

Algoritmos e Estruturas de Dados II

Ordenação simples

Prof. Flávio José M. Coelho

fcoelho@uea.edu.br

Objetivos

1. Entender o Problema da Ordenação
2. Conhecer e entender três algoritmos simples de ordenação.

Ordenação

Problema da Ordenação

Entrada: lista de inteiros $\langle a_1, a_2, \dots, a_n \rangle$.

Saída: Uma permutação (reordenada)
 $\langle a'_1, a'_2, \dots, a'_n \rangle$ da entrada tal que
 $a'_1 \leq a'_2 \leq \dots \leq a'_n$.

Ordenação interna

Lista cabe na memória principal.



Ordenação externa

Lista não cabe na memória principal; usa memória secundária.

Com comparação de chaves

Chaves são comparadas para processar a ordenação.



Sem comparação de chaves

Ordenação digital, radixsort, bucketsort.

Ordenação *in situ* (“*in place*”)

Ordenação ocorre na própria lista.



Ordenação com espaço extra

Cópias extras de partes da lista precisam ser criadas.

Ordenação estável (estabilidade)

Ordem relativa dos itens com chaves iguais não se altera durante a ordenação.



Ordenação não-estável

Itens ordenados e com chaves iguais podem mudar de posição durante a ordenação.

Estabilidade

sorted by time

Chicago	09:00:00
Phoenix	09:00:03
Houston	09:00:13
Chicago	09:00:59
Houston	09:01:10
Chicago	09:03:13
Seattle	09:10:11
Seattle	09:10:25
Phoenix	09:14:25
Chicago	09:19:32
Chicago	09:19:46
Chicago	09:21:05
Seattle	09:22:43
Seattle	09:22:54
Chicago	09:25:52
Chicago	09:35:21
Seattle	09:36:14
Phoenix	09:37:44

sorted by location (not stable)

Chicago	09:25:52
Chicago	09:03:13
Chicago	09:21:05
Chicago	09:19:46
Chicago	09:19:32
Chicago	09:00:00
Chicago	09:35:21
Chicago	09:00:59
Houston	09:01:10
Houston	09:00:13
Phoenix	09:37:44
Phoenix	09:00:03
Phoenix	09:14:25
Seattle	09:10:25
Seattle	09:36:14
Seattle	09:22:43
Seattle	09:10:11
Seattle	09:22:54

*no
longer
sorted
by time*

sorted by location (stable)

Chicago	09:00:00
Chicago	09:00:59
Chicago	09:03:13
Chicago	09:19:32
Chicago	09:19:46
Chicago	09:21:05
Chicago	09:25:52
Chicago	09:35:21
Houston	09:00:13
Houston	09:01:10
Phoenix	09:00:03
Phoenix	09:14:25
Phoenix	09:37:44
Seattle	09:10:11
Seattle	09:10:25
Seattle	09:22:43
Seattle	09:22:54
Seattle	09:36:14

*still
sorted
by time*

Métodos simples

Métodos com eficiência $O(N^2)$ (caso médio).



Métodos eficientes

Métodos com eficiência $O(N \log N)$ (caso médio).



Implementação

Implementação

- ▶ Algoritmos usam uma lista de itens com **chaves** (possivelmente, não únicas).

Implementação

- ▶ Algoritmos usam uma lista de itens com **chaves** (possivelmente, não únicas).
- ▶ Chaves são comparáveis entre si (“redutíveis” a inteiros).

Implementação

- ▶ Algoritmos usam uma lista de itens com **chaves** (possivelmente, não únicas).
- ▶ Chaves são comparáveis entre si (“redutíveis” a inteiros).
- ▶ Para simplificar, usaremos listas de inteiros com implementação estática.

Ordenação simples

1. **Bubblesort** (ordenação bolha).
2. **Selection sort** (ordenação por seleção).
3. **Insertion sort** (ordenação por inserção).

Bubblesort (“bolha”)

Algoritmo

1. Compare chaves adjacentes e permuta-as se estiverem fora de ordem.
2. Repita o passo anterior da penúltima posição até a primeira posição da lista.

Bubblesort

O R D E N A

Bubblesort

	O	R	D	E	N	A
1	O	R	D	E	N	A

Bubblesort

	O	R	D	E	N	A
1	O	R	D	E	N	A
2	O	R	D	E	N	A

Bubblesort

	O	R	D	E	N	A
1	O	R	D	E	N	A
2	O	R	D	E	N	A
3	O	D	R	E	N	A

Bubblesort

	O	R	D	E	N	A
1	O	R	D	E	N	A
2	O	R	D	E	N	A
3	O	D	R	E	N	A
4	O	D	E	R	N	A

Bubblesort

	O	R	D	E	N	A
1	O	R	D	E	N	A
2	O	R	D	E	N	A
3	O	D	R	E	N	A
4	O	D	E	R	N	A
5	O	D	E	N	R	A

Bubblesort

	O	R	D	E	N	A
1	O	R	D	E	N	A
2	O	R	D	E	N	A
3	O	D	R	E	N	A
4	O	D	E	R	N	A
5	O	D	E	N	R	A
6	O	D	E	N	A	R

Bubblesort

	O	R	D	E	N	A	
1	O	R	D	E	N	A	
2	O	R	D	E	N	A	
3	O	D	R	E	N	A	
4	O	D	E	R	N	A	
5	O	D	E	N	R	A	
6	O	D	E	N	A	R	
7	O	D	E	N	A	R	

Bubblesort

	O	R	D	E	N	A	
1	O	R	D	E	N	A	
2	O	R	D	E	N	A	
3	O	D	R	E	N	A	
4	O	D	E	R	N	A	
5	O	D	E	N	R	A	
6	O	D	E	N	A	R	
7	O	D	E	N	A	R	
8	D	O	E	N	A	R	

Bubblesort

9 D E **O** **N** A | R

Bubblesort

9	D	E	O	N	A		R
10	D	E	N	O	A		R

Bubblesort

9	D	E	O	N	A		R
10	D	E	N	O	A		R
11	D	E	N	A		O	R

Bubblesort

9	D	E	O	N	A		R
10	D	E	N	O	A		R
11	D	E	N	A	O		R
12	D	E	N	A	O		R

Bubblesort

9	D	E	O	N	A	R
10	D	E	N	O	A	R
11	D	E	N	A	O	R
12	D	E	N	A	O	R
13	D	E	N	A	O	R

Bubblesort

9	D	E	O	N	A	R
10	D	E	N	O	A	R
11	D	E	N	A	O	R
12	D	E	N	A	O	R
13	D	E	N	A	O	R
14	D	E	N	A	O	R

Bubblesort

9	D	E	O	N	A		R
10	D	E	N	O	A		R
11	D	E	N	A	O		R
12	D	E	N	A	O		R
13	D	E	N	A	O		R
14	D	E	N	A	O		R
15	D	E	A	N	O		R

Bubblesort

9	D	E	O	N	A		R
10	D	E	N	O	A		R
11	D	E	N	A	O		R
12	D	E	N	A	O		R
13	D	E	N	A	O		R
14	D	E	N	A	O		R
15	D	E	A	N	O		R
16	D	E	A	N	O		R

Bubblesort

9	D	E	O	N	A	R
10	D	E	N	O	A	R
11	D	E	N	A	O	R
12	D	E	N	A	O	R
13	D	E	N	A	O	R
14	D	E	N	A	O	R
15	D	E	A	N	O	R
16	D	E	A	N	O	R
17	D	E	A	N	O	R

Bubblesort

18 **D A** | E N O R

Bubblesort

18	D	A		E	N	O	R
19	A		D	E	N	O	R

Bubblesort

18	D	A		E	N	O	R
19	A	D		E	N	O	R
	A	D		E	N	O	R

BUBBLESORT(A)

```
1  para  $i = A.tam - 1$  até 1 passo  $-1$   
2      para  $j = 1$  até  $i$   
3          se  $A[j] > A[j + 1]$   
4              troque  $A[j]$  com  $A[j + 1]$ 
```

Selection sort

Ordenação por seleção

1. Ache a chave mínima na lista de $1..n$ e troque-a com a 1º chave.
2. Ache a chave mínima na lista de $2..n$ e troque-a com a 2º chave.
3. Etc, até a lista de $n - 1..n$.

Selection sort

O R D E N A

Selection sort

	O	R	D	E	N	A
1	O	R	D	E	N	A

Selection sort

	O	R	D	E	N	A
1	O	R	D	E	N	A
2	A	R	D	E	N	O

Selection sort

	O	R	D	E	N	A	
1	O	R	D	E	N	A	
2	A	R	D	E	N	O	
3	A		R	D	E	N	O

Selection sort

	O	R	D	E	N	A
1	O	R	D	E	N	A
2	A	R	D	E	N	O
3	A	R	D	E	N	O
4	A	D	R	E	N	O

Selection sort

	O	R	D	E	N	A
1	O	R	D	E	N	A
2	A	R	D	E	N	O
3	A	R	D	E	N	O
4	A	D	R	E	N	O
5	A	D	R	E	N	O

Selection sort

	O	R	D	E	N	A
1	O	R	D	E	N	A
2	A	R	D	E	N	O
3	A	R	D	E	N	O
4	A	D	R	E	N	O
5	A	D	R	E	N	O
6	A	D	E	R	N	O

Selection sort

	O	R	D	E	N	A
1	O	R	D	E	N	A
2	A	R	D	E	N	O
3	A	R	D	E	N	O
4	A	D	R	E	N	O
5	A	D	R	E	N	O
6	A	D	E	R	N	O
7	A	D	E	R	N	O

Selection sort

	O	R	D	E	N	A
1	O	R	D	E	N	A
2	A	R	D	E	N	O
3	A	R	D	E	N	O
4	A	D	R	E	N	O
5	A	D	R	E	N	O
6	A	D	E	R	N	O
7	A	D	E	R	N	O
8	A	D	E	N	R	O

Selection sort

9 A D E N | R **O**

Selection sort

9	A	D	E	N		R	O
10	A	D	E	N		O	R

Selection sort

9	A	D	E	N		R	O
10	A	D	E	N		O	R
11	A	D	E	N	O		R

Selection sort

9	A	D	E	N	R	O
10	A	D	E	N	O	R
11	A	D	E	N	O	R
	A	D	E	N	O	R

SELECTION-SORT(A)

```
1  para  $i = 1$  até  $A.tam - 1$   
2       $min = i$   
3      para  $j = i + 1$  até  $A.tam$   
4          se  $A[j] < A[min]$   
5               $min = j$   
6      troque  $A[i]$  com  $A[min]$ 
```

Insertion sort

Ordenação por inserção

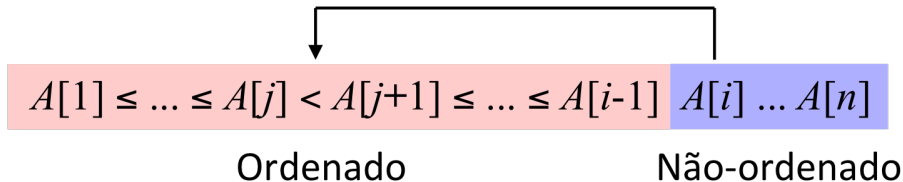
Cartas na mesa estão desordenadas. Cada carta da mesa é colocada na mão de forma ordenada.



Insertion sort

Ordenação por inserção

i -ésimo elemento no bloco não-ordenado é movido para a posição correta no bloco ordenado.



Insertion sort

O R D E N A

Insertion sort

	O	R	D	E	N	A	
1	O		R	D	E	N	A

Insertion sort

	O	R	D	E	N	A	
1	O		R	D	E	N	A
2	O	R		D	E	N	A

Insertion sort

	O	R	D	E	N	A	
1	O		R	D	E	N	A
2	O	R		D	E	N	A
3	D	O	R		E	N	A

Insertion sort

	O	R	D	E	N	A	
1	O		R	D	E	N	A
2	O	R		D	E	N	A
3	D	O	R		E	N	A
4	D	E	O	R		N	A

Insertion sort

	O	R	D	E	N	A	
1	O		R	D	E	N	A
2	O	R		D	E	N	A
3	D	O	R		E	N	A
4	D	E	O	R		N	A
5	D	E	N	O	R		A

Insertion sort

	O	R	D	E	N	A	
1	O		R	D	E	N	A
2	O	R		D	E	N	A
3	D	O	R		E	N	A
4	D	E	O	R		N	A
5	D	E	N	O	R		A
	A	D	E	N	O	R	

INSERTION-SORT(A)

1 **para** $i = 2$ **até** $A.tam$

2 $x = A[i]$

3 $j = i - 1$

4 $A[0] = x$ // *sentinela*





5 **enquanto** $x < A[j]$

6 $A[j + 1] = A[j]$

7 $j = j - 1$

8 $A[j + 1] = x$

Referências

-  T. H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, 3rd edition, MIT Press, 2010
-  A. Levitin. Introduction to the Design and Analysis of Algorithms. 3rd edition. Addison-Wesley, 2007
-  R. Sedgewick, K. Wayne. Algorithms. 4th edition, Addison-Wesley Professional, 2011
-  N. Ziviani. Projeto de Algoritmos com Implementação em Pascal C. Cengage Learning, 2012

Onde obter este material:

`est.uea.edu.br/fcoelho`