



Universidade Federal de São João del Rei
Departamento de Ciência da Computação
Curso de Ciência da Computação

Roteiro 10

Adélson de Oliveira Carmo Júnior
212050019

1 Algoritmos de Ordenação

1.1 Reimplantação

Código

```
1 #ifndef ORDENACAO_PT1
2 #define ORDENACAO_PT1
3
4 int getComp();
5 int getMov();
6 void setComp(int);
7 void setMov(int);
8 int* copiaVetor(int*, int);
9 void imprimeVetor(int*, int);
10 void preencheAleatorio(int*, int, int,
```

```
int);
11 void troca(int*, int *);
12 void BubbleSort(int *, int);
13 void BubbleSort2(int *, int);
14 void InsertionSort(int *, int);
15 void SelectionSort(int *, int);
16
17 #endif
```

codigos/questao11/questao11.h

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4 #include "questao11.h"
5
6 //Medidas de Complexidade
7 int comp; //Num. de comparacoes
8 int mov; //Num. de movimentacoes
9
10
11 int getComp(){
12     return comp;
13 }
14
15 int getMov(){
16     return mov;
17 }
18
19 void setComp(int valor){
20     comp = valor;
21 }
22
```

```
23 void setMov(int valor){
24     mov = valor;
25 }
26
27 int* copiaVetor(int* v, int n){
28     int i;
29     int *v2;
30     v2 = (int*) malloc (n*sizeof(int));
31     for(i=0; i<n; i++) v2[i] = v[i];
32     return v2;
33 }
34
35 void imprimeVetor(int* v, int n){
36     int i, prim = 1;
37     printf("[");
38     for(i=0; i<n; i++)
39         if(prim){ printf("%d", v[i]);
40                     prim = 0; }
41         else printf(", %d", v[i]);
42     printf("]\n");
43 }
```

```

44 void preencheAleatorio(int* v, int n,
    int ini, int fim){
45     int i;
46     for(i=0; i<n; i++)
47         v[i] = ini + rand() % (fim-ini
            + 1);
48 }
49
50 void troca(int* a, int *b){
51     int aux = *a;
52     *a = *b;
53     *b = aux;
54 }
55
56 void BubbleSort(int *v, int n){
57     int i, j;
58     for(i=0; i<n-1; i++){
59         for(j=0; j<n-i-1; j++){
60             comp++;
61             if (v[j]>v[j+1]) {
62                 troca(&v[j], &v[j+1]);
63                 mov++;
64             }
65         }
66 }
67
68 void BubbleSort2(int *v, int n){
69     int j, continua, fim = n;
70     do{
71         continua = 0;
72         for(j=0; j < fim-1; j++){
73             comp++;
74             if (v[j]>v[j+1]) {
75                 troca(&v[j], &v[j+1]);
76                 mov++;
77                 continua = j;
78             }
79         }
80         fim--;
81     }while(continua != 0);
82 }
83
84 void InsertionSort(int *v, int n){
85     int i, j, atual;
86     for(i=1; i < n; i++){
87         atual = v[i];
88         comp++;
89         for(j=i; (j>0) && (atual < v[j-1]);
            j--){
90             v[j] = v[j-1];
91             comp++;
92             mov++;
93         }
94         v[j] = atual;
95     }
96 }
97
98 void SelectionSort(int *v, int n){
99     int i, j, menor;
100    for(i=0; i < n-1; i++){
101        menor = i;
102        for(j=i+1; j < n; j++){
103            comp++;
104            if (v[j] < v[menor])
105                menor = j;
106        }
107        if(i != menor){
108            troca(&v[i], &v[menor]);
109            mov++;
110        }
111    }
112 }

```

codigos/questao11/questao11.c

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4  #include "questao11.h"
5
6  int main(){
7
8      //Atribuicoes iniciais
9      srand(time(NULL));
10     setComp(0);
11     setMov(0);
12     clock_t t;
13
14     int *v1, *v2, *v3;
15     int n;
16
17     /* Recebe valores e preenche */
18     printf("Digite o tamanho do
        vetor:\n");
19     scanf("%d", &n);
20     v1 = (int*) malloc (n*sizeof(int));
21
22     preencheAleatorio(v1, n, 1, 100);
23     imprimeVetor(v1, n);
24
25     v2 = copiaVetor(v1, n);
26     v3 = copiaVetor(v1, n);
27
28     /* BubbleSort */
29     t = clock();
30     BubbleSort(v1, n);
31     t = clock() - t;
32     printf("\nInformacoes da Ordenacao
        por BubbleSort:\n");
33     printf("Tempo Execucao:  %f
        seconds.\n",

```

```

34         ((float)t)/CLOCKS_PER_SEC);
35     printf("Comparacoes: %d\n",
36           getComp());
37     printf("Movimentacoes: %d\n",
38           getMov());
39     printf("Memoria (bytes): %ld\n",
40           n*sizeof(int));
41     imprimeVetor(v1, n);
42
43     /* InsertionSort */
44     setComp(0);
45     setMov(0);
46     t = clock();
47     InsertionSort(v2, n);
48     t = clock() - t;
49     printf("\nInformacoes da Ordenacao
50           por InsertionSort:\n");
51     printf("Tempo Execucao: %f
52           seconds.\n",
53           ((float)t)/CLOCKS_PER_SEC);
54     printf("Comparacoes: %d\n",
55           getComp());
56     printf("Movimentacoes: %d\n",
57           getMov());
58     printf("Memoria (bytes): %ld\n",
59           n*sizeof(int));
60     imprimeVetor(v2, n);
61
62     /* SelectionSort */
63     setComp(0);
64     setMov(0);
65     t = clock();
66     SelectionSort(v3, n);
67     t = clock() - t;
68     printf("\nInformacoes da Ordenacao
69           por SelectionSort:\n");
70     printf("Tempo Execucao: %f
71           seconds.\n",
72           ((float)t)/CLOCKS_PER_SEC);
73     printf("Comparacoes: %d\n",
74           getComp());
75     printf("Movimentacoes: %d\n",
76           getMov());
77     printf("Memoria (bytes): %ld\n",
78           n*sizeof(int));
79     imprimeVetor(v3, n);
80     free(v1);
81     free(v2);
82     free(v3);
83     return 0;
84 }

```

codigos/questao11/main.c

```

1 all: questao11.o
2   gcc questao11.o main.c -o main
3
4 questao11.o: questao11.h questao11.c
5   gcc -c questao11.c
6
7 clean:
8   rm -f questao11.o main

```

codigos/questao11/Makefile

Saída

```

Digite o tamanho do vetor:
10
[11, 83, 88, 8, 95, 37, 71, 95, 97, 36]

Informacoes da Ordenacao por BubbleSort:
Tempo Execucao: 0.000015 seconds.
Comparacoes: 45
Movimentacoes: 16
Memoria (bytes): 40
[8, 11, 36, 37, 71, 83, 88, 95, 95, 97]

```

Figura 1: Questão 1.1 - Saída 1

```

Informacoes da Ordenacao por InsertionSort:
Tempo Execucao: 0.000012 seconds.
Comparacoes: 25
Movimentacoes: 16
Memoria (bytes): 40
[8, 11, 36, 37, 71, 83, 88, 95, 95, 97]

Informacoes da Ordenacao por SelectionSort:
Tempo Execucao: 0.000013 seconds.
Comparacoes: 45
Movimentacoes: 7
Memoria (bytes): 40
[8, 11, 36, 37, 71, 83, 88, 95, 95, 97]

```

Figura 2: Questão 1.1 - Saída 2

1.2 Aplicando modificações para decrescer

Código

```
1 #ifndef ORDENACAO_PT1
2 #define ORDENACAO_PT1
3
4 int getComp();
5 int getMov();
6 void setComp(int);
7 void setMov(int);
8 int* copiaVetor(int*, int);
9 void imprimeVetor(int*, int);
10 void preencheAleatorio(int*, int, int,
```

```
int);
11 void troca(int*, int *);
12 void BubbleSort(int *, int);
13 void BubbleSort2(int *, int);
14 void InsertionSort(int *, int);
15 void SelectionSort(int *, int);
16
17 #endif
```

codigos/questao12/questao12.h

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4 #include "questao12.h"
5
6 //Medidas de Complexidade
7 int comp; //Num. de comparacoes
8 int mov; //Num. de movimentacoes
9
10
11 int getComp(){
12     return comp;
13 }
14
15 int getMov(){
16     return mov;
17 }
18
19 void setComp(int valor){
20     comp = valor;
21 }
22
23 void setMov(int valor){
24     mov = valor;
25 }
26
27 int* copiaVetor(int* v, int n){
28     int i;
29     int *v2;
30     v2 = (int*) malloc (n*sizeof(int));
31     for(i=0; i<n; i++) v2[i] = v[i];
32     return v2;
33 }
34
35 void imprimeVetor(int* v, int n){
36     int i, prim = 1;
37     printf("[");
38     for(i=0; i<n; i++)
39         if(prim){ printf("%d", v[i]);
40                     prim = 0; }
41         else printf(", %d", v[i]);
```

```
41     printf("]\n");
42 }
43
44 void preencheAleatorio(int* v, int n,
45     int ini, int fim){
46     int i;
47     for(i=0; i<n; i++)
48         v[i] = ini + rand() % (fim-ini
49             + 1);
50 }
51
52 void troca(int* a, int *b){
53     int aux = *a;
54     *a = *b;
55     *b = aux;
56 }
57
58 void BubbleSort(int *v, int n) {
59     int i, j;
60     for (i = 0; i < n - 1; i++) {
61         for (j = 0; j < n - i - 1; j++)
62             {
63                 comp++;
64                 if (v[j] < v[j + 1]) {
65                     troca(&v[j], &v[j + 1]);
66                     mov++;
67                 }
68             }
69     }
70 }
71
72 // Bubble Sort 2
73 void BubbleSort2(int *v, int n) {
74     int j, continua, fim = n;
75     do {
76         continua = 0;
77         for (j = 0; j < fim - 1; j++) {
78             comp++;
79             if (v[j] < v[j + 1]) {
80                 troca(&v[j], &v[j + 1]);
81                 mov++;
82             }
83         }
84         continua = 1;
85         fim = j;
86     } while (continua);
87 }
```

```

79         continua = j;
80     }
81 }
82 fim--;
83 } while (continua != 0);
84 }
85
86 // Insertion Sort
87 void InsertionSort(int *v, int n) {
88     int i, j, atual;
89     for (i = 1; i < n; i++) {
90         atual = v[i];
91         comp++;
92         for (j = i; (j > 0) && (atual >
93             v[j - 1]); j--) {
94             v[j] = v[j - 1];
95             comp++;
96             mov++;
97         }
98         v[j] = atual;
99     }
100
101 // Selection Sort
102 void SelectionSort(int *v, int n) {
103     int i, j, maior;
104     for (i = 0; i < n - 1; i++) {
105         maior = i;
106         for (j = i + 1; j < n; j++) {
107             comp++;
108             if (v[j] > v[maior])
109                 maior = j;
110         }
111         if (i != maior) {
112             troca(&v[i], &v[maior]);
113             mov++;
114         }
115     }
116 }

```

codigos/questao12/questao12.c

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4 #include "questao12.h"
5
6 int main(){
7     //Atribuicoes iniciais
8     srand(time(NULL));
9     setComp(0);
10    setMov(0);
11    clock_t t;
12
13
14    int *v1, *v2, *v3;
15    int n;
16
17    /* Recebe valores e preenche */
18    printf("Digite o tamanho do
19        vetor:\n");
20    scanf("%d", &n);
21    v1 = (int*) malloc (n*sizeof(int));
22
23    preencheAleatorio(v1, n, 1, 100);
24    imprimeVetor(v1, n);
25
26    v2 = copiaVetor(v1, n);
27    v3 = copiaVetor(v1, n);
28
29    /* BubbleSort */
30    t = clock();
31    BubbleSort(v1, n);
32    t = clock() - t;
33    printf("\nInformacoes da Ordenacao
34        por BubbleSort:\n");
35    printf("Tempo Execucao: %f
36        seconds.\n",
37        ((float)t)/CLOCKS_PER_SEC);
38    printf("Comparacoes: %d\n",
39        getComp());
40    printf("Movimentacoes: %d\n",
41        getMov());
42    printf("Memoria (bytes): %ld\n",
43        n*sizeof(int));
44
45    imprimeVetor(v2, n);
46
47    /* InsertionSort */
48    setComp(0);
49    setMov(0);
50    t = clock();
51    InsertionSort(v2, n);
52    t = clock() - t;
53    printf("\nInformacoes da Ordenacao
54        por InsertionSort:\n");
55    printf("Tempo Execucao: %f
56        seconds.\n",
57        ((float)t)/CLOCKS_PER_SEC);
58    printf("Comparacoes: %d\n",
59        getComp());
60    printf("Movimentacoes: %d\n",
61        getMov());
62    printf("Memoria (bytes): %ld\n",
63        n*sizeof(int));
64
65    imprimeVetor(v3, n);
66
67    /* SelectionSort */
68    setComp(0);
69    setMov(0);
70    t = clock();
71    SelectionSort(v3, n);
72    t = clock() - t;
73    printf("\nInformacoes da Ordenacao
74        por SelectionSort:\n");
75    printf("Tempo Execucao: %f
76        seconds.\n",
77        ((float)t)/CLOCKS_PER_SEC);
78    printf("Comparacoes: %d\n",
79        getComp());
80    printf("Movimentacoes: %d\n",
81        getMov());
82    printf("Memoria (bytes): %ld\n",
83        n*sizeof(int));
84
85    imprimeVetor(v1, n);
86
87    return 0;
88 }

```

```

58     SelectionSort(v3, n);
59     t = clock() - t;
60     printf("\nInformacoes da Ordenacao
        por SelectionSort:\n");
61     printf("Tempo Execucao:  %f
        seconds.\n",
        ((float)t)/CLOCKS_PER_SEC);
62     printf("Comparacoes:  %d\n",
        getComp());
63     printf("Movimentacoes:  %d\n",
        getMov());
64     printf("Memoria (bytes):  %ld\n",
        n*sizeof(int));
65
66     imprimeVetor(v3, n);
67
68     free(v1);
69     free(v2);
70     free(v3);
71
72     return 0;
73 }

```

codigos/questao12/main.c

```

1 all: questao12.o
2     gcc questao12.o main.c -o main
3
4 questao12.o: questao12.h questao12.c
5     gcc -c questao12.c
6
7 clean:
8     rm -f questao12.o main

```

codigos/questao12/Makefile

Saída

```

Digite o tamanho do vetor:
10
[46, 46, 33, 49, 13, 62, 42, 20, 13, 37]

Informacoes da Ordenacao por BubbleSort:
Tempo Execucao:  0.00018 seconds.
Comparacoes: 45
Movimentacoes: 15
Memoria (bytes): 40
[62, 49, 46, 46, 42, 37, 33, 20, 13, 13]

```

Figura 3: Questão 1.1 - Saída 1

```

Informacoes da Ordenacao por InsertionSort:
Tempo Execucao:  0.00013 seconds.
Comparacoes: 24
Movimentacoes: 15
Memoria (bytes): 40
[62, 49, 46, 46, 42, 37, 33, 20, 13, 13]

Informacoes da Ordenacao por SelectionSort:
Tempo Execucao:  0.00017 seconds.
Comparacoes: 45
Movimentacoes: 7
Memoria (bytes): 40
[62, 49, 46, 46, 42, 37, 33, 20, 13, 13]

```

Figura 4: Questão 1.1 - Saída 2

1.3 Utilizando entradas grandes

Código

```

1 #ifndef ORDENACAO_PT1
2 #define ORDENACAO_PT1
3
4 int getComp();
5 int getMov();
6 void setComp(int);
7 void setMov(int);
8 int* copiaVetor(int*, int);
9 void imprimeVetor(int*, int);
10 void preencheAleatorio(int*, int, int,
        int);
11 void troca(int*, int *);
12 void BubbleSort(int *, int);
13 void BubbleSort2(int *, int);
14 void InsertionSort(int *, int);
15 void SelectionSort(int *, int);
16
17 #endif

```

codigos/questao13/questao13.h

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4  #include "questao13.h"
5
6  //Medidas de Complexidade
7  int comp; //Num. de comparacoes
8  int mov; //Num. de movimentacoes
9
10
11 int getComp(){
12     return comp;
13 }
14
15 int getMov(){
16     return mov;
17 }
18
19 void setComp(int valor){
20     comp = valor;
21 }
22
23 void setMov(int valor){
24     mov = valor;
25 }
26
27 int* copiaVetor(int* v, int n){
28     int i;
29     int *v2;
30     v2 = (int*) malloc (n*sizeof(int));
31     for(i=0; i<n; i++) v2[i] = v[i];
32     return v2;
33 }
34
35 void imprimeVetor(int* v, int n){
36     int i, prim = 1;
37     printf("[");
38     for(i=0; i<n; i++)
39         if(prim){ printf("%d", v[i]);
40                     prim = 0; }
41         else printf(", %d", v[i]);
42     printf("]\n");
43 }
44
45 void preencheAleatorio(int* v, int n,
46     int ini, int fim){
47     int i;
48     for(i=0; i<n; i++)
49         v[i] = ini + rand() % (fim-ini
50             + 1);
51 }
52
53 void troca(int* a, int *b){
54     int aux = *a;
55     *a = *b;
56     *b = aux;
57 }
58
59 void BubbleSort(int *v, int n){

```

```

57     int i, j;
58     for(i=0; i<n-1; i++){
59         for(j=0; j<n-i-1; j++){
60             comp++;
61             if (v[j]>v[j+1]) {
62                 troca(&v[j], &v[j+1]);
63             }
64             mov++;
65         }
66     }
67
68 void BubbleSort2(int *v, int n){
69     int j, continua, fim = n;
70     do{
71         continua = 0;
72         for(j=0; j < fim-1; j++){
73             comp++;
74             if (v[j]>v[j+1]) {
75                 troca(&v[j], &v[j+1]);
76             }
77             mov++;
78             continua = j;
79         }
80         fim--;
81     }while(continua != 0);
82 }
83
84 void InsertionSort(int *v, int n){
85     int i, j, atual;
86     for(i=1; i < n; i++){
87         atual = v[i];
88         comp++;
89         for(j=i; (j>0) && (atual < v[j-1]);
90             j--){
91             v[j] = v[j-1];
92             comp++;
93             mov++;
94         }
95         v[j] = atual;
96     }
97
98 void SelectionSort(int *v, int n){
99     int i, j, menor;
100    for(i=0; i < n-1; i++){
101        menor = i;
102        for(j=i+1; j < n; j++){
103            comp++;
104            if (v[j] < v[menor])
105                menor = j;
106        }
107        if(i != menor){
108            troca(&v[i], &v[menor]);
109            mov++;
110        }
111    }
112 }

```

codigos/questao13/questao13.c

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4  #include "questao13.h"
5
6  int main(int argc, char *argv[]){
7
8      //Atribuicoes iniciais
9      setComp(0);
10     setMov(0);
11     clock_t t;
12
13     int n;
14
15     /* Leitura de arquivo e
16        verificacoes */
17     if (argc != 2) {
18         fprintf(stderr, "Uso: %s
19             <nome_do_arquivo>\n",
20                 argv[0]);
21         return 1;
22     }
23
24     FILE *arquivo = fopen(argv[1], "r");
25
26     if (arquivo == NULL) {
27         fprintf(stderr, "Erro ao abrir
28             o arquivo %s.\n", argv[1]);
29         return 1;
30     }
31
32     /* Pega o tamanho do arquivo e cria
33        um vetor com esse tamanho */
34     fscanf(arquivo, "%d", &n);
35
36     int *v1 = (int*)malloc(n *
37         sizeof(int));
38     int *v2 = (int*)malloc(n *
39         sizeof(int));
40     int *v3 = (int*)malloc(n *
41         sizeof(int));
42
43     /* Confere se o vetor nao esta
44        vazio */
45     if (v1 == NULL) {
46         fprintf(stderr, "Erro ao alocar
47             memoria para o vetor.\n");
48         fclose(arquivo);
49         return 1;
50     }
51
52     /* Copia os dados do arquivo para
53        esse vetor */
54     for (int i = 0; i < n; i++) {
55         fscanf(arquivo, "%d", &v1[i]);
56     }
57
58     v2 = copiaVetor(v1, n);
59     v3 = copiaVetor(v1, n);
60
61     /* BubbleSort */
62     t = clock();
63     BubbleSort(v1, n);
64     t = clock() - t;
65     printf("\nInformacoes da Ordenacao
66         por BubbleSort:\n");
67     printf("Tempo Execucao: %f
68         seconds.\n",
69         ((float)t)/CLOCKS_PER_SEC);
70     printf("Comparacoes: %d\n",
71         getComp());
72     printf("Movimentacoes: %d\n",
73         getMov());
74     printf("Memoria (bytes): %ld\n",
75         n*sizeof(int));
76
77     /* InsertionSort */
78     setComp(0);
79     setMov(0);
80     t = clock();
81     InsertionSort(v2, n);
82     t = clock() - t;
83     printf("\nInformacoes da Ordenacao
84         por InsertionSort:\n");
85     printf("Tempo Execucao: %f
86         seconds.\n",
87         ((float)t)/CLOCKS_PER_SEC);
88     printf("Comparacoes: %d\n",
89         getComp());
90     printf("Movimentacoes: %d\n",
91         getMov());
92     printf("Memoria (bytes): %ld\n",
93         n*sizeof(int));
94
95     /* SelectionSort */
96     setComp(0);
97     setMov(0);
98     t = clock();
99     SelectionSort(v3, n);
100    t = clock() - t;
101    printf("\nInformacoes da Ordenacao
102        por SelectionSort:\n");
103    printf("Tempo Execucao: %f
104        seconds.\n",
105        ((float)t)/CLOCKS_PER_SEC);
106    printf("Comparacoes: %d\n",
107        getComp());
108    printf("Movimentacoes: %d\n",
109        getMov());
110    printf("Memoria (bytes): %ld\n",
111        n*sizeof(int));
112
113    free(v1);
114    free(v2);
115    free(v3);
116
117    fclose(arquivo);

```



```
90     return 0;
```

```
91 }
```

codigos/questao13/main.c

```
1 all: questao13.o
2   gcc questao13.o main.c -o main
3
4 questao13.o: questao13.h questao13.c
5   gcc -c questao13.c
```

```
6
7 clean:
8   rm -f questao13.o main
```

codigos/questao13/Makefile

Saída:

Devido à expressiva quantidade de arquivos de entrada e considerando o tempo significativo que seria demandado para processar um milhão de elementos, optou-se por utilizar exclusivamente os conjuntos que continham o segundo maior número de elementos, totalizando cem mil números.

```
Informacoes da Ordenacao por BubbleSort:
Tempo Execucao: 47.829521 seconds.
Comparacoes: 704982704
Movimentacoes: 704982704
Memoria (bytes): 400000

Informacoes da Ordenacao por InsertionSort:
Tempo Execucao: 23.967834 seconds.
Comparacoes: 705082703
Movimentacoes: 704982704
Memoria (bytes): 400000

Informacoes da Ordenacao por SelectionSort:
Tempo Execucao: 18.944656 seconds.
Comparacoes: 704982704
Movimentacoes: 50000
Memoria (bytes): 400000
```

Figura 5: Questão 2.1 - Ordem decrescente

```
Informacoes da Ordenacao por BubbleSort:
Tempo Execucao: 58.377834 seconds.
Comparacoes: 704982704
Movimentacoes: -1800273297
Memoria (bytes): 400000

Informacoes da Ordenacao por InsertionSort:
Tempo Execucao: 11.892709 seconds.
Comparacoes: -1800173298
Movimentacoes: -1800273297
Memoria (bytes): 400000

Informacoes da Ordenacao por SelectionSort:
Tempo Execucao: 18.458241 seconds.
Comparacoes: 704982704
Movimentacoes: 99988
Memoria (bytes): 400000
```

Figura 6: Questão 2.1 - Ordem misturada

```
Informacoes da Ordenacao por BubbleSort:
Tempo Execucao: 20.242315 seconds.
Comparacoes: 704982704
Movimentacoes: 0
Memoria (bytes): 400000

Informacoes da Ordenacao por InsertionSort:
Tempo Execucao: 0.000584 seconds.
Comparacoes: 99999
Movimentacoes: 0
Memoria (bytes): 400000

Informacoes da Ordenacao por SelectionSort:
Tempo Execucao: 18.408474 seconds.
Comparacoes: 704982704
Movimentacoes: 0
Memoria (bytes): 400000
```

Figura 7: Questão 2.1 - Ordem crescente

```
Informacoes da Ordenacao por BubbleSort:
Tempo Execucao: 21.178679 seconds.
Comparacoes: 704982704
Movimentacoes: 316426
Memoria (bytes): 400000

Informacoes da Ordenacao por InsertionSort:
Tempo Execucao: 0.002068 seconds.
Comparacoes: 416425
Movimentacoes: 316426
Memoria (bytes): 400000

Informacoes da Ordenacao por SelectionSort:
Tempo Execucao: 18.890879 seconds.
Comparacoes: 704982704
Movimentacoes: 12
Memoria (bytes): 400000
```

Figura 8: Questão 2.1 - Quase em ordem crescente

1.4 Reimplementação usando TAD Pessoa

Código

```
1 #ifndef ORDENACAO_PT1
2 #define ORDENACAO_PT1
3
4 typedef struct pessoa{
5     int idade;
6     char* nome;
7 }Pessoa;
8
9
10 int getComp();
11 int getMov();
12 void setComp(int);
13 void setMov(int);
14 Pessoa* criaPessoa();
15 void liberaPessoa(Pessoa*);
```

```
16 Pessoa* copiaVetor(Pessoa*, int);
17 void imprimeVetor(Pessoa*, int);
18 void preencheAleatorio(int*, int, int,
19     int);
20 void troca(Pessoa*, Pessoa*);
21 void BubbleSort(Pessoa*, int);
22 void InsertionSort(Pessoa*, int);
23 void SelectionSort(Pessoa*, int);
24 void InsertionDecres(Pessoa*, int);
25 void SelectionDecres(Pessoa*, int);
26
27 #endif
```

codigos/questao14/questao14.h

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4 #include <string.h>
5 #include "questao14.h"
6
7 //Medidas de Complexidade
8 int comp; //Num. de comparacoes
9 int mov; //Num. de movimentacoes
10
11
12 int getComp(){
13     return comp;
14 }
15
16 int getMov(){
17     return mov;
18 }
19
20 void setComp(int valor){
21     comp = valor;
22 }
23
24 void setMov(int valor){
25     mov = valor;
26 }
27
28 Pessoa* criaPessoa(){
29     Pessoa *pessoa = (Pessoa*) malloc
30         (sizeof(Pessoa));
31     pessoa->idade = 0;
32     pessoa->nome = "";
33     return pessoa;
34 }
35
```

```
36 Pessoa* copiaVetor(Pessoa* v, int n){
37     int i;
38     Pessoa* v2 = (Pessoa*)malloc(n *
39         sizeof(Pessoa));
40
41     if (v2 == NULL){
42         fprintf(stderr, "Erro ao alocar
43             memoria para o vetor de
44             copia.\n");
45         exit(1);
46     }
47
48     for (i = 0; i < n; i++){
49         v2[i].idade = v[i].idade;
50         v2[i].nome = strdup(v[i].nome);
51         if (v2[i].nome == NULL){
52             fprintf(stderr, "Erro ao
53                 alocar memoria para a
54                 copia do nome.\n");
55             exit(1);
56         }
57     }
58     return v2;
59 }
60
61 void liberaPessoa(Pessoa* pessoa){
62     if (pessoa != NULL){
63         free(pessoa->nome);
64         free(pessoa);
65     }
66 }
67
68 void imprimeVetor(Pessoa* v, int n){
69     int i, prim = 1;
70     printf("[");
71     for (i = 0; i < n; i++){
```

```

67         if (prim){
68             printf("{idade: %d, nome:
                %s}", v[i].idade,
                v[i].nome);
69             prim = 0;
70         } else {
71             printf(", {idade: %d, nome:
                %s}", v[i].idade,
                v[i].nome);
72         }
73     }
74     printf("]\n");
75 }
76
77 void preencheAleatorio(int* v, int n,
    int ini, int fim){
78     int i;
79     for(i=0; i<n; i++)
80         v[i] = ini + rand() % (fim-ini
            + 1);
81 }
82
83 void troca(Pessoa* a, Pessoa* b){
84     Pessoa aux = *a;
85     *a = *b;
86     *b = aux;
87     mov++;
88 }
89
90 void BubbleSort(Pessoa* v, int n){
91     int i, j;
92     for(i = 0; i < n - 1; i++){
93         for(j = 0; j < n - i - 1; j++){
94             comp++;
95             // Compara primeiro os
                nomes, em caso de empate
                compara as idades
96             if (strcmp(v[j].nome, v[j +
                1].nome) > 0 ||
                (strcmp(v[j].nome, v[j +
                1].nome) == 0 &&
                v[j].idade > v[j +
                1].idade)){
97                 troca(&v[j], &v[j + 1]);
98             }
99         }
100     }
101 }
102
103 void InsertionSort(Pessoa* v, int n){
104     int i, j;
105     Pessoa atual;
106     for(i = 1; i < n; i++){
107         atual = v[i];
108         comp++;
109         for(j = i; (j > 0) &&
            (strcmp(atual.nome, v[j -
                1].nome) < 0 ||
            (strcmp(atual.nome, v[j -
                1].nome) == 0 && atual.idade
                1].nome) == 0 && atual.idade
                < v[j - 1].idade)); j--){
            v[j] = v[j - 1];
            comp++;
            mov++;
        }
        v[j] = atual;
    }
}

118 void SelectionSort(Pessoa* v, int n){
119     int i, j, menor;
120     for(i = 0; i < n - 1; i++){
121         menor = i;
122         for(j = i + 1; j < n; j++){
123             comp++;
124             // Compara primeiro os
                nomes, em caso de empate
                compara as idades
125             if (strcmp(v[j].nome,
                v[menor].nome) < 0 ||
                (strcmp(v[j].nome,
                v[menor].nome) == 0 &&
                v[j].idade <
                v[menor].idade)){
126                 menor = j;
127             }
128         }
129         if(i != menor){
130             troca(&v[i], &v[menor]);
131         }
132     }
133 }
134
135 void InsertionDecres(Pessoa* v, int n){
136     int i, j;
137     Pessoa atual;
138     for(i = 1; i < n; i++){
139         atual = v[i];
140         comp++;
141         for(j = i; (j > 0) &&
            (strcmp(atual.nome, v[j -
                1].nome) > 0 ||
            (strcmp(atual.nome, v[j -
                1].nome) == 0 && atual.idade
                > v[j - 1].idade)); j--){
            v[j] = v[j - 1];
            comp++;
            mov++;
        }
        v[j] = atual;
    }
}

150 void SelectionDecres(Pessoa* v, int n){
151     int i, j, menor;
152     for(i = 0; i < n - 1; i++){
153         menor = i;
154         for(j = i + 1; j < n; j++){

```

```

155         comp++;
156         // Compara primeiro os
            nomes, em caso de empate
            compara as idades
157         if (strcmp(v[j].nome,
            v[menor].nome) > 0 ||
            (strcmp(v[j].nome,
            v[menor].nome) == 0 &&
            v[j].idade >
            v[menor].idade)){

```

```

            menor = j;
        }
    }
    if(i != menor){
        troca(&v[i], &v[menor]);
    }
}

```

codigos/questao14/questao14.c

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4 #include <string.h>
5 #include "questao14.h"
6
7 int main(){
8     srand(time(NULL));
9     setComp(0);
10    setMov(0);
11    clock_t t;
12
13    Pessoa *v1, *v2, *v3, *v4;
14    Pessoa *p1, *p2, *p3;
15    int n = 3;
16
17    v1 = (Pessoa*) malloc(n *
        sizeof(Pessoa));
18
19    p1 = criaPessoa();
20    p1->idade = 20;
21    p1->nome = strdup("Fulano");
22    v1[0] = *p1;
23
24    p2 = criaPessoa();
25    p2->idade = 50;
26    p2->nome = strdup("Fulano");
27    v1[1] = *p2;
28
29    p3 = criaPessoa();
30    p3->idade = 30;
31    p3->nome = strdup("Beltrano");
32    v1[2] = *p3;
33
34    imprimeVetor(v1, n);
35
36    v2 = copiaVetor(v1, n);
37    v3 = copiaVetor(v1, n);
38    v4 = copiaVetor(v1, n);
39
40    /* InsertionSort */
41    setComp(0);
42    setMov(0);
43    t = clock();
44    InsertionSort(v1, n);

```

```

45    t = clock() - t;
46    printf("\nInformacoes da ordenacao
        por InsertionSort crescente:\n");
47    printf("Tempo Execucao: %f
        seconds.\n", ((float)t) /
        CLOCKS_PER_SEC);
48    printf("Comparacoes: %d\n",
        getComp());
49    printf("Movimentacoes: %d\n",
        getMov());
50    printf("Memoria (bytes): %ld\n", n
        * sizeof(Pessoa));
51
52    imprimeVetor(v1, n);
53
54    /* SelectionSort */
55    setComp(0);
56    setMov(0);
57    t = clock();
58    SelectionSort(v2, n);
59    t = clock() - t;
60    printf("\nInformacoes da Ordenacao
        por SelectionSort crescente:\n");
61    printf("Tempo Execucao: %f
        seconds.\n", ((float)t) /
        CLOCKS_PER_SEC);
62    printf("Comparacoes: %d\n",
        getComp());
63    printf("Movimentacoes: %d\n",
        getMov());
64    printf("Memoria (bytes): %ld\n", n
        * sizeof(Pessoa));
65
66    imprimeVetor(v2, n);
67
68    /* InsertionSort Descrescente*/
69    setComp(0);
70    setMov(0);
71    t = clock();
72    InsertionDecres(v1, n);
73    t = clock() - t;
74    printf("\nInformacoes da ordenacao
        por InsertionSort
        decrescente:\n");
75    printf("Tempo Execucao: %f

```

```

        seconds.\n", ((float)t) /
        CLOCKS_PER_SEC);
76 printf("Comparacoes: %d\n",
        getComp());
77 printf("Movimentacoes: %d\n",
        getMov());
78 printf("Memoria (bytes): %ld\n", n
        * sizeof(Pessoa));
79
80 imprimeVetor(v1, n);
81
82 /* SelectionSort Descrescente*/
83 setComp(0);
84 setMov(0);
85 t = clock();
86 SelectionDecres(v2, n);
87 t = clock() - t;
88 printf("\nInformacoes da Ordenacao
        por SelectionSort
        decrescente:\n");
89 printf("Tempo Execucao: %f
        seconds.\n", ((float)t) /
        CLOCKS_PER_SEC);
90 printf("Comparacoes: %d\n",

        getComp());
91 printf("Movimentacoes: %d\n",
        getMov());
92 printf("Memoria (bytes): %ld\n", n
        * sizeof(Pessoa));
93
94 imprimeVetor(v2, n);
95
96 free(v1);
97 free(v2);
98 free(v3);
99 free(v4);
100
101 // Certifique-se de liberar a
        memoria alocada para cada Pessoa
102 liberaPessoa(p1);
103 liberaPessoa(p2);
104 liberaPessoa(p3);
105
106 return 0;
107 }

```

codigos/questao14/main.c

```

1 all: questao14.o
2 gcc questao14.o main.c -o main
3
4 questao14.o: questao14.h questao14.c
5 gcc -c questao14.c

6
7 clean:
8 rm -f questao14.o main

```

codigos/questao14/Makefile

Saída

```

[[{idade: 20, nome: Fulano}, {idade: 50, nome: Fulano}, {idade: 30, nome: Beltrano}]

Informacoes da ordenacao por InsertionSort crescente:
Tempo Execucao: 0.000004 seconds.
Comparacoes: 4
Movimentacoes: 2
Memoria (bytes): 48
[[{idade: 30, nome: Beltrano}, {idade: 20, nome: Fulano}, {idade: 50, nome: Fulano}]

Informacoes da Ordenacao por SelectionSort crescente:
Tempo Execucao: 0.000005 seconds.
Comparacoes: 3
Movimentacoes: 2
Memoria (bytes): 48
[[{idade: 30, nome: Beltrano}, {idade: 20, nome: Fulano}, {idade: 50, nome: Fulano}]

```

Figura 9: Questão 1.1 - Saída 1

```
Informacoes da ordenacao por InsertionSort decrescente:
Tempo Execucao: 0.000004 seconds.
Comparacoes: 5
Movimentacoes: 3
Memoria (bytes): 48
[{idade: 50, nome: Fulano}, {idade: 20, nome: Fulano}, {idade: 30, nome: Beltrano}]

Informacoes da Ordenacao por SelectionSort decrescente:
Tempo Execucao: 0.000003 seconds.
Comparacoes: 3
Movimentacoes: 1
Memoria (bytes): 48
[{idade: 50, nome: Fulano}, {idade: 20, nome: Fulano}, {idade: 30, nome: Beltrano}]
```

Figura 10: Questão 1.1 - Saída 2