Proz	**Material Complementar**
	Professor: Gleison Brito
Turma: BHZ231AMTDS	
Ordenação	***gleison.batista@prozeducacao.com.br***

Algoritmos de Ordenação

Bubblesort

O algoritmo Bubblesort é uma das abordagens mais simplistas para a ordenação de dados. A ideia básica consiste em percorrer o vetor diversas vezes, em cada passagem, fazendo flutuar para o topo da lista (posição mais à direita possível) o maior elemento da sequência. Para um vetor com N elementos, esse algoritmo realiza NxN operações. Portanto, diz-se que a complexidade do método é de ordem quadrática. Por essa razão, ele **NÃO** é recomendado para programas que precisem de velocidade e operem com quantidade elevada de dados.

O exemplo a seguir mostra o passo a passo necessário para a ordenação do seguinte vetor [5, 2, 13, 7, -3, 4, 15, 10, 1, 6]

1ª passagem (levar maior elemento para última posição)

[5, 2, 13, 7, -3, 4, 15, 10, 1, 6]	em amarelo, não troca
[2, 5, 13, 7, -3, 4, 15, 10, 1, 6]	em azul, troca
[2, 5, 7, 13, -3, 4, 15, 10, 1, 6]	
[2, 5, 7, -3, 13, 4, 15, 10, 1, 6]	
[2, 5, 7, -3, 4, 13, 15, 10, 1, 6]	
[2, 5, 7, -3, 4, 13, 15, 10, 1, 6]	
[2, 5, 7, -3, 4, 13, 10, 15, 1, 6]	
[2, 5, 7, -3, 4, 13, 10, 1, 15, 6]	
[5, 2, 7, -3, 4, 13, 10, 1, 6, 15]	

2ª passagem (levar segundo maior para penúltima posição)

[2, 5, 7, -3, 4, 13, 10, 1, 6, 15]
[2, 5, 7, -3, 4, 13, 10, 1, 6, 15]
[2, 5, -3, 7, 4, 13, 10, 1, 6, 15]
[2, 5, -3, 4, 7, 13, 10, 1, 6, 15]
[2, 5, -3, 4, 7, 13, 10, 1, 6, 15]
[2, 5, -3, 4, 7, 10, 13, 1, 6, 15]
[2, 5, -3, 4, 7, 10, 1, 13, 6, 15]
[2, 5, -3, 4, 7, 10, 1, 6, 13, 15]

3ª passagem (levar terceiro maior para antepenúltima posição)

[2, 5, -3, 4, 7, 10, 1, 6, 13, 15]
[2, -3, 5, 4, 7, 10, 1, 6, 13, 15]
[2, -3, 4, 5, 7, 10, 1, 6, 13, 15]
[2, -3, 4, 5, 7, 10, 1, 6, 13, 15]
[2, -3, 4, 5, 7, 10, 1, 6, 13, 15]
[2, -3, 4, 5, 7, 1, 10, 6, 13, 15]
[2, -3, 4, 5, 7, 1, 6, 10, 13, 15]

4ª passagem

[-3, 2, 4, 5, 7, 1, 6, 10, 13, 15]
[-3, 2, 4, 5, 7, 1, 6, 10, 13, 15]
[-3, 2, 4, 5, 7, 1, 6, 10, 13, 15]
[-3, 2, 4, 5, 7, 1, 6, 10, 13, 15]
[-3, 2, 4, 5, 1, 7, 6, 10, 13, 15]
[-3, 2, 4, 5, 1, 6, 7, 10, 13, 15]

5ª passagem

[-3, 2, 4, 5, 1, 6, 7, 10, 13, 15]
[-3, 2, 4, 5, 1, 6, 7, 10, 13, 15]
[-3, 2, 4, 5, 1, 6, 7, 10, 13, 15]
[-3, 2, 4, 1, 5, 6, 7, 10, 13, 15]
[-3, 2, 4, 1, 5, 6, 7, 10, 13, 15]

6ª passagem

[-3, 2, 4, 1, 5, 6, 7, 10, 13, 15]
[-3, 2, 4, 1, 5, 6, 7, 10, 13, 15]
[-3, 2, 1, 4, 5, 6, 7, 10, 13, 15]
[-3, 2, 1, 4, 5, 6, 7, 10, 13, 15]

7ª passagem

[-3, 2, 1, 4, 5, 6, 7, 10, 13, 15]
[-3, 1, 2, 4, 5, 6, 7, 10, 13, 15]
[-3, 1, 2, 4, 5, 6, 7, 10, 13, 15]

8ª passagem

[-3, 1, 2, 4, 5, 6, 7, 10, 13, 15]
[-3, 1, 2, 4, 5, 6, 7, 10, 13, 15]

9ª passagem

[-3, 1, 2, 4, 5, 6, 7, 10, 13, 15]

Insertion Sort

Insertion Sort, ou ordenação por inserção, é o algoritmo de ordenação que, dado um vetor inicial constrói um vetor final com um elemento de cada vez, uma inserção por vez. É bastante eficiente para problemas com pequenas entradas, sendo o mais eficiente entre os algoritmos desta ordem de classificação. Podemos fazer uma comparação do Insertion sort com o modo de como algumas pessoas organizam um baralho num jogo de cartas. Imagine que você está jogando cartas. Você está com as cartas na mão e elas estão ordenadas. Você recebe uma nova carta e deve colocá-la na posição correta da sua mão de cartas, de forma que as cartas obedeçam a ordenação. A cada nova carta adicionada a sua mão de cartas, a nova carta pode ser menor que algumas das cartas que você já tem na mão ou maior, e assim, você começa a comparar a nova carta com todas as cartas na sua mão até encontrar sua posição correta. Você insere a nova carta na posição correta, e, novamente, sua mão é composta de cartas totalmente ordenadas. Então, você recebe outra carta e repete o mesmo procedimento. Então outra carta, e outra, e assim por diante, até você não receber mais cartas. Esta é a ideia por trás da ordenação por inserção. Percorra as posições do vetor, começando com o índice um. Cada nova posição é como a nova carta que você recebeu, e você precisa inseri-la no lugar correto na fatia do vetor ordenado à esquerda da posição do elemento que se deseja inserir na posição correta da fatia ordenada.

O exemplo a seguir ilustra o funcionamento do algoritmo em um exemplo passo a passo. Suponha o seguinte vetor de entrada. [5, 2, 13, 7, -3, 4, 15, 10, 1, 6]

1ª passagem: [5, 2, 13, 7, -3, 4, 15, 10, 1, 6] → [2, 5, 13, 7, -3, 4, 15, 10, 1, 6]

2ª passagem: [2, 5, 13, 7, -3, 4, 15, 10, 1, 6] → [2, 5, 13, 7, -3, 4, 15, 10, 1, 6]

3ª passagem: [2, 5, 13, 7, -3, 4, 15, 10, 1, 6] → [2, 5, 7, 13, -3, 4, 15, 10, 1, 6]

4ª passagem: [2, 5, 7, 13, -3, 4, 15, 10, 1, 6] → [-3, 2, 5, 7, 13, 4, 15, 10, 1, 6]

5ª passagem: [-3, 2, 5, 7, 13, 4, 15, 10, 1, 6] → [-3, 2, 4, 5, 7, 13, 15, 10, 1, 6]

6ª passagem: [-3, 2, 4, 5, 7, 13, 15, 10, 1, 6] → [-3, 2, 4, 5, 7, 13, 15, 10, 1, 6]

7ª passagem: [-3, 2, 4, 5, 7, 13, 15, 10, 1, 6] → [-3, 2, 4, 5, 7, 10, 13, 15, 1, 6]

8ª passagem: [-3, 2, 4, 5, 7, 10, 13, 15, 1, 6] → [-3, 1, 2, 4, 5, 7, 10, 13, 15, 6]

9ª passagem: [-3, 1, 2, 4, 5, 7, 10, 13, 15, 6] → [-3, 1, 2, 4, 5, 6, 7, 10, 13, 15]

Os algoritmos Bubble Sort e Insertion Sort se encontram disponíveis em:

https://github.com/gleisonbt/Material_Desenv_Sistemas_Proz/tree/main

Bibliografia:

Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Algoritmos: Teoria e Prática. 3a edição. Elsevier, 2012. ISBN 978853523699