**Definição de Algoritmos**

O Algoritmo é um esquema de resolução de um problema. Pode ser implementado com qualquer sequência de valores ou objetos que tenham uma lógica infinita, ou seja, qualquer coisa que possa fornecer uma sequência lógica.

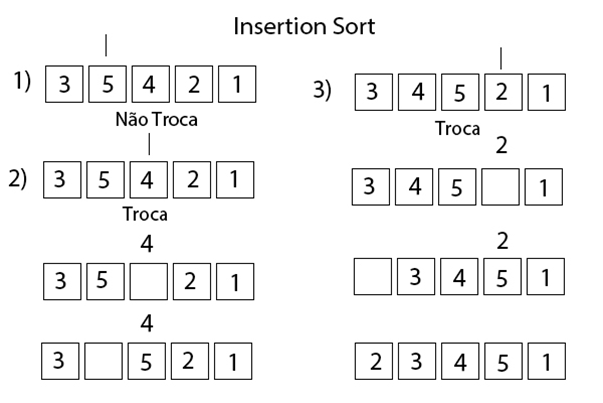
Podemos ilustrar um algoritmo pelo exemplo de uma receita culinária, embora muitos algoritmos sejam mais complexos. Um Algoritmo mostra passo a passo os procedimentos necessários para resolução de um problema.

**Insertion Sort**

Insertion Sort ou ordenação por inserção é o método que percorre um vetor de elementos da esquerda para a direita e à medida que avança vai ordenando os elementos à esquerda. Possui complexidade C(n) = O(n) no melhor caso e C(n) = O(n²) no caso médio e pior caso. É considerado um método de ordenação estável.

Um método de ordenação é estável se a ordem relativa dos itens iguais não se altera durante a ordenação.

O funcionamento do algoritmo é bem simples: consiste em cada passo a partir do segundo elemento selecionar o próximo item da sequência e colocá-lo no local apropriado de acordo com o critério de ordenação.

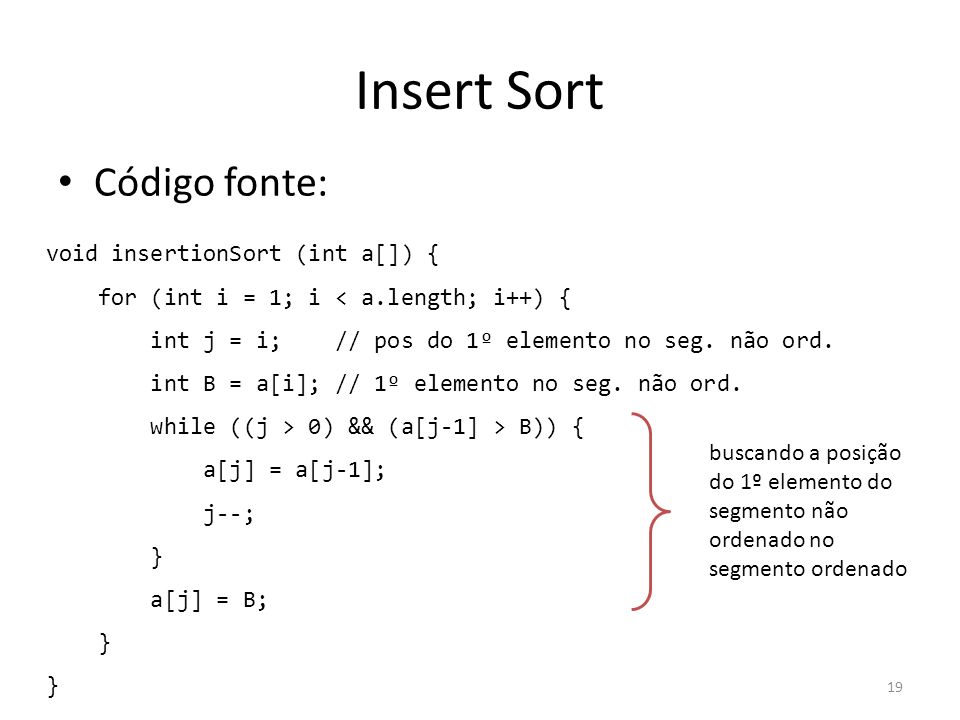


**Esquema de funcionamento do Insertion Sort:**

* Neste passo é verificado se o 5 é menor que o 3, como essa condição é falsa, então não há troca.
* É verificado se o quatro é menor que o 5 e o 3, ele só é menor que o 5, então os dois trocam de posição.
* É verificado se o 2 é menor que o 5, 4 e o 3, como ele é menor que 3, então o 5 passa a ocupar a posição do 2, o 4 ocupa a posição do 5 e o 3 ocupa a posição do 4, assim a posição do 3 fica vazia e o 2 passa para essa posição.

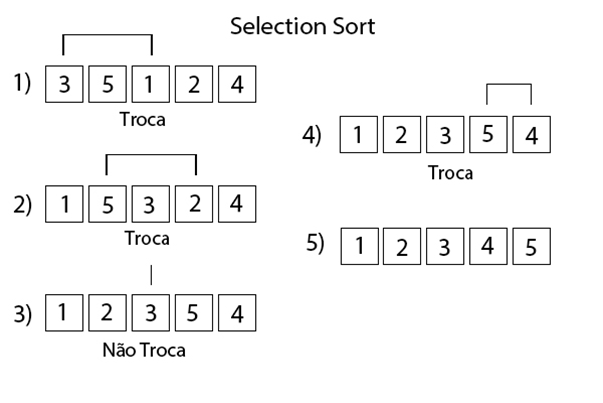
O mesmo processo de comparação acontece com o número 1, após esse processo o vetor fica ordenado.

**Implementação:**



**Selection Sort**

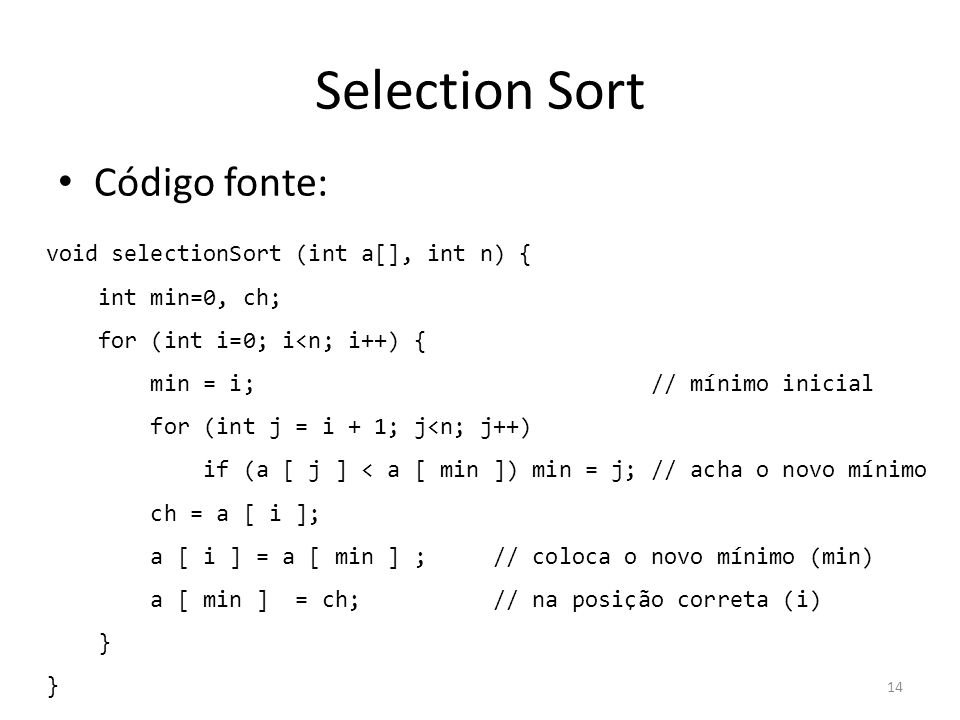
A ordenação por seleção ou selection sort consiste em selecionar o menor item e colocar na primeira posição, selecionar o segundo menor item e colocar na segunda posição, segue estes passos até que reste um único elemento. Para todos os casos (melhor, médio e pior caso) possui complexidade C(n) = O(n²) e não é um algoritmo estável.



**Esquema de funcionamento do Bubble Sort:**

* Neste passo o primeiro número escolhido foi o 3, ele foi comparado com todos os números à sua direita e o menor número encontrado foi o 1, então os dois trocam de lugar.
* O mesmo processo do passo 1 acontece, o número escolhido foi o 5 e o menor número encontrado foi o 2.
* Não foi encontrado nenhum número menor que 3, então ele fica na mesma posição.
* O número 5 foi escolhido novamente e o único número menor que ele à sua direita é o 4, então eles trocam.
* Vetor já ordenado.

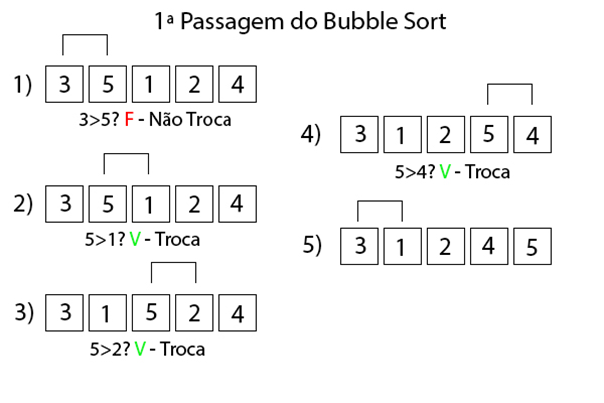
**Implementação:**



**Bubble Sort**

O bubble sort, ou ordenação por flutuação, é um algoritmo de ordenação dos mais simples. A ideia é comparar dois elementos e trocá-los de posição, até que os elementos de maior valor sejam levados para o final do vetor. O processo continua até a ordenação total do vetor lembrando a forma como as bolhas em um tanque de água procuram seu próprio nível, e disso vem o nome do algoritmo.

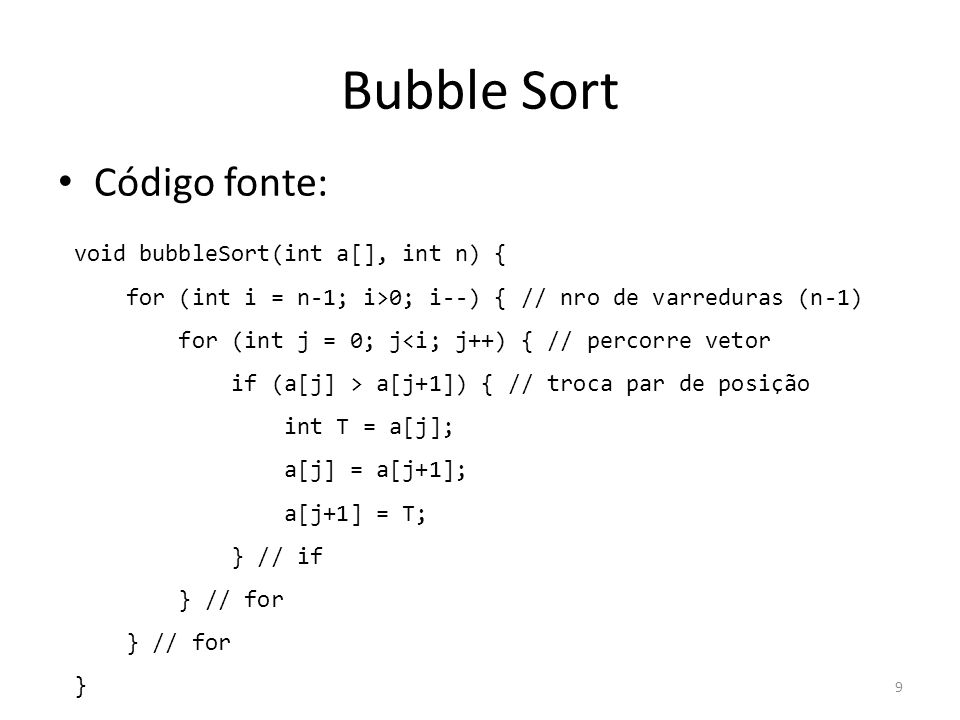
A complexidade desse algoritmo é de ordem quadrática (O(n²)). Por isso, ele não é recomendado para programas que precisem de velocidade e operem com quantidade elevada de dados. Também é necessária uma condição de parada, geralmente uma flag ou variável que armazena se houve troca ou não na passagem. Se uma passagem chega ao seu final sem troca a ordenação cessa.



**Esquema de funcionamento do Bubble Sort:**

* É verificado se o 3 é maior que 5, por essa condição ser falsa, não há troca.
* É verificado se o 5 é maior que 1, por essa condição ser verdadeira, há uma troca.
* É verificado se o 5 é maior que 2, por essa condição ser verdadeira, há uma troca.
* É verificado se o 5 é maior que 4, por essa condição ser verdadeira, há uma troca.
* O método retorna ao início do vetor realizando os mesmos processos de comparações, isso é feito até que o vetor esteja ordenado.

**Implementação:**

****

**Comb Sort**

O comb sort é uma melhoria do bubble sort. O bubble sort sempre compara valores adjacentes. Então todas as inversões são removidas uma por uma. comb sort adicionou uma melhoria em bubble sort, essa melhoria usa um gap de tamanho maior que 1. O gap começa com um valor grande e encolhe por um fator de 1,3 em cada iteração até atingir o valor 1. Assim, comb sort remove mais de uma inversão, fazendo com que tenha um desempenho melhor do que o bublle sort.

O fator de encolhimento foi encontrado na prática como 1,3.

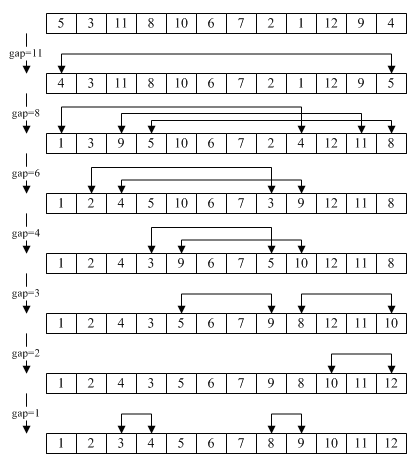
Embora funcione melhor que o bubble sort, em média, o pior caso continua sendo O (n 2).

**Complexidade do Tempo:** A complexidade do pior caso deste algoritmo é

O (n 2) e a complexidade do Melhor Caso é O (n).

**Espaço Auxiliar:** O (1).

**Funcionamento:**



**Implementação:**

