Universidade Federal de Goiás
Goiânia 15 de Abril de 2018
Edson Júnior Frota Silva – 201515412
Engenharia de Computação
Atividade de Estrutura de Dados II

QUESTÃO 01: Implemente na linguagem C o algoritmo de ordenação Shell sort. Utilize uma função auxiliar para implementar a ordenação.

```c
#include<stdio.h>
void Shell_Sort(int *ponteiro, int tamanho);

int main(int argc, char** argv)
{
    int dimensao;
    printf("Informe a dimensao do vetor: ");
    scanf("%i ",&dimensao);

    int i, vetor[dimensao];

    for(i = 0; i <dimensao; i++)
    {
        scanf("%i", &vetor[i]);
    }

    Shell_Sort(vetor, dimensao);

    printf("\nValores ordenados:\n");
    for(i = 0; i < dimensao; i++)
    {
        printf("%i\n", vetor[i]);
    }
    return (0);
}
void Shell_Sort(int *ponteiro, int tamanho)
{
    int i , j , contador;
    int posicao = 1;

    do
    {
        posicao = 3 * posicao + 1;
    }
    while(posicao < tamanho);

    do
    {
        posicao /= 3;
        for(i = posicao; i < tamanho; i++)
        {
            contador = ponteiro[i];
            j = i - posicao;

            while (j >= 0 && contador < ponteiro[j])
            {
                ponteiro[j + posicao] = ponteiro[j];
                j -= posicao;
            }
```

```
50              ponteiro[j + posicao] = contador;
51          }
52      }
53      while(posicao > 1);
54  }
```

QUESTÃO 02: Implemente na linguagem C o algoritmo de ordenação Quick sort. Utilize uma função auxiliar para implementar a ordenação.

```
 1  #include<stdio.h>
 2  void QuickSort(int *vetor, int inicio, int fim)
 3  {
 4      int i, j, meio, contador;
 5
 6      i = inicio;
 7      j = fim;
 8      meio = vetor[(inicio + fim) / 2];
 9      do
10      {
11          while(vetor[i] < meio)
12          {
13              i++;
14          }
15          while(vetor[j] > meio)
16          {
17              j--;
18          }
19          if(i <= j)
20          {
21              contador = vetor[i];
22              vetor[i] = vetor[j];
23              vetor[j] = contador;
24              i++;
25              j--;
26          }
27      }
28      while(i <= j);
29
30      if(inicio < j)
31      {
32          QuickSort(vetor, inicio, j);
33      }
34      if(i < fim)
35      {
36          QuickSort(vetor, i, fim);
37      }
38  }
39  int main()
40  {
41      int i, tamanho;
42      printf("Informe a dimensao do vetor: ");
43      scanf("%i", &tamanho);
44
45      int vetor[tamanho];
46
47      for(i = 0; i < tamanho; i++)
48      {
49          scanf("%i", &vetor[i]);
50      }
51      QuickSort(vetor,0,tamanho);
```

```c

    printf("Vetor Ordenado: ");
    for( i = 0; i < tamanho; i++)
    {
        printf("\n%i ", vetor[i]);
    }
    return(0);
}
```

QUESTÃO 03: Implemente na linguagem C o algoritmo de ordenação Merge sort. Utilize uma função auxiliar para implementar a ordenação.

```c
#include<stdio.h>
void merge(int vetor[], int inicio, int meio, int fim)
{
    int inicio_A, inicio_B, inicio_Aux, tamanho;
    int *vetAux;

    inicio_A = inicio;
    inicio_B = meio+1;
    inicio_Aux = 0;
    tamanho = fim-inicio+1;

    vetAux = (int*)malloc(tamanho * sizeof(int));
    while(inicio_A <= meio && inicio_B <= fim)
    {
        if(vetor[inicio_A] < vetor[inicio_B])
        {
            vetAux[inicio_Aux] = vetor[inicio_A];
            inicio_A++;
        }
        else
        {
            vetAux[inicio_Aux] = vetor[inicio_B];
            inicio_B++;
        }
        inicio_Aux++;
    }
    while(inicio_A <= meio)
    {
        vetAux[inicio_Aux] = vetor[inicio_A];
        inicio_Aux++;
        inicio_A++;
    }
    while(inicio_B <= fim)
    {
        vetAux[inicio_Aux] = vetor[inicio_B];
        inicio_Aux++;
        inicio_B++;
    }
    for(inicio_Aux = inicio; inicio_Aux <= fim; inicio_Aux++)
    {
        vetor[inicio_Aux] = vetAux[inicio_Aux-inicio];
    }
    free(vetAux);
}
void mergeSort(int vetor[], int inicio, int fim)
{
    if (inicio < fim)
    {
```

```
49          int meio = ( fim + inicio ) / 2;
50          mergeSort(vetor, inicio, meio);
51          mergeSort(vetor, meio+1, fim);
52          merge(vetor, inicio, meio, fim);
53      }
54  }
55  int main()
56  {
57      int i, dimensao;
58
59      printf("Digite a dimensao do vetor: ");
60      scanf("%i", &dimensao);
61      int vetor[dimensao];
62
63      for(i = 0; i < dimensao; i++)
64      {
65          scanf("%i", &vetor[i]);
66      }
67      mergeSort(vetor, 0, dimensao);
68
69      for( i = 0; i < dimensao; i++)
70      {
71          printf("%i ", vetor[i]);
72      }
73      return(0);
74  }
```

QUESTÃO 04: O trecho de algoritmo a seguir corresponde ao método de ordenação do tipo:

```
1   declare X[5], n, i, aux numérico
2   para n ← 1 até 5 faça
3       início
4           para i ← 0 até 3 faça
5               início
6                   se (X [i] > X[i+1] )
7                       então início
8                           aux X[i]
9                           X [i] ← X[i + 1]
10                          X[i+1] ← aux
11              fim
12      fim
13  fim
```

b) Bubble Sort

QUESTÃO 05: O seguinte algoritmo, chamado ordena, implementa um conhecido método de ordenação para listas sequenciais:

```
ordena (int vet[], int n) {
        int i, j, pos, aux;
        para ( i = 0; i < n - 1; i++ ){
                pos = i;
                para ( j = i + 1; j < n; j++ )
                        se ( vet [pos] > vet [j] )
                                pos = j;
                se ( pos <> i ) {
                        aux = vet[i];
                        vet[i] = vet[pos];
                        vet[pos] = aux;
                }
        }
}
```

e) 7, decrescente

QUESTÃO 06: Considere o seguinte algoritmo, responsável por realizar a ordenação de um array de dados.

```
public int[] mySortingAlgorithm (int[] data){
 int size = data.length;
 int tmp = 0;
 for(int i = 0;i<size;i++){
  for(int j = (size-1);j>=(i+1);j--){
   if(data[j]<data[j-1]){
    tmp = data[j];
    data[j]=data[j-1];
    data[j-1]=tmp;
   }
  }
 }
 return data;
}
```

d) ShellSort

QUESTÃO 07: Observe o código abaixo, que busca o maior elemento de um vetor v[0..n - 1].

```
int max(int n, int v[])
{
  int j, x = v[0];
  for (j = 1; j < n; j += 1)
    if (x < v[j]) x = v[j];
  return x;
}
```

b) O(n)

QUESTÃO 08: Qual é o tipo de algoritmo de ordenação que tem como princípio percorrer o vetor diversas vezes, a cada passagem fazendo o maior elemento se mover para o final da estrutura?
d) BubbleSort

QUESTÃO 09: Correlacione os algoritmos internos de ordenação de listas da coluna à esquerda com sua descrição, na coluna à direita.

1) Bubblesort.
2) Ordenação por Seleção
3) Ordenação por Inserção
4) Shellsort
5) Quicksort

( ) Escolhe-se um pivot e particiona-se a lista em duas sublistas: uma com os elementos menores que ele e outra com os maiores, que, ao serem ordenadas e combinadas com o pivot, geram uma lista ordenada. O processo é aplicado às partições para ordená-las. Embora tenha uma complexidade de pior caso de O(n2), no caso médio é de O(n log n).

( ) Encontra-se o menor item do vetor. Troca-se com o item da primeira posição do vetor. Repetem-se essas duas operações com os n − 1 itens restantes, depois com os n − 2 itens, até que reste apenas um elemento.

( ) Método preferido dos jogadores de cartas. A cada momento existem duas partes na lista: uma ordenada (destino) e outra não ordenada (fonte). Inicialmente a lista destino tem apenas o primeiro elemento, e a fonte os demais elementos. Em cada passo a partir de i=2, seleciona-se o i-ésimo item da lista fonte. Deve-se colocá-lo no lugar apropriado na lista destino, de acordo com o critério de ordenação.

( ) É uma extensão de um outro algoritmo de ordenação conhecido e permite trocas de elementos distantes um do outro, não necessariamente adjacentes. Os itens separados de h posições são rearranjados. Todo h-ésimo item leva a uma lista ordenada. Tal lista é dita estar h-ordenada.

( ) Varre-se a lista trocando-se de posição os elementos adjacentes fora de ordem. Varre-se a lista até que não haja mais trocas e, neste caso, a lista está ordenada.

b) 5, 2, 3, 4, 1.

QUESTÃO 10: Quantas comparações e trocas de posição ocorrerão se utilizarmos o algoritmo Bubble Sort para ordenar do menor para o maior valor o vetor [60,32,45,5,6,2], respectivamente:
c) 25 e 15

QUESTÃO 11: O algoritmo de ordenação denominado quicksort é baseado na partição do arquivo em duas partes, a partir de um elemento arbitrariamente escolhido que termina localizado na sua posição final. Cada uma das partes é então ordenada independentemente, aplicando-se o algoritmo recursivamente, até que todo o arquivo esteja ordenado. Analise as mudanças na disposição dos elementos de um vetor com 10 elementos que é submetido ao processo de partição.

1	2	3	4	5	6	7	8	9	10
A	S	O	R	T	I	N	B	C	E
A	C	O	R	T	I	N	B	S	E
A	C	B	R	T	I	N	O	S	E
A	C	B	E	T	I	N	O	S	R

O elemento arbitrariamente escolhido foi aquele que estava na posição:

b) 5

QUESTÃO 12: Utilizando o algoritmo Shell sort, obtenha o número de comparações e movimentações em cada passo (h e i) para os seguintes vetores

- 45,56,12,43,95,19,8,67
- 8,12,19,43,45,56,67,95
- 95,67,56,45,43,19,12,8
- 19,12,8,45,43,56,67,95

```
1  /*mudanças em i: 20; em h: 4;
2  n de movimentações: 24;
3  n de comparações: 37;
4
5  *mudanças em i: 20; em h: 4;
6  n de movimentações: 17;
7  n de comparações: 37;
8
9  *mudanças em i: 20; em h: 4;
10 n de movimentações: 29;
11 n de comparações: 37;
12
13 *mudanças em i: 20; em h: 4;
14 n de movimentações: 19;
15 n de comparações: 37;
16 */
17 #include<stdio.h>
18 int main(){
19     int v[8];
20
21     v[0] = 45;
22     v[1] = 56;
23     v[2] = 12;
24     v[3] = 43;
25     v[4] = 95;
26     v[5] = 19;
27     v[6] = 8;
28     v[7] = 67;
29
30     int aux, i, j;
31     int conti = 0;
32     int conth = 0;
33     int contm = 0;
34     int contc = 0;
35
36     int h = 8/2;
37     conth++;
38
39     while(h > 0){
40         contc++;
41         i = h;
42         conti++;
43         while(i < 8){
44             contc++;
45             aux = v[i];
46             j = i;
```

```
47              contc++;
48              while(j >= h && aux < v[j - h]){
49                  v[j] = v[j - h];
50                  contm++;
51                  j = j -h;
52              }
53              v[j] = aux;
54              contm++;
55              i = i +1;
56              conti++;
57          }
58          h = h/2;
59          conth++;
60      }
61          printf("- %i - %i - %i - %i -", conti, conth, contm, contc);
62      return (0);
63  }
```

QUESTÃO 13: Escreva uma função que receba vetores disjuntos x[0..m-1] e y[0..n-1], ambos em ordem crescente, e produza um vetor z[0..m+n-1] que contenha o resultado da intercalação dos dois vetores dados. Escreva duas versões da função: uma iterativa e uma recursiva.

```
 1  #include<stdio.h>
 2
 3  void BubbleSort(int vetor[], int tamanho_Vetor)
 4  {
 5      int i, j, aux;
 6      for(i = 0; i < tamanho_Vetor - 1; i++)
 7      {
 8
 9          for( j = 0; j < (tamanho_Vetor -(i +1)); j++)
10          {
11
12              if(vetor[j] > vetor[j+1])
13              {
14                  aux = vetor[j];
15                  vetor[j] = vetor[j+1];
16                  vetor[j+1] = aux;
17              }
18          }
19      }
20  }
21
22  void Concatenar(int vetor_1[], int vetor_2[], int tamanho_Vetor)
23  {
24      int j, vetor_3[tamanho_Vetor];
25      for(j = 0; j < tamanho_Vetor; j++)
26      {
27          vetor_3[j] = vetor_1[j] + vetor_2[j];
28          printf("\n%i ", vetor_3[j]);
29      }
30  }
31
32  int main()
33  {
34      int tamanho , i;
35
36      printf("Digite a dimensao do vetor: ");
37      scanf("%i", &tamanho );
```

```
38
39      int vetor_1[tamanho];
40      int vetor_2[tamanho];
41
42      printf("\nInforme os elementos dos vetores:\n");
43
44      printf("\nPRIMEIRO VETOR: \n");
45      for(i = 0; i < tamanho; i++)
46      {
47          scanf("%i", &vetor_1[i]);
48      }
49
50      printf("\nSEGUNDO VETOR: \n");
51      for(i = 0; i < tamanho; i++)
52      {
53          scanf("%i", &vetor_1[i]);
54      }
55
56      printf("\nResultado: \n");
57      BubbleSort(vetor_1, tamanho);
58      BubbleSort(vetor_1, tamanho);
59      Concatenar(vetor_1, vetor_1, tamanho);
60  }
```

Edson Júnior Frota Silva
Engenharia de Computação