

Algoritmos e Estrutura de Dados II

Algoritmos de Ordenação

prof. Frederico Santos de Oliveira

Universidade Federal de Mato Grosso
Instituto de Engenharia



Roteiro

- 1 Objetivos
- 2 Referências bibliográficas
- 3 Introdução
- 4 Bubblesort
- 5 Selectionsort
- 6 Insertionsort
- 7 Exercício
- 8 Conclusão

Objetivos


Esta aula tem como objetivos:


- 1 Apresentar os conceitos básicos sobre ordenação;
- 2 Explicitar os métodos mais simples de ordenação por comparação;
- 3 Exemplificar a execução dos algoritmos.

Referências bibliográficas

 OLIVEIRA, S. L. G. *Algoritmos e seus fundamentos*. 1. ed. Lavras: Editora UFLA, 2011.

 PIVA, D. et al. *Estrutura de Dados e Técnicas de Programação*. São Paulo: Elsevier, 2014. ISBN 9788535274387.

 SEDGEWICK, R.; WAYNE, K. *Algorithms, 4th Edition*. Boston: Addison-Wesley, 2011. I-XII, 1-955 p. ISBN 978-0-321-57351-3.

 ZIVIANI, N. *Projeto de Algoritmos: com implementações em Pascal e C*. São Paulo: Cengage Learning, 2011. ISBN 9788522110506.

Ordenar

Segundo Ziviani (2011), ordenar é o processo de rearranjar um conjunto de objetos em uma ordem ascendente ou decendente.

- A ordenação visa facilitar a recuperação posterior de itens do conjunto ordenado;
- As técnicas de ordenação permitem apresentar um amplo de algoritmos distintos para resolver uma mesma tarefa.

Introdução

Segundo Sedgewick e Wayne (2011), existem três razões práticas para estudar os algoritmos de ordenação:

- Analisar os algoritmos de ordenação é uma introdução completa as técnicas de comparação de desempenho de algoritmos;
- Técnicas semelhantes são eficazes no tratamento de outros problemas;
- Muitas vezes usamos algoritmos de ordenação como ponto de partida para resolver outros problemas.

Introdução

Segundo Sedgewick e Wayne (2011), existem três razões práticas para estudar os algoritmos de ordenação:

- Analisar os algoritmos de ordenação é uma introdução completa as técnicas de comparação de desempenho de algoritmos;
- Técnicas semelhantes são eficazes no tratamento de outros problemas;
- Muitas vezes usamos algoritmos de ordenação como ponto de partida para resolver outros problemas.

Mais importante do que esses motivos práticos é que os algoritmos são elegantes, clássicos, e eficazes.

Notação

- Os métodos trabalham sobre os registros de um arquivo;
- Cada registro possui uma chave para controlar a ordenação;
- Podem existir outros componentes em um registro;
- Exemplo de estrutura, em C:

Algoritmo 1: TAD Registro

1 **início**

```
2   | registro {  
3   |   | int chave ;  
4   |   | Ponteiro dados;  
5   | } ItemVetor;
```

-
- A chave é usada como critério para ordenação.
 - O ponteiro dados aponta para a informação indexada pela chave.

Características

- Qualquer tipo de chave, sobre o qual exista uma regra de ordenação bem definida, pode ser utilizada;
 - As ordens mais usadas são a numérica e a lexicográfica.
- Um método de ordenação é estável se a ordem relativa dos itens com chaves iguais não se altera durante a ordenação.
 - Alguns dos métodos mais eficientes não são estáveis;
 - A estabilidade pode ser forçada quando o método é não-estável.

Características

Os métodos de ordenação podem ser classificados como:

- **Ordenação interna:** o arquivo a ser ordenado cabe todo na memória principal;
 - Qualquer registro pode ser imediatamente acessado.
- **Ordenação externa:** o arquivo a ser ordenado não cabe todo na memória principal;
 - Registros são acessados sequencialmente ou em blocos.

Características

- A maioria dos métodos de ordenação é baseada em comparações de chaves;
- Existem métodos que utilizam o princípio da distribuição;
- Exemplo: ordenar um baralho com 52 cartas, pelo valor e pelo naipe seguindo as seguintes instruções:
 - Separe as cartas em 13 montes (valores das cartas);
 - Colete os montes na ordem desejada;
 - Distribua cada monte em 4 montes (naipes das cartas);
 - Colete os montes na ordem desejada.
- Qual é o custo desse algoritmo?

Cr terios de an lise

- Dado que n   a quantidade de registros em um arquivo, as medidas de complexidade relevantes s o:
 - n mero de compara  es $C(n)$ entre chaves;
 - n mero de movimentac  es $M(n)$ de registros dos arquivo.
- O uso da mem ria   um requisito primordial na ordena  o interna:
 - os m todos de ordena  o *in situ* s o os preferidos;
 - tamb m existem aqueles que utilizam listas encadeadas;
 - e os m todos que fazem c pias dos itens a serem ordenados.

Ordenação interna por comparação

Métodos simples:

- adequados para pequenos arquivos;
- requerem $O(n^2)$ comparações;
- produzem programas pequenos.

Métodos eficientes:

- adequados para arquivos maiores;
- requerem $O(n \log n)$ comparações;
- usam menos comparações;
- as comparações são mais complexas nos detalhes.

O que estudaremos?

Na disciplina:

Estudaremos os algoritmos de **ordenação interna** que utilizam o princípio de **comparação e por contagem**.

Nesta aula:

- Bubblesort
- Selectionsort
- Insertionsort

O melhor algoritmo

Atenção

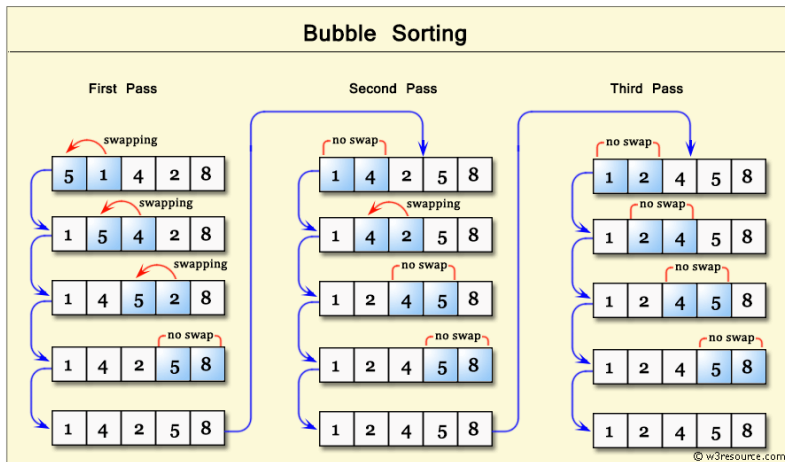
- Não existe um método de ordenação considerado superior a todos os outros.
- É necessário analisar o problema e, com base nas características dos dados, decidir qual método melhor se aplica à ele.

Bubblesort

Bubblesort

- Os elementos vão “**borbulhando**” a cada iteração do método até a posição correta para ordenação da lista;
- Como os elementos são trocados (borbulhados) frequentemente, há um alto custo com troca de elementos.

Exemplo



Pseudo-código Bubblesort

Algoritmo 2: Bubblesort

Entrada: Vetor $V[0..n-1]$, tamanho n

Saída: Vetor V ordenado

```
1 início
2   para ( $i \leftarrow 1$  até  $n-1$ ) faça
3       para ( $j \leftarrow 0$  até  $n-i-1$ ) faça
4           se ( $V[j] > V[j+1]$ ) então
5               Trocar  $V[j] \leftrightarrow V[j+1]$ 
```

Análise

Bubblesort

Algoritmo 3: Bubblesort

Entrada: Vetor $V[0..n-1]$, tamanho n

Saída: Vetor V ordenado

1 **início**

```
2   para ( $i \leftarrow 1$  até  $n-1$ ) faça
3       para ( $j \leftarrow 0$  até  $n-i-1$ ) faça
4           se ( $V[j] > V[j+1]$ ) então
5               Trocar  $V[j] \leftrightarrow V[j+1]$ 
```

Nº execuções

Análise

Bubblesort

Algoritmo 4: Bubblesort

Entrada: Vetor $V[0..n-1]$, tamanho n

Saída: Vetor V ordenado

```
1 início
2   para ( $i \leftarrow 1$  até  $n-1$ ) faça
3       para ( $j \leftarrow 0$  até  $n-i-1$ ) faça
4           se ( $V[j] > V[j+1]$ ) então
5               Trocar  $V[j] \leftrightarrow V[j+1]$ 
```

Nº execuções

$$L2 = n$$

$$\begin{aligned} L3 &= n + (n-1) + \dots + 1 \\ &= \frac{(1+n)(n)}{2} = \frac{n^2 + n}{2} \end{aligned}$$

$$\begin{aligned} L4 &= (n-1) + (n-2) + \dots + 0 \\ &= \frac{(0+n-1)n}{2} = \frac{n^2 - n}{2} \end{aligned}$$

$$L5 = \frac{n^2 - n}{2}$$

Tabela: Custo Bubblesort

Linha	Melhor	Pior	Médio
2	n	n	n
3	$\frac{n^2+n}{2}$	$\frac{n^2+n}{2}$	$\frac{n^2+n}{2}$
4	$\frac{n^2-n}{2}$	$\frac{n^2-n}{2}$	$\frac{n^2-n}{2}$
5	0	$\frac{n^2-n}{2}$	$\frac{n^2-n}{4}$

- Melhor caso: $T(n) = n + \frac{n^2+n}{2} + \frac{n^2-n}{2} + 0 = n^2 + n$
- Pior caso: $T(n) = n + \frac{n^2+n}{2} + 2\frac{n^2-n}{2} = \frac{3n^2}{2} + 2$
- Caso médio: $T(n) = n + \frac{n^2+n}{2} + \frac{n^2-n}{2} + \frac{n^2-n}{4} = \frac{5n^2}{4} - \frac{3n}{4}$

- Número de Comparações $C(n)$ é definido pela linha 4.
 - Possui custo $C(n) = \frac{n^2-n}{2}$ em qualquer situação.
 - Portanto, o número de comparações é $C(n) = \Theta(n^2)$.
- Número de Movimentos $M(n)$ é definido pela linha 5.
 - No melhor caso, não realiza trocas.
 - No pior caso, realiza $\frac{n^2-n}{2}$ trocas, ou seja, é $O(n^2)$ em relação ao n^o de movimentos.
 - No caso médio, realiza $\frac{n^2-n}{4}$ trocas.

Características

Vantagens

- Algoritmo **simples**;
- Algoritmo **estável**

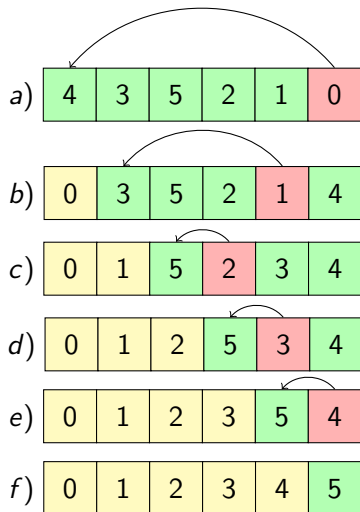
Desvantagens

- O fato do arquivo já estar ordenado não ajuda a reduzir o número de comparações (o custo continua quadrático), porém o número de movimentações cai para zero.

Selectionsort

Selectionsort

- A cada iteração, seleciona o menor elemento da lista e troque-o com o item na posição correta;
- É realizada uma troca a cada iteração.



Pseudo-código Selectionsort

Algoritmo 5: Selectionsort

Entrada: Vetor $V[0..n-1]$, tamanho n

Saída: Vetor V ordenado

```
1 início
2   para  $i \leftarrow 0$  até  $n-2$  faça
3      $min \leftarrow i$ 
4     para  $j \leftarrow i+1$  até  $n-1$  faça
5       se  $V[j] < V[min]$  então
6          $min \leftarrow j$ 
7     trocar  $V[min] \leftrightarrow V[i]$ 
```

Adaptado de Oliveira (2011, p. 73).

Análise

Selectionsort

Algoritmo 6: Selectionsort

Entrada: Vetor $V[0..n-1]$, tamanho n

Saída: Vetor V ordenado

```
1 início
2   para  $i \leftarrow 0$  até  $n-2$  faça
3      $min \leftarrow i$ 
4     para  $j \leftarrow i+1$  até  $n-1$  faça
5       se  $V[j] < V[min]$  então
6          $min \leftarrow j$ 
7     trocar  $V[min] \leftrightarrow V[i]$ 
```

Nº execuções

Análise

Selectionsort

Algoritmo 7: Selectionsort

Entrada: Vetor $V[0..n-1]$, tamanho n

Saída: Vetor V ordenado

1 **início**

```
2   para  $i \leftarrow 0$  até  $n-2$  faça
3        $min \leftarrow i$ 
4       para  $j \leftarrow i+1$  até  $n-1$  faça
5           se  $V[j] < V[min]$  então
6                $min \leftarrow j$ 
7       trocar  $V[min] \leftrightarrow V[i]$ 
```

Nº execuções

$$L2 = n$$

$$L3 = n - 1$$

$$L4 = n + (n-1) + \dots + 1$$
$$= \frac{(1+n)n}{2} = \frac{n^2 + n}{2}$$

$$L5 = \frac{n^2 - n}{2}$$

$$L6 = n - 1$$

Tabela: Custo Selectionsort

Linha	Melhor	Pior	Médio
2	n	n	n
3	$n - 1$	$n - 1$	$n - 1$
4	$\frac{n^2+n}{2}$	$\frac{n^2+n}{2}$	$\frac{n^2+n}{2}$
5	$\frac{n^2-n}{2}$	$\frac{n^2-n}{2}$	$\frac{n^2-n}{2}$
6	0	$\frac{n^2-n}{2}$	$\frac{n^2-n}{4}$
7	$n - 1$	$n - 1$	$n - 1$

- Melhor caso: $T(n) = n + 2(n - 1) + n^2 = n^2 + 3n - 2$
- Pior caso: $T(n) = n + 2(n - 1) + \frac{n^2+n}{2} + (n^2 - n) = \frac{3n^2}{2} + \frac{5n}{2} - 2$
- Caso médio: $T(n) = n + 2(n - 1) + \frac{n^2+n}{2} + \frac{n^2-n}{2} + \frac{n^2-n}{4} = \frac{5n^2}{4} + \frac{11n}{4} - 2$

- Número de Comparações $C(n)$ é definido pela linha 5.
 - Possui custo $C(n) = \frac{n^2 - n}{2}$ em qualquer situação.
 - Portanto, o número de comparações é $C(n) = \Theta(n^2)$.
- Número de Movimentos $M(n)$ é definido pela linha 7.
 - Em qualquer situação, realiza $n - 1$ trocas.

Características

Vantagens

- Custo linear no tamanho da entrada para o número de movimentos de registros;
- É o algoritmo a ser utilizado para arquivos com registros muito grandes (que possuem alto custo de movimentação);
- É muito interessante para arquivos pequenos.

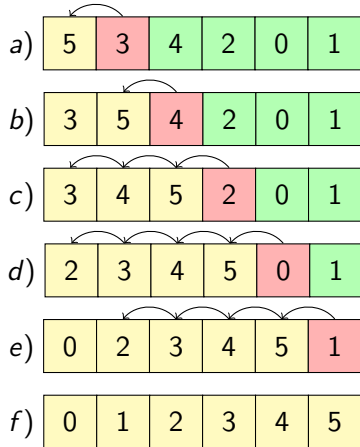
Desvantagens

- O fato do arquivo já estar ordenado não ajuda em nada, pois o custo continua quadrático;
- O algoritmo **não é estável**.

Insertionsort

Insertionsort

- Algoritmo utilizado pelos jogadores de cartas:
 - Inicia-se com a mão esquerda vazia e as cartas viradas com a face para baixo na mesa;
 - Em seguida, remove-se uma carta de cada vez na mesa, inserindo-a na posição correta na mão esquerda;
 - Para encontrar a posição correta de uma carta, compare-a sequencialmente a cada uma das cartas que já estão na mão.



Pseudo-código Insertionsort

Algoritmo 8: Insertionsort

Entrada: Vetor $V[0..n-1]$, tamanho n

Saída: Vetor V ordenado

```
1 início
2   para ( $i \leftarrow 1$  até  $n-1$ ) faça
3       chave  $\leftarrow V[i]$ 
4        $j \leftarrow i-1$ 
5       enquanto ( $j \geq 0$  AND  $V[j] > \text{chave}$ ) faça
6            $V[j+1] \leftarrow V[j]$ 
7            $j \leftarrow j-1$ 
8        $V[j+1] \leftarrow \text{chave}$ 
```

Adaptado de Oliveira (2011).

Análise

Insertionsort

Algoritmo 9: Insertionsort

Entrada: Vetor $V[0..n-1]$, tamanho n

Saída: Vetor V ordenado

1 **início**

```
2   para ( $i \leftarrow 1$  até  $n-1$ ) faça
3        $chave \leftarrow V[i]$ 
4        $j \leftarrow i-1$ 
5       enquanto ( $j \geq 0$  AND  $V[j] > chave$ )
6           faça
7                $V[j+1] \leftarrow V[j]$ 
8                $j \leftarrow j-1$ 
9        $V[j+1] \leftarrow chave$ 
```

Nº execuções

Análise

Insertionsort

Algoritmo 10: Insertionsort

Entrada: Vetor $V[0..n-1]$, tamanho n

Saída: Vetor V ordenado

```
1 início
2   para  $(i \leftarrow 1 \text{ até } n-1)$  faça
3       chave  $\leftarrow V[i]$ 
4        $j \leftarrow i-1$ 
5       enquanto  $(j \geq 0 \text{ AND } V[j] > \text{chave})$ 
6           faça
7                $V[j+1] \leftarrow V[j]$ 
8                $j \leftarrow j-1$ 
9        $V[j+1] \leftarrow \text{chave}$ 
```

Nº execuções

$$L2 = n$$

$$L3 \text{ e } L4 = n-1$$

$$\begin{aligned} L5 &= 2 + 3 + \dots + n \\ &= \frac{(2+n)(n-1)}{2} \end{aligned}$$

$$= \frac{n^2 + n - 2}{2}$$

$$\begin{aligned} L6 \text{ e } L7 &= 1 + 2 + \dots + (n-1) \\ &= \frac{(1+n-1)(n-1)}{2} \end{aligned}$$

$$= \frac{n^2 - n}{2}$$

$$L8 = n-1$$

Tabela: Custo Insertionsort

Linha	Melhor	Pior	Médio
2	n	n	n
3 e 4	$2(n - 1)$	$2(n - 1)$	$2(n - 1)$
5	n	$\frac{n^2+n-2}{2}$	$\frac{n^2+3n-2}{4}$
6 e 7	0	$2\frac{n^2-n}{2}$	$2\frac{n^2-n}{4}$
8	$n - 1$	$n - 1$	$n - 1$

- Melhor caso: $T(n) = n + 3(n - 1) + n = 5n - 3$
- Pior caso: $T(n) = n + 3(n - 1) + \frac{n^2+n-2}{2} + 2\frac{n^2-n}{2} = \frac{3n^2}{2} + \frac{7n}{2} - 4$
- Caso médio: $T(n) = n + 3(n - 1) + \frac{n^2+3n-2}{4} + \frac{n^2-n}{2} = \frac{3n^2}{4} + \frac{17n}{4} - \frac{7}{2}$

- Número de Comparações $C(n)$ é definido pela linha 5.
 - No melhor caso, $C(n) = n$, ou seja, é $\Omega(n)$.
 - No pior caso, $C(n) = \frac{n^2+n-2}{2}$, ou seja, é $O(n^2)$.
 - No caso médio, $C(n) = \frac{n^2+3n-2}{4}$.
- Número de Movimentos $M(n)$ é definido pelas linhas 6 e 8.
 - No melhor caso, a linha 6 é executada nenhuma vez, mas a linha 8 $n - 1$ vezes.
 - No pior caso, a linha 6 é executada $\frac{n^2-n}{2}$ e a linha 8 $n - 1$, o que dá um total de $\frac{n^2}{2} + \frac{n}{2} - 1$ trocas.
 - No caso médio, a linha 6 é executada $\frac{n^2-n}{4}$, o que dá $\frac{n^2}{4} + \frac{3n}{4} - 1$ trocas.

Insertionsort

Características

Vantagens e desvantagens

- O número mínimo de comparações e movimentos ocorre quando os itens estão originalmente em ordem;
- O número máximo ocorre quando os itens estão originalmente na ordem reversa;
- É o método a ser utilizado quando o arquivo está “quase ordenado”;
- É um bom método quando se deseja adicionar uns poucos itens a um arquivo ordenado, pois o custo é linear;
- O algoritmo de ordenação por inserção é **estável**.

Tarefa

- Dada a sequência de números em ordem decrescente:
6 5 4 3 2 1;
- Ordene em ordem crescente utilizando os três algoritmos estudados em sala (**BubbleSort**, **SelectionSort** e **InsertionSort**), apresentando a sequência dos números a cada passo.

Conclusão

Objetivos alcançados

- Nesta aula, tivemos o primeiro contato com algoritmos de ordenação;
- Foram vistos os algoritmos **BubbleSort**, **SelectionSort** e **InsertionSort**;

Qual o melhor?

- Cada algoritmo possui suas características particulares;
- Não existe um método de ordenação considerado universalmente superior a todos os outros;
- É necessário analisar o problema e, com base nas características dos dados, decidir qual método melhor se aplica à ele.

Conclusão

Quadro comparativo dos métodos de ordenação:

Algoritmo	Comparações			Movimentações			Espaço	Estável	In situ
	Melhor	Médio	Pior	Melhor	Médio	Pior			
Bubble	$O(n^2)$			$O(n)$	$O(n^2)$		$O(1)$	Sim	Sim
Selection	$O(n^2)$			$O(n)$			$O(1)$	Não *	Sim
Insertion	$O(n)$	$O(n^2)$		$O(n)$	$O(n^2)$		$O(1)$	Sim	Sim

* Existem versões estáveis.

Conclusão

Ordenação

- A tarefa de ordenação é muito importante, ela é uma necessidade básica para a solução de muitos problemas.
- Piva et al. (2014) sugere alguns materiais para reforçarem o aprendizagem:
 - Simulador didático de testes de algoritmos de ordenação
 - Vídeo demonstrando os métodos de ordenação

Próxima aula

Estudar os algoritmos considerados eficientes.

Dúvidas

