Algoritmos de Ordenação:

Insertion e Bubble Sort

Ana Cardoso
Márcia Saraiva
E.P.G.I

Sumário:

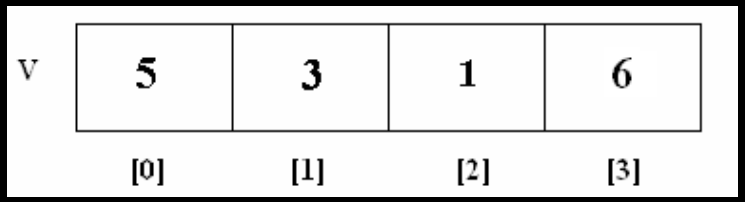
- Algoritmo de ordenação por inserção:
  - Funcionamento
  - Implementação
- Algoritmo de ordenação por borbulhagem:
  - Funcionamento
  - Implementação
- Conclusões

Ordenação por Inserção

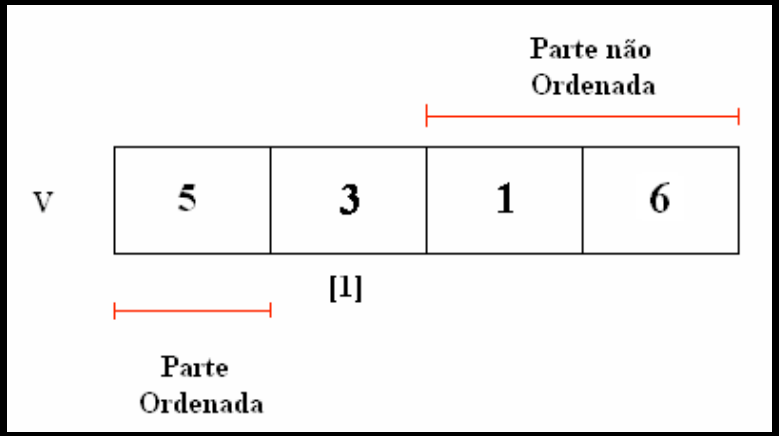
Ordena um array da esquerda para a direita, por ordem crescente, ou seja, à medida que avança vai deixando os elementos à esquerda ordenados, usando para isso, uma variável temporária.

Funcionamento:

Ordenar o seguinte vector por inserção:

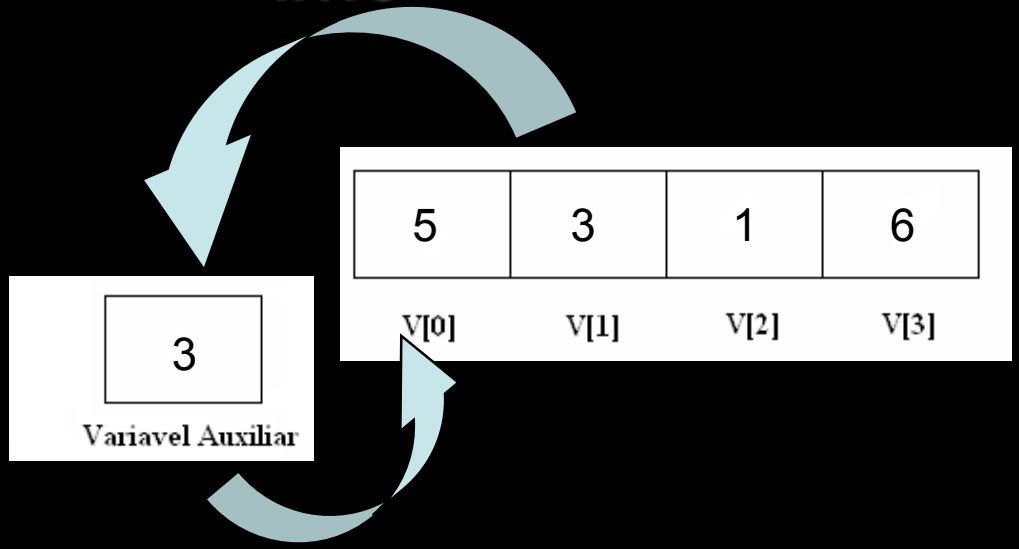


Começa-se em V[1] do vector, porque assume que v[0] já está ordenado:



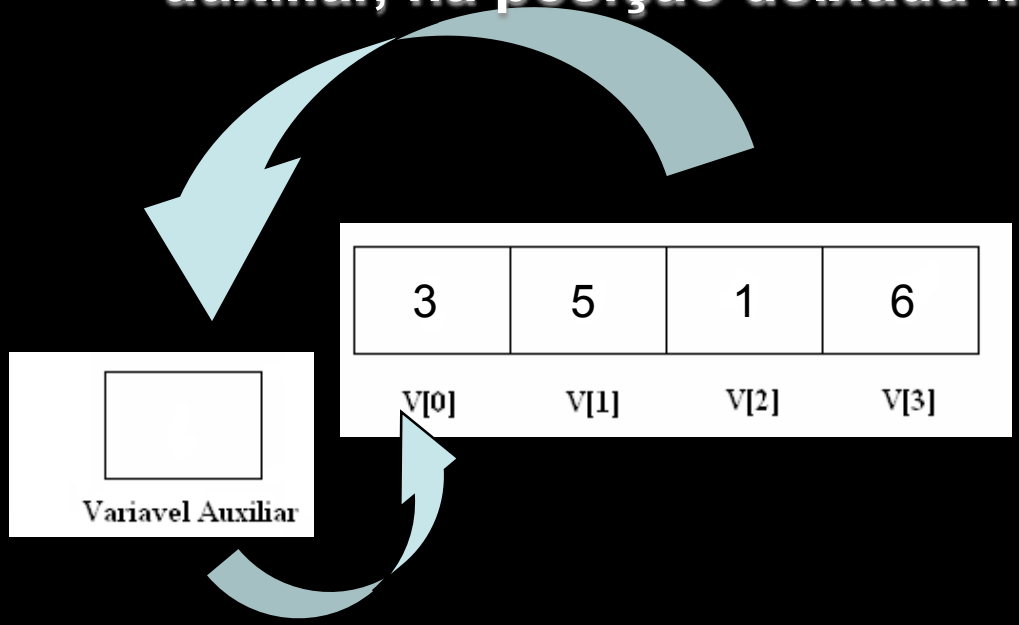
1ºPasso

- Compara-se o elemento v[1] com v[0]
  - Como o elemento v[1] < elemento v[0]
    - Guarda-se o elemento v[1] numa variável auxiliar
      - Move-se o elemento v[0] para a direita
        - Insere-se o elemento que está na variável auxiliar, na posição deixada livre



2º Passo:

- Compara-se o elemento v[2] com v[1] e v[0]
  - Como o elemento de v[2] < elemento de v[1] e v[0]
    - Guarda-se o elemento de v[2] numa variável auxiliar
      - Move-se o elemento de v[1] e de v[0] para a direita
        - Insere-se o elemento que está na variável auxiliar, na posição deixada livre

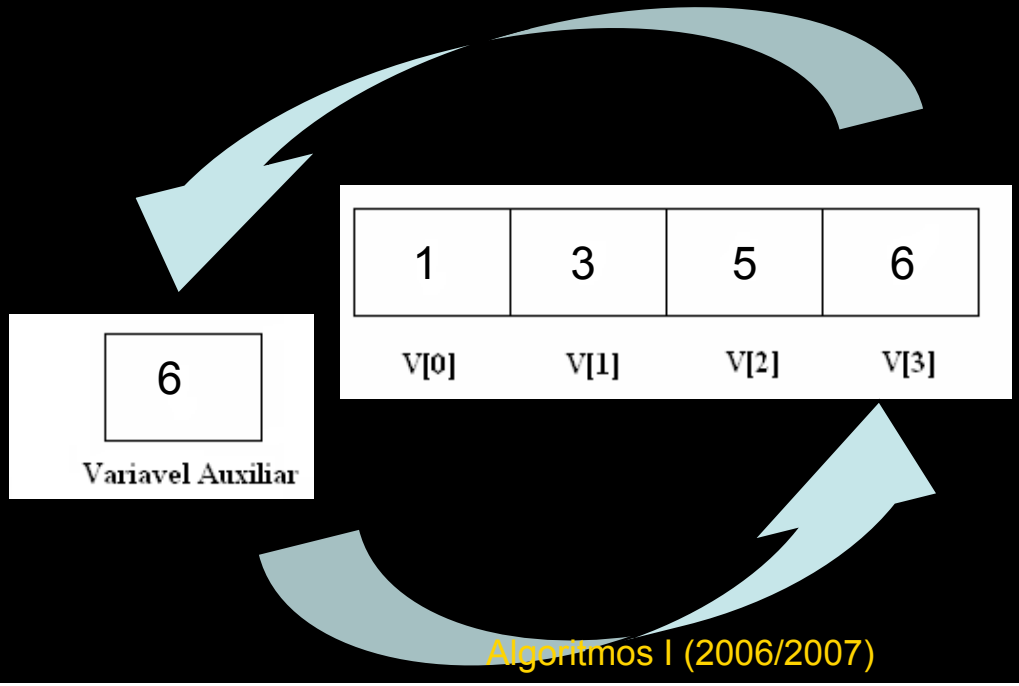


3º Passo:

- Compara-se o elemento de v[3] com os elementos das posições anteriores

  - Como o elemento v[3] > elemento v[2] , v[1], v[0] então não há deslocamento de elementos e este é colocado sobre ele próprio.

Obtendo-se assim o vector ordenado



Implementação

```
2.  Void ordena_insercao (int v[], int tam)
3.  {
4.        int j, i, temp;
5.        for (i=1; i<tam; i++)
6.        {
7.          temp = v[i];
8.          for (j=i-1; v[j] > temp && j >= 0; j--)
9.                v[j+1] = v[j];
10.         v[j+1] = temp;
11.       }
12. }
```

Legenda:

V - Vector a ordenar

TAM - Tamanho da lista

i, j – Contadores

Temp – Realiza Trocas

Ordenação por Borbulhagem

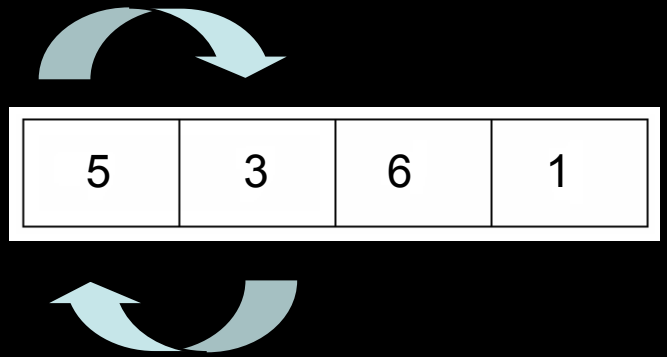
Compara dois elementos consecutivos de um vector e se o da esquerda é maior que o da direita trocam de posição. Quando existem trocas, os elementos maiores tendem a deslocar-se para a direita e os menores para a esquerda.

Funcionamento:

1º Passo:

Compara os dois primeiros elementos.

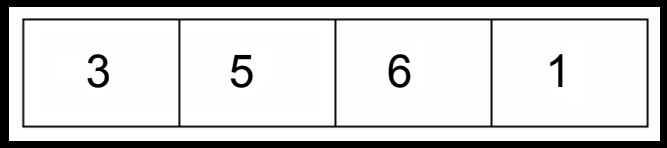
Como o elemento da esquerda é maior que o da direita, trocam.



Continuação:

Compara os elementos seguintes

Como os elementos não estão desordenados não há trocas

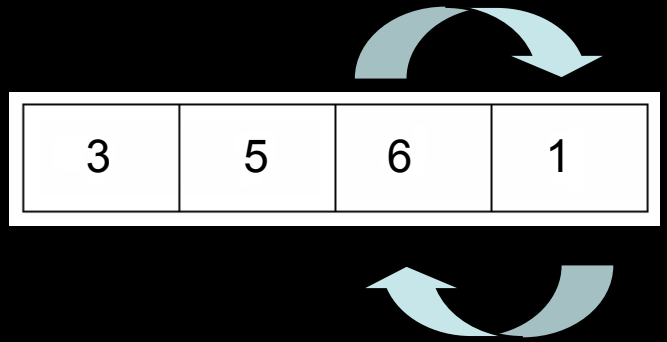


Continuação:

Compara os últimos 2 elementos

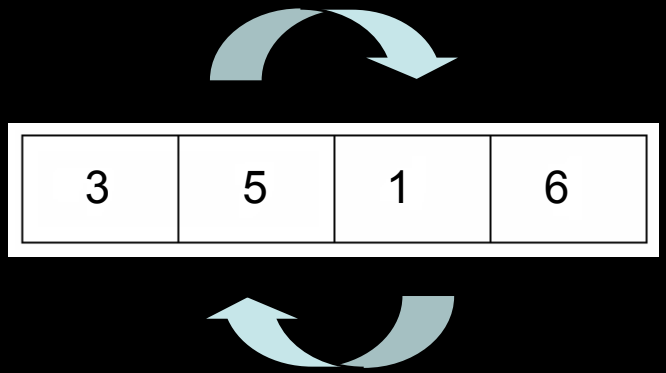
Como o elemento da esquerda é maior que o da direita, trocam.

Ficando assim o último elemento ordenado



2ºPasso:

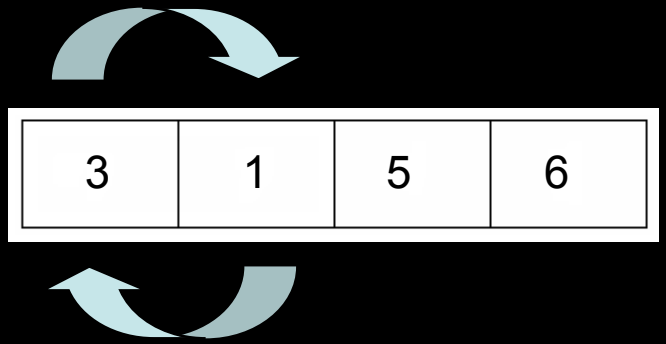
Comparam-se novamente os elementos na parte não ordenada do array



3º Passo:

Repete-se o 2º Passo e processo termina quando todos os elementos forem comparados.

Ficando assim o vector ordenado



Implementação

```
1.  Void ordena_borbulhagem(int v[], int TAM)
2.  {     int j, fim, t;
4.        fim = TAM-1;
5.        do
6.        {
7.           t=0;
8.           for (j=0 ; j< fim; j++)
9.                 if (v[j] > v[j+1])
10.                 {
11.                    troca (&v[j],&v[j+1]);
12.                    t=j;
13.                 }
14.          fim= t;
15.       }
16.       while (t!=0);
17. }
```

Legenda:

V - Vector a ordenar

TAM - Tamanho da lista

i, j, fim – Contadores

Conclusões

- Os algoritmos de ordenação por inserção e de ordenação por borbulhagem:

  → São muito simples

  → Eficazes apenas para uma pequena quantidade de dados.

  → Tem complexidade quadrática *0(n^2)*

- A inserção é geralmente melhor que o de borbulhagem, porque realiza um menor número de comparações pelo vector para este ser ordenado

- A única característica compensadora da borbulhagem é que requer pouco espaço adicional, ao contrário da inserção que utiliza uma variável auxiliar para realizar as trocas.

Algoritmos de Ordenação:

Insertion e Bubble Sort

FIM

Ana Cardoso
Márcia Saraiva
E.P.G.I