Algoritmos - Aula 3

Fernando Raposo

Vamos ver

- Ordenação
- Algoritmos de Ordenação

- Bubble sort
- Insertion sort

Ordenação

- (Re) Organizar conjunto de elementos de **uma forma específica**;
- Esta reordenação visa facilitar a recuperação futura dos dados;
  - Pode ser **numérica**
    - 3,8,9,7,10,15,0 -> 0,3,7,8,9,10,15
  - Pode ser **alfabética** (lexicográfica)
    - Duas palavras são lexicograficamente iguais se:
      1. Tem o mesmo comprimento;
      2. Possuem os mesmos caracteres nas mesmas posições.

Ordenação (cont…)

- Comparação Lexicográfica (Algoritmo)
  - Para determinar qual palavra vem na frente analise os caracteres da **esquerda** para a **direita**;
  - O primeiro caractere das duas que for diferente determinará qual irá primeiro;
  - Os caracteres são comparados utilizando a tabela Unicode;
    - Obs: Todas as letras maiúsculas vem antes das minúsculas.

Java

	Relação		`palavraA.compareTo(palavraB)`
`palavraA`	Menor que	`palavraB`	Número negativo
`palavraA`	Igual	`palavraB`	zero
`palavraA`	Maior que	`palavraB`	Número positivo

Comparação Lexicográfica

- Quem é maior que quem?
  - “Casa” ou “casa”
  - “universidade” ou “universidadE”
  - “TRABALHO” ou “trabalho”

  Parte Prática

- Vamos fazer alguns testes em JAVA usando o **`compareTo`**

Bubble Sort

- Vantagem: Simples de implementar;
- Ordenação por “flutuação”. Percorremos o array e fazemos com que os elementos maiores flutuem para o topo.
- Estratégia para array de tamanho *n*:
  - Percorrer os elementos do array par a par;
  - Se o elemento da esquerda é maior que o da direita, trocar de posição;
  - Repita os dois passos acima (n - 1) vezes, depois (n-2) vezes e assim por diante...

Bubble Sort

Bubble Sort

6 > 5?

Bubble Sort

6 > 5? Sim, trocar de posição..

Bubble Sort

Trocado!

Bubble Sort

6 > 7?

Bubble Sort

6 > 7?, Não! Manter posição...

Bubble Sort

7 > 9?

Bubble Sort

7 > 9? Não, Manter posição!

Bubble Sort

9 > 8?

Bubble Sort

9 > 8? Sim! Trocar posição!

Bubble Sort

Trocado!

Bubble Sort

9 > 0?

Bubble Sort

9 > 0? Sim!, Trocar posição!

Bubble Sort

Trocado!

Bubble Sort

9 Está Ordenado!, podemos marcá-lo.

Bubble Sort

9 Está Ordenado!, podemos marcá-lo.

Bubble Sort

Vamos percorrer n-1 agora...

Bubble Sort

5 > 6?

Bubble Sort

5 > 6? Não, manter posição!

Bubble Sort

6 > 7?

Bubble Sort

6 > 7? Não, manter posição!

Bubble Sort

7 > 8?

Bubble Sort

7 > 8? Não, manter posição!

Bubble Sort

8 > 0?

Bubble Sort

8 > 0? Sim! Trocar posição.

Bubble Sort

Trocado!

Bubble Sort

8 Está Ordenado!, podemos marcá-lo.

Bubble Sort

Marcado!

Bubble Sort

Repetindo para (n-2)

Bubble Sort

5 > 6?

Bubble Sort

5 > 6? Não! Manter posição...

Bubble Sort

6 > 7?

Bubble Sort

6 > 7? Não! Manter posição...

Bubble Sort

7 > 0?

Bubble Sort

7 > 0? Sim! Trocar de posição...

Bubble Sort

Trocado!

Bubble Sort

7 está ordenado, podemos marcá-lo.

Bubble Sort

Marcado!

Bubble Sort

5 > 6?

Bubble Sort

5 > 6? Não! Manter posição...

Bubble Sort

6 > 0?

Bubble Sort

6 > 0? Sim! Trocar posição!

Bubble Sort

Trocado!

Bubble Sort

6 está ordenado, podemos marcá-lo.

Bubble Sort

Marcado!

Bubble Sort

5 > 0?

Bubble Sort

5 > 0? Sim! Trocar posição!

Bubble Sort

Trocado!

Bubble Sort

5 está ordenado, podemos marcá-lo.

Bubble Sort

Marcado!

Bubble Sort

0 está ordenado, podemos marcá-lo.

Bubble Sort

Marcado! E fim do Algoritmo.

Análise de Complexidade Bubble Sort

- No melhor caso: O(n)
  - Ocorre quando o array já está ordenado;
  - Como assim O(n)?
    - Percorremos todo um array, n-1 posições, sem fazer nenhuma troca de posições;
    - Na notação Big-O, desconsideramos o (-1), dizemos O(n).

Análise de Complexidade Bubble Sort

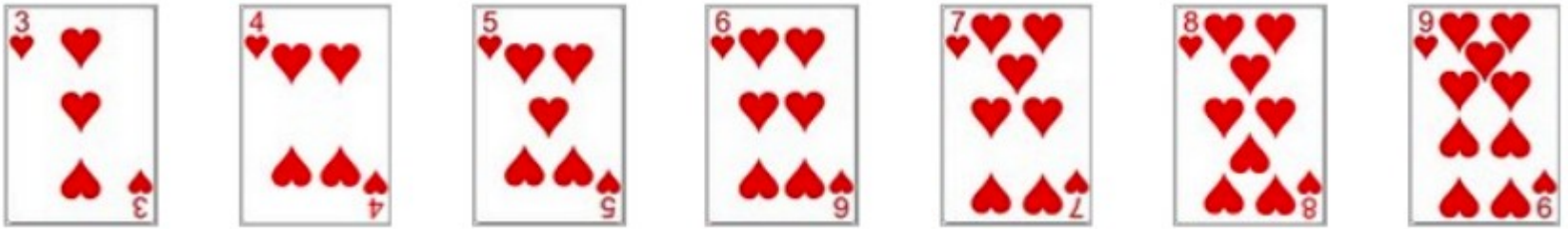
- No pior caso e caso médio: $O(n^2)$
  - O pior caso acontece quando o array está na ordem reversa;
  - Primeiro (n-1) comparações, depois (n-2), …

[4,3,2,1]
[3,4,2,1]
[3,2,4,1]
[3,2,1,4] → 4 foi para posição n-1
[2,3,1,4]
[2,1,3,4] → 3 foi para posição n-2
[1,2,3,4] → 2 foi para posição n-3

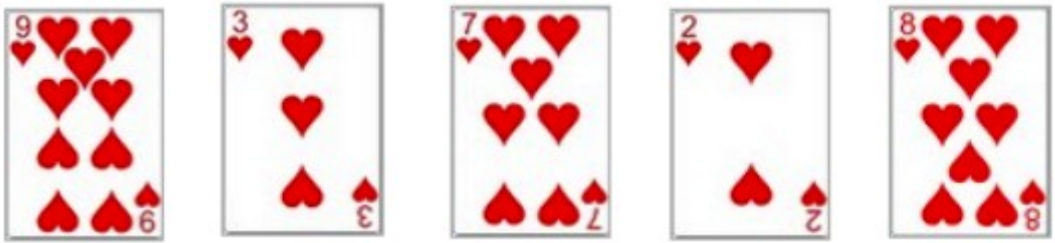
- Fizemos (n-1) loops para chegar à ordenação;
- Fizemos n(n-1)/2 comparações, ou $n^2/2-n/2$. Na notação Big-O: (n^2)

Insertion Sort

- Funciona da mesma forma com que organizamos um baralho de cartas;



Insertion Sort

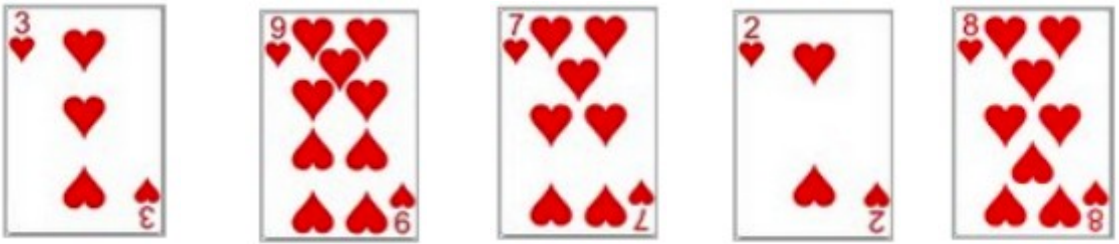


Começamos do segundo elemento, 3 < 9?, então trocar...

Insertion Sort

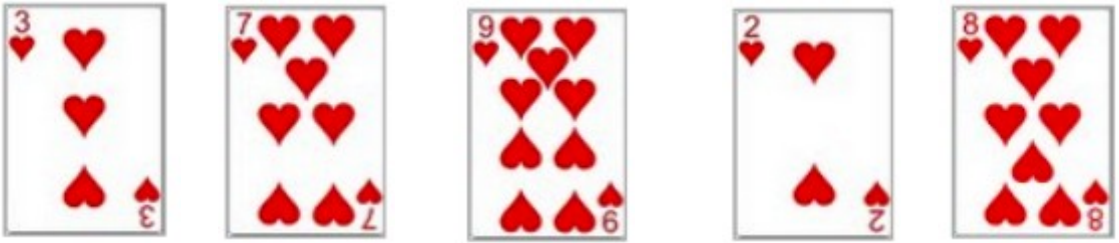
Trocado!

Insertion Sort



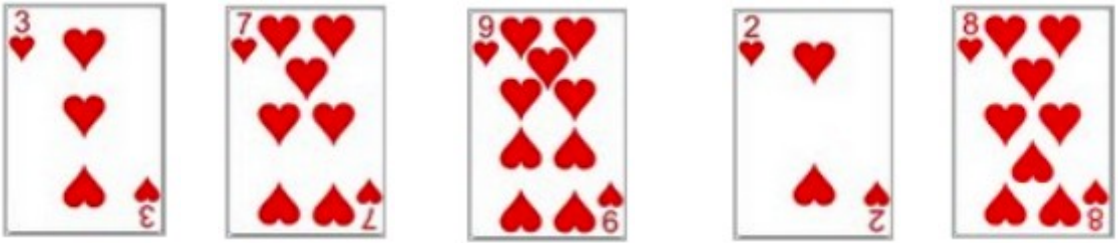
7 < 9? Sim! E é maior que 3 então trocar.

Insertion Sort



Trocado!

Insertion Sort



2 < 9? Sim! Mas é menor que 3 e 7. Então colocá-lo na frente.

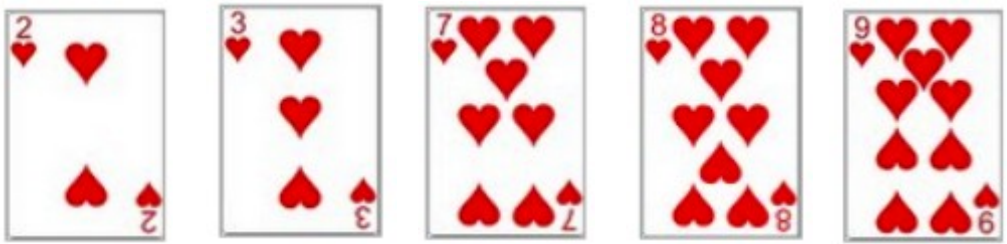
Insertion Sort

Trocado!

Insertion Sort

8 < 9? Sim! e é maior que 7, coloca-lo após este.

Insertion Sort



Ordenado!

Análise de Complexidade Insertion Sort

- No melhor caso: O(n)
- No pior caso: $O(n^2)$

Mas… a complexidade dele parece similar à do Bubble Sort, ele vale a pena?

- Existe alguma vantagem sob o Bubble Sort…
- Se você souber que o array está um pouco ordenado, ele fica mais rápido que o Bubble sort: O(n)


Thank You!