

Estrutura de Dados

Métodos de Ordenação

Bubble Sort

Selection Sort


Prof.(a): Me. William P. Santos Júnior

Conceitos Introdutório

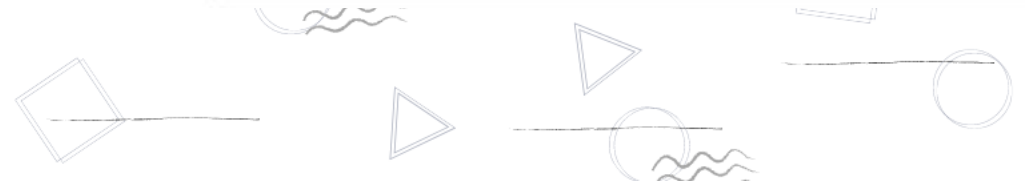
Por que Ordenar?

É comum, ordenar conjunto de coisas em nossas vidas, e em se tratando de dados armazenados em computadores, como geralmente lidamos com grandes volumes de dados, é desejável que esses estejam ordenados de alguma forma para facilitar nossa vida na hora da manipulação.

Vamos pensar em um exemplo de nosso cotidiano. Em sua playlist é muito mais fácil achar sua música preferida se ela estiver ordenada de acordo com suas preferencias. Outro exemplo seria encontrar a mediana em uma lista de dados, se esta lista estivesse ordenada, ou, se em uma lista constarem dados iguais ou itens com o mesmo valor é mais fácil fazer um rearranjo para que esses itens ou dados fiquem um do lado do outro. Tudo isso pode ser utilizado também com conjunto de dados.



50	40	30	20	10
40	30	20	10	50
30	20	10	40	50
20	10	30	40	50
10	20	30	40	50
10	20	30	40	50




Conceitos Introductório

Por que Ordenar?

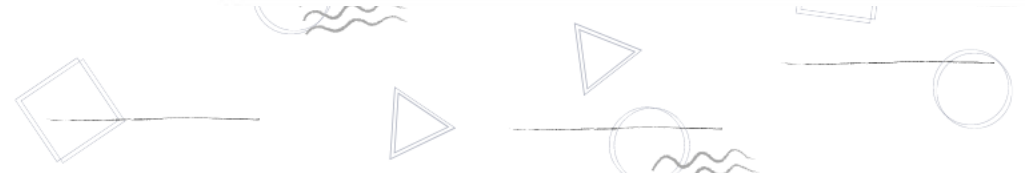
Outros exemplos que podemos considerar após uma base de dados estar construída:

- Alunos por nota
- Clientes por CEP
- Vendas por preço
- Cidades em ordem crescente de população.

E fazer a ordenação de dados pode ser um passo inicial para pesquisa-los quando necessário, utilizando uma **pesquisa binária** ou **pesquisa linear**.



50	40	30	20	10
40	30	20	10	50
30	20	10	40	50
20	10	30	40	50
10	20	30	40	50
10	20	30	40	50



Passos para Ordenação

Podemos definir um problema de ordenação da seguinte forma:

Dados de Entrada: podemos definir uma sequência n , com R_1, R_2, \dots, R_n , que chamaremos de registros.

Processamento dos Dados: É a descrição das tarefas que o algoritmo deverá executar, essas tarefas podem ser ordenar o conjunto de registros em ordem crescente ou decrescente.

Descrição da Saída: Retornar a sequência de entrada já ordenada, na ordem definida, crescente ou decrescente.

Cada registro mencionado na definição ao lado possui uma chave que conduz o processo de ordenação. Outros dados, podem existir, mas não têm efeito na ordenação, exceto se fizerem parte do registro. Entretanto, em nosso exemplo, os registros serão números inteiros ou strings, ou seja, cada registro será composto unicamente da chave. Além disso, para simplificar, faremos nossa ordenação dos registros em ordem crescente.

Redefinição do Problema

Dados de
Entrada:

Sequência S de n números inteiros ou de Strings;

Processamento
dos dados:

Os elementos deverão ser ordenados de forma Crescente;

Descrição da
Saída:

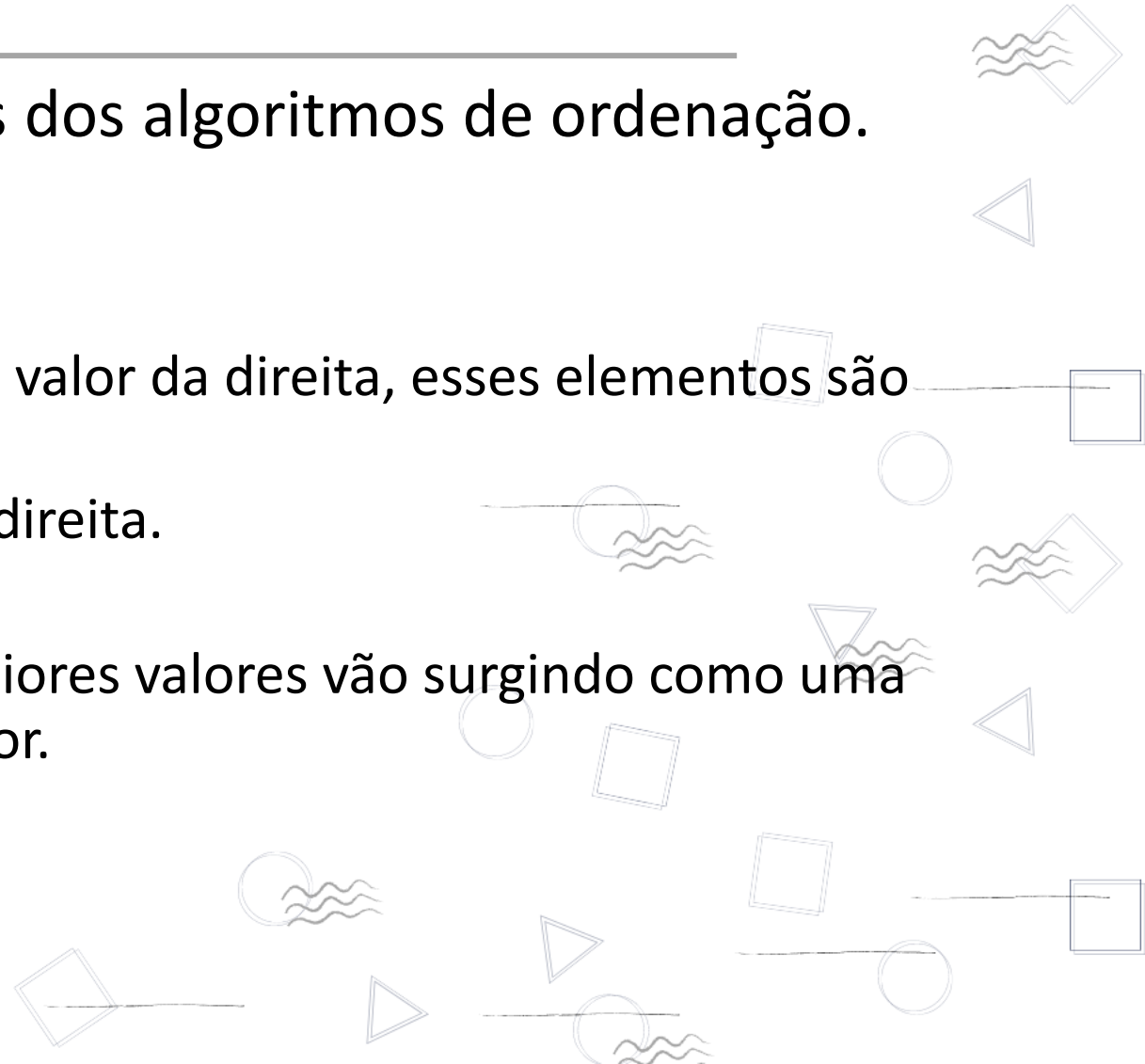
Retornar S , dos dados de entrada em ordem crescente;

Bubble Sort

- Muito lento, porém é o mais simples dos algoritmos de ordenação.
- Execução:
 - Comparação de dois valores
 - Se o valor da esquerda for maior que o valor da direita, esses elementos são trocados de posição;
 - Faz o deslocamento de uma posição a direita.

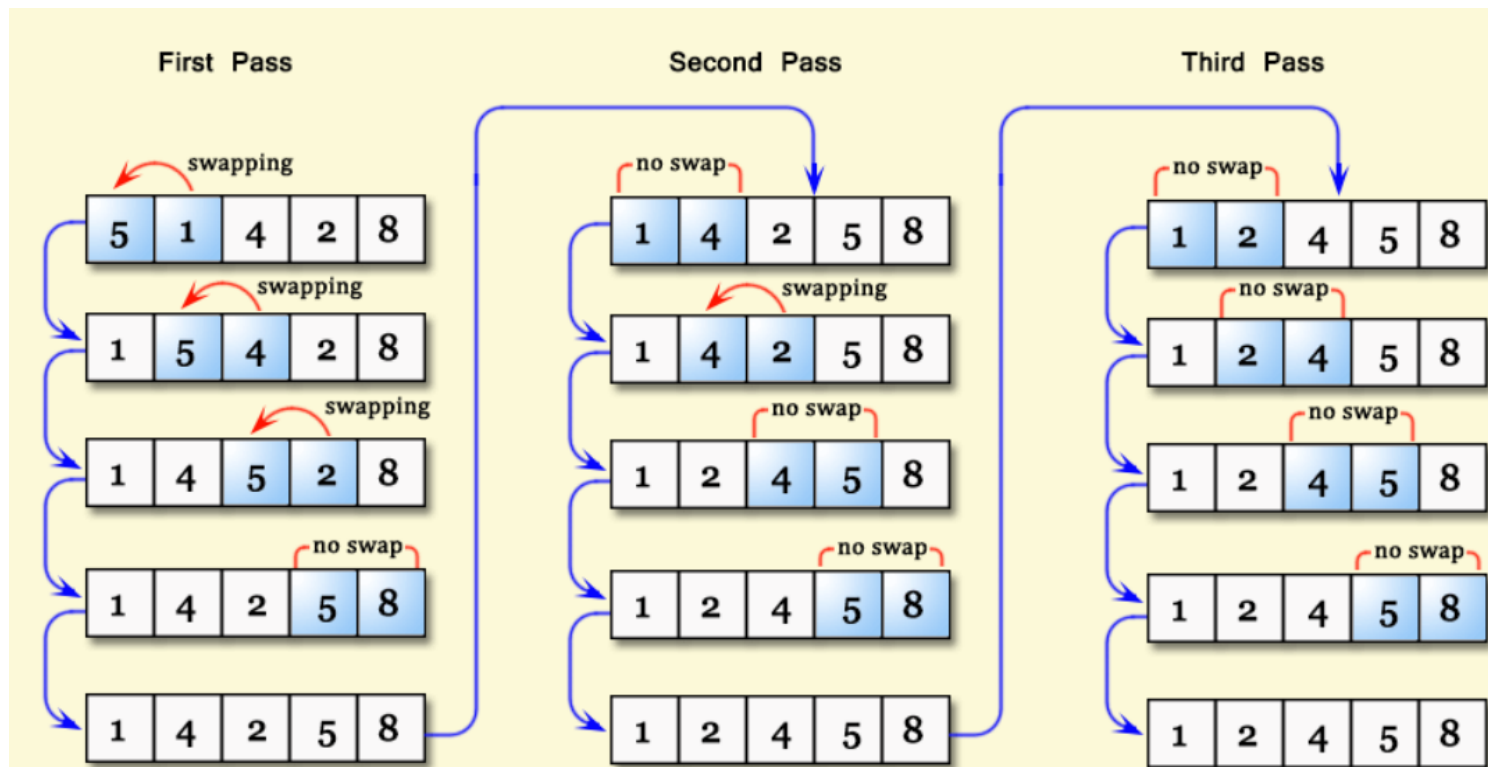
A medida que o algoritmo avança, os maiores valores vão surgindo como uma “bolha”, na extremidade superior do vetor.

Veja exemplo: <https://visualgo.net/en/sorting>



Bubble Sort

Implementação.



```
for (i = 0; i < n-1; i++) {  
    for (j = 0; j < n-i-1; j++) {  
        if (vet[j] > vet[j+1]) {  
            temp = vet[j];  
            vet[j] = vet[j+1];  
            vet[j+1] = temp;  
        }  
    }  
}
```

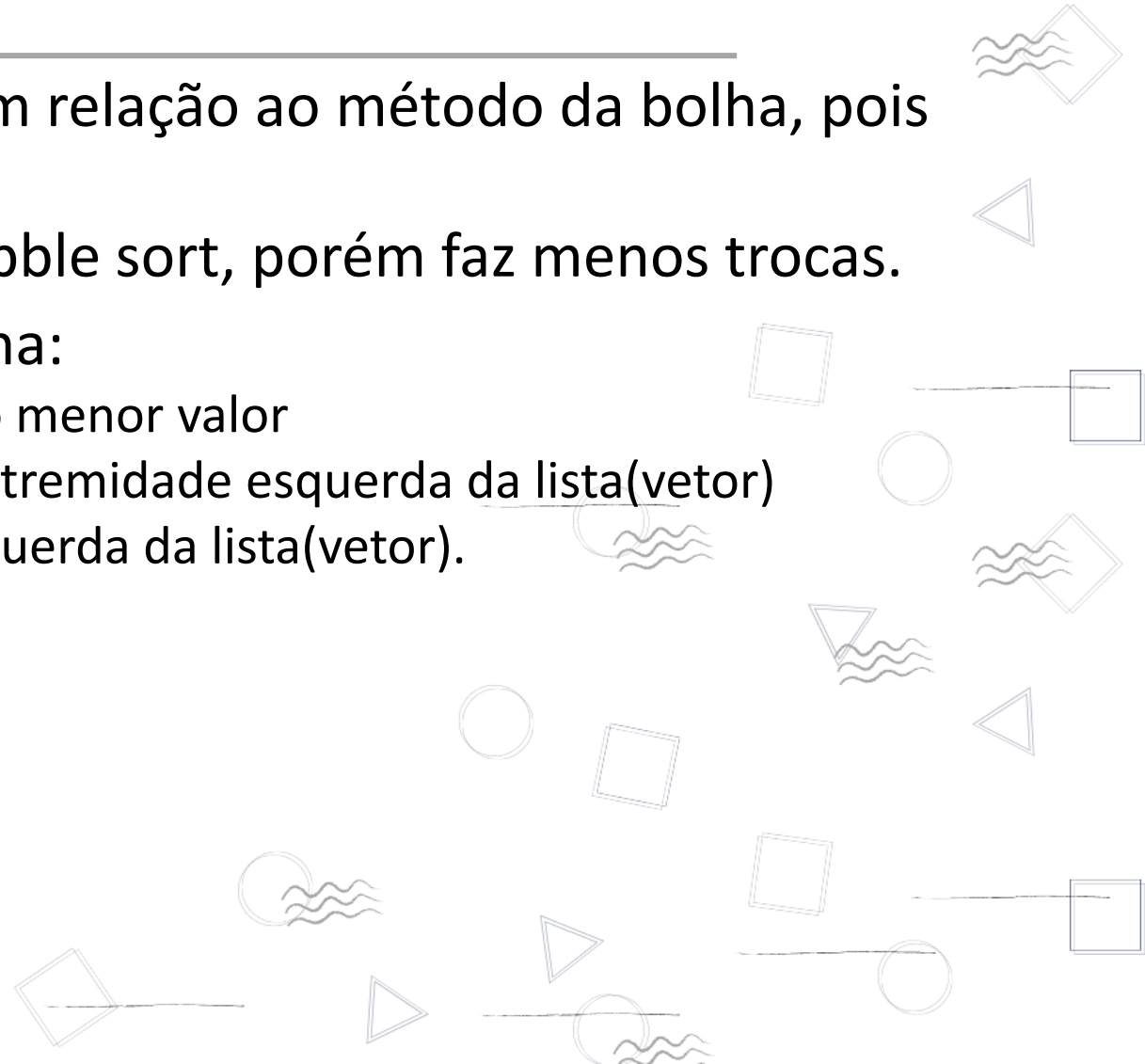
Selection Sort

- Esse algoritmo melhora a ordenação em relação ao método da bolha, pois reduz o numero de trocas.
- O número de comparações igual ao bubble sort, porém faz menos trocas.

Esse algoritmo funciona da seguinte forma:

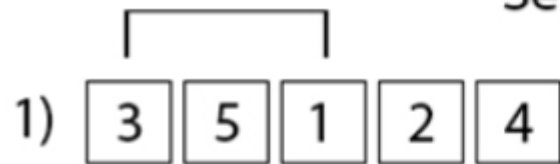
- percorre todos os elementos e seleciona o menor valor
- o menor valor é trocado com o valor da extremidade esquerda da lista(vetor)
- os valores ordenados se concentram a esquerda da lista(vetor).

Veja exemplo: <https://visualgo.net/en/sorting>

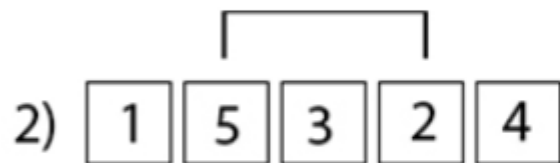


Selection Sort

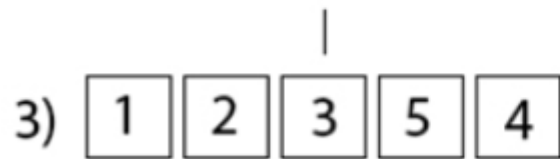
Selection Sort



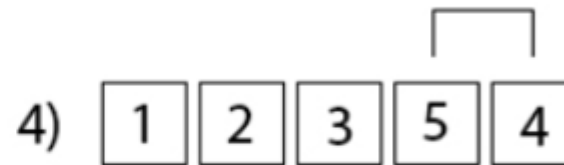
Troca



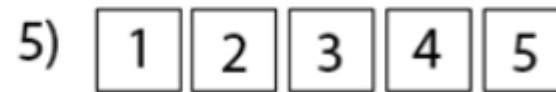
Troca



Não Troca



Troca



```
for (i = 0; i < n-1; i++) {  
    min_idx = i;  
    for (j = i+1; j < n; j++) {  
        if (arr[j] < arr[min_idx]) {  
            min_idx = j;  
        }  
    }  
    int temp = arr[min_idx];  
    arr[min_idx] = arr[i];  
    arr[i] = temp;  
}
```

UniEVANGÉLICA
UNIVERSIDADE EVANGÉLICA DE GOIÁS

