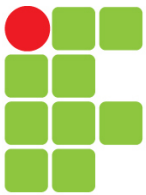
Fundamentos de ordenação e ordenação por troca

Programação de computadores II

Prof. Renan Augusto Starke

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

18 de outubro 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 troca
- Conhecer o *Bubble Sort*
- Aplicar ordenação nas estruturas de dados conhecidas

Tópico

Fundamentos de ordenação

- Considere uma sequencia arbitrária $S = \{s_1, s_2, s_3, ...s_n\}$ composta por $n \geq 0$ elementos formados de um conjunto universal U.

Ordenação

Ordenar significa rearranjar os elementos de S para produzir uma nova sequencia, S', onde os elementos aparecem *em ordem*.

- O que significa os elementos de S' estarem em ordem?
- Vamos assumir que há uma relação, $<$, definida sobre o universo U.
- A relação $<$ deve ser uma *ordem total*.

Fundamentos de ordenação

Definição

A relação de **ordem total**, $<$, *é definida em um conjunto universo* U *com as seguintes propriedades:*

1. *Para todos os pares* $(i, j) \in U \times U$, *exatamente uma das seguintes preposições é verdadeira:* $i < j$, $i = j$ *ou* $j < i$.
   (Todos os elementos são comensuráveis).
2. *Para todas as triplas* $(i, j, k) \in U \times U \times U$:
   $i < j \wedge j < k \Leftrightarrow i < k$.
   (A relação $<$ *é transitiva).*

- Para ordenar os elementos de uma sequencia S, determina-se as *permutações* $P = \{p_1, p_2, ..., p_n\}$ de elementos de S onde:

$s_{p_1} \leq s_{p_2} \leq s_{p_3} \leq ... \leq s_{p_n}$

Fundamentos de ordenação

- Na prática não estamos interessados exatamente nas permutações P.
- Nosso objetivo é calcular a sequencia ordenada $S' = \{s'_1, s'_2, s'_3, ...s'_n\}$ onde:

  $s'_i = s_{p_i}$ para todo $1 \leq i \leq n$.

- Às vezes a sequencia S contém duplicatas, há valores de i e j, $1 \leq i < j \leq n$ tal que $s_i = s_j$.
- Quando a sequencia possui duplicatas não há garantias que as duplicatas mantenham suas posições relativas originais.
- Se as posições relativas originais são mantidas, a ordenação é conhecida como **estável**.

Fundamentos de ordenação

- Há inúmeros algoritmos de ordenação ou classificação (*sorting*):
  - Ordenação por troca
  - Ordenação por seleção
  - Ordenação por árvore
  - Ordenação por inserção
  - Ordenação por intercalação

- A diferença entre eles está relacionada com:
  - Eficiência
  - Implementação
  - Aplicação

Tópico

Ordenação por troca

Ordenação por troca

Compreende em algoritmos onde a ordenação é realizada por *trocas entre pares* de elementos.

- Troca entre elementos adjacentes
- Troca entre elementos mais distantes

Algoritmos mais conhecidos:

- *Bubble Sort*
- *Quick Sort*

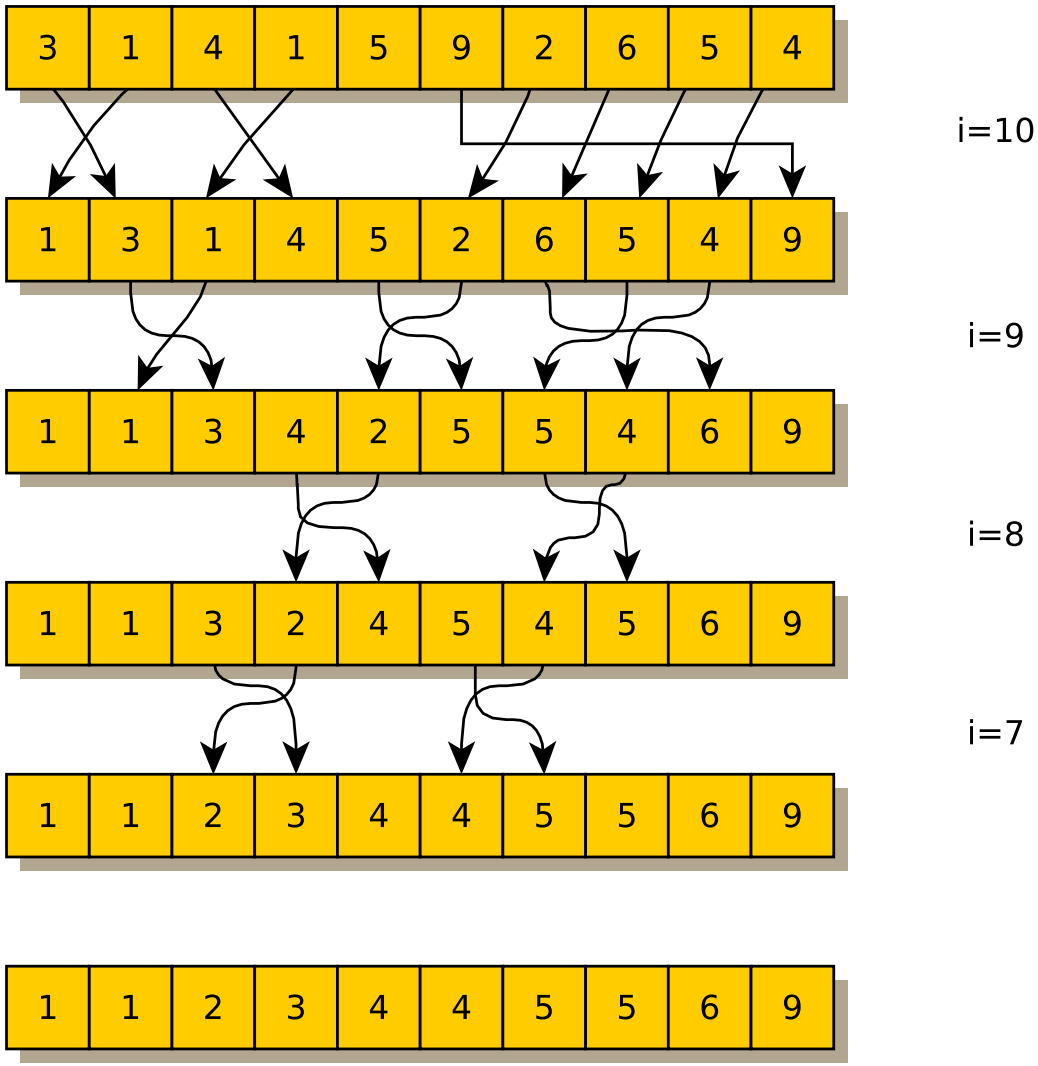
Bubble Sort

Bubble Sort

Uma sequencia $S = \{s_1, s_2, ...s_3\}$ é ordenada realizando $n - 1$ passos pelos dados. Em cada passo, elementos adjacentes são comparados e trocados (*swapped*) se necessário.

- *Bubble sort* é o algoritmo mais conhecido e mais simples para ordenação.
- Note que no primeiro passo, o maior elemento é *bubble up* (borbulhado) para a última posição.
- No geral, após k passos pelos dados, os últimos k elementos na posição correta não precisam ser mais considerados.

Bubble Sort


3 1 4 1 5 9 2 6 5 4
i=10
1 3 1 4 5 2 6 5 4 9
i=9
1 1 3 4 2 5 5 4 6 9
i=8
1 1 3 2 4 5 4 5 6 9
i=7
1 1 2 3 4 4 5 5 6 9
1 1 2 3 4 4 5 5 6 9

Bubble Sort

```
1     @  for (i = n; i > 1; i--)
2          for (j = 0; j < i - 1; j++)
3             if (array[j] > array[j + 1])
4               swap(j, j + 1);
```

- Laço externo: é exec. por $n-1$ iterações no total.
- Cada i iteração do laço externo provoca $i-1$ iterações do laço interno.

Total número de iterações é:

$$\sum_{i=2}^{n}(i-1) = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} \tag{1}$$

Portanto, *Bubble Sort* é $O(n^2)$.

Bubble Sort

- Qual o pior caso do *Bubble Sort*?

Tópico

Exercícios

- Implemente o *Bubble Sort* para um vetor de inteiros.
  - Teste seu algoritmo para um vetor de 100.000 de elementos alocados dinamicamente.
  - Inicialize-o com números aleatórios.
  - Meça o tempo de execução para 20 execuções do *Bubble Sort*.
  - Calcule o tempo de execução médio.
  - Pesquise como medir o tempo de execução de partes de um programa em C.
- Estenda a implementação da lista duplamente encadeada com uma função de ordenação por *Bubble Sort*.