Escola Superior de Tecnologia e Gestão

Instituto Politécnico da Guarda

VETORES – PESQUISA E ORDENAÇÃO

2011-11-10

2011/2012, A1, S1

PAULO NUNES

AV. DR. FRANCISCO SÁ CARNEIRO, 50 - 6301-559 GUARDA

TELF. 271220161, EXT. 161, GAB:20

GPS: LATITUDE: 40.5416236730513, LONGITUDE: -7.28243350982666

VOIP: pnunes@ipg.pt, MSN: pnunes@ipg.pt, SKYPE: pnunes.ipg.pt

EMAIL: Mailto:pnunes@ipg.pt, WEB: http://www.ipg.pt/user/~pnunes/

SUMÁRIO

- Técnicas de resolução de problemas
- Pesquisa
  - Sequencial e Binária
- Algoritmos de ordenação
  - Conceitos sobre algoritmos de ordenação
  - Aplicações
  - Tipos de algoritmos de ordenação
    - Adaptativos e não adaptativos
    - Estável e não estável
    - Interno e Externo
    - Direto e Indireto
- Algoritmos de ordenação de vetores
  - Selection sort, Insertion sort, Bubble sort, Quick sort

TÉCNICAS RESOLUÇÃO PROBLEMAS

1. **Força bruta de pesquisa ou busca exaustiva**
   - "brute-force search or exhaustive search"
   - Também conhecida como gerar e testar, é uma técnica de resolução de problemas trivial mas muito geral, que consiste:
     - Enumerar de forma sistematica todos os candidatos possíveis para a solução e verificar se cada candidato satisfaz declaração do problema.
2. **Dividir para conquistar**
   - Dividir um problema grande e complexo em pequenos problemas que podem se facilmente resolvidos.
   - O probela fica resolvido com a resolução dos problemas mais pequenos.

MÉTODOS DE PESQUISA

- Pesquisa de um elemento num vetor com N elementos.
- Pesquisa sequencial ou linear
  - Vetor ordenado ou não.
    - Comparar o elemento a procurar com cada um dos elementos do vetor, começando no início do vetor até ao final.
    - O número de comparações varia entre 1 e N
  - Técnica: Força bruta de pesquisa
- Pesquisa binária
  - Requer que o vetor esteja ordenado.
  - O número de comparações varia entre 1 e $\log_2(N)$
  - Técnica: Dividir para conquistar

ESPAÇO MEMÓRIA (N=1E4)

VARIÁVEIS DE ENTRADA

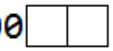
e (Inteiro T2) (>= 0, <= 20)

N (Inteiro T2) (>= 1)

V (Inteiro T2) [10000] (>= 0, <= 20)

1 
2 
3 
4 
5 
6 
7 
8 

..

10000 

VARIÁVEIS DE SAÍDA

existe (Texto T3) (∈ {"Sim","Não"})

PESQUISA SEQUENCIAL 1/3


```
Algoritmo: PerquisaSequencial
Objetivo:
Permite verificar a existência de um dado elemento num
vetor de números [0,20], sem necessidade de estar ordenado.
Percorre todo o vetor.
Variáveis
  Entrada:
    e (Inteiro T2) - Elemento a pesquisar (Uni0) (>= 0, <= 20)
    N (Inteiro T2) - Dimensão do vetor (>= 1)
    V [N] (Inteiro T2) - Vetor (>= 0, <= 20)
  Auxiliares:
    iL ( T2) - Desc0 (Uni0) (>= Li0, <= Ls0)
  Saída:
    existe (Texto T3) - O elemento existe (∈ {"Sim","Não"})
Data: 2011-11-5 20:58:54
Autor: Paulo Nunes
Versão: 1.1
Obs:
```

PESQUISA SEQUENCIAL 2/3


```
Início:
    /* Entrada de dados (INPUT) */
    FAZER
      ESCREVER "Elemento a pesquisar (Uni0)?"
      LER e
    ATÉ ( (e >= 0) E (e <= 20) )
    FAZER
      ESCREVER "Dimensão do vetor?"
      LER N
    ATÉ (N >= 1)
    PARA iL=1 ATÉ N FAZER
      FAZER
        ESCREVER "Vetor ", "[", iL, "]", " ?"
        LER V[iL]
      ATÉ ( (V[iL] >= 0) E (V[iL] <= 20) )
    FIMPARA /* iL */
```

PESQUISA SEQUENCIAL 3/3


```
    /* Processamento (PROCESSING) */
    existe = "Não"
    PARA iL=1 ATÉ N FAZER
      SE (e = V[iL]) ENTÃO
         existe = "Sim"
      FIMSE
    FIMPARA
    /* Saída de resultados (OUTPUT) */
    ESCREVER "O elemento existe: ", existe
Fim.
```

PESQUISA SEQUENCIAL: V2 1/3


```
Algoritmo: PerquisaSequencialMelhorada
Objetivo:
Permite verificar a existência de um dado elemento num
vetor de números [0,20], sem necessidade de estar ordenado.
Quando encontra o elemento pára a pesquisa.
Variáveis
  Entrada:
    e (Inteiro T2) - Elemento a pesquisar (Uni0) (>= 0, <= 20)
    N (Inteiro T2) - Dimensão do vetor (>= 1)
    V [N] (Inteiro T2) - Vetor (>= 0, <= 20)
  Auxiliares:
    iL ( T2) - Desc0 (Uni0) (>= Li0, <= Ls0)
  Saída:
    existe (Texto T3) - O elemento existe (∈ {"Sim","Não"})
Data: 2011-11-5 20:58:54
Autor: Paulo Nunes
Versão: 1.1
Obs:
```

PESQUISA SEQUENCIAL: V2 2/3


```
Início:
    /* Entrada de dados (INPUT) */
    FAZER
      ESCREVER "Elemento a pesquisar (Uni0)?"
      LER e
    ATÉ ( (e >= 0) E (e <= 20) )
    FAZER
      ESCREVER "Dimensão do vetor?"
      LER N
    ATÉ (N >= 1)
    PARA iL=1 ATÉ N FAZER
      FAZER
        ESCREVER "Vetor ", "[", iL, "]", " ?"
        LER V[iL]
      ATÉ ( (V[iL] >= 0) E (V[iL] <= 20) )
    FIMPARA /* iL */
```

PESQUISA SEQUENCIAL: V2 3/3


```
    /* Processamento (PROCESSING) */
    iL = 1
    ENQUANTO ((iL ≤ N) E (e ≠ V[iL]))
      iL ← iL + 1
    FIMENQUANTO
    SE (iL ≤ N) ENTÃO
       existe = "Sim"
    SENÃO
       existe = "Não"
    FIMSE
    /* Saída de resultados (OUTPUT) */
    ESCREVER "O elemento existe: ", existe
Fim.
```

PESQUISA SEQUENCIAL: V3 1/3


```
Algoritmo: PerquisaSequencialVetorOrdenado
Objetivo:
Permite verificar a existência de um dado elemento num
vetor de números [0,20], sem necessidade de estar ordenado.
Quando encontra o elemento ou enconta um elemento maior a pesquisa pára.
Variáveis
  Entrada:
    e (Inteiro T2) - Elemento a pesquisar (Uni0) (>= 0, <= 20)
    N (Inteiro T2) - Dimensão do vetor (>= 1)
    V [N] (Inteiro T2) - Vetor (>= 0, <= 20)
  Auxiliares:
    iL ( T2) - Desc0 (Uni0) (>= Li0, <= Ls0)
  Saída:
    existe (Texto T3) - O elemento existe (∈ {"Sim","Não"})
Data: 2011-11-5 20:58:54
Autor: Paulo Nunes
Versão: 1.0
Obs:
```

PESQUISA SEQUENCIAL: V3 2/3


```
Início:
    /* Entrada de dados (INPUT) */
    FAZER
      ESCREVER "Elemento a pesquisar (Uni0)?"
      LER e
    ATÉ ( (e >= 0) E (e <= 20) )
    FAZER
      ESCREVER "Dimensão do vetor?"
      LER N
    ATÉ (N >= 1)
    PARA iL=1 ATÉ N FAZER
      FAZER
        ESCREVER "Vetor ", "[", iL, "]", " ?"
        LER V[iL]
      ATÉ ( (V[iL] >= 0) E (V[iL] <= 20) )
    FIMPARA /* iL */
```

PESQUISA SEQUENCIAL: V3 3/3


```
  /* Processamento (PROCESSING) */
  iL = 1
  ENQUANTO ((iL ≤ N) E (e < V[iL]))   /* sai quanto encontra o elemento */
    iL ← iL + 1                       /* ou um elemento maior */
  FIMENQUANTO                         /* ou termina o vetor */
  SE ((iL ≤ N) E (e = V[iL])) ENTÃO
    existe = "Sim"
  SENÃO
     existe = "Não"
  FIMSE
  /* Saída de resultados (OUTPUT) */
  ESCREVER "O elemento existe: ", existe
Fim.
```

ANÁLISE PESQUISA SEQUENCIAL

Pesquisa Sequencial			
	Caso		
N	Pior	Média (N)	Melhor
100	100	100	1
1.000	1.000	1.000	1
10.000	10.000	10.000	1
100.000	100.000	100.000	1
1.000.000	1.000.000	1.000.000	1
1.000.000.000	1.000.000.000	1.000.000.000	1
2.000.000.000	2.000.000.000	2.000.000.000	1

Pesquisa Sequencial Melhorada			
	Caso		
N	Pior	Média (N+1)/2	Melhor
100	100	50,5	1
1.000	1.000	500,5	1
10.000	10.000	5.000,5	1
100.000	100.000	50.000,5	1
1.000.000	1.000.000	500.000,5	1
1.000.000.000	1.000.000.000	500.000.001	1
2.000.000.000	2.000.000.000	1.000.000.001	1

PESQUISA BINÁRIA

- Vetor ordenado.
- Realiza sucessivas divisões do espaço de busca (divisão e conquista) comparando o elemento buscado (chave) com o elemento no meio do vetor.
- Se o elemento do meio do vetor for à chave, a busca termina com sucesso.
  - Caso contrário, se o elemento do meio vier antes do elemento buscado, então a busca continua na metade posterior do vetor.
  - E finalmente, se o elemento do meio vier depois da chave, a busca continua na metade anterior do vetor.

PESQUISA BINÁRIA 1/3


```
Algoritmo: PerquisaBinaria
Objetivo:
Permite verificar a existência de um dado elemento num
vetor de números [0,20].
O vetor deve estar ordenado.
Variáveis
  Entrada:
    e (Inteiro T2) - Elemento a pesquisar (Uni0) (>= 0, <= 20)
    N (Inteiro T2) - Dimensão do vetor (>= 1)
    V [N] (Inteiro T2) - Vetor (>= 0, <= 20)
  Auxiliares:
    meio (Inteiro T6) - Representam o meio vetor (>= 1, <= N)
    inicio (Inteiro T6) - Representam o inicio o vetor (>= 1, <= N)
    fim (Inteiro T6) - Representam o fimo vetor (>= 1, <= N)
  Saída:
    existe (Texto T3) - O elemento existe (∈ {"Sim","Não"})
Data: 2011-11-5 20:58:54
Autor: Paulo Nunes
Versão: 1.2
Obs:
```

PESQUISA BINÁRIA 2/3


```
Início:
    /* Entrada de dados (INPUT) */
    FAZER
      ESCREVER "Elemento a pesquisar (Uni0)?"
      LER e
    ATÉ ( (e >= 0) E (e <= 20) )
    FAZER
      ESCREVER "Dimensão do vetor?"
      LER N
    ATÉ (N >= 1)
    PARA iL=1 ATÉ N FAZER
      FAZER
        ESCREVER "Vetor ", "[", iL, "]", " ?"
        LER V[iL]
      ATÉ ( (V[iL] >= 0) E (V[iL] <= 20) )
    FIMPARA /* iL */
```

PESQUISA BINÁRIA 3/3


```
  /* Processamento (PROCESSING) */
  fim ← N                          /* O valor do último índice do vetor   */
  inicio ← 1                       /* O valor do primeiro índice do vetor */
  existe ← "Não"
  FAZER
    meio ← (inicio + fim) div 2
    SE (e = V[meio]) ENTÃO         /* encontrou */
      existe ← "Sim"
      inicio ← fim + 1             /* força a saída */
    FIMSE
    SE (E < V[meio]) ENTÃO         /* descarta a parte direira do vetor */
      fim ← meio - 1
    FIMSE
    SE (e > V[meio]) ENTÃO         /* descarta a parte esquerda do vetor */
      inicio ← meio +1
    FIMSE
  ATÉ (inicio > fim)
  /* Saída de resultados (OUTPUT) */
  ESCREVER "O elemento existe: ", existe
Fim.
```

ANÁLISE

Pesquisa Binária			
N	Caso		
	Pior	Média (1+log2(N)/2	Melhor
100	7	3,8	1
1.000	10	5,5	1
10.000	13	7,1	1
100.000	17	8,8	1
1.000.000	20	10,5	1
1.000.000.000	30	15,4	1
2.000.000.000	31	15,9	1
4.000.000.000	32	16,4	1
8.000.000.000	33	16,9	1
1.000.000.000.000	40	20,4	1
1.000.000.000.000.000	50	25,4	1

SEQUENCIAL - BINÁRIA

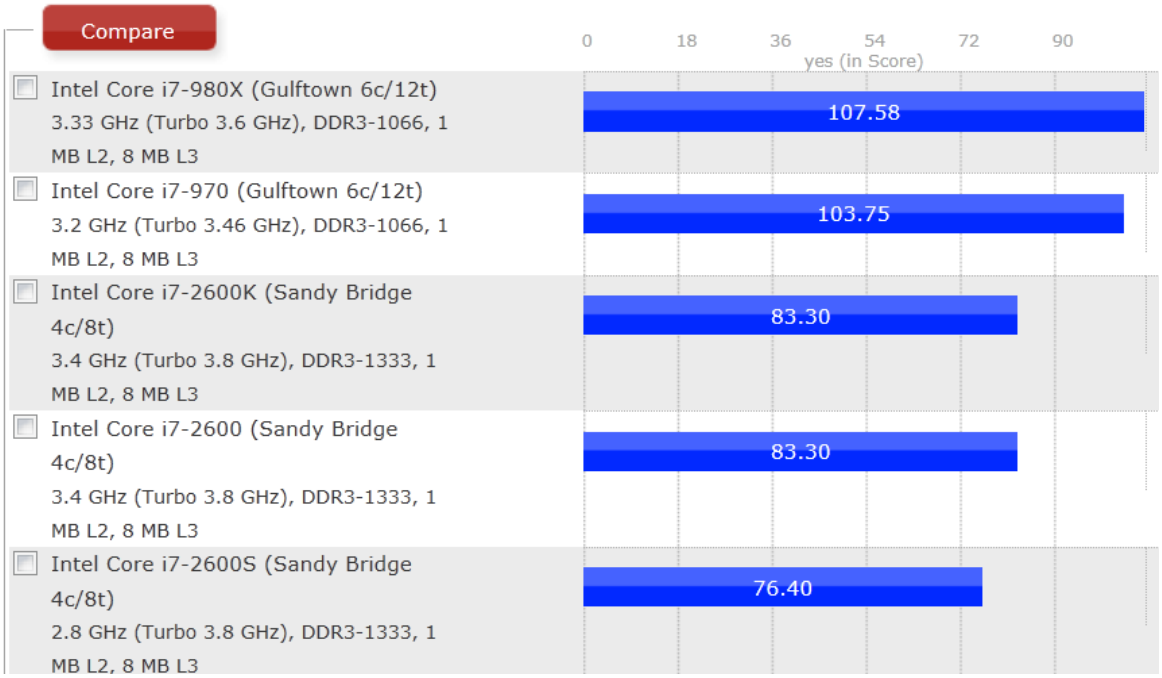
Número de operações de pesquisa				
N	Pesquisa sequencial		Pesquisa Binária	
	Pior	Média	Pior	Média
100	100	51	8	4
1.000	1.000	501	11	5
10.000	10.000	5.001	14	7
100.000	100.000	50.001	18	9
1.000.000	1.000.000	500.001	21	10
1.000.000.000	1.000.000.000	500.000.001	31	15
2.000.000.000	2.000.000.000	1.000.000.001	32	16
4.000.000.000	4.000.000.000	2.000.000.001	33	16
7.000.000.000	7.000.000.000	3.500.000.001	34	17
8.000.000.000	8.000.000.000	4.000.000.001	34	17

SEQUENCIAL - BINÁRIA

Número de vezes mais tempo(Mais rápido=1)		
	Pesquisa sequencial	Pesquisa Binária
N	**Média**	**Média**
100	13	1
1.000	100	1
10.000	714	1
100.000	5.556	1
1.000.000	50.000	1
1.000.000.000	33.333.333	1
2.000.000.000	62.500.000	1
4.000.000.000	125.000.000	1
7.000.000.000	205.882.353	1
8.000.000.000	235.294.118	1

CPU - OPERAÇÕES

http://www.tomshardware.com/charts/desktop-cpu-charts-2010/Raw-Performance-SiSoftware-Sandra-2010-Pro-GFLOPS,2409.html



Escola Superior de Tecnologia e Gestão
Instituto Politécnico da Guarda

ALGORITMOS DE ORDENAÇÃO DE VETORES

CONCEITOS

- Algoritmo de ordenação
  - Algoritmo que permite organizar um conjunto de dados por uma certa ordem efetuando apenas comparações e trocas entre eles.
    - Existem métodos de ordenação que utilizam princípio da **distribuição**.
- Ordem
  - Regra bem definida pela qual os elementos devem ser colocados.

EXEMPLO: CARTAS

- Exemplo de ordenação por distribuição:
  - considere o problema de ordenar um baralho com 52 cartas na ordem:
    1. A < 2 < 3 < … < 10 < J < Q < K
    2. ♣ < ♦ < ♥ < ♠

EXEMPLO: CARTAS: ALGORITMO

1. Distribuir as cartas abertas em treze montes:
2. ases, dois, três, ..., reis.
3. Colete os montes na ordem especificada.
4. Distribua novamente as cartas abertas em quatro montes: paus, ouros, copas e espadas.
5. Colete os montes na ordem especificada.

APLICAÇÕES

- Apresentar listas de dados ordenados pelo item mais conveniente.
- Localizar dados de modo mais eficiente.
  - Gestores de bases de dados.
  - Jogos.
  - Utilizado em milhões de programas/aplicações.

TIPOS DE ORDENS

- Numérica: valor numérico.
- Lexicográfica: ordem pré-definida.

Numérica	Lexicográfica
1	1
2	10
10	102
21	2
35	21
45	250
102	35
250	45

Numérica	Lexicográfica
2009-01-05	2009-01-05
2009-09-12	2009-09-12
2010-05-20	2010-05-20
2011-01-25	2011-01-25
2011-10-02	2011-10-02
2012-06-08	2012-06-08
2013-02-13	2013-02-13
2013-10-21	2013-10-21

Numérica	Lexicográfica
05-01-2009	02-10-2011
12-09-2009	05-01-2009
20-05-2010	08-06-2012
25-01-2011	12-09-2009
02-10-2011	13-02-2013
08-06-2012	20-05-2010
13-02-2013	21-10-2013
21-10-2013	25-01-2011

CHAVES DE ORDENAÇÃO

- Chave
  - item pelo qual os dados estão ordenados.
- Tipos de chave
  - **Simples**, um item.
    - Exemplos: Nome, Nota, Data e Idade.
  - **Compostas**, mais de um item.
    - Exemplos: Nota + Nome, Género + Nome.

EXEMPLO: 1

Registar						Resultados de avaliações						
Aluno	Visitas	Presente	Nome	Presenças	Faltas	C1	C2	C3	C4	C5	C6	C7
1009413	0	☐	Aarão Coelho			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010277	0	☐	Ana Andrade	2	16	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1000506	0	☐	Ana Ferreira			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010248	0	☐	André Faustino	4	14	13.00	0.00	0.00	0.00	0.00	0.00	0.00
1008356	0	☐	André Costa			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010066	0	☐	André Madeira	3	15	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010523	0	☐	André Terras	9	9	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1008921	0	☐	André Gonçalves			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010074	0	☐	André Gomes			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1000631	0	☐	António Luís			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010521	0	☐	António Reis	2	16	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1006440	0	☐	António Fernandes			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1009499	0	☐	António Fortunato			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1009448	0	☐	Belmiro Bernardo			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010391	0	☐	Bruno Gomes			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1009251	0	☐	Bruno Almeida			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1007280	0	☐	Carina Cardoso			0.00	0.00	0.00	0.00	0.00	0.00	0.00

Qual a regra ?

Primeiro nome.

EXEMPLO: 2

Registar						Resultados de avaliações						
Aluno	Visitas	Presente	Nome/No.Ap	Presenças	Faltas	C1	C2	C3	C4	C5	C6	C7
1009413	0	☐	Aarão Emanuel de Jesus Gaspar Coelho			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010277	0	☐	Ana Andrade	2	16	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1000506	0	☐	Ana Luisa Paiva Ferreira			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010248	0	☐	André Antunes Faustino	4	14	13.00	0.00	0.00	0.00	0.00	0.00	0.00
1008356	0	☐	André Correia da Costa			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010066	0	☐	André Daniel Pacheco Madeira	3	15	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010523	0	☐	André Filipe Morgado Terras	9	9	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1008921	0	☐	André Martins Gonçalves			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010074	0	☐	André Monteiro Gomes			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1000631	0	☐	António Fernandes Luís			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010521	0	☐	António Filipe Madeira dos Reis	2	16	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1006440	0	☐	António Jorge de Almeida Fernandes			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1009499	0	☐	António Luís Pina Fortunato			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1009448	0	☐	Belmiro Santos Bernardo			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010391	0	☐	Bruno Andrade Gomes			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1009251	0	☐	Bruno José Oliveira Almeida			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1007280	0	☐	Carina Gomes Cardoso			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1009986	0	☐	Carlos Miguel Boto Figueiredo			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1009449	0	☐	Claudia Morgado	10	8	12.00	0.00	0.00	0.00	0.00	0.00	0.00
1007283	0	☐	Clodomiro Manuel Nobre Gonçalves			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1009543	0	☐	Daniel Almeida Rodrigues			0.00	0.00	0.00	0.00	0.00	0.00	0.00

Qual a regra ?

Nome.

EXEMPLO: 3

Registar						Resultados de avaliações						
Aluno	Visitas	Presente	Nome	Presenças	Faltas	C1	C2	C3	C4	C5	C6	C7
1010198	0	☐	Renato Terras	15	3	17.00	0.00	0.00	0.00	0.00	0.00	0.00
1010395	0	☐	Micael Martins	14	4	12.00	0.00	0.00	0.00	0.00	0.00	0.00
1010253	0	☐	Nuno Santos	13	5	13.00	0.00	0.00	0.00	0.00	0.00	0.00
1010607	0	☐	Eduardo Dias	13	5	19.00	0.00	0.00	0.00	0.00	0.00	0.00
1010194	0	☐	Daniel Carvalhinho	12	6	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010314	0	☐	Miguel Almeida	11	7	17.00	0.00	0.00	0.00	0.00	0.00	0.00
1009449	0	☐	Claudia Morgado	10	8	12.00	0.00	0.00	0.00	0.00	0.00	0.00
1010523	0	☐	André Terras	9	9	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010547	0	☐	Luís Gonçalves	9	9	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010580	0	☐	Edson Varela	8	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010548	0	☐	Tiago Manso	8	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1009686	0	☐	João Delgado	6	12	16.00	0.00	0.00	0.00	0.00	0.00	0.00
1010746	0	☐	Diogo Pascoal	6	12	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010210	0	☐	Elson Pina	5	13	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010248	0	☐	André Faustino	4	14	13.00	0.00	0.00	0.00	0.00	0.00	0.00
1010287	0	☐	Nuno Pinto	4	14	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010193	0	☐	Filipe Monteiro	4	14	13.50	0.00	0.00	0.00	0.00	0.00	0.00
1010111	0	☐	Rui Fernandes	4	14	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010937	0	☐	Davide Alves	4	14	14.00	0.00	0.00	0.00	0.00	0.00	0.00

Qual a regra ?

Presenças

EXEMPLO: 4

Registar						Resultados de avaliações						
Aluno	Visitas	Presente	Nome	Presenças	Faltas	C1	C2	C3	C4	C5	C6	C7
1010607	0	☐	Eduardo Dias	13	5	19.00	0.00	0.00	0.00	0.00	0.00	0.00
1010314	0	☐	Miguel Almeida	11	7	17.00	0.00	0.00	0.00	0.00	0.00	0.00
1010198	0	☐	Renato Terras	15	3	17.00	0.00	0.00	0.00	0.00	0.00	0.00
1009686	0	☐	João Delgado	6	12	16.00	0.00	0.00	0.00	0.00	0.00	0.00
1010937	0	☐	Davide Alves	4	14	14.00	0.00	0.00	0.00	0.00	0.00	0.00
1010193	0	☐	Filipe Monteiro	4	14	13.50	0.00	0.00	0.00	0.00	0.00	0.00
1010253	0	☐	Nuno Santos	13	5	13.00	0.00	0.00	0.00	0.00	0.00	0.00
1010248	0	☐	André Faustino	4	14	13.00	0.00	0.00	0.00	0.00	0.00	0.00
1010395	0	☐	Micael Martins	14	4	12.00	0.00	0.00	0.00	0.00	0.00	0.00
1010766	0	☐	Ivo Costa	1	17	12.00	0.00	0.00	0.00	0.00	0.00	0.00
1009449	0	☐	Claudia Morgado	10	8	12.00	0.00	0.00	0.00	0.00	0.00	0.00
1006050	0	☐	Patricia Marques			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1008637	0	☐	João Figueiredo			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1009088	0	☐	Vitor Pereira			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1009413	0	☐	Aarão Coelho			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1009687	0	☐	João Salvador			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1009990	0	☐	Fábio Monteiro			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010580	0	☐	Edson Varela	8	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010275	0	☐	Renato Concha	3	15	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1007288	0	☐	Frederico Santos			0.00	0.00	0.00	0.00	0.00	0.00	0.00

Qual a regra ?

C1

EXEMPLO: 4

Registar						Resultados de avaliações						
Aluno	Visitas	Presente	Nome/No.Ap	Presenças	Faltas	C1	C2	C3	C4	C5	C6	C7
1010607	0	☐	Eduardo Dias	13	5	19.00	0.00	0.00	0.00	0.00	0.00	0.00
1010314	0	☐	Miguel Almeida	11	7	17.00	0.00	0.00	0.00	0.00	0.00	0.00
1010198	0	☐	Renato Terras	15	3	17.00	0.00	0.00	0.00	0.00	0.00	0.00
1009686	0	☐	João Delgado	6	12	16.00	0.00	0.00	0.00	0.00	0.00	0.00
1010937	0	☐	Davide Alves	4	14	14.00	0.00	0.00	0.00	0.00	0.00	0.00
1010193	0	☐	Filipe Monteiro	4	14	13.50	0.00	0.00	0.00	0.00	0.00	0.00
1010248	0	☐	André Faustino	4	14	13.00	0.00	0.00	0.00	0.00	0.00	0.00
1010253	0	☐	Nuno Santos	13	5	13.00	0.00	0.00	0.00	0.00	0.00	0.00
1009449	0	☐	Claudia Morgado	10	8	12.00	0.00	0.00	0.00	0.00	0.00	0.00
1010766	0	☐	Ivo Costa	1	17	12.00	0.00	0.00	0.00	0.00	0.00	0.00
1010395	0	☐	Micael Martins	14	4	12.00	0.00	0.00	0.00	0.00	0.00	0.00
1009413	0	☐	Aarão Coelho			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010277	0	☐	Ana Andrade	2	16	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1000506	0	☐	Ana Ferreira			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1008356	0	☐	André Costa			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010066	0	☐	André Madeira	3	15	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010523	0	☐	André Terras	9	9	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1008921	0	☐	André Gonçalves			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010074	0	☐	André Gomes			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1000631	0	☐	António Luis			0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010521	0	☐	António Reis	2	16	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1006440	0	☐	António Fernandes			0.00	0.00	0.00	0.00	0.00	0.00	0.00

Qual a regra ?

C1 + Primeiro Nome

EXEMPLO: 4

Registar

Aluno	Visitas	Presente	Nome/No.Ap	Presenças	Faltas	Resultados de avaliações C1	C2	C3	C4	C5	C6	C7
1010546	0	☐	Henrique Carvalho	1	17	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3334444	0	☐	Hristiyan Stefanov	1	17	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010766	0	☐	Ivo Costa	1	17	12.00	0.00	0.00	0.00	0.00	0.00	0.00
Nelson	0	☐	Nelson Sousa	1	17	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010747	0	☐	Rui Martins	1	17	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010246	0	☐	Silvia Rocha	1	17	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010277	0	☐	Ana Andrade	2	16	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010521	0	☐	António Reis	2	16	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010274	0	☐	David Concha	2	16	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010066	0	☐	André Madeira	3	15	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1009254	0	☐	Hugo Quina	3	15	0.00	0.00	0.00	0.00	0.00	0.00	0.00
JoseBarrio	0	☐	José Barrio	3	15	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010282	0	☐	Nelson Sousa	3	15	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1008043	0	☐	Nuno Galinho	3	15	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010275	0	☐	Renato Concha	3	15	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010810	0	☐	S. Salomão	3	15	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010248	0	☐	André Faustino	4	14	13.00	0.00	0.00	0.00	0.00	0.00	0.00
1010937	0	☐	Davide Alves	4	14	14.00	0.00	0.00	0.00	0.00	0.00	0.00
1010193	0	☐	Filipe Monteiro	4	14	13.50	0.00	0.00	0.00	0.00	0.00	0.00
1010287	0	☐	Nuno Pinto	4	14	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010111	0	☐	Rui Fernandes	4	14	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1010210	0	☐	Elson Pina	5	13	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Qual a regra ?

Presenças + Primeiro nome.

MÉTODOS ORDENAÇÃO VETORES

- Métodos simples
  - Selecção - *Selection sort*
  - Inserção - *Insertion sort*
  - Bolha - *Bubble sort*
  - Comb sort
- Métodos complexos
  - Quick sort, Merge sort, Heapsort,Shell sort, Radix sort, Gnome sort, Count sort, Bucket sort, Cocktail sort,Timsort

SELEÇÃO: DOC


```
Algoritmo: OrdenaSelecao
Objetivo:
Permite ordenar um vetor de números [0,20]
por seleção (Selection sort)
Algoritmo:
- procurar menor elemento e trocar com o elemento na 1º posição
- procurar 2ª menor elemento e trocar com o elemento na 2ª posição
- proceder assim até ordenação estar completa
Variáveis
  Entrada:
    N (Inteiro T6) - Dimensão do vetor (>= 1, <= 999999)
    V [N] (Inteiro T6) - Vetor (>= 0, <= 20)
  Auxiliares:
    posicao_menor (Inteiro T6) - Posição do menor elemento (>= 1, <= N)
    iL (Inteiro T6) - Índice vector (>= 1, <= N)
    j (Inteiro T6) - Índice vector (>= 1, <= N)
    auxiliar (Inteiro T2) - Guarda elemento do vetor (>= 0, <= 20)
  Saída:
    V [N] (Inteiro T2) - Vetor ordenado (>= 0, <= 20)
Data: 2011-11-5 20:58:54
Autor: Paulo Nunes
Versão: 1.2
Obs: A saída é mesmo vetor de entrada. Este é uma permutação ou reordenação do
vetor de entrada.
```

ORD: SELEÇÃO: ENTRADA


```
Início:
    /* Entrada de dados (INPUT) */
    FAZER
      ESCREVER "Dimensão do vetor?"
      LER N
    ATÉ ( (N >= 1) E (N <= 999999) )
    PARA iL=1 ATÉ N FAZER
      FAZER
        ESCREVER "Vetor ", "[", iL, "]", " ?"
        LER V[iL]
      ATÉ ( (V[iL] >= 0) E (V[iL] <= 20) )
    FIMPARA /* iL */
```

SELEÇÃO: ORDENAÇÃO

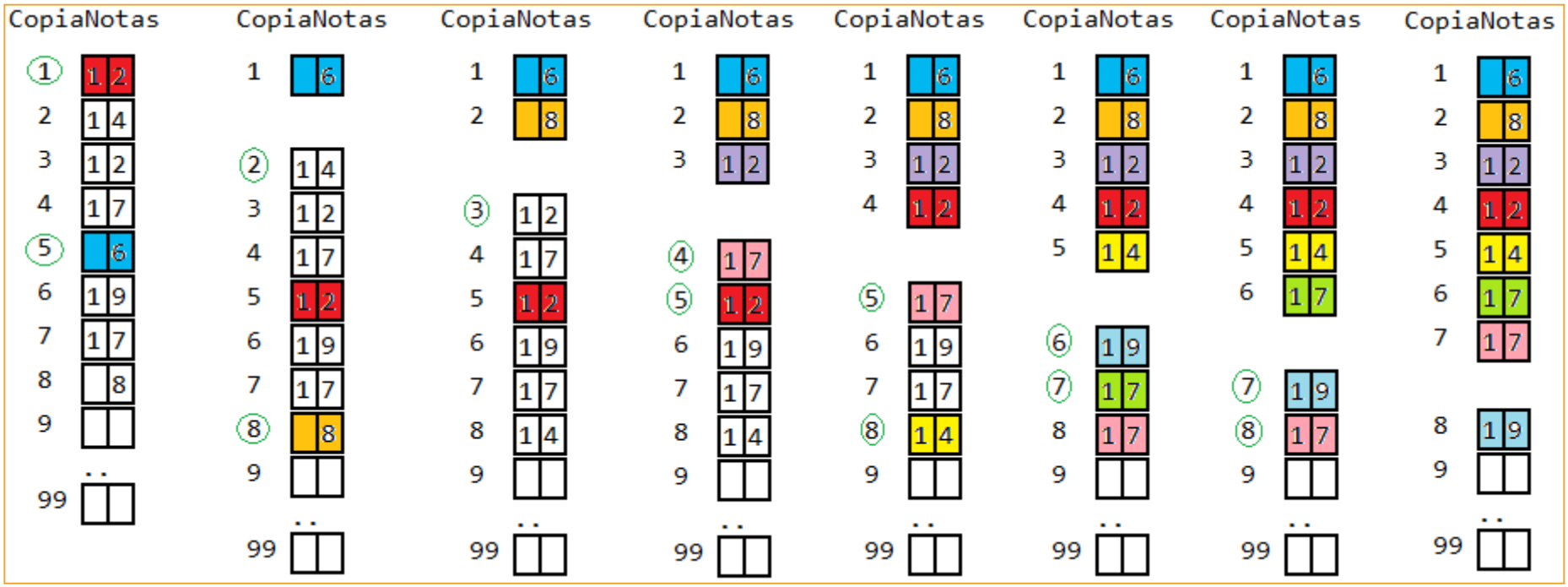

```
 /* Processamento (PROCESSING) */
PARA iL=1 ATÉ N - 1 FAZER
  posicao_menor = i
  PARA j=iL + 1 ATÉ N FAZER
    SE V[j] < V[posicao_menor] ENTÃO
      posicao_menor ← j
    FIMSE
  FIMPARA
  auxiliar ← V[iL]
  V[iL] ← V[posicao_menor]
  V[posicao_menor] ← auxiliar
FIMPARA
```

VLink1 VLink2

SELEÇÃO: SAÍDA


```
    /* Saída de resultados (OUTPUT) */
    ESCREVER "Vetor ordenado:"
    PARA iL=1 ATÉ N FAZER
      ESCREVER V[iL] /* Muda de linha */
    FIMPARA /* iL */
Fim.
```

EXEMPLO


CopiaNotas
CopiaNotas
CopiaNotas
CopiaNotas
CopiaNotas
CopiaNotas
CopiaNotas
CopiaNotas

ORDENAÇÃO POR INSERÇÃO

- **Método preferido dos jogadores de cartas.**
  1. Começamos com uma mão esquerda vazia e as cartas de face para baixo sobre a mesa.
  2. Em seguida, retire um carta da mesa, e insira-o na posição correta na mão esquerda.
     - Para encontrar a posição correta para a carta, tem que compará-lo com cada uma das cartas já na mão esquerda.
     - Note que em todos os momentos, as cartas na mão esquerda estão ordenadas, e estas cartas foram originalmente as cartas do topo da pilha das cartas na mesa.
- Mão esquerda-vetor ordenado
- Mão direita-vetor ainda desordenado

INSERÇÃO: DOC


```
Algoritmo: OrdenaInsercao
Objetivo:
Permite ordenar um vetor de números [0,20]
por inserção(Insertion sort)
Algoritmo:
Método preferido dos jogadores de cartas.
1.Começamos com uma mão esquerda vazia e as cartas de face para baixo sobre a mesa.
2.Em seguida, retire um carta da mesa, e insira-o na posição correta na mão esquerda.
  - Para encontrar a posição correta para a carta, tem que compará-lo com cada uma
    das cartas já na mão esquerda.
  - Note que em todos os momentos, as cartas na mão esquerda estão ordenadas,
    e estas cartas foram originalmente as cartas do topo da pilha das cartas na mesa.
- Mão esquerda-vetor ordenado
- Mão direita-vetor ainda desordenado
Variáveis
  Entrada:
    N (Inteiro T6) - Dimensão do vetor (>= 1, <= 999999)
    V [N] (Inteiro T6) - Vetor (>= 0, <= 20)
  Auxiliares:
    iL (Inteiro T6) - Índice vector (>= 1, <= N)
    j (Inteiro T6) - Índice vector (>= 1, <= N)
    auxiliar (Inteiro T2) - Guarda elemento do vetor (>= 0, <= 20)
  Saída:
    V [N] (Inteiro T2) - Vetor ordenado (>= 0, <= 20)
Data: 2011-11-6 11:58:54
Autor: Paulo Nunes
Versão: 1.2
Obs: A saída é mesmo vetor de entrada. Este é uma permutação ou reordenação do vetor de
entrada.
```

INSERÇÃO: ORDENAÇÃO


```
 /* Processamento (PROCESSING) */
PARA iL=2 ATÉ N FAZER
  PARA j=iL ATÉ 2 PASSO -1 FAZER
    SE (V[j-1] > V[j] ENTÃO
      auxiliar ← V[iL]
      V[iL] ← V[j]
      V[j] ← auxiliar
    FIMSE
  FIMPARA
FIMPARA
```

VLink1 VLink2

INSERÇÃO: COMENTÁRIOS

- Faz muitas **trocas.**
- **Comparar e trocar** não é o melhor processo de mover vários dados uma posição para a direita.

INSERÇÃO MELHORADO

- Reduzir as trocas.
- Para encontrar a posição correta para a carta, tem que compará-lo com cada uma das cartas já na mão esquerda até encontrar uma menor.

- Podemos sair do ciclo interno se (V[j-1] < V[j]) é verdadeira.
  - Remover instruções irrelevantes
    - ***Comparar e trocar*** não é o melhor processo de mover vários dados uma posição para a direita.
    - **Solução**:
      - Fazer uma cópia o elemento mais à direita.
      - Mover os elementos mais à esquerda uma posição para a direita até encontrar a posição de inserção.
      - Colocar lá o elemento previamente copiado.
  - Esta modificação torna o algoritmo adaptativo.

INSERÇÃO MELHORADO: DOC


```
Algoritmo: OrdenaInsercaoMelhorado
Objetivo:
Permite ordenar um vetor de números [0,20]
por inserção(Insertion sort)
Algoritmo:
Método preferido dos jogadores de cartas.
1.Começamos com uma mão esquerda vazia e as cartas de face para baixo sobre a mesa.
2.Em seguida, retire um carta da mesa, e insira-o na posição correta na mão esquerda.
  - Para encontrar a posição correta para a carta, tem que compará-lo com cada uma
    das cartas já na mão esquerda até encontrar uma menor.
    - Podemos sair do ciclo interno se  (V[j-1] < V[j]) é verdadeira.
    - Remover instruções irrelevantes
      - Comparar e trocar não é o melhor processo de mover vários dados
        uma posição para a direita.
    - Solução:
      - Fazer uma cópia o elemento mais à direita.
      - Mover os elementos mais à esquerda uma posição para a direita
        até encontrar a posição de inserção.
      -Colocar lá o elemento previamente copiado.
  - Note que em todos os momentos, as cartas na mão esquerda estão ordenadas,
    e estas cartas foram originalmente as cartas do topo da pilha das cartas na mesa.
- Mão esquerda-vetor ordenado
- Mão direita-vetor ainda desordenado
Variáveis
  Entrada:
    N (Inteiro T6) - Dimensão do vetor (>= 1, <= 999999)
    V [N] (Inteiro T6) - Vetor (>= 0, <= 20)
  Auxiliares:
    iL (Inteiro T6) - Índice vector (>= 1, <= N)
    j (Inteiro T6) - Índice vector (>= 1, <= N)
    auxiliar (Inteiro T2) - Guarda elemento do vetor (>= 0, <= 20)
  Saída:
    V [N] (Inteiro T2) - Vetor ordenado (>= 0, <= 20)
Data: 2011-11-6 15:35
Autor: Paulo Nunes
Versão: 1.1
Obs: A saída é mesmo vetor de entrada. Este é uma permutação ou reordenação do vetor de entrada.
```

INSERÇÃO MELHORADO: ORDENAÇÃO


```
 /* Processamento (PROCESSING) */
PARA j=2 ATÉ N FAZER
  auxiliar ← V[j]
  iL = j - 1
  ENQUANTO ((iL > 0) E (V[iL] > auxiliar)) FAZER
    V[iL + 1] ← V[iL]
    iL ← iL - 1
  FIMENQUANTO
  v[iL + 1] ← auxiliar
FIMPARA
```

VLink1 VLink2

BUBBLESORT: DOC


```
Algoritmo: OrdenaBolha
Objetivo:
Permite ordenar um vetor de números [0,20]
pelo método da bolha (Bubble sort)
Algoritmo:
 - Fazer múltiplas passagens pelos dados trocando de cada
   vez dois elementos adjacentes que estejam fora de ordem,
   até não haver mais trocas.
Algoritmo básico de ordenação.
Variáveis
  Entrada:
    N (Inteiro T6) - Dimensão do vetor (>= 1, <= 999999)
    V [N] (Inteiro T6) - Vetor (>= 0, <= 20)
  Auxiliares:
    iL (Inteiro T6) - Índice vector (>= 1, <= N)
    j (Inteiro T6) - Índice vector (>= 1, <= N)
    auxiliar (Inteiro T2) - Guarda elemento do vetor (>= 0, <= 20)
  Saída:
    V [N] (Inteiro T2) - Vetor ordenado (>= 0, <= 20)
Data: 2011-11-6 15:56:55
Autor: Paulo Nunes
Versão: 1.2
Obs: A saída é mesmo vetor de entrada. Este é uma permutação ou
reordenação do vetor de entrada.
```

BUBBLESORT: ORDENAÇÃO


```
 /* Processamento (PROCESSING) */
PARA iL=1 ATÉ N FAZER
  PARA j=N ATÉ iL+1 PASSO -1 FAZER
    SE (A[j] < A[j-1]) ENTÃO
      auxiliar ← A[j]
      A[j]     ← A[j-1]
      A[j-1]   ← auxiliar
    FIMSE
  FIMENQUANTO
FIMPARA
```

VLink1 VLink2

QUICKSORT

- É um método de ordenação muito rápido e eficiente, inventado por C.A.R. Hoare em 1960, quando visitou a Universidade de Moscovo como estudante.
- Naquela época, Hoare trabalhou em um projeto de tradução de máquina para o National Physical Laboratory.
- Ele criou o 'Quicksort ao tentar traduzir um dicionário de inglês para russo, ordenando as palavras, tendo como objetivo reduzir o problema original em subproblemas que possam ser resolvidos mais fácil e rapidamente.

http://pt.wikipedia.org/wiki/Quicksort#cite_note-1

QUICKSORT: DOC

- O Quicksort adota a estratégia de divisão e conquista.
- A estratégia consiste em dividir o conjunto de dados em dois conjuntos separados por um único elemento, denominado por pivot:
  - Conjunto esquerda | pivot | Conjunto direita
    - Essa operação é denominada de partição.
    - O pivot encontra-se na sua posição final.
    - Todos os elementos da esquerda são ≤ do que o pivot.
    - Todos os elementos da direita são ≥ do que o pivot.
  - Repetir o processo para os dois conjuntos até que os conjuntos tenham apenas um elemento.

QUICKSORT: DOC


```
Algoritmo: QuickSort(e, d, V)
Objetivo:
Permite ordenar um vetor de números [0,20]
pelo método denominado Quicksort.
Algoritmo:
 - Dado o vetor V[e, d].
   - Dividir em dois sub-vetores: V[e, p-1] e V[p+1, e]
     e um pivot V[p].
 - Recursivamente efetuar os mesmo para os sub-vetores.
 - O valor de p depende dos dados.
Parâmetros
  Entrada:
     e (Inteiro T6) - Posição esquerda do vetor (>= 1, <= 999999)
     d (Inteiro T6) - Posição direita do vetor (>= 1, <= 999999)
     V [N] (Inteiro T6) - Vetor (>= 0, <= 20)
  Saída:
     V [N] (Inteiro T2) - Vetor ordenado (>= 0, <= 20)
Variáveis
  Auxiliares:
     p (Inteiro T6) - Índice da partição (>= 1, <= N)
Data: 2011-11-06 17:47
Autor: Paulo Nunes
Versão: 1.1
Obs: A saída é mesmo vetor de entrada. Este é uma permutação ou reordenação do vetor de entrada.
```

VLink1 VLink2

QUICKSORT: ORDENAÇÃO


```
Início:
   /* Processamento (PROCESSING) */
   SE (e < d) ENTÃO
      p = ParticaoQuickSort(V, e, d)
      Quicksort(V, e, p-1)
      Quicksort(V, p+1, d)
   FIMSE
Fim.
```

VLink1 VLink2

QUICKSORT: DOC


```
Algoritmo: ParticaoQuickSort(e, d, V)
Objetivo:
Permite dividir um vetor V[e, d] em dois sub-vetores:
 - V[e, p-1] e V[p+1, e]
 - e um pivot V[p].
 - Todos os elementos do sub-vetor V[e, p-1] são ≤ do que o pivot.
 - Todos os elementos da sub-vetor V[p+1, e] são ≥ do que o pivot.
Valor de retorno:
     p (Inteiro T6) - Posição do pivot (>= 1, <= 999999)
Parâmetros
  Entrada:
     e (Inteiro T6) - Posição esquerda do vetor (>= 1, <= 999999)
     d (Inteiro T6) - Posição direita do vetor (>= 1, <= 999999)
     V [N] (Inteiro T6) - Vetor (>= 0, <= 20)
  Saída:
     V [N] (Inteiro T2) - Vetor ordenado (>= 0, <= 20)
Variáveis
  Auxiliares:
     i (Inteiro T6) - Índice do vetor - encontra elementos > pivot (>= 1, <= d)
     j ( T2) - Índice do vetor - encontra elementos < pivot (>= 1, <= d)
     pivot ( T2) - Elemento pivot (>= 0, <= 20)
     aux ( T2) - Usada para troca de elementos (>= 0, <= 20)
Data: 2011-11-06 18:41
Autor: Paulo Nunes
Versão: 1.0
Obs: A saída é mesmo vetor de entrada. Este é uma permutação ou reordenação do vetor de entrada.
```

VLink1 VLink2

QUICKSORT: ALG


```
Início:
    /* Processamento (PROCESSING) */
    i ← e
    j ← d
    pivot ← V[(e + d) div 2]          /* Escolhe o elemento central como pivot */
    ENQUANTO (i <= j)                 /* para quando se cruzam */
      ENQUANTO (V[i] < pivot) FAZER
        i ← i + 1
      FIMENQUANTO
      ENQUANTO (V[J] > pivot) FAZER
        j ← j - 1
      FIMENQUANTO
      SE (i <= j) ENTÃO               /* Troca i com j */
        aux  ← V[i]
        V[i] ← V[j]
        V[j] ← aux
        i    ← i + 1
        j    ← j - 1
      FIMSE
    FIMENQUANTO
    p ← i-1                           /* ou j+1 */
    RETORNA p
Fim.
```

VLink1 VLink2

SHELLSORT

- Insertion sort:
  - Apenas envolve trocas entre itens adjacentes.
  - Lento se o menor item está no final do vetor, serão necessários N passos para o colocar na posição correta
- Shell sort
  - Permitir trocas entre elementos que estão afastados. Acelera o processo de deslocamento dos elementos para a sua posição final.

SHELLSORT

- Efetuar comparações entre elementos com distâncias (h).
  - No início começar com distâncias (h) da ordem de grandeza de N.
    - Usando valores de h grandes é possível mover elementos no vetor a grandes distâncias o que torna mais fácil h-ordenar mais tarde com h pequenos.
  - Em cada passo os dados a distâncias h estão ordenados.
    - Diz-se que estão h-ordenados.
    - É equivalente a h sequências ordenadas entrelaçadas.
- Usando este procedimento para qualquer sequência de h's que termine em 1 vão produzir um vetor ordenado.
- Cada passo torna o próximo mais simples, porque:
  - desloca rapidamente elementos pequenos da direita para a esquerda e vice-versa.

COMPARAÇÃO

Name	Best	Average	Worst	Memory	Stable	Method
Quicksort	$n \log n$	$n \log n$	n^2	$\log n$	Depends	Partitioning
Insertion sort	n	n^2	n^2	1	Yes	Insertion
Selection sort	n^2	n^2	n^2	1	Depends	Selection
Bubble sort	n	n^2	n^2	1	Yes	Exchanging

COMPARAÇÃO - MÉTODOS

Tempo de execução:

- Observação: O método que levou menos tempo real para executar recebeu o valor 1 e os outros receberam valores relativos a ele.
- Registros na ordem aleatória:

	5.00	5.000	10.000	30.000
Inserção	11,3	87	161	–
Seleção	16,2	124	228	–
Shellsort	1,2	1,6	1,7	2
Quicksort	1	1	1	1
Heapsort	1,5	1,6	1,6	1,6

COMPARAÇÃO

Name	Best	Average	Worst	Memory	Stable	Method	Other notes
Quicksort	$n \log n$	$n \log n$	n^2	$\log n$	Depends	Partitioning	Quicksort can be done in place with O(log(*n*)) stack space, but the sort is unstable[citation needed]. Naïve variants use an O(*n*) space array to store the partition. An O(*n*) space implementation can be stable[citation needed].
Merge sort	$n \log n$	$n \log n$	$n \log n$	Depends	Yes	Merging	Used to sort this table in Firefox [2].
In-place Merge sort	—	—	$n (\log n)^2$	1	Yes	Merging	Implemented in Standard Template Library (STL): [3]; can be implemented as a stable sort based on stable in-place merging: [4]
Heapsort	$n \log n$	$n \log n$	$n \log n$	1	No	Selection	
Insertion sort	n	n^2	n^2	1	Yes	Insertion	Average case is also $\mathcal{O}(n + d)$, where *d* is the number of inversions
Introsort	$n \log n$	$n \log n$	$n \log n$	$\log n$	No	Partitioning & Selection	Used in SGI STL implementations
Selection sort	n^2	n^2	n^2	1	Depends [5].	Selection	Its stability depends on the implementation. Used to sort this table in Safari or other Webkit web browser [6].
Timsort	n	$n \log n$	$n \log n$	n	Yes	Insertion & Merging	n comparisons when the data is already sorted or reverse sorted.
Shell sort	n	$n(\log n)^2$ *or* $n^{3/2}$	depends on gap sequence. Best known: $O(n \log^2 n)$	1	No	Insertion	
Bubble sort	n	n^2	n^2	1	Yes	Exchanging	Tiny code size
Binary tree sort	n	$n \log n$	$n \log n$	n	Yes	Insertion	When using a self-balancing binary search tree
Cycle sort	—	n^2	n^2	1	No	Insertion	In-place with theoretically optimal number of writes
Library sort	—	$n \log n$	n^2	n	Yes	Insertion	
Patience sorting	—	—	$n \log n$	n	No	Insertion & Selection	Finds all the longest increasing subsequences within O(*n* log *n*)
Smoothsort	n	$n \log n$	$n \log n$	1	No	Selection	An adaptive sort - n comparisons when the data is already sorted, and 0 swaps.
Strand sort	n	n^2	n^2	n	Yes	Selection	
Tournament sort	—	$n \log n$	$n \log n$			Selection	
Cocktail sort	n	n^2	n^2	1	Yes	Exchanging	
Comb sort	—	—	n^2	1	No	Exchanging	Small code size
Gnome sort	n	n^2	n^2	1	Yes	Exchanging	Tiny code size
Bogosort	n	$n \cdot n!$	$n \cdot n! \to \infty$	1	No	Luck	Randomly permute the array and check if sorted.

Name	O(average)	O(max)
Bubble sort	$1. \times 10^{12}$	$1. \times 10^{12}$
Selection sort	$1. \times 10^{12}$	$1. \times 10^{12}$
Insertion sort	$1. \times 10^{12}$	$1. \times 10^{12}$
Shell sort	3.97267×10^{8}	3.97267×10^{8}
Binary Tree sort	1.99316×10^{7}	1.99316×10^{7}
Library Sort	1.99316×10^{7}	$1. \times 10^{12}$
Merge Sort	1.99316×10^{7}	1.99316×10^{7}
Heapsort	1.99316×10^{7}	1.99316×10^{7}
Quicksort	1.99316×10^{7}	$1. \times 10^{12}$
Introsort	1.99316×10^{7}	1.99316×10^{7}
Pigeonhole sort	1.00002×10^{6}	1.00002×10^{6}
Bucket sort	$4. \times 10^{6}$	$4. \times 10^{12}$
Counting sort	$1. \times 10^{6}$	$1. \times 10^{6}$
LSD Radix sort	$4. \times 10^{6}$	$4. \times 10^{6}$
MSD Radio sort	$4. \times 10^{6}$	$8. \times 10^{6}$
Spreadsort	$4. \times 10^{6}$	$5. \times 10^{6}$
Simple pancake sort	$1. \times 10^{6}$	$1. \times 10^{6}$
Bead Sort	1000.	$1. \times 10^{6}$
Sorting networks	19.9316	19.9316
Cocktail Sort	$1. \times 10^{12}$	$1. \times 10^{12}$
Comb sort	$1. \times 10^{12}$	$1. \times 10^{12}$
Gnome sort	$1. \times 10^{12}$	$1. \times 10^{12}$
Smoothsort	1.99316×10^{7}	1.99316×10^{7}
Patience Sorting	1.99316×10^{7}	1.99316×10^{7}
Strand sort	1.99316×10^{7}	$1. \times 10^{12}$
Tourament sort	1.99316×10^{7}	1.99316×10^{7}
Han's Sort	4.31698×10^{6}	4.31698×10^{6}
Thorup's Sort	4.31698×10^{6}	4.31698×10^{6}
Fast Interger Sort	2.07774×10^{6}	2.07774×10^{6}
Bogosort	$8.263931688331240 \times 10^{5565\,714}$	∞
Stooge Sort	1.80746×10^{16}	1.80746×10^{16}

ESTÁVEL E NÃO ESTÁVEL

- Um método de ordenação é estável se a ordem relativa dos itens com chaves iguais não se altera durante a ordenação.
  - Alguns dos métodos de ordenação mais eficientes não são estáveis.
    - A estabilidade pode ser forçada quando o método é não-estável.
    - Sedgewick (1988) sugere agregar um pequeno índice a cada chave antes de ordenar, ou então aumentar a chave de alguma outra forma.

Nome	Nota
André Silva	13
André Gomes	13
Carlos Santos	14

Nome	Nota
André Gomes	13
André Silva	13
Carlos Santos	14

ADAPTATIVOS E NÃO ADAPTATIVOS

- Quando a ordem inicial dos dados afeta o número de operações (comparações e trocas) diz-se adaptativo.

Tempo de Execução

- Registros na ordem ascendente:

	500	5.000	10.000	30.000
Inserção	1	1	1	1
Seleção	128	1.524	3.066	–
Shellsort	3,9	6,8	7,3	8,1
Quicksort	4,1	6,3	6,8	7,1
Heapsort	12,2	20,8	22,4	24,6

Tempo de Execução

- Registros na ordem descendente:

	500	5.000	10.000	30.000
Inserção	40,3	305	575	–
Seleção	29,3	221	417	–
Shellsort	1,5	1,5	1,6	1,6
Quicksort	1	1	1	1
Heapsort	2,5	2,7	2,7	2,9

DIRETO E INDIRETO

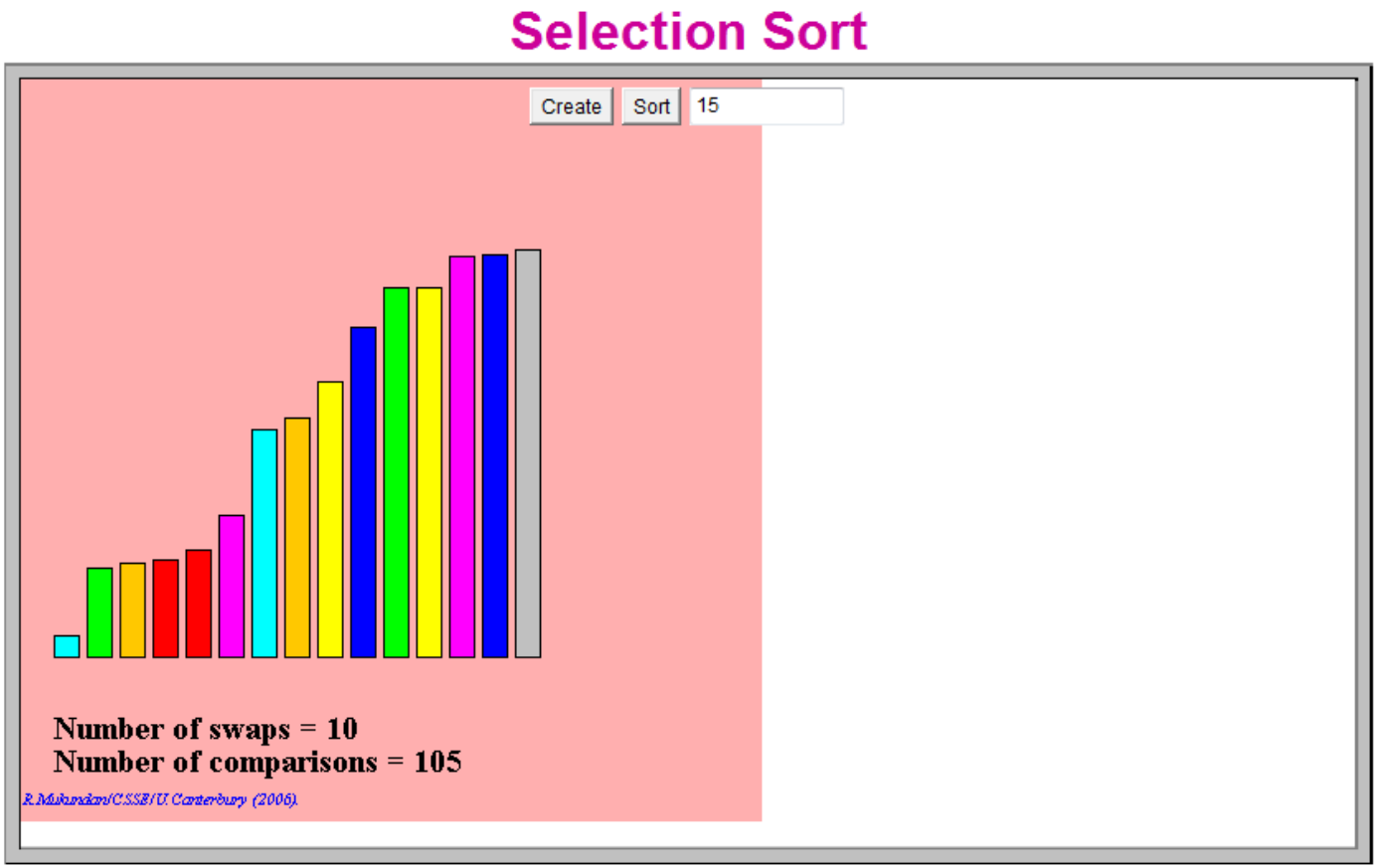
- Um algoritmo de ordenação é dito **direto** se os dados são acedidos diretamente nas operações de comparação e troca; caso contrário é dito **indireto.**

INTERNO E EXTERNO

- Algoritmo de ordenação interno
  - O conjunto de dados a ser ordenado cabe todo na memória principal.
- Ordenação de ordenação externo
  - O conjunto de dados a ser ordenado não cabe todo na memória principal.
    - São colocados na memória principal as chaves de ordenação e a sua localização, número de registo.
    - Procede-se à ordenação das chaves e quando se trocam duas chaves também se trocam os números de registos.
    - Para listar o conjunto de dados externo estes são lidos pela ordem dos números de registos.

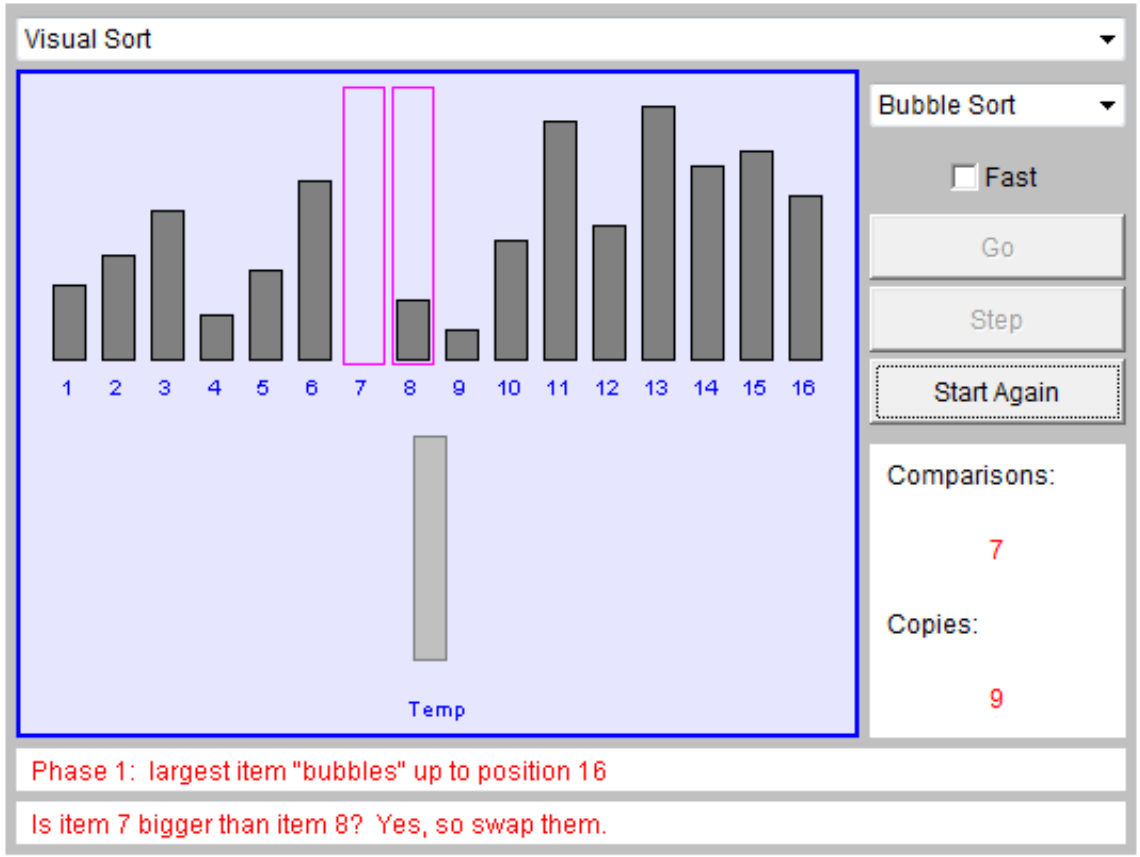
ANIMAÇÃO – BARRAS-FIXO

http://www.cosc.canterbury.ac.nz/mukundan/dsal/BSort.html



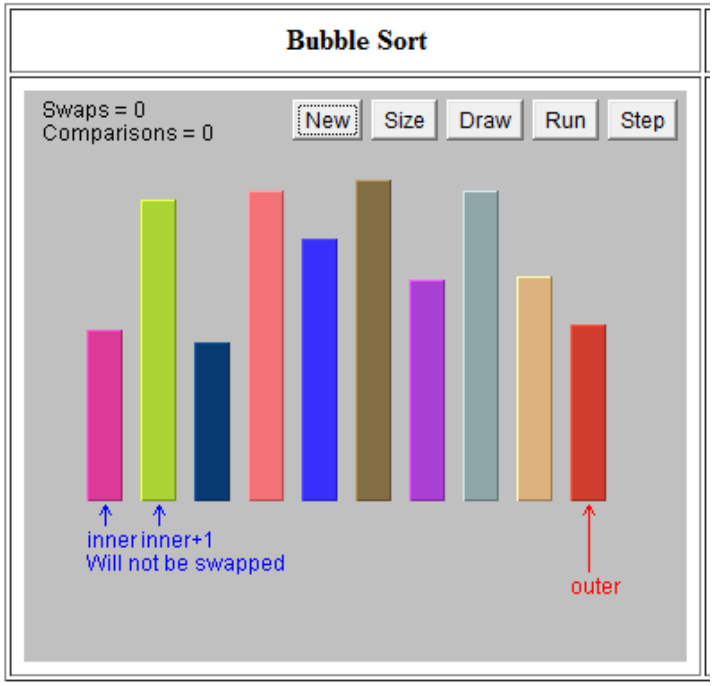
ANIMAÇÃO – BARRAS-FIXO

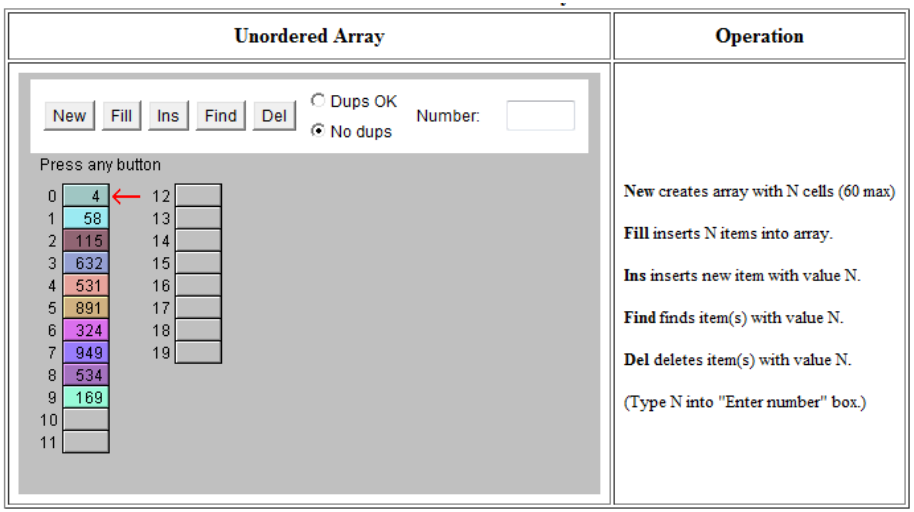
http://math.hws.edu/TMCM/java/xSortLab/



ANIMAÇÃO

- Animação de dezenas de algoritmos e estruturas de dados
  - Em algumas animações permite escolher os dados
    - O livro "Data Structures and Algorithms in Java," Second Edition. Robert Lafore, 2002, vem acompanhado de diversas animações de algoritmos vistos no curso.
      - http://mainline.brynmawr.edu/Courses/cs206/spring2004/lafore.html





BIBLIOGRAFIA

- Knuth, Donald E. (1993). The Art of Computer Programming – VOLUME 3 -Sorting and Searching. Third Edition, Prentice Hall.
- Livro "Projeto de Algoritmos" – Nívio Ziviani
  - http://www2.dcc.ufmg.br/disciplinas/aeds2_turmaA1/cap4.pdf
- Centenas de algoritmos
  - http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Sorting/bubbleSort.htm

LINKS: ORDENAÇÃO VISUAL

- Ordenação:
  - Algoritmos de ordenação: passo a passo - Pode ser usado no browser e permite a execução dos algoritmos passo a passo.
  - Algoritmos de ordenação: execução - Grupo da Unicamp - Programa para Windows - visualização da animação
  - Algoritmos de ordenação: execução - Departamento de Computação da University of British Columbia (UBC) - Vancouver. Animação no browser.
  - Diversas estruturas de Dados e algoritmos, incluindo ordenação (indica o que vai fazer na execução); listas encadeadas, buscas, etc. - Departamento de Computação e Engenharia de Software, University of Canterbury, Nova Zelândia
  - Veja no Google outros sites para animação de algoritmos