CPU dos vetores randômico, ordenando e crescente e invertido.

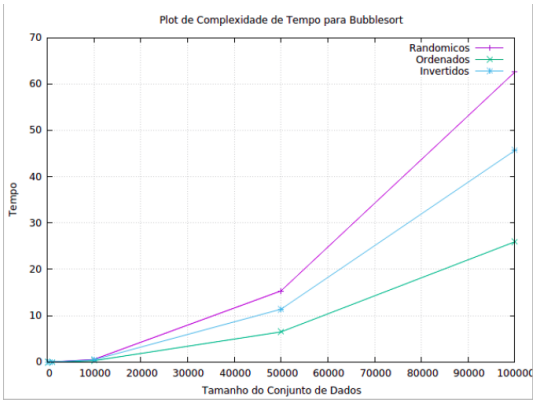

Figura 1.1: *Gráfico de Desempenho do Bubble Sort*

Tabela 1.1: *Vetor em Ordem Randômicos*

Tamanho	Tempo CPU	Comparações	Trocas
100	0.000074	4950	2334
1000	0.004377	499500	254452
10000	0.532368	49995000	24869446
50000	15.331760	1249975000	623999240
100000	62.561832	4999950000	2491777535

Tabela 1.2: *Vetor Ordenado Crescente*

Tamanho	Tempo CPU	Comparações	Trocas
100	0.000028	4950	0
1000	0.002597	499500	0
10000	0.264488	49995000	0
50000	6.486772	1249975000	0
100000	25.934586	4999950000	0

Tabela 1.3: *Vetor Ordenado Inverso*

Tamanho	Tempo	Comparações	Trocas
100	0.000049	4950	4943
1000	0.005810	499500	499005
10000	0.470248	49995000	49945311
50000	11.380366	1249975000	1248726361
100000	45.641796	4999950000	4994953036

No pior caso, quando o vetor é decrescente, o algoritmo precisará percorrer o vetor inteiro e irá fazer o máximo de trocas possıveis, diminuindo consideravelmente o desempenho. As posições dos elementos no bubbleSort desempenham um papel fundamental para determinar a performance da execução do algoritmo.

1.1.2 Insertion Sort

O Insertion Sort é eficiente quando aplicado à um vetor com poucos elementos. Em cada passo, a partir de i=2, o i-ésimo item da sequência fonte é apanhado e transferido para a sequência destino, sendo inserido no seu lugar apropriado. O algoritmo assemelha-se com a maneira que os jogadores de cartas ordenam as cartas na mão em um jogo, como o pôquer, por exemplo.

A complexidade do insertion Sort no melhor caso é de $\Omega(n)$ de acordo com[Ziviani 2004], visto que ele irá fazer n comparações e ver que o vetor já está ordenado. No pior caso e no no caso médio a complexidade do insertion Sort é de $\Theta(n^2)$.

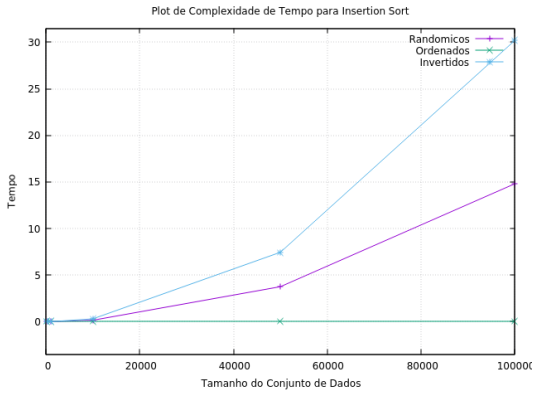

Figura 1.2: *Gráfico de desempenho do Insertion Sort*

Através do gráco percebe-se o desempenho incrivelmente superior para os vetores ordenados, já que o algoritmo tem complexidade de Θ (n) no melhor caso. Considerando que o vetor já está ordenado, a execução apenas compara todos os elementos com o elemento anterior, e não faz nenhuma troca. Dados mostrado nas tabelas abaixo:

Tabela 1.4: *Vetor em Ordem Randômica*

Tamanho	Tempo	Comparações	Trocas
100	0.000019	2334	2433
1000	0.001756	254452	255451
10000	0.149007	24869446	24879445
50000	3.753903	623999240	624049239
100000	14.816816	2491777535	2491877534

Tabela 1.5: *Vetor Ordenado Crescente*

Tamanho	Tempo	Comparações	Trocas
100	0.000002	99	0
1000	0.000008	999	0
10000	0.000089	9999	0
50000	0.000383	49999	0
100000	0.000750	999999	0

Tabela 1.6: *Vetor Ordenado Invertido*

Tamanho	Tempo	Comparações	Trocas
100	0.000046	4943	5042
1000	0.003015	499005	500004
10000	0.302580	49945311	49955310
50000	7.450999	1248726361	1248776360
100000	30.207895	4994953036	4995053035

No vetor invertido (pior caso) ocorre o contrário do vetor crescente, já que a complexidade é de $\theta(n^2)$ o algoritmo compara todos os elementos do vetor com o elemento anterior e faz todas as trocas possíveis, assim o desempenho cai consideravelmente, por exemplo, no vetor crescente com 100000 elementos o tempo de execução foi de 0.000758 segundos enquanto no invertido o tempo subiu para 30.059 segundos.

No vetor aleatório, que possui desempenho $\theta(n^2)$ foi analisado através do gráfico que o desempenho ca entre o crescente e o decrescente.

1.1.3 Selection sort

O Selection Sort tem como principio de funcionamento selecionar o menor item do vetor e a seguir trocá-lo pela primeira posição do vetor. Isto ocorre para os n-1 elementos restantes, depois com os n-2 itens, até que reste apenas um elemento. A principal diferença destes métodos em relação ao Bubble Sort e ao Insertion Sort é que ele realiza apenas uma troca por interação. Vejamos abaixo o gráfico e a tabela com os resultados.

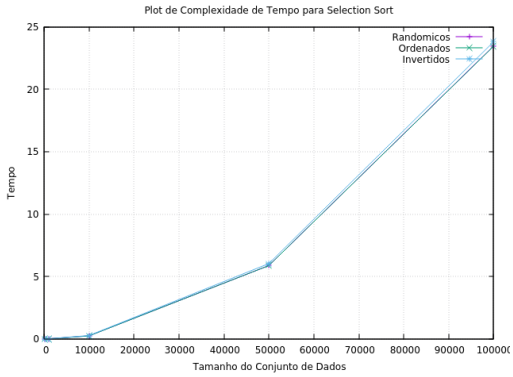

Figura 1.3: *Gráfico de desempenho do Selection Sort*

Tabela 1.7: *Vetor em Ordem Randômica*

Tamanho	Tempo	Comparações	Trocas
100	0.000070	4950	95
1000	0.004793	499500	992
10000	0.239561	49995000	9987
50000	5.867622	1249975000	49948
100000	23.466951	4999950000	99879

Tabela 1.8: *Vetor Ordenado Crescente*

Tamanho	Tempo	Comparações	Trocas
100	0.000035	4950	0
1000	0.002769	499500	0
10000	0.237034	49995000	0
50000	5.871239	1249975000	0
100000	23.442564	4999950000	0

Tabela 1.9: *Vetor Ordenado Invertido*

Tamanho	Tempo	Comparações	Trocas
100	0.000033	4950	54
1000	0.002763	499500	632
10000	0.267355	49995000	7089
50000	6.031119	1249975000	36791
100000	23.805443	4999950000	74944

O algoritmo selection sort tem complexidade $\theta(n^2)$ em todos os casos, então o desempenho vai ser bastante parecido em todos os tipos de vetores. A principal vantagem do

selection sort é que ela funciona bem em uma lista pequena. Além disso, por ser um algoritmo de ordenação de local, não precisa de armazenamento temporário além do necessário para guardar a lista original. A principal desvantagem é sua baixa eficiência em listas grandes. Podemos observar também que o fato de o arquivo já estar ordenado não ajuda em nada, ordem de complexidade quadrática, algoritmo não estável.

1.1.4 Shell Sort

Este algoritmo é uma extensão do método Insertion Short proposto por Donald Shellem 1959. O algoritmo de inserção troca itens adjacentes quando está procurando oponto de inserção na sequência destino. Se o menor item estiver na posição mais à direita no vetor, então o número de comparações e movimentações é igual a n-1para encontrar o seu ponto de inserção. O Shell Sort contorna este problema,permitindo trocas de registros distantes um do outro. De maneira geral ele passa várias vezes no vetor dividindo-o em vetores menores, e nestes são aplicados o algoritmo de ordenação por inserção tradicional. Dentre os programas de ordenação interna que tem ordem de complexidade quadrática,o ShellSort é o mais eficiente.

O metodo possui vantagens por que tem o código simples, interessante para arquivos de tamanho moderado. A desvantagens é que o algoritmo é não estável, tempo de execução sensível à ordem inicial do arquivo.

Figura 1.4: *Gráfico de desempenho do Shell Sort*

Tabela 1.10: *Vetor em Ordem Randômica*

Tamanho	Tempo	Comparações	Trocas
100	0.000027	2746	439
1000	0.000533	60421	8503
10000	0.009999	1215982	134790
50000	0.078373	11114824	898875
100000	0.183851	26417757	1976805

Tabela 1.11: *Vetor Ordenado Crescente*

Tamanho	Tempo	Comparações	Trocas
100	0.000004	505	0
1000	0.000089	8009	0
10000	0.000634	120009	0
50000	0.003578	700010	0
100000	0.007660	1500011	0

Tabela 1.12: *Vetor Ordenado Invertido*

Tamanho	Tempo	Comparações	Trocas
100	0.000010	1061	247
1000	0.000143	18501	4261
10000	0.001775	264386	52210
50000	0.009926	1571799	271989
100000	0.020669	3343336	543324

1.1.5 Merge Sort

É outro algoritmo de ordenação do tipo dividir para conquistar. Sua ideia básica é criar uma sequência ordenada a partir de duas outras também ordenadas. Para isso, ele divide a sequência original em pares de dados, ordena-as; depois as agrupa em sequencial de quatro elementos, e assim por diante, até ter toda a sequência dividida em apenas duas partes.

Os três passos úteis dos algoritmos dividir-para-conquistar, que se aplicam ao merge sort são:

- Dividir: Dividir os dados em subsequências pequenas;
- Conquistar: Classicar as duas metades recursivamente aplicando o merge sort;
- Combinar: Juntar as duas metades em um único conjunto já classicado.

Vantagens: Passível de ser transformado em estável, fácil implementação, complexidade O(nlogn).

Desvantagens: utiliza memória auxiliar

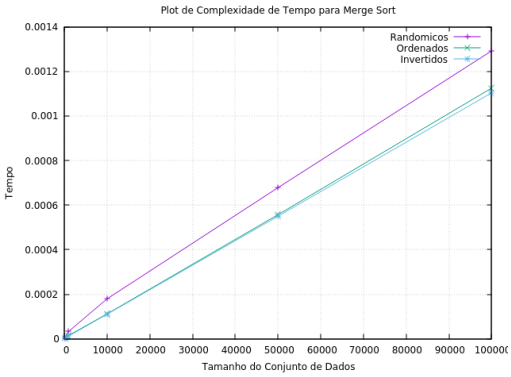

Figura 1.5: *Gráfico de desempenho do Merge Sort*

Tabela 1.13: *Vetor em Ordem Randômica*

Tamanho	Tempo	Comparações	Trocas
100	0.000004	780	1360
1000	0.000046	11767	21334
10000	0.000181	155398	288596
50000	0.000703	989879	1857558
100000	0.001345	2758825	5195450

Tabela 1.14: *Vetor Ordenado Crescente*

Tamanho	Tempo	Comparações	Trocas
100	0.000002	2759605	5196810
1000	0.000015	2770592	5216784
10000	0.000112	2914223	5484046
50000	0.000556	3748704	7053008
100000	0.001121	5517650	10390900

Tabela 1.15: *Vetor Ordenado Invertido*

Tamanho	Tempo	Comparações	Trocas
100	0.000002	5518430	10392260
1000	0.000020	5529417	10412234
10000	0.000110	5673048	10679496
50000	0.000552	6507529	12248458
100000	0.001103	8276475	15586350

1.1.6 Quick Sort

É o algoritmo mais rápido que se conhece entre os de interna para ampla variedade de situações. Foi escrito em 1960 e publicado em 1962 por C.A.Hoare após vários refinamentos . Adotando a estratégia dividir para conquistar o funcionamento resume-se a dividir o problema de ordenar um vetor de n posições em dois outros menores. O quick sort possui um tempo de execução de pior caso $\Theta(n^2)$ sobre um arranjo de n números. Apesar der possuir um tempo de execução lento no pior caso, o quick sort com frequência é a melhor opção prática para ordenação, devido a sua notável eciência na média o seu tempo de execução esperado é $O(nlogn)$.

O quick sort baseia-se na divisão do vetor em dois sub-vetores, dependendo de um elemento chamado pivô, normalmente o 1º elemento do vetor. Um dos sub-vetores contém os elementos menores que o pivô enquanto a outra contém os maiores. O pivô é colocado

entre ambas, ficando na posição correta. Os dois sub-vetores são ordenadas de forma idêntica, até que se chegue à um vetor com um só elemento. A Complexidade do caso médio é de O(nlogn)

Vejamos abaixo os gráfico de desempenho e as tabelas contendo dados sobre o tempo CPU, comparações e trocas para cada tipo de pivô escolhido(Extremidade da direita, centro e esquerda).

Pivô - Elemento da Direita

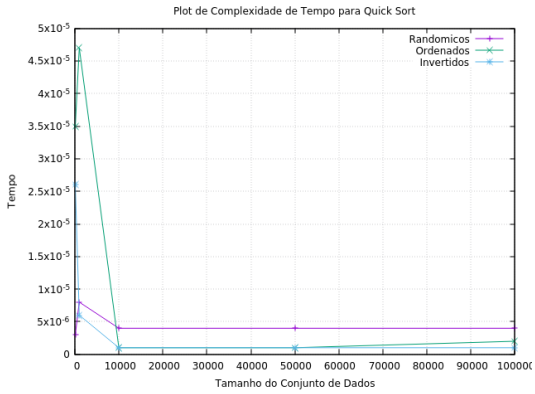

Figura 1.6: *Gráfico de desempenho do Quick Sort*

Tabela 1.16: *Vetor em Ordem Randômica*

Tamanho	Tempo	Comparações	Trocas
100	0.000003	605	192
1000	0.000008	8183	2957
10000	0.000004	107684	41276
50000	0.000004	591520	286937
100000	0.000004	1604961	822945

Tabela 1.17: *Vetor Ordenado Crescente*

Tamanho	Tempo	Comparações	Trocas
100	0.000035	1609904	823044
1000	0.000047	2071079	824057
10000	0.000001	21058476	840259
50000	0.000001	172176084	969267
100000	0.000002	520903953	1274832

Tabela 1.18: *Vetor Ordenado Invertido*

Tamanho	Tempo	Comparações	Trocas
100	0.000026	520908670	1274935
1000	0.000006	521282843	1276157
10000	0.000001	532834620	1296495
50000	0.000001	620379304	1447052
100000	0.000001	822085783	1795911

Pivô - Elemento central

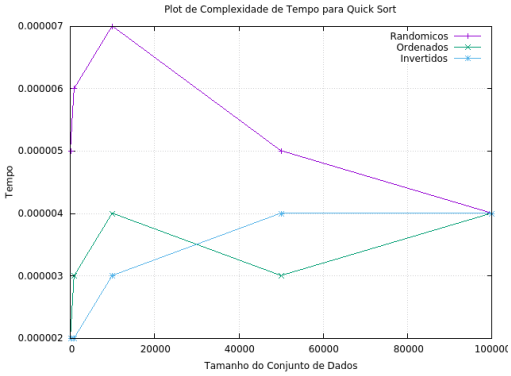

Figura 1.7: *Gráfico de desempenho do Quick Sort*

Tabela 1.19: *Vetor em Ordem Randômica*

Tamanho	Tempo	Comparações	Trocas
100	0.000005	488	192
1000	0.000006	6943	2924
10000	0.000007	87025	41778
50000	0.000005	562845	285285
100000	0.000004	1471410	821246

Tabela 1.20: *Vetor Ordenado Crescente*

Tamanho	Tempo	Comparações	Trocas
100	0.000002	1471896	821309
1000	0.000003	1479921	822106
10000	0.000004	1578971	838306
50000	0.000003	2093321	967314
100000	0.000004	3129483	1272879

Tabela 1.21: *Vetor Ordenado Invertido*

Tamanho	Tempo	Comparações	Trocas
100	0.000002	3129875	1272991
1000	0.000002	3136899	1274285
10000	0.000003	3225951	1295478
50000	0.000004	3690370	1449470
100000	0.000004	4626608	1804975

Pivô - Elemento Esquerdo

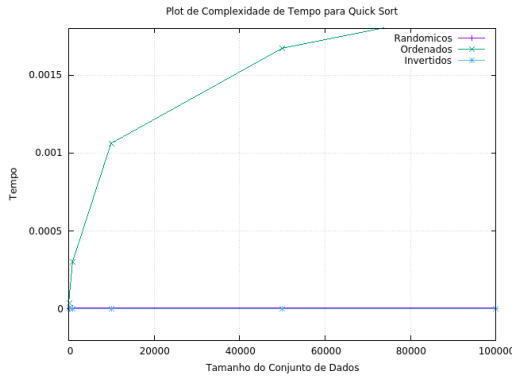

Figura 1.8: *Gráfico de desempenho do Quick Sort*

Tabela 1.22: *Vetor em Ordem Randômica*

Tamanho	Tempo	Comparações	Trocas
100	0.000003	1653625388	3587234
1000	0.000005	1653633108	3589951
10000	0.000004	1653731417	3628475
50000	0.000003	1654232108	3872310
100000	0.000004	1655239756	4407578

Tabela 1.23: *Vetor Ordenado Crescente*

Tamanho	Tempo	Comparações	Trocas
100	0.000040	1655244699	4407677
1000	0.000303	1655698809	4408690
10000	0.001061	1674891948	4424892
50000	0.001669	1824972310	4553900
100000	0.001943	2175744177	4859465

Tabela 1.24: *Vetor Ordenado Invertido*

Tamanho	Tempo	Comparações	Trocas
100	0.000008	2175748936	4859567
1000	0.000000	2176122376	4860779
10000	0.000001	2187936250	4881218
50000	0.000002	2276479281	5031986
100000	0.000002	2480437317	5380572

Como podemos observar o quick sort utilizando o um elemento central do array obteve ótimos resultados em relação aos elementos da extrema direita e da extrema esquerda. O quick sort na três tipos de pivô utilizado, apresentou excelente resultados sobre os outros métedos abordados nesse trabalho.

1.1.7 Conclusão

Feitas todas as análises pode-se concluir que o Quick sort se mantem constante para todos os tamanhos, sendo mais eficiente. O algoritmo Insertion sort quando o vetor já se encontra ordenando crescente, apresenta bons resultados. Entre os algoritmos de complexidade $O(n^2)$, o insertion sort é melhor para todos os tamanhos experimentados. O Bubble sort foi que obteve os péssimos resultados para vetores de tamanho grandes..

1.2 Código-fonte

O projeto poderá ser baixado no seguinte link:
https://github.com/nayron/OrdenacaoPAA

1.2.1 Arquivo ordena.h

```
1
2  /* Codigo com a funcao de Ordenacao */
3  #ifndef ORDENA_H_INCLUDED
4  #define ORDENA_H_INCLUDED
5  #include <time.h>
6
7  clock_t tempoInicio, tempoFim;
8  FILE *gnuplot;
9  unsigned long  troca=0,comparacao=0;
10 /*-----------------------------------------------*/
11 int *LendoArquivo(char *nomeArq, long int *tamanho)
12 {
```

```
13      FILE *arquivo;
14      int *vetor;
15      long int i, LinhasdoArquivo;
16      char linha[200];
17      //abre arquivo e descobre quantidade de numeros para o vetor
18      arquivo = fopen(nomeArq,"r");
19      if (!arquivo){
20          printf("Arquivo nao existe\n");
21          exit(0);
22      }
23      LinhasdoArquivo = 0;
24      while (fgets(linha,200,arquivo))
25      {
26          LinhasdoArquivo++;
27      }
28      fclose(arquivo);
29      *tamanho = LinhasdoArquivo;
30      //escreve o arquivo o tamanho do vetor
31      fprintf( gnuplot, "%d\t\t", LinhasdoArquivo);
32      vetor = malloc(sizeof(int) * LinhasdoArquivo);
33
34      arquivo = fopen(nomeArq,"r");
35      i = 0;
36      while (fgets(linha,200,arquivo))
37      {
38          //percorrer arquivo para gravar numeros no vetor
39          vetor[i] = atoi(linha);
40          i++;
41      }
42      fclose(arquivo);
43      return vetor;
44  }
45  /*-----------------------------------------------*/
46  void BubbleSort(int *vetor, long int n)
47  {
48      unsigned long long int i, j;
49      troca = 0;
50      comparacao = 0;
51      tempoInicio = clock();
52      for (j=0; j<n-1; j++)
53      {
```

```
54          for (i=0; i<n-(j+1); i++)
55          {
56              comparacao++;
57              if (vetor[i]>vetor[i+1])
58              {
59                  int aux = vetor[i+1];
60                  vetor[i+1] = vetor[i];
61                  vetor[i]= aux;
62                  troca++;
63              }
64          }
65      }
66      tempoFim = clock();
67  }
68  /*------------------------------------------------*/
69  void shellSort(int *Vetor, long int tamanho)
70  {
71      long int i = (tamanho - 1) / 2;
72      int chave, k, aux;
73      troca = 0;
74      comparacao = 0;
75      tempoInicio = clock();
76      while(i != 0)
77      {
78          do
79          {
80              chave = 1;
81              for(k = 0; k < tamanho - i; ++k)
82              {
83                  comparacao++;
84                  if(Vetor[k] > Vetor[k + i])
85                  {
86                      aux = Vetor[k];
87                      Vetor[k] = Vetor[k + i];
88                      Vetor[k + i] = aux;
89                      chave = 0;
90                      troca++;
91                  }
92              }
93          }
94          while(chave == 0);
```

```
95          i = i / 2;
96      }
97       tempoFim = clock();
98  }
99  /*------------------------------------------------*/
100 void selectSort(int *vetor,long int n)
101 {
102     long int i, j, posMenor;
103     int aux;
104     troca = 0;
105     comparacao = 0;
106
107     tempoInicio = clock();
108     for (i = 0; i < n - 1; i++)
109     {
110         posMenor = i;
111         for (j = i + 1; j < n; j++)
112         {
113             comparacao++;
114             if (vetor[j] < vetor[posMenor]) posMenor = j;
115         }
116
117         if (posMenor != i)
118         {
119             troca++;
120             aux = vetor[i];
121             vetor[i] = vetor[posMenor];
122             vetor[posMenor] = aux;
123         }
124     }
125     tempoFim = clock();
126 }
127 /*------------------------------------------------*/
128 void insertionSort(int *vetor, long int n)
129 {
130     long long int i, j;
131     int aux;
132     troca = 0;
133     comparacao = 0;
134
135     tempoInicio = clock();
```

```
136     for (i = 1; i < n; i++)
137     {
138         aux = vetor[i];
139         for (j = i - 1; j >= 0 && aux < vetor[j]; --j,
                comparacao++)
140         {
141             vetor[j + 1] = vetor[j];
142             troca++;
143         }
144         vetor[j + 1] = aux;
145         troca++;
146     }
147     tempoFim = clock();
148 }
149 /*------------------------------------------------*/
150 void mergee (int *vetor,int v2[],int inicio1, int inicio2,int
    fim2)
151 {
152     long int n1=inicio1;
153     long int n2=inicio2;
154     long int fim1=n2-1;
155     long int aux=0,i;
156
157     tempoInicio = clock();
158     comparacao++;
159     while((n1<=fim1) && (n2<=fim2))
160     {
161
162         if (vetor[n1]<vetor[n2])
163         {
164             troca++;
165             comparacao++;
166             v2[aux++] = vetor[n1++];
167
168         }
169         else
170         {
171             troca++;
172             comparacao++;
173
174             v2[aux++] = vetor[n2++];
```

```
175         }
176     }
177     while(n1<=fim1)
178     {
179         v2[aux++] = vetor[n1++];
180         troca++;
181         comparacao++;
182     }
183     while(n2<=fim2)
184     {
185         v2[aux++] = vetor[n2++];
186         troca++;
187         comparacao++;
188     }
189     for(i=0; i<aux; i++)
190     {
191         vetor[i+inicio1]=v2[i];
192          troca++;
193     }
194     tempoFim = clock();
195 }
196 void mergeSort (int *vetor, int v2[],long int esq, long int dir)
197 {
198     long int meio,valor,valor2,res;
199     if(esq<dir)
200     {
201         meio=(esq+dir)/2;
202         mergeSort(vetor,v2,esq,meio);
203         mergeSort(vetor,v2,meio+1,dir);
204         mergee(vetor,v2,esq,meio+1,dir);
205     }
206 }
207 /*-----------------------------------------------*/
208 void quicksort(int *vetor, int esq, int dir) {
209
210     int i, j, pivo, aux;
211     i = esq;
212     j = dir;
213
214     tempoInicio = clock();
215     pivo = vetor[(esq + dir) / 2]; //Elemento central como piv
```

```
216     // pivo = vetor[esq];
217     // pivo = vetor[dir];
218      while(i <= j) {
219          while(vetor[i] < pivo && i < dir) {
220              i++;
221              comparacao++;
222          }
223          while(vetor[j] > pivo && j > esq) {
224              j--;
225              comparacao++;
226          }
227          if(i <= j) {
228              aux = vetor[i];
229              vetor[i] = vetor[j];
230              vetor[j] = aux;
231              i++;
232              j--;
233               troca++;
234            //  comparacao++;
235          }
236      }
237      if(j > esq) {
238          quicksort(vetor, esq, j);
239      }
240      if(i < dir) {
241          quicksort(vetor, i, dir);
242      }
243      tempoFim = clock();
244  }
245  /*------------------------------------------------*/
246  void GravarArquivo(int *vetor, long int tamanho)
247  {//Essa fun   o grava o vetor ordenando em um arquivo txt
248
249      long int i;
250      FILE *gravar;
251      char Nome_arquivo[200];
252
253      sprintf(Nome_arquivo, "Vetor-%ld-Ordenado.txt", tamanho);
254      gravar = fopen(Nome_arquivo,"w");
255      for (i = 0; i < tamanho; i++)
256      {
```

```
257         fprintf( gravar," %d\n", vetor[i]);
258         fflush(gravar);
259     }
260     fclose(gravar);
261     printf("\n");
262 }
263 void MostarVetor(int *vetor, long int tamanho)
264 {//funcao responsavel por mostar no console.
265     long int i;
266 
267     for (i = 0; i<tamanho; i++)
268     {
269         printf("%d\t", vetor[i]);
270     }
271     printf("\n");
272 }
273 /*---------------------------------------------*/
274 void printResultados(){
275     //grava os resultados de comparacoess, trocas e tempo de cpu
            em um arquivo.
276     fprintf( gnuplot," %f\t", (float)(tempoFim - tempoInicio)/
           CLOCKS_PER_SEC);
277     fprintf( gnuplot, "%ld\t\t",comparacao);
278     fprintf( gnuplot, "%ld\n", troca);
279 
280     //mostas as trocas e compracoes no console
281      printf("comparacao: %ld\n Trocas %ld\n",comparacao,troca);
282      printf("Tempo: %f\n", (float)(tempoFim - tempoInicio)/
            CLOCKS_PER_SEC);
283 }
284 
285 #endif // ORDENA_H_INCLUDED
```

1.2.2 Arquivo main.c

```
1 /* Codigo com a Funcao main */
2 
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <time.h>
6 #include "ordena.h"
```

```
7
8  int main()
9  {
10
11      int *VetorArquivo;
12      long int tamanho;
13      long int vetor2[100000];//vetor auxilixar para verge sort
14      int op;
15      do
16      {
17          //system ("clear");
18          printf("\n\t|--------| MENU DE OPCOES | ----------|");
19          printf("\n\t|-------------------------------------|");
20          printf("\n\t| 1 -----| BUBBLE  SORT |-------------|");
21          printf("\n\t| 2 -----| SELECT  SORT |-------------|");
22          printf("\n\t| 3 -----| INSERT  SORT |-------------|");
23          printf("\n\t| 4 -----| QUICK   SORT |-------------|");
24          printf("\n\t| 5 -----| MERGE   SORT |-------------|");
25          printf("\n\t| 6 -----| SHELL   SORT |-------------|");
26          printf("\n\t|-------------------------------------|\n");
27          scanf("\t %d",&op);
28          switch(op)
29          {
30          case 1 :
31
32              gnuplot = fopen("relatorios/bubble-sorte-rond.dat",
                    "w");
33              printf("Ordenando vetor com 100 elementos\n");
34              fprintf(gnuplot, "#RANDOMICOS\n#Tamanho\t Tempo\t\t
                    Comparacao\t trocas\n");
35              VetorArquivo = LendoArquivo("vetores/N100.txt",&
                    tamanho);
36              BubbleSort(VetorArquivo, tamanho);
37              GravarArquivo(VetorArquivo, tamanho);
38              printResultados();
39             /*-------------------------------------------*/
40              printf("Ordenando vetor com 1.000 elementos\n");
41              VetorArquivo = LendoArquivo("vetores/N1000.txt",&
                    tamanho);
42              BubbleSort(VetorArquivo, tamanho);
43              GravarArquivo(VetorArquivo, tamanho);
```

```
44              printResultados();
45             /*-----------------------------------------------*/
46              printf("Ordenando vetor com 10.000 elementos\n");
47              VetorArquivo = LendoArquivo("vetores/N10000.txt",&
                   tamanho);
48              BubbleSort(VetorArquivo, tamanho);
49              GravarArquivo(VetorArquivo, tamanho);
50              printResultados();
51             /*-----------------------------------------------*/
52              printf("Ordenando vetor de 50.000 elementos\n");
53              VetorArquivo = LendoArquivo("vetores/N50000.txt",&
                   tamanho);
54              BubbleSort(VetorArquivo, tamanho);
55              GravarArquivo(VetorArquivo, tamanho);
56              printResultados();
57             /*-----------------------------------------------*/
58              printf("Ordenando vetor de 100.000 elementos \n");
59              VetorArquivo = LendoArquivo("vetores/N100000.txt",&
                   tamanho);
60              BubbleSort(VetorArquivo, tamanho);
61              GravarArquivo(VetorArquivo, tamanho);
62              printResultados();
63              fclose(gnuplot);
64              /*-----------------------------------------------*/
65              printf("|==================================|\n");
66              gnuplot = fopen("relatorios/bubbleSorte-ord.dat", "w
                   ");
67              printf("Vetor ja ordenado com 100\n");
68              fprintf(gnuplot, "#ORDENANDOS\n#Tamanho\t Tempo\t\t
                   Comparacao\t trocas\n");
69              VetorArquivo = LendoArquivo("vetores/N100-ordenado.
                   txt",&tamanho);
70              BubbleSort(VetorArquivo, tamanho);
71              GravarArquivo(VetorArquivo, tamanho);
72              printResultados();
73              /*-----------------------------------------------*/
74              printf("Vetor ja ordenado com 1.000\n");
75              VetorArquivo = LendoArquivo("vetores/N1000-ordenado.
                   txt",&tamanho);
76              BubbleSort(VetorArquivo, tamanho);
77              GravarArquivo(VetorArquivo, tamanho);
```

```
78              printResultados();
79             /*-----------------------------------------------*/
80              printf("Vetor ja ordenado com 10.000\n");
81              VetorArquivo = LendoArquivo("vetores/N10000-ordenado
                   .txt",&tamanho);
82              BubbleSort(VetorArquivo, tamanho);
83              GravarArquivo(VetorArquivo, tamanho);
84              printResultados();
85             /*-----------------------------------------------*/
86              printf("Vetor ja ordenado com 50.000\n");
87              VetorArquivo = LendoArquivo("vetores/N50000-ordenado
                   .txt",&tamanho);
88              BubbleSort(VetorArquivo, tamanho);
89              GravarArquivo(VetorArquivo, tamanho);
90              printResultados();
91             /*-----------------------------------------------*/
92              printf("Vetor ja ordenado com 100.000\n");
93              VetorArquivo = LendoArquivo("vetores/N100000-
                   ordenado.txt",&tamanho);
94              BubbleSort(VetorArquivo, tamanho);
95              GravarArquivo(VetorArquivo, tamanho);
96              printResultados();
97              fclose(gnuplot);
98             /*-----------------------------------------------*/
99               printf("
                    |=========================================================
                    n");
100
101              gnuplot = fopen("relatorios/bubbleSorte-inv.dat", "
                    w");
102             printf("Vetor ordenando inverso com 100\n");
103             fprintf(gnuplot, "#ORDENANDO INVERSO\n#Tamanho\t
                   Tempo\t\t Comparacao\t trocas\n");
104             VetorArquivo = LendoArquivo("vetores/N100-inverso.
                   txt",&tamanho);
105             BubbleSort(VetorArquivo, tamanho);
106             GravarArquivo(VetorArquivo, tamanho);
107             printResultados();
108             /*-----------------------------------------------*/
109             printf("Vetor ordenando inverso com 1.000\n");
```

```
110             VetorArquivo = LendoArquivo("vetores/N1000-inverso.
                    txt",&tamanho);
111             BubbleSort(VetorArquivo, tamanho);
112             GravarArquivo(VetorArquivo, tamanho);
113             printResultados();
114            /*-----------------------------------------------*/
115             printf("Vetor ordenando inverso com 10.000\n");
116             VetorArquivo = LendoArquivo("vetores/N10000-inverso.
                    txt",&tamanho);
117             BubbleSort(VetorArquivo, tamanho);
118             GravarArquivo(VetorArquivo, tamanho);
119             printResultados();
120             /*-----------------------------------------------*/
121             printf("Vetor ordenando inverso com 50.000\n");
122             VetorArquivo = LendoArquivo("vetores/N50000-inverso.
                    txt",&tamanho);
123             BubbleSort(VetorArquivo, tamanho);
124             GravarArquivo(VetorArquivo, tamanho);
125             printResultados();
126             /*-----------------------------------------------*/
127             printf("Vetor ordenando inverso com 100.000\n");
128             VetorArquivo = LendoArquivo("vetores/N100000-inverso
                    .txt",&tamanho);
129             BubbleSort(VetorArquivo, tamanho);
130             GravarArquivo(VetorArquivo, tamanho);
131             printResultados();
132             fclose(gnuplot);
133            /*-----------------------------------------------*/
134
135             system("gnuplot -bg gray90 script_gnuplot/plota.gnu"
                    );
136
137             break;
138
139         case 2:
140
141             gnuplot = fopen("relatorios/Select-sort-rond.dat", "
                    w");
142
143             printf("Ordenando vetor com 100 elementos\n");
```

```
144             fprintf(gnuplot, "#RANDOMICOS\n#Tamanho\t Tempo\t\t
                   Comparacao\t trocas\n");
145             VetorArquivo = LendoArquivo("vetores/N100.txt",&
                   tamanho);
146             selectSort(VetorArquivo, tamanho);
147             GravarArquivo(VetorArquivo, tamanho);
148             printResultados();
149            /*-----------------------------------------------*/
150             printf("Ordenando vetor com 1.000 elementos\n");
151             VetorArquivo = LendoArquivo("vetores/N1000.txt",&
                   tamanho);
152             selectSort(VetorArquivo, tamanho);
153             GravarArquivo(VetorArquivo, tamanho);
154             printResultados();
155            /*-----------------------------------------------*/
156             printf("Ordenando vetor com 10.000 elementos\n");
157             VetorArquivo = LendoArquivo("vetores/N10000.txt",&
                   tamanho);
158             selectSort(VetorArquivo, tamanho);
159             GravarArquivo(VetorArquivo, tamanho);
160             printResultados();
161            /*-----------------------------------------------*/
162             printf("Ordenando vetor de 50.000 elementos\n");
163             VetorArquivo = LendoArquivo("vetores/N50000.txt",&
                   tamanho);
164             selectSort(VetorArquivo, tamanho);
165             GravarArquivo(VetorArquivo, tamanho);
166             printResultados();
167            /*-----------------------------------------------*/
168             printf("Ordenando vetor de 100.000 elementos \n");
169             VetorArquivo = LendoArquivo("vetores/N100000.txt",&
                   tamanho);
170             selectSort(VetorArquivo, tamanho);
171             GravarArquivo(VetorArquivo, tamanho);
172             printResultados();
173             fclose(gnuplot);
174             /*-----------------------------------------------*/
175             printf("
                   |=============================================================
                   n");
176
```

```
177             gnuplot = fopen("relatorios/Select-sort-ord.dat", "w
                    ");
178             printf("Vetor ja ordenado com 100\n");
179             fprintf(gnuplot, "#ORDENANDOS\n#Tamanho\t Tempo\t\t
                    Comparacao\t trocas\n");
180             VetorArquivo = LendoArquivo("vetores/N100-ordenado.
                    txt",&tamanho);
181             selectSort(VetorArquivo, tamanho);
182             GravarArquivo(VetorArquivo, tamanho);
183             printResultados();
184             /*-----------------------------------------------*/
185             printf("Vetor ja ordenado com 1.000\n");
186             VetorArquivo = LendoArquivo("vetores/N1000-ordenado.
                    txt",&tamanho);
187             selectSort(VetorArquivo, tamanho);
188             GravarArquivo(VetorArquivo, tamanho);
189             printResultados();
190             /*-----------------------------------------------*/
191             printf("Vetor ja ordenado com 10.000\n");
192             VetorArquivo = LendoArquivo("vetores/N10000-ordenado
                    .txt",&tamanho);
193             selectSort(VetorArquivo, tamanho);
194             GravarArquivo(VetorArquivo, tamanho);
195             printResultados();
196             /*-----------------------------------------------*/
197             printf("Vetor ja ordenado com 50.000\n");
198             VetorArquivo = LendoArquivo("vetores/N50000-ordenado
                    .txt",&tamanho);
199             selectSort(VetorArquivo, tamanho);
200             GravarArquivo(VetorArquivo, tamanho);
201             printResultados();
202             /*-----------------------------------------------*/
203             printf("Vetor ja ordenado com 100.000\n");
204             VetorArquivo = LendoArquivo("vetores/N100000-
                    ordenado.txt",&tamanho);
205             selectSort(VetorArquivo, tamanho);
206             GravarArquivo(VetorArquivo, tamanho);
207             printResultados();
208             fclose(gnuplot);
209             /*-----------------------------------------------*/
```

```
210             printf("
                    |=========================================================
                    n");
211
212             gnuplot = fopen("relatorios/Select-sort-inv.dat", "
                    w");
213             printf("Vetor ordenando inverso com 100\n");
214             fprintf(gnuplot, "#ORDENANDO INVERSO\n#Tamanho\t
                    Tempo\t\t Comparacao\t trocas\n");
215             VetorArquivo = LendoArquivo("vetores/N100-inverso.
                    txt",&tamanho);
216             selectSort(VetorArquivo, tamanho);
217             GravarArquivo(VetorArquivo, tamanho);
218             printResultados();
219            /*-----------------------------------------------*/
220             printf("Vetor ordenando inverso com 1.000\n");
221             VetorArquivo = LendoArquivo("vetores/N1000-inverso.
                    txt",&tamanho);
222             selectSort(VetorArquivo, tamanho);
223             GravarArquivo(VetorArquivo, tamanho);
224             printResultados();
225           /*-----------------------------------------------*/
226             printf("Vetor ordenando inverso com 10.000\n");
227             VetorArquivo = LendoArquivo("vetores/N10000-inverso.
                    txt",&tamanho);
228             selectSort(VetorArquivo, tamanho);
229             GravarArquivo(VetorArquivo, tamanho);
230             printResultados();
231            /*-----------------------------------------------*/
232             printf("Vetor ordenando inverso com 50.000\n");
233             VetorArquivo = LendoArquivo("vetores/N50000-inverso.
                    txt",&tamanho);
234             selectSort(VetorArquivo, tamanho);
235             GravarArquivo(VetorArquivo, tamanho);
236             printResultados();
237            /*-----------------------------------------------*/
238             printf("Vetor ordenando inverso com 100.000\n");
239             VetorArquivo = LendoArquivo("vetores/N100000-inverso
                    .txt",&tamanho);
240             selectSort(VetorArquivo, tamanho);
241             GravarArquivo(VetorArquivo, tamanho);
```

```
242             printResultados();
243             fclose(gnuplot);
244             /*-----------------------------------------------*/
245 
246             system("gnuplot -bg gray90 script_gnuplot/plota2.gnu
                   ");
247             break;
248         case 3:
249 
250             gnuplot = fopen("relatorios/Insert-sort-rond.dat", "
                   w");
251 
252             printf("Ordenando vetor com 100 elementos\n");
253             fprintf(gnuplot, "#RANDOMICOS\n#Tamanho\t Tempo\t\t
                   Comparacao\t trocas\n");
254             VetorArquivo = LendoArquivo("vetores/N100.txt",&
                   tamanho);
255             insertionSort(VetorArquivo, tamanho);
256             GravarArquivo(VetorArquivo, tamanho);
257             printResultados();
258             /*-----------------------------------------------*/
259             printf("Ordenando vetor com 1.000 elementos\n");
260             VetorArquivo = LendoArquivo("vetores/N1000.txt",&
                   tamanho);
261             insertionSort(VetorArquivo, tamanho);
262             GravarArquivo(VetorArquivo, tamanho);
263             printResultados();
264            /*-----------------------------------------------*/
265             printf("Ordenando vetor com 10.000 elementos\n");
266             VetorArquivo = LendoArquivo("vetores/N10000.txt",&
                   tamanho);
267             insertionSort(VetorArquivo, tamanho);
268             GravarArquivo(VetorArquivo, tamanho);
269             printResultados();
270             //
                   /-------------------------------------------------------------
271             printf("Ordenando vetor de 50.000 elementos\n");
272             VetorArquivo = LendoArquivo("vetores/N50000.txt",&
                   tamanho);
273             insertionSort(VetorArquivo, tamanho);
```

```
274             GravarArquivo(VetorArquivo, tamanho);
275             printResultados();
276            /*-----------------------------------------------*/
277             printf("Ordenando vetor de 100.000 elementos \n");
278             VetorArquivo = LendoArquivo("vetores/N100000.txt",&
                   tamanho);
279             insertionSort(VetorArquivo, tamanho);
280             GravarArquivo(VetorArquivo, tamanho);
281             printResultados();
282             fclose(gnuplot);
283            /*-----------------------------------------------*/
284             printf("
                   |=============================================================
                   n");
285
286             gnuplot = fopen("relatorios/Insert-sort-ord.dat", "w
                   ");
287
288             printf("Vetor ja ordenado com 100\n");
289             fprintf(gnuplot, "#ORDENANDOS\n#Tamanho\t Tempo\t\t
                   Comparacao\t trocas\n");
290             VetorArquivo = LendoArquivo("vetores/N100-ordenado.
                   txt",&tamanho);
291             insertionSort(VetorArquivo, tamanho);
292             GravarArquivo(VetorArquivo, tamanho);
293             printResultados();
294            /*-----------------------------------------------*/
295             printf("Vetor ja ordenado com 1.000\n");
296             VetorArquivo = LendoArquivo("vetores/N1000-ordenado.
                   txt",&tamanho);
297             insertionSort(VetorArquivo, tamanho);
298             GravarArquivo(VetorArquivo, tamanho);
299             printResultados();
300             /*-----------------------------------------------*/
301             printf("Vetor ja ordenado com 10.000\n");
302             VetorArquivo = LendoArquivo("vetores/N10000-ordenado
                   .txt",&tamanho);
303             insertionSort(VetorArquivo, tamanho);
304             GravarArquivo(VetorArquivo, tamanho);
305             printResultados();
306             /*-----------------------------------------------*/
```

```
307             printf("Vetor ja ordenado com 50.000\n");
308             VetorArquivo = LendoArquivo("vetores/N50000-ordenado
                   .txt",&tamanho);
309             insertionSort(VetorArquivo, tamanho);
310             GravarArquivo(VetorArquivo, tamanho);
311             printResultados();
312             /*-----------------------------------------------*/
313             printf("Vetor ja ordenado com 100.000\n");
314             VetorArquivo = LendoArquivo("vetores/N100000-
                   ordenado.txt",&tamanho);
315             insertionSort(VetorArquivo, tamanho);
316             GravarArquivo(VetorArquivo, tamanho);
317             printResultados();
318             fclose(gnuplot);
319             /*-----------------------------------------------*/
320              printf("
                    |=========================================================
                    n");
321
322             gnuplot = fopen("relatorios/Insert-sort-inv.dat", "w
                   ");
323
324             printf("Vetor ordenando inverso com 100\n");
325             fprintf(gnuplot, "#ORDENANDO INVERSO\n#Tamanho\t
                   Tempo\t\t Comparacao\t trocas\n");
326             VetorArquivo = LendoArquivo("vetores/N100-inverso.
                   txt",&tamanho);
327             insertionSort(VetorArquivo, tamanho);
328             GravarArquivo(VetorArquivo, tamanho);
329             printResultados();
330             /*-----------------------------------------------*/
331             printf("Vetor ordenando inverso com 1.000\n");
332             VetorArquivo = LendoArquivo("vetores/N1000-inverso.
                   txt",&tamanho);
333             insertionSort(VetorArquivo, tamanho);
334             GravarArquivo(VetorArquivo, tamanho);
335             printResultados();
336            /*-----------------------------------------------*/
337             printf("Vetor ordenando inverso com 10.000\n");
338             VetorArquivo = LendoArquivo("vetores/N10000-inverso.
                   txt",&tamanho);
```

```
339             insertionSort(VetorArquivo, tamanho);
340             GravarArquivo(VetorArquivo, tamanho);
341             printResultados();
342            /*-----------------------------------------------*/
343             printf("Vetor ordenando inverso com 50.000\n");
344             VetorArquivo = LendoArquivo("vetores/N50000-inverso.
                   txt",&tamanho);
345             insertionSort(VetorArquivo, tamanho);
346             GravarArquivo(VetorArquivo, tamanho);
347             printResultados();
348            /*-----------------------------------------------*/
349             printf("Vetor ordenando inverso com 100.000\n");
350             VetorArquivo = LendoArquivo("vetores/N100000-inverso
                   .txt",&tamanho);
351             insertionSort(VetorArquivo, tamanho);
352             GravarArquivo(VetorArquivo, tamanho);
353             printResultados();
354             fclose(gnuplot);
355            /*-----------------------------------------------*/
356             system("gnuplot -bg gray90 script_gnuplot/plota3.gnu
                   ");
357             break;
358         case 4:
359
360              gnuplot = fopen("relatorios/quick-sort-rand.dat", "
                    w");
361              printf("Ordenando vetor com 100 elementos\n");
362             fprintf(gnuplot, "#RANDOMICOS\n#Tamanho\t Tempo\t\t
                   Comparacao\t trocas\n");
363             VetorArquivo = LendoArquivo("vetores/N100.txt",&
                   tamanho);
364             quicksort(VetorArquivo,0,tamanho-1);
365            // MostarVetor(VetorArquivo,tamanho);
366             printResultados();
367             //GravarArquivo(VetorArquivo, tamanho);
368             //
                   /-----------------------------------------------------------

369             printf("Ordenando vetor com 1.000 elementos\n");
370             VetorArquivo = LendoArquivo("vetores/N1000.txt",&
                   tamanho);
```

```
371             quicksort(VetorArquivo,0,tamanho-1);
372             //GravarArquivo(VetorArquivo, tamanho);
373              printResultados();
374             //
                   /-----------------------------------------------------------

375             printf("Ordenando vetor com 10.000 elementos\n");
376             VetorArquivo = LendoArquivo("vetores/N10000.txt",&
                   tamanho);
377             quicksort(VetorArquivo,0,tamanho-1);
378             printResultados();
379             GravarArquivo(VetorArquivo, tamanho);
380            /*-----------------------------------------------*/
381             printf("Ordenando vetor de 50.000 elementos\n");
382             VetorArquivo = LendoArquivo("vetores/N50000.txt",&
                   tamanho);
383             quicksort(VetorArquivo,0,tamanho-1);
384             printResultados();
385             //GravarArquivo(VetorArquivo, tamanho);
386            /*-----------------------------------------------*/
387             printf("Ordenando vetor de 100.000 elementos \n");
388             VetorArquivo = LendoArquivo("vetores/N100000.txt",&
                   tamanho);
389             quicksort(VetorArquivo,0,tamanho-1);
390             printResultados();
391             //GravarArquivo(VetorArquivo, tamanho);
392             fclose(gnuplot);
393            /*-----------------------------------------------*/
394             printf("
                   |=============================================================
                   n");
395
396             gnuplot = fopen("relatorios/quick-sort-ord.dat", "w"
                   );
397             printf("Vetor ja ordenado com 100\n");
398             fprintf(gnuplot, "#ORDENANDOS\n#Tamanho\t Tempo\t\t
                   Comparacao\t trocas\n");
399             VetorArquivo = LendoArquivo("vetores/N100-ordenado.
                   txt",&tamanho);
400             quicksort(VetorArquivo,0,tamanho-1);
401             printResultados();
```

```
402             //GravarArquivo(VetorArquivo, tamanho);
403
404             //
                   /-----------------------------------------------------------
405             printf("Vetor ja ordenado com 1.000\n");
406             VetorArquivo = LendoArquivo("vetores/N1000-ordenado.
                   txt",&tamanho);
407             quicksort(VetorArquivo,0,tamanho-1);
408             printResultados();
409             //GravarArquivo(VetorArquivo, tamanho);
410             /*-----------------------------------------------*/
411             printf("Vetor ja ordenado com 10.000\n");
412             VetorArquivo = LendoArquivo("vetores/N10000-ordenado
                   .txt",&tamanho);;
413             quicksort(VetorArquivo,0,tamanho-1);
414             printResultados();
415             //GravarArquivo(VetorArquivo, tamanho);
416             /*-----------------------------------------------*/
417             printf("Vetor ja ordenado com 50.000\n");
418             VetorArquivo = LendoArquivo("vetores/N50000-ordenado
                   .txt",&tamanho);
419             quicksort(VetorArquivo,0,tamanho-1);
420             printResultados();
421             //GravarArquivo(VetorArquivo, tamanho);
422            /*-----------------------------------------------*/
423             printf("Vetor ja ordenado com 100.000\n");
424             VetorArquivo = LendoArquivo("vetores/N100000-
                   ordenado.txt",&tamanho);
425             quicksort(VetorArquivo,0,tamanho-1);
426             printResultados();
427             //GravarArquivo(VetorArquivo, tamanho);
428             fclose(gnuplot);
429            /*-----------------------------------------------*/
430              printf("
                    |=============================================================
                    n");
431
432              gnuplot = fopen("relatorios/quick-sort-inv.dat", "w
                    ");
433             printf("Vetor ordenando inverso com 100\n");
```

```
434             fprintf(gnuplot, "#ORDENANDO INVERSO\n#Tamanho\t
                   Tempo\t\t Comparacao\t trocas\n");
435             VetorArquivo = LendoArquivo("vetores/N100-inverso.
                   txt",&tamanho);
436             quicksort(VetorArquivo,0,tamanho-1);
437             printResultados();
438            // GravarArquivo(VetorArquivo, tamanho);
439            /*-----------------------------------------------*/
440             printf("Vetor ordenando inverso com 1.000\n");
441             VetorArquivo = LendoArquivo("vetores/N1000-inverso.
                   txt",&tamanho);
442             quicksort(VetorArquivo,0,tamanho-1);
443             printResultados();
444             //GravarArquivo(VetorArquivo, tamanho);
445            /*-----------------------------------------------*/
446             printf("Vetor ordenando inverso com 10.000\n");
447             VetorArquivo = LendoArquivo("vetores/N10000-inverso.
                   txt",&tamanho);
448             quicksort(VetorArquivo,0,tamanho-1);
449             printResultados();
450            // GravarArquivo(VetorArquivo, tamanho);
451             /*-----------------------------------------------*/
452             printf("Vetor ordenando inverso com 50.000\n");
453             VetorArquivo = LendoArquivo("vetores/N50000-inverso.
                   txt",&tamanho);
454             quicksort(VetorArquivo,0,tamanho-1);
455             printResultados();
456           //  GravarArquivo(VetorArquivo, tamanho);
457             /*-----------------------------------------------*/
458             printf("Vetor ordenando inverso com 100.000\n");
459             VetorArquivo = LendoArquivo("vetores/N100000-inverso
                   .txt",&tamanho);
460             quicksort(VetorArquivo,0,tamanho-1);
461             printResultados();
462            // GravarArquivo(VetorArquivo, tamanho);
463             fclose(gnuplot);
464             /*-----------------------------------------------*/
465
466             system("gnuplot -bg gray90 script_gnuplot/plota4.gnu
                   ");
467
```

```
468
469             break;
470         case 5:
471             gnuplot = fopen("relatorios/merge-sort-rand.dat", "w
                    ");
472              printf("Ordenando vetor com 100 elementos\n");
473             fprintf(gnuplot, "#RANDOMICOS\n#Tamanho\t Tempo\t\t
                    Comparacao\t trocas\n");
474             VetorArquivo = LendoArquivo("vetores/N100.txt",&
                    tamanho);
475             mergeSort(VetorArquivo,vetor2,0,tamanho);
476             printResultados();
477             MostarVetor(VetorArquivo,tamanho);
478             //GravarArquivo(VetorArquivo, tamanho);
479             /*-----------------------------------------------*/
480             printf("Ordenando vetor com 1.000 elementos\n");
481             VetorArquivo = LendoArquivo("vetores/N1000.txt",&
                    tamanho);
482              mergeSort(VetorArquivo,vetor2,0,tamanho);
483             printResultados();
484             //GravarArquivo(VetorArquivo, tamanho);
485             /*-----------------------------------------------*/
486             printf("Ordenando vetor com 10.000 elementos\n");
487             VetorArquivo = LendoArquivo("vetores/N10000.txt",&
                    tamanho);
488              mergeSort(VetorArquivo,vetor2,0,tamanho);
489             printResultados();
490             //GravarArquivo(VetorArquivo, tamanho);
491             /*-----------------------------------------------*/
492             printf("Ordenando vetor de 50.000 elementos\n");
493             VetorArquivo = LendoArquivo("vetores/N50000.txt",&
                    tamanho);
494              mergeSort(VetorArquivo,vetor2,0,tamanho);
495             printResultados();
496            // GravarArquivo(VetorArquivo, tamanho);
497            /*-----------------------------------------------*/
498             printf("Ordenando vetor de 100.000 elementos \n");
499             VetorArquivo = LendoArquivo("vetores/N100000.txt",&
                    tamanho);
500             mergeSort(VetorArquivo,vetor2,0,tamanho);
501             printResultados();
```

```
502             // GravarArquivo(VetorArquivo, tamanho);
503              fclose(gnuplot);
504             /*-----------------------------------------------*/
505              printf("
                    |=============================================================
                    n");
506
507              gnuplot = fopen("relatorios/merge-sort-ord.dat", "w"
                    );
508              printf("Vetor ja ordenado com 100\n");
509              fprintf(gnuplot, "#ORDENANDOS\n#Tamanho\t Tempo\t\t
                    Comparacao\t trocas\n");
510              VetorArquivo = LendoArquivo("vetores/N100-ordenado.
                    txt",&tamanho);
511               mergeSort(VetorArquivo,vetor2,0,tamanho);
512              printResultados();
513              //GravarArquivo(VetorArquivo, tamanho);
514
515             /*-----------------------------------------------*/
516              printf("Vetor ja ordenado com 1.000\n");
517              VetorArquivo = LendoArquivo("vetores/N1000-ordenado.
                    txt",&tamanho);
518               mergeSort(VetorArquivo,vetor2,0,tamanho);
519             printResultados();
520             // GravarArquivo(VetorArquivo, tamanho);
521             /*-----------------------------------------------*/
522              printf("Vetor ja ordenado com 10.000\n");
523              VetorArquivo = LendoArquivo("vetores/N10000-ordenado
                    .txt",&tamanho);
524               mergeSort(VetorArquivo,vetor2,0,tamanho);
525              printResultados();
526              //GravarArquivo(VetorArquivo, tamanho);
527              /*-----------------------------------------------*/
528              printf("Vetor ja ordenado com 50.000\n");
529              VetorArquivo = LendoArquivo("vetores/N50000-ordenado
                    .txt",&tamanho);
530               mergeSort(VetorArquivo,vetor2,0,tamanho);
531             printResultados();
532             // GravarArquivo(VetorArquivo, tamanho);
533              /*-----------------------------------------------*/
534              printf("Vetor ja ordenado com 100.000\n");
```

```
535             VetorArquivo = LendoArquivo("vetores/N100000-
                   ordenado.txt",&tamanho);
536              mergeSort(VetorArquivo,vetor2,0,tamanho);
537            printResultados();
538            // GravarArquivo(VetorArquivo, tamanho);
539             fclose(gnuplot);
540             /*-----------------------------------------------*/
541              printf("
                   |=============================================================
                   n");
542
543              gnuplot = fopen("relatorios/merge-sort-inv.dat", "w
                   ");
544             printf("Vetor ordenando inverso com 100\n");
545             fprintf(gnuplot, "#ORDENANDO INVERSO\n#Tamanho\t
                   Tempo\t\t Comparacao\t trocas\n");
546             VetorArquivo = LendoArquivo("vetores/N100-inverso.
                   txt",&tamanho);
547              mergeSort(VetorArquivo,vetor2,0,tamanho);
548            printResultados();
549            // GravarArquivo(VetorArquivo, tamanho);
550
551            /*-----------------------------------------------*/
552             printf("Vetor ordenando inverso com 1.000\n");
553             VetorArquivo = LendoArquivo("vetores/N1000-inverso.
                   txt",&tamanho);
554              mergeSort(VetorArquivo,vetor2,0,tamanho);
555           printResultados();
556           //  GravarArquivo(VetorArquivo, tamanho);
557             /*-----------------------------------------------*/
558             printf("Vetor ordenando inverso com 10.000\n");
559             VetorArquivo = LendoArquivo("vetores/N10000-inverso.
                   txt",&tamanho);
560              mergeSort(VetorArquivo,vetor2,0,tamanho);
561            printResultados();
562            // GravarArquivo(VetorArquivo, tamanho);
563             /*-----------------------------------------------*/
564             printf("Vetor ordenando inverso com 50.000\n");
565             VetorArquivo = LendoArquivo("vetores/N50000-inverso.
                   txt",&tamanho);
566              mergeSort(VetorArquivo,vetor2,0,tamanho);
```

```
567             printResultados();
568             //GravarArquivo(VetorArquivo, tamanho);
569            /*-----------------------------------------------*/
570             printf("Vetor ordenando inverso com 100.000\n");
571             VetorArquivo = LendoArquivo("vetores/N100000-inverso
                   .txt",&tamanho);
572              mergeSort(VetorArquivo,vetor2,0,tamanho);
573            printResultados();
574            // GravarArquivo(VetorArquivo, tamanho);
575             fclose(gnuplot);
576            /*-----------------------------------------------*/
577
578             system("gnuplot -bg gray90 script_gnuplot/plota5.gnu
                   ");
579             break;
580         case 6:
581
582             gnuplot = fopen("relatorios/shell-sort-rond.dat", "w
                   ");
583
584             printf("Ordenando vetor com 100 elementos\n");
585             fprintf(gnuplot, "#RANDOMICOS\n#Tamanho\t Tempo\t\t
                   Comparacao\t trocas\n");
586             VetorArquivo = LendoArquivo("vetores/N100.txt",&
                   tamanho);
587             shellSort(VetorArquivo,tamanho);
588             GravarArquivo(VetorArquivo, tamanho);
589             printResultados();
590            /*-----------------------------------------------*/
591             printf("Ordenando vetor com 1.000 elementos\n");
592             VetorArquivo = LendoArquivo("vetores/N1000.txt",&
                   tamanho);
593             shellSort(VetorArquivo,tamanho);
594             GravarArquivo(VetorArquivo, tamanho);
595             printResultados();
596            /*-----------------------------------------------*/
597             printf("Ordenando vetor com 10.000 elementos\n");
598             VetorArquivo = LendoArquivo("vetores/N10000.txt",&
                   tamanho);
599             shellSort(VetorArquivo,tamanho);
600             GravarArquivo(VetorArquivo, tamanho);
```

```
601             printResultados();
602             /*-----------------------------------------------*/
603             printf("Ordenando vetor de 50.000 elementos\n");
604             VetorArquivo = LendoArquivo("vetores/N50000.txt",&
                   tamanho);
605             shellSort(VetorArquivo,tamanho);
606             GravarArquivo(VetorArquivo, tamanho);
607             printResultados();
608            /*-----------------------------------------------*/
609             printf("Ordenando vetor de 100.000 elementos \n");
610             VetorArquivo = LendoArquivo("vetores/N100000.txt",&
                   tamanho);
611             shellSort(VetorArquivo,tamanho);
612             GravarArquivo(VetorArquivo, tamanho);
613             printResultados();
614             fclose(gnuplot);
615            /*-----------------------------------------------*/
616             printf("
                   |=============================================================
                   n");
617
618             gnuplot = fopen("relatorios/shell-sort-ord.dat", "w"
                   );
619             printf("Vetor ja ordenado com 100\n");
620             fprintf(gnuplot, "#ORDENANDOS\n#Tamanho\t Tempo\t\t
                   Comparacao\t trocas\n");
621             VetorArquivo = LendoArquivo("vetores/N100-ordenado.
                   txt",&tamanho);
622             shellSort(VetorArquivo,tamanho);
623             GravarArquivo(VetorArquivo, tamanho);
624             printResultados();
625            /*-----------------------------------------------*/
626             printf("Vetor ja ordenado com 1.000\n");
627             VetorArquivo = LendoArquivo("vetores/N1000-ordenado.
                   txt",&tamanho);
628             shellSort(VetorArquivo,tamanho);
629             GravarArquivo(VetorArquivo, tamanho);
630             printResultados();
631            /*-----------------------------------------------*/
632             printf("Vetor ja ordenado com 10.000\n");
```

```
633             VetorArquivo = LendoArquivo("vetores/N10000-ordenado
                    .txt",&tamanho);
634             shellSort(VetorArquivo,tamanho);
635             GravarArquivo(VetorArquivo, tamanho);
636             printResultados();
637            /*-----------------------------------------------*/
638             printf("Vetor ja ordenado com 50.000\n");
639             VetorArquivo = LendoArquivo("vetores/N50000-ordenado
                    .txt",&tamanho);
640             shellSort(VetorArquivo,tamanho);
641             GravarArquivo(VetorArquivo, tamanho);
642             printResultados();
643             /*-----------------------------------------------*/
644             printf("Vetor ja ordenado com 100.000\n");
645             VetorArquivo = LendoArquivo("vetores/N100000-
                    ordenado.txt",&tamanho);
646             shellSort(VetorArquivo,tamanho);
647             GravarArquivo(VetorArquivo, tamanho);
648             printResultados();
649             fclose(gnuplot);
650            /*-----------------------------------------------*/
651              printf("
                     |=============================================================
                     n");
652
653              gnuplot = fopen("relatorios/shell-sort-inv.dat", "w
                     ");
654             printf("Vetor ordenando inverso com 100\n");
655             fprintf(gnuplot, "#ORDENANDO INVERSO\n#Tamanho\t
                    Tempo\t\t Comparacao\t trocas\n");
656             VetorArquivo = LendoArquivo("vetores/N100-inverso.
                    txt",&tamanho);
657             shellSort(VetorArquivo,tamanho);
658             GravarArquivo(VetorArquivo, tamanho);
659             printResultados();
660             /*-----------------------------------------------*/
661             printf("Vetor ordenando inverso com 1.000\n");
662             VetorArquivo = LendoArquivo("vetores/N1000-inverso.
                    txt",&tamanho);
663             shellSort(VetorArquivo,tamanho);
664             GravarArquivo(VetorArquivo, tamanho);
```

```
665             printResultados();
666             /*-----------------------------------------------*/
667             printf("Vetor ordenando inverso com 10.000\n");
668             VetorArquivo = LendoArquivo("vetores/N10000-inverso.
                   txt",&tamanho);
669             shellSort(VetorArquivo,tamanho);
670             GravarArquivo(VetorArquivo, tamanho);
671             printResultados();
672            /*-----------------------------------------------*/
673             printf("Vetor ordenando inverso com 50.000\n");
674             VetorArquivo = LendoArquivo("vetores/N50000-inverso.
                   txt",&tamanho);
675             shellSort(VetorArquivo,tamanho);
676             GravarArquivo(VetorArquivo, tamanho);
677             printResultados();
678
679             printf("Vetor ordenando inverso com 100.000\n");
680             VetorArquivo = LendoArquivo("vetores/N100000-inverso
                   .txt",&tamanho);
681             shellSort(VetorArquivo,tamanho);
682             GravarArquivo(VetorArquivo, tamanho);
683             printResultados();
684             fclose(gnuplot);
685             /*-----------------------------------------------*/
686
687             system("gnuplot -bg gray90 script_gnuplot/plota6.gnu
                   ");
688             break;
689         case 0:
690
691             printf("Obrigado por usar o programa\n");
692             break;
693         }
694     }
695     while(op!=0);
696      system("gnuplot -bg gray90 script_gnuplot/plota7.gnu");
697     return 1;
698 }
```

Referências Bibliográficas

1 Algorithms in C [SEDGEWICK Robert (1946)] Princeton University, Addison Wesley Public Shing Company.

2 F. LORENZI, P. N. DE MATTOS, T. P. DE CARVALHO. Estruturas de Dados. Thomson, 2007.

3 Algoritmos Teoria e Pratica [CORMEN H. Thomas, LEISERSON E. Charles, RIVEST L. Ronald, STEIN Clifford] Tradução da 2 Edição Amerciana, Editora Campus.

4 Complexidade de Algoritmos: análise, projeto emétodos. Porto Alegre: Editora Sagra Luzzatto, 2002.