CENTRO DE CIÊNCIA E TECNOLOGIA
LABORATÓRIO DE CIÊNCIAS MATEMÁTICAS
UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE

Algoritmos de Ordenação Elementares

Disciplina: Estrutura de Dados I

Prof. Fermín Alfredo Tang Montané

Curso: Ciência da Computação

Algoritmo de Bolha (Bubblesort)
Descrição do Algoritmo

- Os algoritmos elementares são apropriados para ordenar um número pequeno de elementos.
- O algoritmo de ordenação Bubblesort é um algoritmo de ordenação popular.
- Consiste em trocar de maneira repetida elementos adjacentes que estão fora de ordem.

Algoritmo de Bolha (Bubblesort)

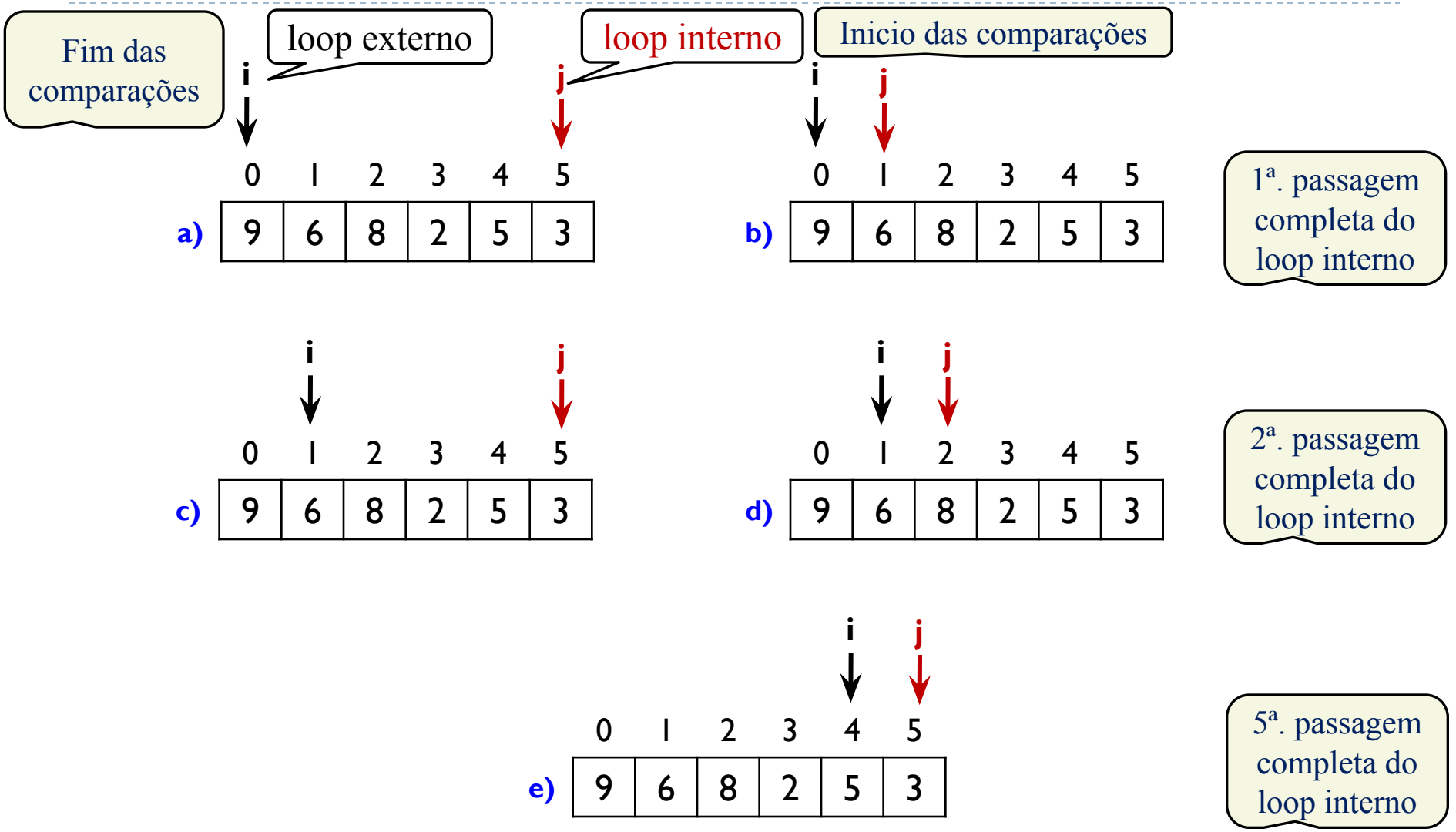
Descrição do Algoritmo – Versão1

- O algoritmo de bolha ordena um vetor A de tamanho N.
- Considere que a função Comprimento(A) retorna o número de elementos do vetor A.
- Esta versão faz comparações e trocas (começando pelos últimos elementos do vetor) de maneira a concentrar os menores elementos no inicio do vetor.

```
BUBBLESORT(A)                          /* Ordena o arranjo A[N] */
1.  para i ← 0 até Comprimento[A]-2 fazer
2.      para j ← Comprimento[A]-1 até i + 1 fazer
3.          se A[ j ] < A[ j – 1] então          /* Comparação */
4.              Trocar A[ j ] ↔ A[ j – 1]         /* Troca */
5.          fim-se
6.      fim-para
7.  fim-para
```

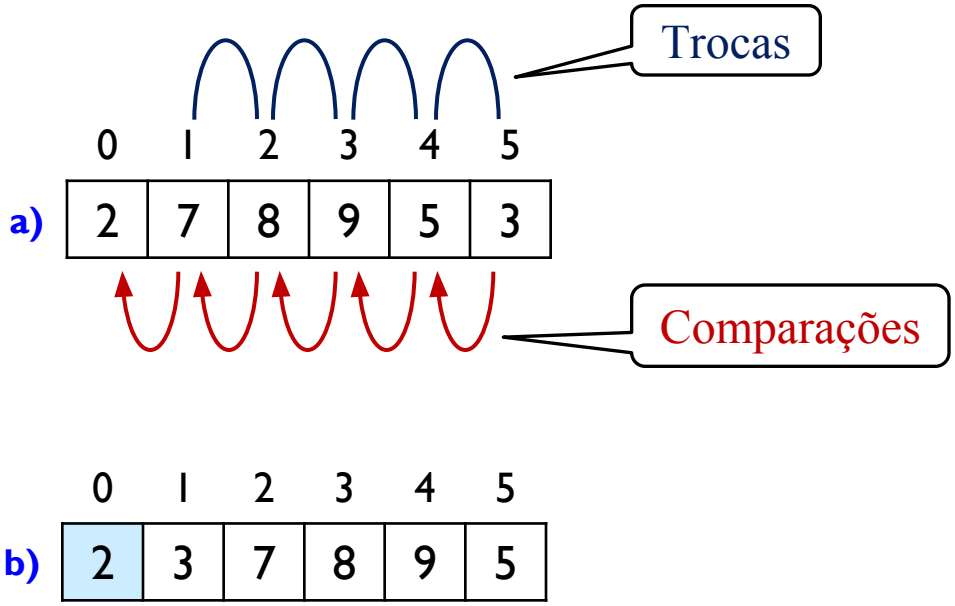
Ordenação por Seleção (Bubblesort)

Exemplo sobre a variação dos índices



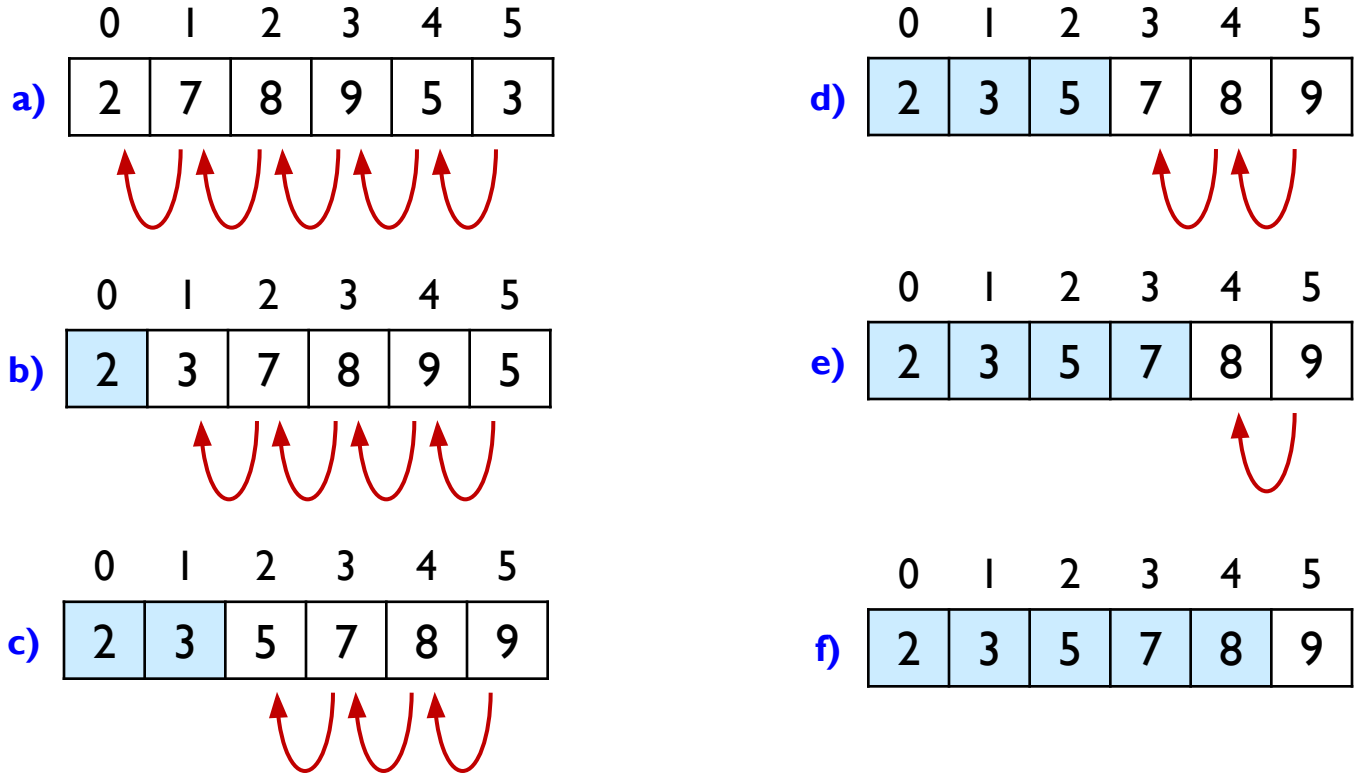
Algoritmo de Bolha (Bubblesort)

Exemplo de uma iteração



Algoritmo de Bolha (Bubblesort)

Exemplo Completo



Algoritmo de Bolha (Bubblesort)

Complexidade do Algoritmo - Comparações

- Como pode ser observado no exemplo, com N=6 elementos, são realizadas 5 comparações na primeira passagem, 4 comparações na segunda e assim sucessivamente até ter uma comparação na última passagem. O número total de comparações é:

$$5+4+3+2+1=15$$

- No caso geral, com N elementos, temos N-1 comparações na primeira passagem, N-2 comparações na segunda e assim sucessivamente até ter 1 comparação na última passagem.

$$(N-1)+(N-2)+\ldots+2+1=\frac{N(N-1)}{2}$$

- É importante ter presente:

$$\sum_{i=1}^{N} i = \frac{N(N+1)}{2}$$

Algoritmo de Bolha (Bubblesort)

Complexidade do Algoritmo - Comparações

- Assim, o algoritmo faz aproximadamente $N^2/2$ comparações.
- Este valor $N^2/2$ difere pouco do valor exato, principalmente se N for grande.

Número de Elementos	$(N^2-N)/2$ Comparações Exato	$N^2/2$ Comparações Aprox.	Var %
10	45	50	10%
100	4.950	5.000	1%
1000	499.500	500.000	0,1%
5000	12.497.500	12.500.000	0,02%

$1Ghz = 10^9$ operações/seg

$10^6/10^9 = 10^{-3} \approx$ milésimos seg

Algoritmo de Bolha (Bubblesort)

Complexidade do Algoritmo - Trocas

- Em geral, o algoritmo realizará menos trocas que comparações.
- O número de trocas depende dos valores contidos no vetor, existem três casos a considerar:
  - **Melhor caso**.- o vetor já se encontra ordenado em ordem ascendente e nenhuma troca é realizada.
  - **Pior caso**.- o vetor se encontra ordenado em ordem descendente resultando em aprox. $N^2/2$ trocas. Uma troca a cada comparação.
  - **Caso Médio**.- acontece com dados aleatórios, neste caso, metade das comparações resultam em troca, temos assim $N^2/4$ trocas.

Algoritmo de Bolha (Bubblesort)

Complexidade do Algoritmo – Pior Caso

- No pior caso, o número de comparações e o número de trocas são proporcionais a N^2.
- Usando a analise de complexidade assintótica, e a notação Big O, sabe-se que podemos ignorar as constantes e os termos de menor ordem, assim o algoritmo executa em tempo $O(N^2)$.

Ordenação por Seleção (Selection Sort)

Descrição do Algoritmo

- A ordenação por seleção melhora a ordenação pelo método de bolha, reduzindo o número de trocas necessárias de $O(N^2)$ para $O(N)$.
- No entanto, o número de comparações permanece em $O(N^2)$.
- Com tudo, a melhoria pode ser significativa no caso de que grandes registros tenham que ser movidos fisicamente na memória.
- Nesse caso, o tempo de troca será muito mais importante que o tempo de comparação (Em java, são movidas as referencias e não os objetos).

Ordenação por Seleção (Selection Sort)

Descrição do Algoritmo

- A cada passagem, o algoritmo percorre um subconjunto de elementos do vetor e realiza comparações de maneira a **selecionar** o menor elemento do subconjunto.
- O menor elemento selecionado em cada passagem é colocado na extremidade esquerda do vetor mediante **uma troca**, ficando assim ordenado.
- O menor elemento selecionado na primeira passagem é colocado na posição 0. O menor elemento selecionado na segunda passagem é colocado na posição 1, e assim sucessivamente.
- Na próxima passagem, o subconjunto de elementos se reduz em um elemento, aqueles mais a direita.

Ordenação por Seleção (Selection Sort)

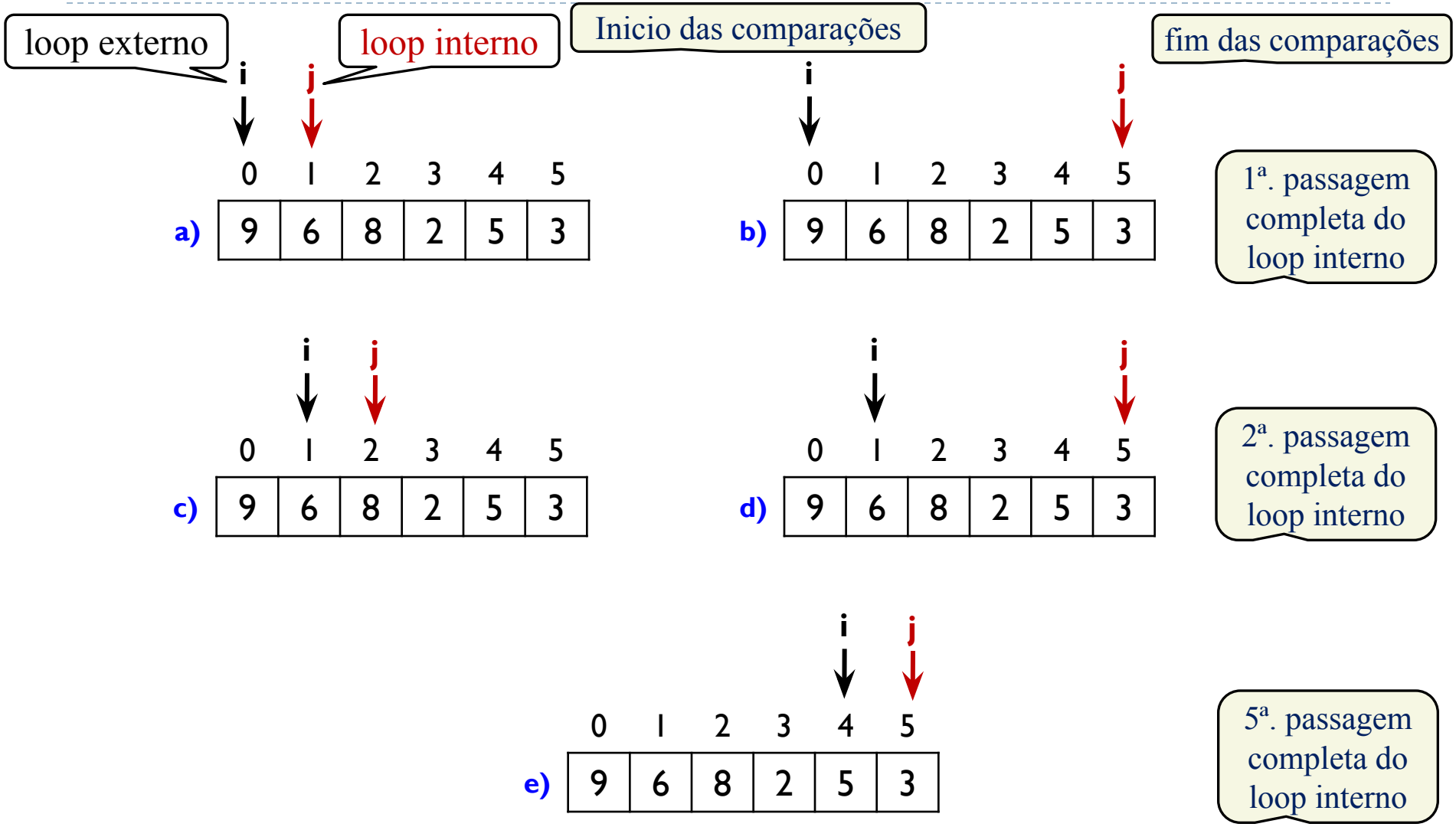
Descrição do Algoritmo

- O Algoritmo de ordenação por seleção ordena um vetor A de tamanho N. Considere que a função Comprimento(A) retorna o número de elementos do vetor A.

```
SELECTION_SORT (A)                                  /* Ordena o arranjo A[N] */
1.  para i ← 0 até  Comprimento(A)-2 fazer
2.        min ← i;                                  /* Índice do menor elemento*/
3.        para j ← i+1 até  Comprimento(A)-1 fazer
4.            se a[j] <a[min] então                 /* Comparações */
5.                min ← j;                          /* Atualização do Índice */
6.            fim-se
6.        fim-enquanto
8.        Trocar A[ i ] ↔ A[ min];                  /* Troca */
9.  fim-para
```

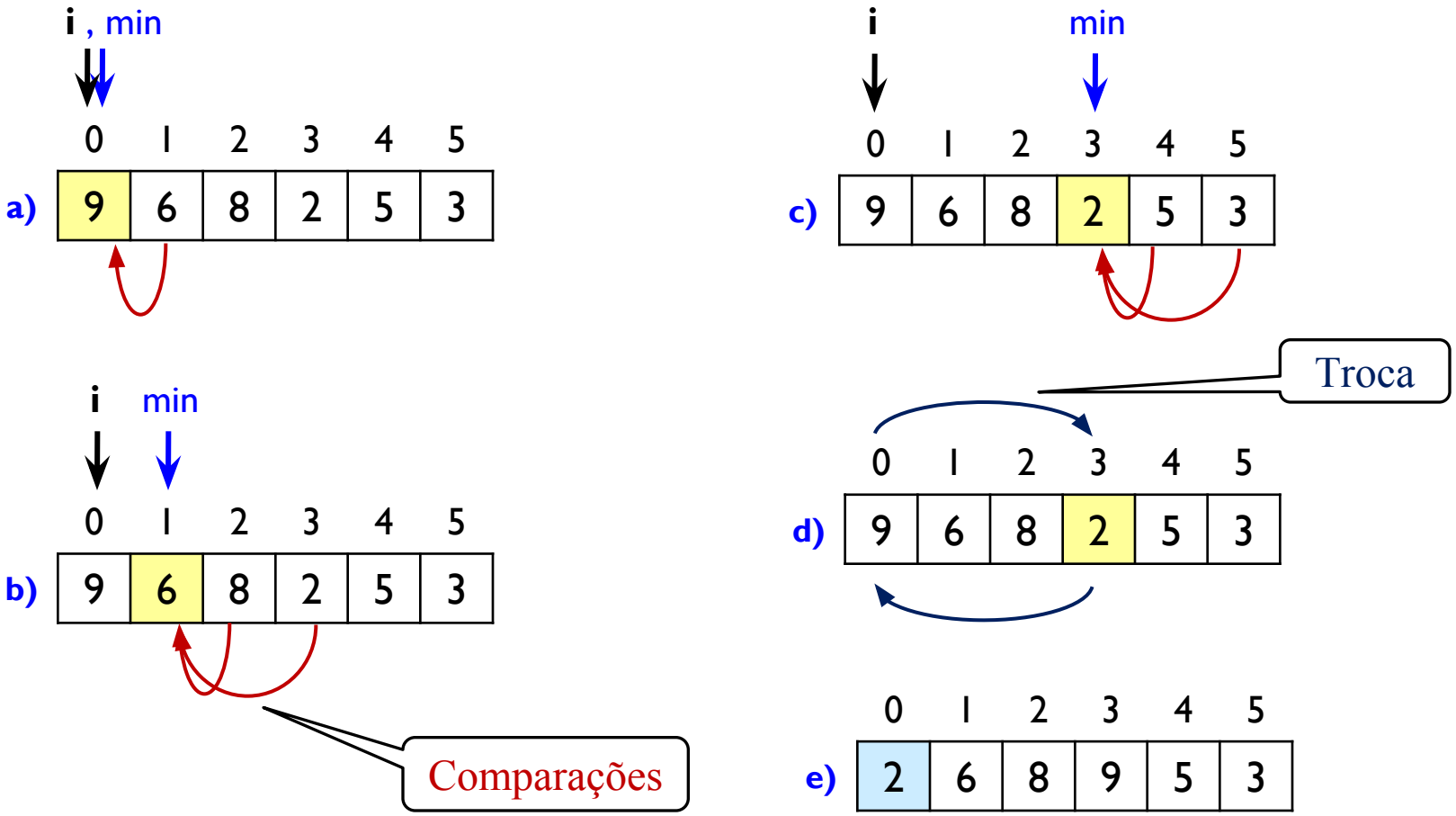
Ordenação por Seleção (Selection Sort)

Exemplo sobre a variação dos índices



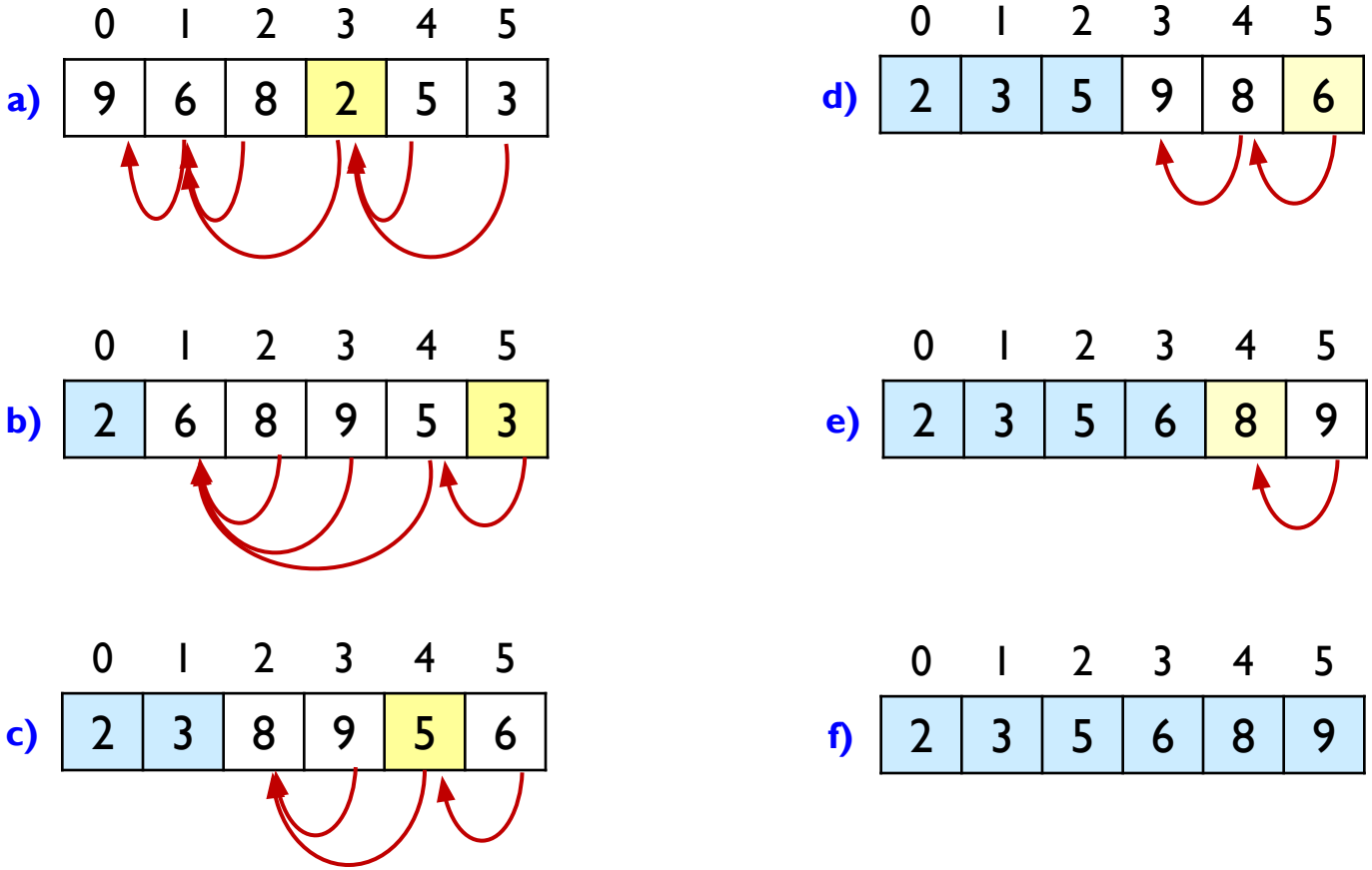
Ordenação por Seleção (Selection Sort)

Exemplo de uma iteração



Ordenação por Seleção (Selection Sort)

Exemplo Completo



Ordenação por Seleção (Selection Sort)

Complexidade do Algoritmo – Comparações e Trocas

- A ordenação por seleção realiza o mesmo número de comparações que a ordenação pelo método de bolha: N(N-1)/2.
- Porém, o número de trocas é muito menor: N-1, **no pior caso.**
- Para grandes valores de N, os tempos de comparação predominarão, com isso a ordenação por seleção **executa em tempo O(N^2)**, exatamente como o método bolha.

Número de Elementos	$N^2/2$ Comparações	N Trocas
10	50	10
100	5.000	100
1000	500.000	1000
5000	12.500.000	5000

Ordenação por Seleção (Selection Sort)

Complexidade do Algoritmo – Comparações e Trocas

- Com tudo, a ordenação por seleção é inquestionavelmente mais rápida que o método bolha porque há muito poucas trocas.
- Em particular, a ordenação por seleção poder ser consideravelmente mais rápida que o método bolha se os tempos de troca forem muito maiores que os tempos de comparação.

	Método Bolha		**Método Seleção**	
Número de Elementos	$N^2/2$ Comparações	$N^2/2$ Trocas	$N^2/2$ Comparações	N Trocas
10	50	50	50	10
100	5.000	5.000	5.000	100
1000	500.000	500.000	500.000	1000
5000	12.500.000	12.500.000	12.500.000	5000

Ordenação por Inserção (Insertion Sort)

Descrição do Algoritmo

- O método de inserção funciona de maneira semelhante ao modo como a maioria das pessoas ordena uma mão em um jogo de cartas.
- Começamos com a mão vazia e cartas na mesa viradas para baixo. Removemos uma carta da mesa, de cada vez, e a inserimos na posição correta na mão.
- Para achar a posição correta de uma carta, compara-se a carta com cada uma das cartas que já se encontram na mão.
- As cartas na mão permanecem ordenadas.

Ordenação por Inserção (Insertion Sort)

Descrição do Algoritmo

- O Algoritmo de ordenação por inserção ordena um vetor A de tamanho N. Considere que a função Comprimento(A) retorna o número de elementos do vetor A.

```
INSERTION_SORT (A)                                  /* Ordena o arranjo A[N] */
1.  para j ← 1 até  Comprimento(A)-1 fazer
2.        key ← A[ j ];                             /* Variável temporária Key */
3.        /* Insere key na sequência ordenada A[0 ... j–1]  */
4.        i ← j – 1;
5.        enquanto (i =>0) e (key<A[i]) fazer
6.              A[i + 1] ← A[ i ];                  /* Deslocamento de elementos */
7.              i ← i – 1;
8.        fim-enquanto
8.        A[i + 1] ← key;                           /* Inserção da Key */
9.  fim-para
```

Ordenação por Inserção (Insertion Sort)

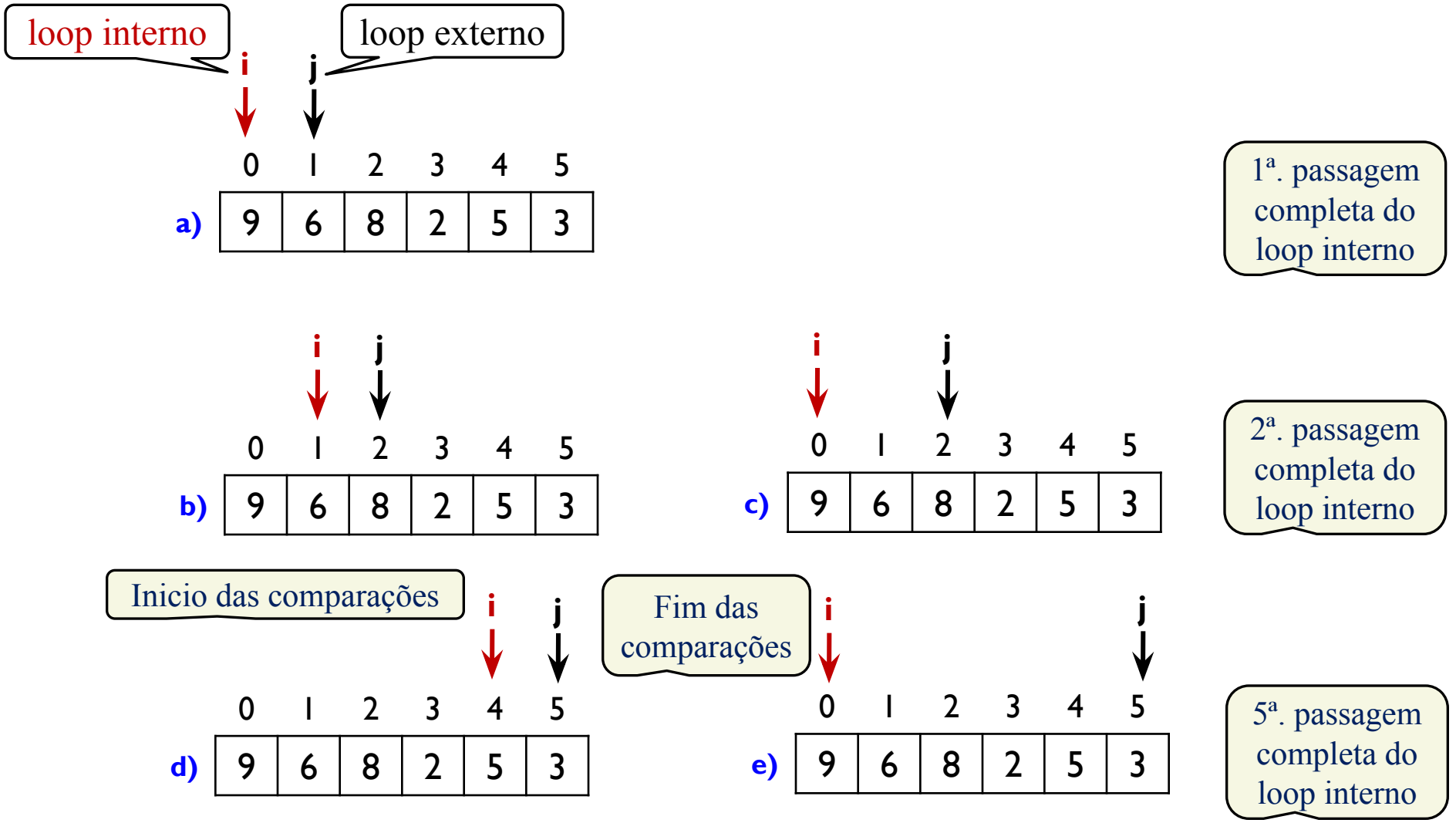
Algoritmo (Versão2)

- Esta versão define os índices de uma maneira um pouco diferente, porém equivalente.

```
INSERTION_SORT (A)                                  /* Ordena o arranjo A[0...N-1] */
1.      para j ← 1 até  Comprimento(A)-1 fazer
2.          key ← A[ j ];                           /* Variável temporária Key */
3.          i ← j;
4.          enquanto (i >0) e (key<=A[i-1]) fazer
5.              A[i] ← A[ i -1];                    /* Deslocamento de elementos */
6.              i ←i –1;
7.          fim-enquanto
8.          A[i] ← key;                             /* Inserção da Key */
9.      fim-para
```

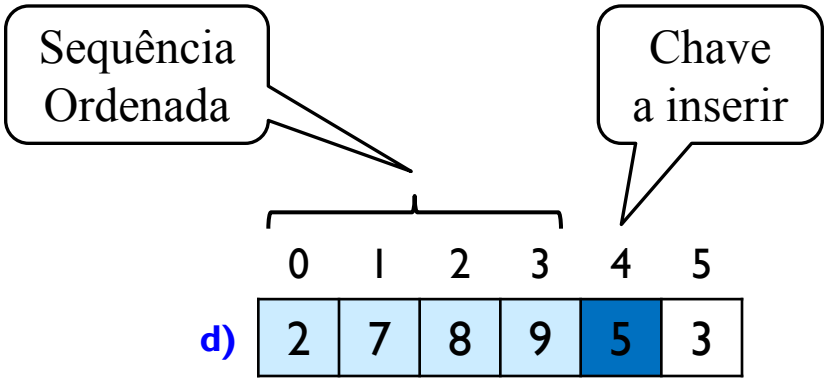
Ordenação por Inserção (Insertion Sort)

Exemplo sobre a variação dos índices



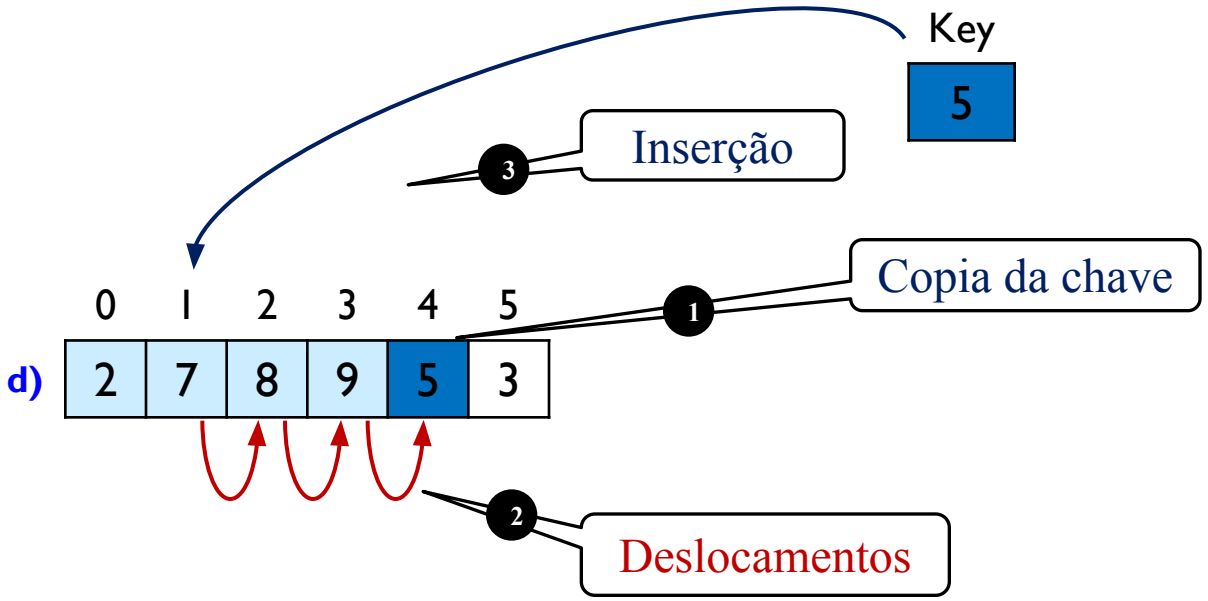
Ordenação por Inserção (Insertion Sort) Descrição do Algoritmo

- O algoritmo por inserção realiza em cada passagem a inserção de um elemento, marcado em azul na figura, na posição certa de uma sequência previamente ordenada no vetor.



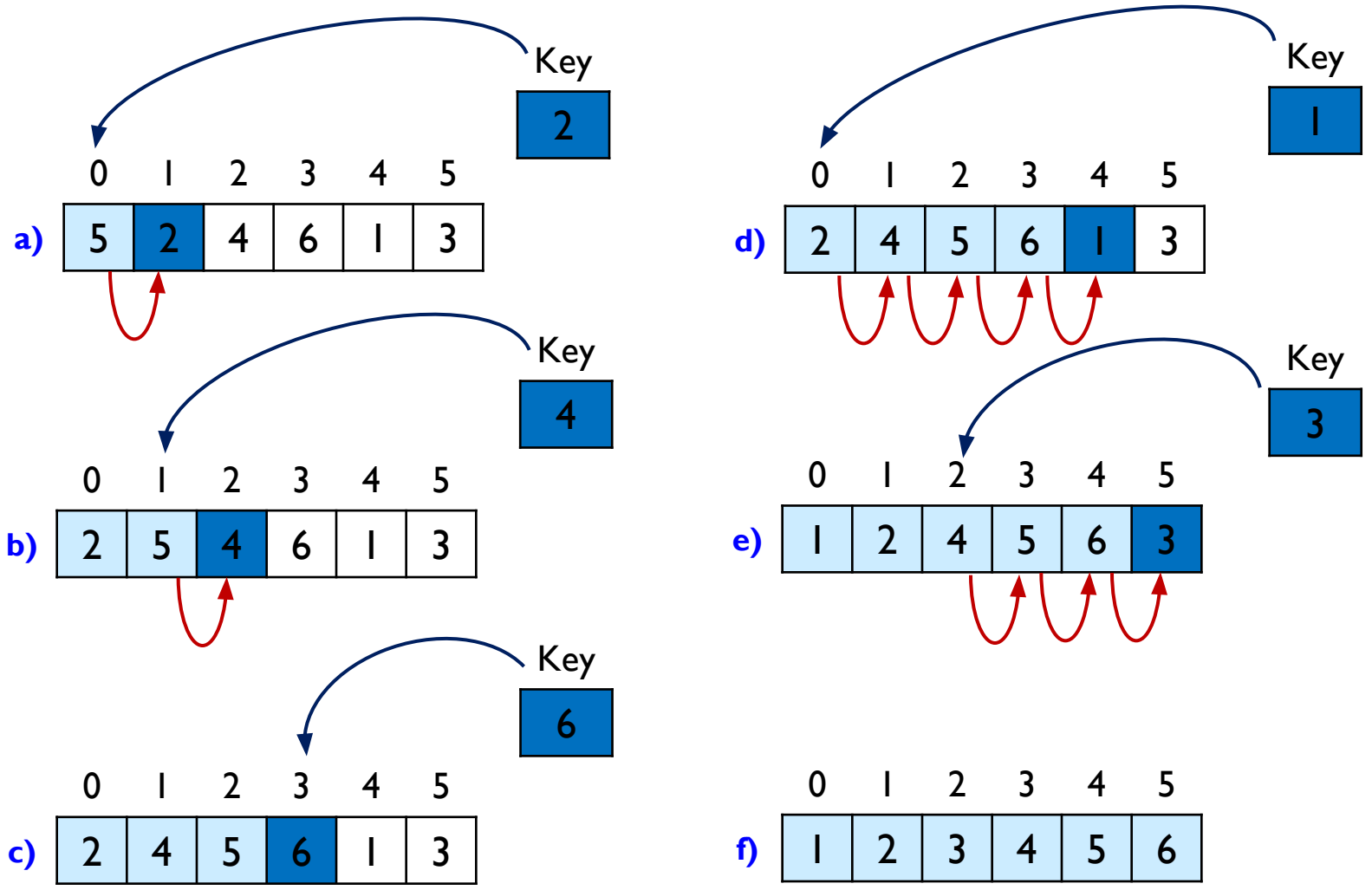
Ordenação por Inserção (Insertion Sort) Exemplo de uma iteração

- Na figura, o elemento a ser inserido, marcado em azul, é armazenado temporariamente na **variável auxiliar** Key, de maneira a permitir os deslocamentos.



Ordenação por Inserção (Insertion Sort)

Exemplo Completo



Ordenação por Inserção (Insertion Sort)

Complexidade do Algoritmo – Comparações

- O método de inserção realiza dois tipos de operações: comparações e deslocamentos (cópias).
- Na primeira passagem, o método faz **no máximo** uma comparação. Na segunda passagem, o método faz no máximo duas comparações. Assim, sucessivamente até que na última passagem, o método faz no máximo N-1 comparações.
- No pior caso, o método realizará:

$$1+2+\ldots+(N-2)+(N-1) = \frac{N(N-1)}{2} \text{ comparações}$$

- O número de cópias é aproximadamente o mesmo que o número de comparações. Contudo, uma copia, não é tão demorada quanto uma troca. O que representa uma vantagem com relação ao método de bolha.

Ordenação por Inserção (Insertion Sort)

Complexidade do Algoritmo – Comparações e trocas

- O número de comparações dependerá dos valores contidos no vetor, existem três casos a considerar:
  - **Pior caso.-** o vetor se encontra ordenado em ordem inverso (descendente) resultando em N*(N-1)/2 comparações e copias. A ordenação **executa em tempo O(N^2)**
  - **Caso Médio.-** acontece com dados aleatórios, neste caso, metade das comparações e copias são realizadas resultando em N*(N-1)/4 comparações e trocas. A ordenação **executa em tempo O(N^2).**
  - **Melhor Caso.-** os dados já se encontram ordenados ou quase ordenados. Neste caso, o loop interno não é executado, já que a condição nunca será satisfeita. O laço externo executa N-1 vezes A ordenação **executa em tempo O(N).**

Algoritmos Elementares
Comparação de Complexidade

- A comparação detalhada entre os três algoritmos elementares:

	Método Bolha		**Método Seleção**		**Método Inserção**	
Caso	Comp.	Trocas	Comp.	Trocas	Comp.	Copias
Pior	$(N^2-N)/2$	$(N^2-N)/2$	$(N^2-N)/2$	N	$(N^2-N)/2$	$(N^2-N)/2$
Médio	$(N^2-N)/2$	$(N^2-N)/4$	$(N^2-N)/2$	N	$(N^2-N)/4$	$(N^2-N)/4$
Melhor	$(N^2-N)/2$	0	$(N^2-N)/2$	N	N-1	0

Algoritmos Elementares
Comparação de Complexidade

- No pior caso, a comparação numérica dos três algoritmos:

	Método Bolha		**Método Seleção**		**Método Inserção**	
Número Elem.	$(N^2-N)/2$ Comp.	$(N^2-N)/2$ Trocas	$(N^2-N)/2$ Comp.	N Trocas	$(N^2-N)/2$ Comp.	$(N^2-N)/2$ Copias
10	45	45	45	10	45	45
100	4.950	4.950	4.950	100	4.950	4.950
1000	499.500	499.500	499.500	1000	499.500	499.500
5000	12.497.500	12.497.500	12.497.500	5000	12.497.500	12.497.500

Algoritmos Elementares
Comparação de Complexidade

- No caso médio, a comparação numérica dos três algoritmos:

	Método Bolha		**Método Seleção**		**Método Inserção**	
Número Elem.	$(N^2-N)/2$ Comp.	$(N^2-N)/4$ Trocas	$(N^2-N)/2$ Comp.	N Trocas	$(N^2-N)/4$ Comp.	$(N^2-N)/4$ Copias
10	45	23	45	10	23	23
100	4.950	2.475	4.950	100	2.475	2.475
1000	499.500	249.750	499.500	1000	249.750	249.750
5000	12.497.500	6.248.750	12.497.500	5000	6.248.750	6.248.750

Algoritmos Elementares
Comparação de Complexidade

- No melhor caso, a comparação numérica dos três algoritmos: :

	Método Bolha		Método Seleção		Método Inserção	
Número Elem.	$(N^2-N)/2$ Comp.	0 Trocas	$(N^2-N)/2$ Comp.	N Trocas	N-1 Comp.	0 Copias
10	45	0	45	10	9	0
100	4.950	0	4.950	100	99	0
1000	499.500	0	499.500	1000	999	0
5000	12.497.500	0	12.497.500	5000	4999	0

Algoritmos Elementares
Comparação de Complexidade

- Em termos de complexidade:

	Método Bolha	**Método Seleção**	**Método Inserção**
Caso	Complex. tempo	Complex. Tempo	Complex. tempo
Pior	$O(N^2)$	$O(N^2)$	$O(N^2)$
Médio	$O(N^2)$	$O(N^2)$	$O(N^2)$
Melhor	$O(N^2)$	$O(N^2)$	$O(N)$

Referências

- Thomas **Cormen**, Charles **Leiserson,** et al.**.** Algoritmos. Teoria e Prática. 2ª Edição. 2002.
- Robert **Lafore**. Estruturas de Dados e Algoritmos em Java. Editora Ciencia Moderna. 2ª Edição. 2004.