

MÉTODOS DE ORDENAÇÃO DE VETORES

Renan Roos - 21/04/2016

Esse manual apresentará rapidamente como funciona cada método de ordenação de dados na teoria com pseudocode e/ou na linguagem C#. Também teremos os prints de como é realizado no nosso software de execução. Todo o software foi feito na linguagem C# e dentro de cada método há o código comentado.

São aplicados nesse trabalho os métodos de seleção, inserção, heap, quick e odd-even (par-ímpar). Todos os exemplos mostrados e printados aqui, foram feitos com vetores de tamanho de 100 caracteres.

• MÉTODO DE SELEÇÃO (SELECTION SORT)

Um dos algoritmos mais simples de ordenação. Funciona da seguinte maneira. O método seleccione o menor número do vetor, trocando de posição com o primeiro da lista.

Assim, o segundo da lista será comparado com o restante, caso encontre um menor que ele, eles trocam de lugar. E assim é realizado todo o processo até todo o vetor estar ordenado.

Vantagens: Custo linear no tamanho da entrada para o número de movimentos de registros – a ser utilizado quando há registros muito grandes.

Desvantagens: Não adaptável. Não importa se o arquivo está parcialmente ordenado. Algoritmo não é estável.

Código em C usado em nosso material de referência, vide seção Referência Bibliográfica.

```
void Selecao (Item *A, Indice *n)
{
    Indice i, j, Min;
    Item x;
    for (i = 1; i <= *n - 1; i++)
    {
        Min = i;
        for (j = i + 1; j <= *n; j++)
            if (A[j].Chave < A[Min].Chave) Min = j;
        x = A[Min];
        A[Min] = A[i];
        A[i] = x;
    }
}
```

No programa de ordenação de dados:

Método: Seleção

Tempo de Execução: 00:00:00.0724301

Ordenação de Vetores

Opções de Ordenação: Seleção

Número de Posições:

Vetor Não-Ordenado:

65	44	40	21	98
56	58	54	48	87
48	16	23	45	5
57	70	10	55	1

Vetor Ordenado:

Tempo de Criação: 00:00:00.0710132

Tempo de Ordenação:

Gerar Vetor Limpar Vetor Ordenar

Ordenação de Vetores

Opções de Ordenação: Seleção

Número de Posições:

Vetor Não-Ordenado:

65	44	40	21	98
56	58	54	48	87
48	16	23	45	5
57	70	10	55	1

Vetor Ordenado:

1	1	2	3	5
6	7	7	8	8
10	11	12	13	16
16	17	10	10	10

Tempo de Criação: 00:00:00.0710132

Tempo de Ordenação: 00:00:00.0724301

Gerar Vetor Limpar Vetor Ordenar

- **MÉTODO DE INSERÇÃO (INSERTION SORT)**

O método de Inserção divide o vetor em duas partes, uma lista classificada e uma não classificada. Em cada movimento, o algoritmo move a variável do vetor não classificado para o classificado, até estar toda classificada. E assim faz todas as comparações, como acontece no método de Seleção.

Vantagens: Laço interno é eficiente, inserção é adequado para ordenar vetores pequenos, pois o custo é linear. É estável.

Desvantagens: Número de comparações tem crescimento quadrático. Alto custo de movimentação de elementos no vetor.

Código em C usado em nosso material de referência, vide seção Referência Bibliográfica.

```
void Insercao(Item *A, Indice *n)
{
    Indice i, j;
    Item x;
    for (i = 2; i <= *n; i++)
    {
        x = A[i]; j = i - 1;
        A[0] = x; /* sentinela */
        while (x.Chave < A[j].Chave)
        {
            A[j+1] = A[j]; j--;
        }
        A[j+1] = x;
    }
}
```

No programa de ordenação de dados:

Método: Inserção

Tempo de Execução: 00:00:00.0753728

Ordenação de Vetores

Opções de Ordenação:

Tempo de Criação: 00:00:00.0753728

Número de Posições:

Tempo de Ordenação:

Vetor Não-Ordenado:

48	81	16	42	81
75	90	89	83	10
30	70	55	81	80
96	79	17	90	17

Vetor Ordenado:

Ordeação de Vetores

Opções de Ordeação:

Inserção

Número de Posições:

Vetor Não-Ordenado:

48	81	16	42	81
75	90	89	83	10
30	70	55	81	80
96	79	47	90	17

Tempo de Criação:

00:00:00.0753728

Tempo de Ordeação:

00:00:00.0745221

Vetor Ordenado:

0	1	1	2	3
5	6	6	7	7
8	8	10	10	13
13	14	14	15	15

Gerar Vetor

Limpar Vetor

Ordenar

Código em C usado em nosso material de referencie, vide seção Referência Bibliográfica.

■ Função Partição:

```
void Particao(Indice Esq, Indice Dir,
            Indice *i, Indice *j, Item *A)
{ Item x, w;
  *i = Esq; *j = Dir;
  x = A[(*i + *j)/2]; /* obtem o pivo x */
  do
  { while (x.Chave > A[*i].Chave) (*i)++;
    while (x.Chave < A[*j].Chave) (*j)--;
    if ((*i) <= (*j))
    { w = A[*i]; A[*i] = A[*j]; A[*j] = w;
      (*i)++; (*j)--;
    }
  } while (*i <= *j);
}
```

Algoritmos e Estrutura de Dados II

No programa de ordenação de dados:

Método: Quick Sort

Tempo de Execução: 00:00:00.0691116

Ordenação de Vetores

Opções de Ordenação: Quick

Tempo de Criação: 00:00:00.0691116

Número de Posições:

Tempo de Ordenação:

Vetor Não-Ordenado:

8	17	98	23	77
19	78	87	68	9
41	73	52	31	20
65	40	16	50	12

Vetor Ordenado:

Gerar Vetor Limpar Vetor Ordenar

Ordenação de Vetores

Opções de Ordenação: Quick

Tempo de Criação: 00:00:00.0691116

Número de Posições:

Tempo de Ordenação: 00:00:00.0748317

Vetor Não-Ordenado:

8	17	98	23	77
19	78	87	68	9
41	73	52	31	20
65	40	16	50	12

Vetor Ordenado:

2	2	2	3	8
8	9	11	12	13
16	17	19	20	20
20	20	23	25	25

Gerar Vetor Limpar Vetor Ordenar

- **MÉTODO PAR/ÍMPAR (ODD-EVEN)**

É uma variação do Bubble Sort, onde Even significada par e Odd é ímpar. Ele funciona através da comparação de todos os números ímpares e pares do vetor. É feita a comparação entre os números, formando pares ordenados. Então, o primeiro é par e o segundo ímpar, caso fuja desse padrão, eles trocam de lugar.

No programa de ordenação de dados:
Método: Par-Ímpar
Tempo de Execução: 00:00:00.0701277

Ordenação de Vetores

Opções de Ordenação: Par/Ímpar

Tempo de Criação: 00:00:00.0701277

Número de Posições:

Tempo de Ordenação:

Vetor Não-Ordenado:

82	60	80	64	21
0	53	18	77	45
91	78	9	77	63
31	15	00	60	27

Vetor Ordenado:

--	--	--	--	--

Gerar Vetor **Limpar Vetor** **Ordenar**

Ordenação de Vetores

Opções de Ordenação: Par/Ímpar

Tempo de Criação: 00:00:00.0701277

Número de Posições:

Tempo de Ordenação: 00:00:00.0791120

Vetor Não-Ordenado:

82	60	80	64	21
0	53	18	77	45
91	78	9	77	63
31	15	00	60	27

Vetor Ordenado:

0	2	5	6	8
8	8	9	10	11
13	15	17	18	18
18	19	20	20	20

Gerar Vetor **Limpar Vetor** **Ordenar**

- **COMPARAÇÃO DE DESEMPENHO ENTRE OS MÉTODOS**

Inserção:

Melhor usar para menos de 20 elementos. Método é estável.

Melhor desempenho que Bubble Sort.

Seleção:

Melhor usar para até mil números.

Melhor desempenho que Bubble Sort e Inserção.

Quicksort:

É recursivo, o que demanda uma pequena quantidade de memória adicional.

Seu auxiliar chama-se pivot. Chama um método de ordenação simples nos arquivos pequenos.

Melhor desempenho que Bubble Sort, Inserção e Seleção.

Heapsort:

Não necessita de nenhuma memória adicional.

Feito com grafos e árvores binárias, dentro da Estrutura de Dados. Dá rapidez aos movimentos.

Ímpar-Par:

Variação do Bubble Sort, mas com características específicas.

É mais complicado do que implementar o Bubble.

Utilizados em casos onde precisamos somente ordenar por par ou ímpar.

- **REFERÊNCIAS BIBLIOGRÁFICAS**

Ordenação: Introdução e métodos elementares -
<http://homepages.dcc.ufmg.br/~cunha/teaching/20121/aeds2/sorting-intro.pdf>
Acessado em: 21/04/2016

Livro "Projeto de Algoritmos" – Nívio Ziviani Capítulo 4 -
<http://www2.dcc.ufmg.br/livros/algoritmos/> Acessado em: 21/04/2016

Algoritmos de Ordenação - http://www.rafaeldiasribeiro.com.br/downloads/ED_4.pdf
Acessado em: 21/04/2016