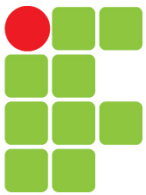
Ordenação por inserção

Programação de computadores II

Prof. Renan Augusto Starke

Instituto Federal de Santa Catarina – IFSC
Campus Florianópolis
renan.starke@ifsc.edu.br

17 de junho de 2016



INSTITUTO FEDERAL
SANTA CATARINA

Ministério da Educação
Secretaria de Educação Profissional e Tecnológica
INSTITUTO FEDERAL DE SANTA CATARINA

Tópicos da aula

Tópico

Objetivos

- Entender alguns fundamentos matemáticos relacionados com algoritmos de ordenação
- Aprender as ordenações por inserção
  - Inserção direta
  - Inserção binária
- Aplicar ordenação nas estruturas de dados conhecidas

Tópico

Ordenação por inserção

Ordenação por inserção

Compreende em algoritmos onde a ordenação é realizada por *inserção* de elementos.

Algoritmos mais conhecidos:

- *Inserção direta*
- *Inserção binária*

Ordenação por inserção

Um algoritmo por inserção recebe uma sequencia $S = \{s_1, s_2, ...s_3\}$ desordenada e computa uma *serie* de sequencias ordenadas $S'_0, S'_1, S'_2, ...S'_n$ da seguinte forma:

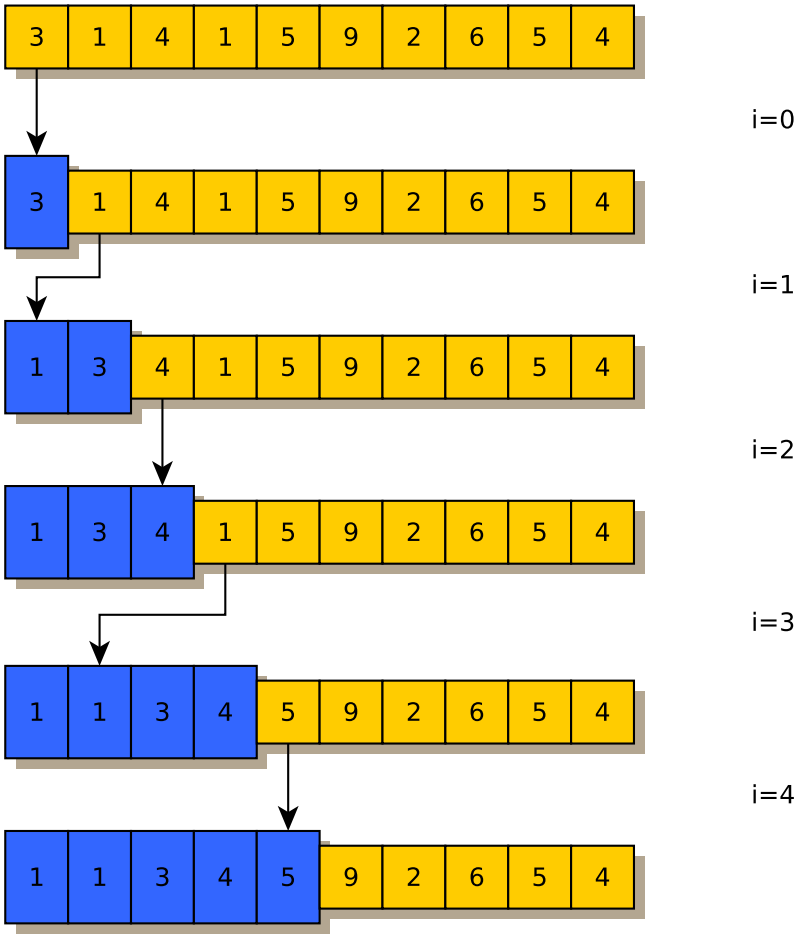
1. A primeira sequencia na série, S'_0, é vazia: $S_0 = \{\}$ ou $|S'_0| = 0$.
2. Dada uma sequencia S'_i, da série, para $0 \leq i < n$, o próximo elemento da série, S'_{i+1}, é obtido pela inserção do $(i+1)$-ésimo elemento da sequencia não ordenada S_{i+1} na posição correta em S'_i.

Cada elemento S'_i, $0 \leq i < n$, contém os primeiros i elementos da sequencia não ordenada S. Assim, o final da sequencia na série, S'_n, é a sequencia que procuramos, isto é $S' = S'_n$.

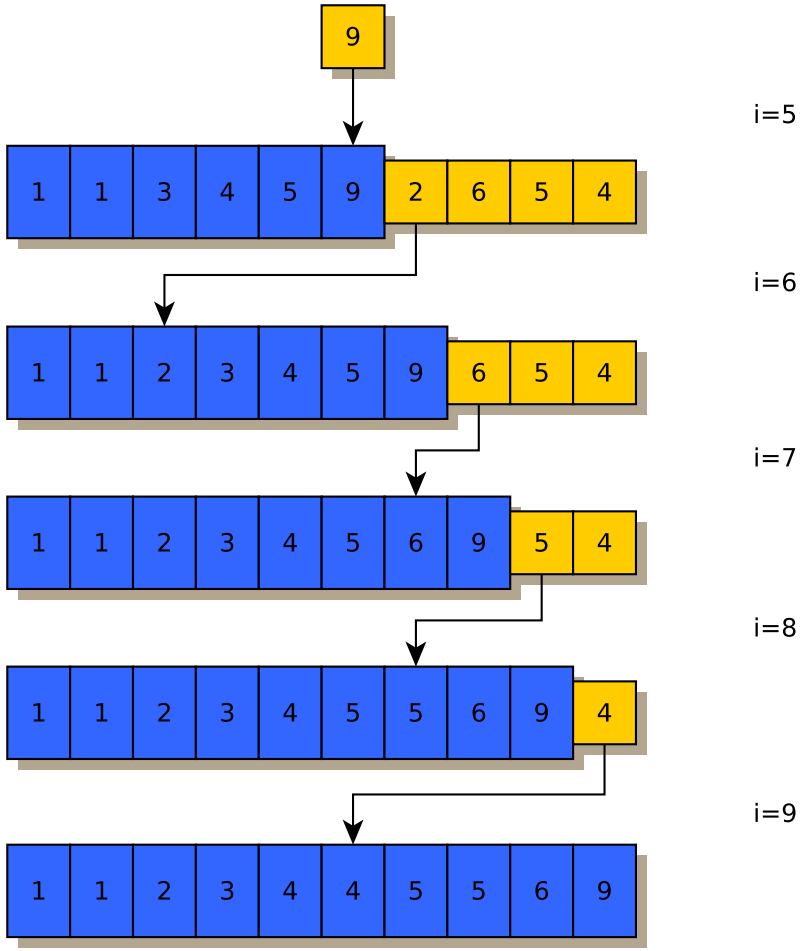
Ordenação por inserção

1. Na ordenação por inserção:
   - Nova sequencia, S', **pode ser um novo vetor ou lista.**
   - Nova sequencia, S', **pode ser o mesmo vetor** S**.**

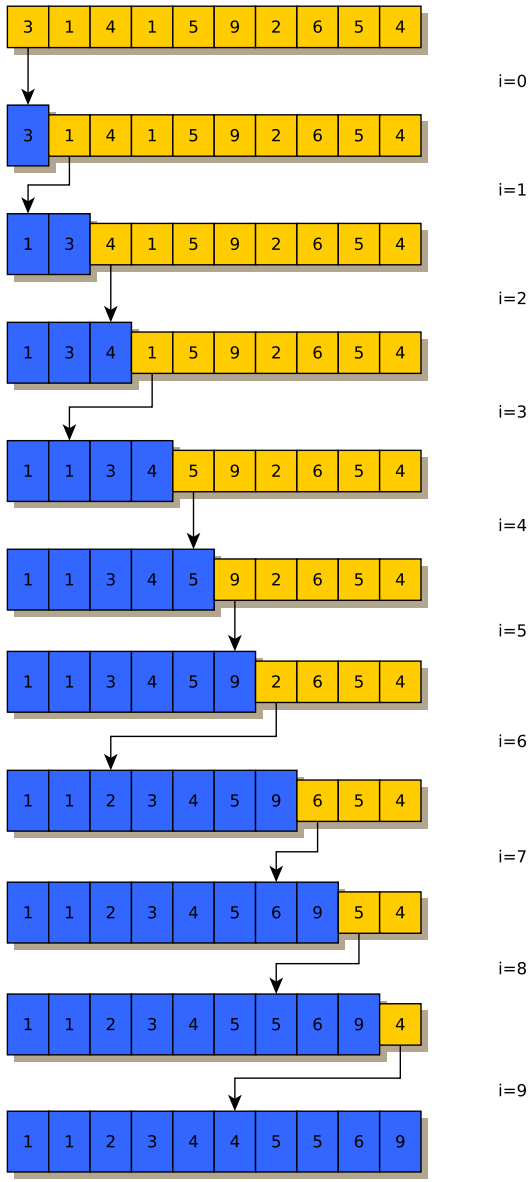
Exemplo


3 1 4 1 5 9 2 6 5 4
i=0
3 1 4 1 5 9 2 6 5 4
i=1
1 3 4 1 5 9 2 6 5 4
i=2
1 3 4 1 5 9 2 6 5 4
i=3
1 1 3 4 5 9 2 6 5 4
i=4
1 1 3 4 5 9 2 6 5 4

Exemplo


9
i=5
1 1 3 4 5 9 2 6 5 4
i=6
1 1 2 3 4 5 9 6 5 4
i=7
1 1 2 3 4 5 6 9 5 4
i=8
1 1 2 3 4 5 5 6 9 4
i=9
1 1 2 3 4 4 5 5 6 9

Exemplo


3 1 4 1 5 9 2 6 5 4
i=0
3 1 4 1 5 9 2 6 5 4
i=1
1 3 4 1 5 9 2 6 5 4
i=2
1 3 4 1 5 9 2 6 5 4
i=3
1 1 3 4 5 9 2 6 5 4
i=4
1 1 3 4 5 9 2 6 5 4
i=5
1 1 3 4 5 9 2 6 5 4
i=6
1 1 2 3 4 5 9 6 5 4
i=7
1 1 2 3 4 5 6 9 5 4
i=8
1 1 2 3 4 5 5 6 9 4
i=9
1 1 2 3 4 4 5 5 6 9

Algoritmo

```
1  insertsort(int array[], int n)
2  {
3    int i, j;
4
5    for (i = 1; i < n; i++) {
6
7      for (j = i; j > 0 && array[j-1] > array[j]; j--)
8        swap( array[j], array[j - 1] );
9  }
```

Complexidade

- Pior caso: $O(n^2)$
- Caso médio: $O(n^2)$
- Melhor caso: $O(n)$

Inserção binária

- A inserção direta que vimos faz uma busca linear para encontrar a posição que queremos inserir o novo elemento
- Vamos inserir o elemento em uma sequencia que está ordenada, certo?
- Logo, podemos utilizar uma busca binária ao invés da linear para descobrir a posição
- Complexidade das buscas no pior caso:
  - Linear $O(n)$
  - Binária $O(\log n)$

Algoritmo

```
1  insertsort(int array[], int n)
2  {
3    int i, tmp, meio;
4
5    for (i = 1; i < n; i++) {
6
7      tmp = array[i];
8      esq = 0;
9      dir = i;
10
11     while (esq < dir) {
12
13       meio = (esq + dir) / 2;
14
15       if (tmp >= array[meio])
16         esq = meio + 1;
17       else
18         dir = meio;
19      }
20
21      for (j = i; j > esq; j--)
22       swap(array[j - 1], j);
23
24 }
```

Complexidade

- Pior caso: $O(n^2)$
- Caso médio: $O(n^2)$
- Melhor médio: $O(n \log n)$

Tópico

Exercícios

- Implemente o ordenação por inserção para um vetor de inteiros.
  - Teste seu algoritmo para um vetor de 10.000 de elementos alocados dinamicamente.
  - Inicialize-o com números aleatórios.
  - Meça o tempo de execução para 20 execuções.
  - Calcule o tempo de execução médio.
  - **OBS: números aleatórios gerados por:** srand() alimentando a semente com srand (getpid() ^time(NULL));
- Estenda a implementação da lista duplamente encadeada com uma função de ordenação por inserção.
- Compare com o tempo de execução do *BubbleSort*
- Compare com o tempo de execução do *QuickSort*