



**ANTONIO MENEGHETTI FACULDADE**  
**CURSO DE SISTEMAS DE INFORMAÇÃO**  
**TÓPICOS AVANÇADOS DE PROGRAMAÇÃO**

**GUILHERME LEMOS VIEIRA**

**ALGORITMOS DE ORDENAÇÃO EM C#**

**RECANTO MAESTRO - RS**







**2016**

# Sumário

1. <b>Manual de Instruções</b> .....	03
2. <b>Algoritmos de Ordenação</b> .....	04
2.1. Insertion Sort.....	05
2.2. Selection Sort.....	06
2.3. Comb Sort.....	07
2.4. Shell Sort.....	09
2.5. Gnome Sort.....	11

## 1. Manual de instruções:

**Passo 1:** Para executar os Algoritmos de Ordenação, de um duplo clique em “ConsoleGuilherme”.

Nome	Data de modificaç...	Tipo	Tamanho
 ConsoleGuilherme	20/04/2016 20:04	Aplicativo	8 KB
 ConsoleGuilherme.exe	19/04/2016 15:22	XML Configuratio...	1 KB
 ConsoleGuilherme.pdb	20/04/2016 20:04	Arquivo PDB	24 KB
 ConsoleGuilherme.vshost	20/04/2016 21:25	Aplicativo	23 KB
 ConsoleGuilherme.vshost.exe	19/04/2016 15:22	XML Configuratio...	1 KB
 ConsoleGuilherme.vshost.exe.manifest	30/10/2015 05:19	Arquivo MANIFEST	1 KB

**Passo 2:** Irá abrir a aplicação, a qual possui um Menu interativo com 5 opções de Algoritmos de Ordenação, para que possa escolher e testar os Algoritmos de Ordenação.

```
E:\Guilherme\A M F\6º Semestre\Tópicos Avançados de Programação\TAP Guilherme\TAP Guilherme\ConsoleGuilherme\ConsoleGuilherme\bin\Debu...
-----MENU-----
Escolha uma das opções abaixo:
1 - Inser Sort;
2 - Selection Sort;
3 - Comb Sort;
4 - Shell Sort;
5 - Gnome Sort;
6 - Sair;

SUA OPÇÃO:
```

**Passo 3:** Selecione entre 1 e 5 o Algoritmo de Ordenação desejado, em “SUA OPÇÃO” irá aparecer o numero selecionado, após isso tecle Enter para que o algoritmo seja executado.

```
SUA OPÇÃO: 3_
```

**Passo 4:** Ao ser executado, o algoritmo irá mostrar o “Vetor original”, o qual apresenta uma numeração de 100 posições desordenada, logo abaixo aparece o “Vetor ordenado” com as mesmas posições, porém em ordem crescente, bem como em “TEMPO” o tempo de execução do algoritmo.

```
E:\Guilherme\A M F\6° Semestre\Tópicos Avançados de Programação\TAP Guilherme\TAP Guilherme\Console\Guilherme\Console\Guilherme\bin\Debu...
-----MENU-----
Escolha uma das opções abaixo:
1 - Inser Sort;
2 - Selection Sort;
3 - Comb Sort;
4 - Shell Sort;
5 - Gnome Sort;
6 - Sair;

SUA OPÇÃO: 3
CombSort

Vetor Original:
23,45,2,91,26,7,24,62,30,61,3,61,18,69,85,84,77,80,25,77,50,80,44,32,15,59,59,17,18,65,74,83,73,91,43,64,75,15,66,76,26,
22,97,89,29,83,44,21,41,23,95,39,74,82,58,43,0,69,76,66,48,6,43,64,86,20,88,27,25,20,8,62,13,49,50,28,82,55,2,31,15,37,7
5,94,70,35,9,90,32,0,64,6,38,0,27,20,78,32,2,8,

Vetor Ordenado:
0,0,0,2,2,2,3,6,6,7,8,8,9,13,15,15,15,17,18,18,20,20,20,20,21,22,23,23,24,25,25,26,26,27,27,28,29,30,31,32,32,32,35,37,38,3
9,41,43,43,43,44,44,45,48,49,50,50,55,58,59,59,61,61,62,62,64,64,64,65,66,66,69,69,70,73,74,74,75,75,76,76,77,77,78,80,8
0,82,82,83,83,84,85,86,88,89,90,91,91,94,95,97,

TEMPO:      00:00:00.0000730

Tecle 'ENTER' para Próximo.
```

**Passo 5:** Para executar outro algoritmo, tecla Enter.

```
Tecle 'ENTER' para Próximo.
```

**Passo 6:** A aplicação irá limpar a tela e exibirá novamente o “Menu” com as opções, repita os passos 2, 3, 4 e 5 para executar os demais algoritmos.

```
-----MENU-----
Escolha uma das opções abaixo:
1 - Inser Sort;
2 - Selection Sort;
3 - Comb Sort;
4 - Shell Sort;
5 - Gnome Sort;
6 - Sair;

SUA OPÇÃO: 6
```

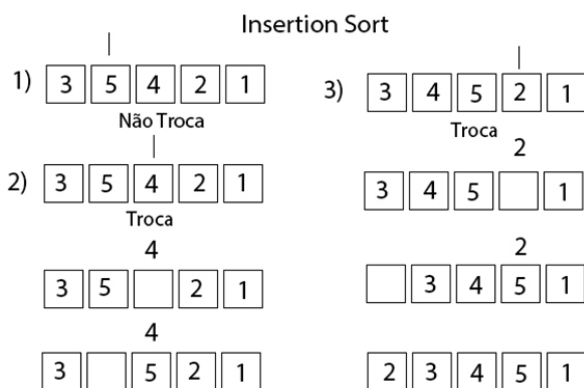
**Passo 7:** Para sair da aplicação, ao executar o passo 5, digite a opção 6 do “Menu” em “SUA OPÇÃO” e tecla Enter.

```
SUA OPÇÃO: 6_
```

## 2. Algoritmos de Ordenação

### 2.1 Insertion Sort

- Método simples de ordenação que percorre um vetor ordenando os elementos a esquerda a medida que avança.
- Algoritmo simples e eficiente quando aplicado em pequenas listas.
- Neste algoritmo a lista é percorrida da esquerda para a direita, à medida que avança vai deixando os elementos mais à esquerda ordenados.
- O algoritmo funciona da mesma forma que as pessoas usam para ordenar cartas em um jogo de baralho.



#### Vetor de ordenação:

```
int[] vetor = new int[100];  
for (int i = 0; i < vetor.Length; i++)  
{  
    vetor[i] = rand.Next(100);  
}
```

```
Console.WriteLine("Vetor Original: ");  
Imprimir(vetor);
```

```
int min, aux;
```

```
for (int i = 0; i < vetor.Length - 1; i++)  
{
```

```
    min = i;
```

```
    for (int j = i + 1; j < vetor.Length; j++)  
        if (vetor[j] < vetor[min])  
            min = j;
```

```
    if (min != i)
```

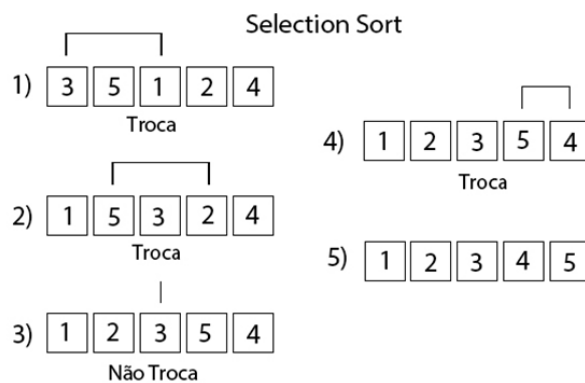
```
    {  
        aux = vetor[min];  
        vetor[min] = vetor[i];  
        vetor[i] = aux;
```

```
    }
```

```
}
```

## 2.2 Selection Sort

- Método de ordenação por seleção. Ele percorre a lista em busca do menor valor e o move para a posição correta precedido sempre do elemento de menor valor.
- Baseado em passar sempre o menor valor do vetor para a primeira posição (ou o maior dependendo da ordem requerida), depois o segundo menor valor para a segunda posição e assim sucessivamente, até os últimos dois elementos.
- É escolhido um número a partir do primeiro, este número escolhido é comparado com os números a partir da sua direita, quando encontrado um número menor, o número escolhido ocupa a posição do menor número encontrado. Este número encontrado será o próximo número escolhido, caso não for encontrado nenhum número menor que este escolhido, ele é colocado na posição do primeiro número escolhido, e o próximo número à sua direita vai ser o escolhido para fazer as comparações.
- O processo é repetido até que a lista esteja ordenada.



### Vetor de ordenação:

```
int[] vetor = new int[100];
for (int i = 0; i < vetor.Length; i++)
{
    vetor[i] = rand.Next(100);
}

Console.WriteLine("Vetor Original: ");
Imprimir(vetor);

int min, aux;

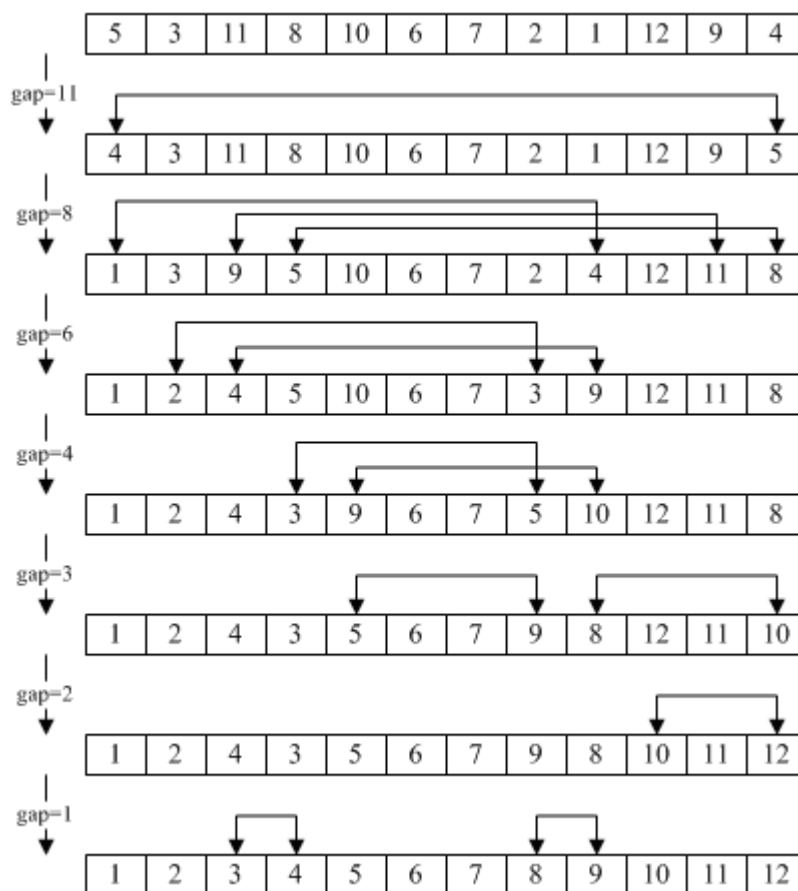
for (int i = 0; i < vetor.Length - 1; i++)
{
    min = i;

    for (int j = i + 1; j < vetor.Length; j++)
        if (vetor[j] < vetor[min])
            min = j;

    if (min != i)
    {
        aux = vetor[min];
        vetor[min] = vetor[i];
        vetor[i] = aux;
    }
}
```

## 2.3 Comb Sort

- Algoritmo de ordenação simples e faz parte da família de ordenação por troca. O Comb Sort é uma melhoria do Bubble Sort e rivaliza com o Quick Sort.
- Repetidamente reordena diferentes pares de itens, separados por um salto, que é calculado a cada passagem, esse método é semelhante ao Bubble Sort, porém mais eficiente.
- Na Bubble Sort, quando quaisquer dois elementos são comparados, eles sempre têm um gap (distancia um do outro) de 1. A idéia básica do Comb Sort é que a diferença pode ser muito mais do que um.
- O gap (intervalo) começa como o comprimento da lista a ser ordenada, dividida pelo fator de encolhimento (em geral 1,3).



### Vetor de ordenação:

```
Stopwatch stopWatch = new Stopwatch(); //declaração da contagem

Random rand = new Random();

int[] vetor = new int[100];
for (int i = 0; i < vetor.Length; i++)
{
    vetor[i] = rand.Next(100);
}

Console.WriteLine("CombSort\n"); //nome do algoritmo
Console.WriteLine("Vetor Original: ");
Imprimir(vetor);

stopWatch.Reset(); //zera o contador
```

```
stopWatch.Start(); //inicia a contagem

int gap = vetor.Length;
bool swapped = true;
while (gap > 1 || swapped)
{
    if (gap > 1)
    {
        gap = (int)(gap / 1.247330950103979);
    }

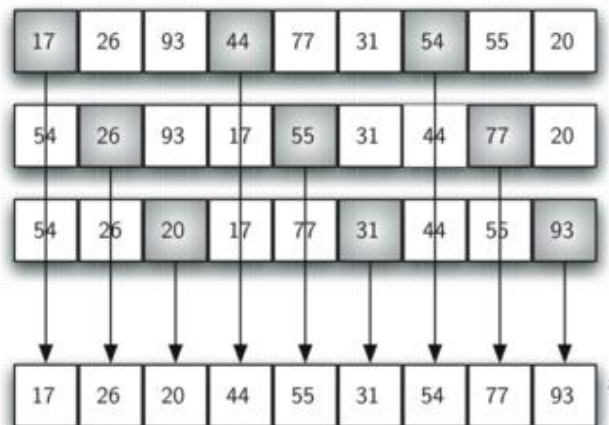
    int i = 0;
    swapped = false;
    while (i + gap < vetor.Length)
    {
        if (vetor[i].CompareTo(vetor[i + gap]) > 0)
        {
            int t = vetor[i];
            vetor[i] = vetor[i + gap];
            vetor[i + gap] = t;
            swapped = true;
        }
        i++;
    }
}

stopWatch.Stop(); //para a contagem
```



## 2.4 Shell sort

- É o mais eficiente algoritmo de classificação dentre os de complexidade quadrática
- O algoritmo passa várias vezes pela lista dividindo o grupo maior em menores, nos grupos menores é aplicado o método da ordenação por inserção.
- Baseia em uma variável chamada de incremento de sequência, ou incremento de shell, que é dado por  $h$  e ao decorrer da execução do algoritmo, é decrementada até 1.
- A ordenação é realizada em vários passos, usando uma sequência de valores do incremento de shell  $\langle h_1, h_2, h_3 \dots h_N \rangle$  onde começando por  $h_N$  selecionamos apenas os valores que estão  $h_N$  elementos distantes um do outro, então ordenamos esses elementos com algum algoritmo de ordenação simples como bolha, seleção ou inserção. Deste modo, apenas os elementos selecionados serão ordenados, os outros são todos ignorados.



### Vetor de Ordenação:

```
int[] vetor = new int[100];
for (int i = 0; i < vetor.Length; i++)
{
    vetor[i] = rand.Next(100);
}

Console.WriteLine("Vetor Original: ");
Imprimir(vetor);

int h = 1;
int n = vetor.Length;

while (h < n)
{
    h = h * 3 + 1;
}

h = h / 3;
int c, j;
while (h > 0)
{
    for (int i = h; i < n; i++)
    {
        c = vetor[i];
        j = i;
        while (j >= h && vetor[j - h] > c)
        {
            vetor[j] = vetor[j - h];
            j = j - h;
        }
    }
}
```

```
        }  
        vetor[j] = c;  
    }  
    h = h / 2;  
}
```

## 2.5 Gnome Sort

- Algoritmo similar ao Insertion Sort, mas com a diferença que leva o elemento para sua posição correta, em uma sequência grande de trocas assim como o Bubble Sort.
- Percorre o vetor comparando seus elementos dois a dois, assim que ele encontra um elemento que está na posição incorreta, ou seja, o número maior antes de um menor, ele troca a posição dos elementos, e volta com esse elemento até que encontre o seu respectivo lugar.

2	3	9	7	5	3 > 2, ok
2	3	9	7	5	9 > 3, ok
2	3	9	7	5	7 > 9, swap
2	3	7	9	5	7 > 3, ok
2	3	7	9	5	5 > 9, swap
2	3	7	5	9	5 > 7, swap
2	3	5	7	9	5 > 3, ok
2	3	5	7	9	sorted

### Vetor de ordenação:

```
int[] vetor = new int[100];
for (int i = 0; i < vetor.Length; i++)
{
    vetor[i] = rand.Next(100);
}

Console.WriteLine("Vetor Original: ");
Imprimir(vetor);

int p = 0;
int aux;
while (p < (vetor.Length - 1))
{
    if (vetor[p] > vetor[p + 1])
    {
        aux = vetor[p];
        vetor[p] = vetor[p + 1];
        vetor[p + 1] = aux;
        if (p > 0)
        {
            p -= 2;
        }
    }
    p++;
}
```