**Ordenação:**

-É a tarefa de colocar um conjunto de dados em uma determina ordem;

-Permite acesso mais eficiente aos dados;

**Algoritmo de ordenação:**

-É o algoritmo que coloca os dados de uma dada sequência em uma certa ordem;

**Tipos de ordenação mais comuns:**

-Numérica:

1,2,3,4,5(crescente)

5,4,3,2,1(decrescente)

-Lexicográfica(Alfabética):

Ana, André, Bianca, Ricardo(crescente)

Ricardo, Bianca, André, Ana(decrescente)

**Classificação dos métodos de ordenação:**

-Ordenação interna:

O arquivo a ser ordenado cabe todo dentro da memória principal;

Qualquer registro pode ser imediatamente acessado;

-Ordenação externa:

O arquivo a ser ordenado não cabe na memória principal;

Os registros são acessados sequencialmente ou em blocos;

**BubbleSort:**

-Compara pares de elementos adjacentes e os troca de lugar se estiverem em ordem diferente da desejada;

-Esse processo se repete até que todos os elementos estejam ordenados;

-“Carrega” o maior(crescente) ou menor(decrescente) elemento até o a última posição do vetor,

-Desse comportamento que vem o nome(Bubble=bolha), igual á bolhas em um recipiente onde as maiores descem ate o fundo;

Performance:

-Melhor caso: O(N)

-Pior caso: O(N^2)

-Não recomendado para grandes conjuntos de dados;

**InsertionSort:**

-Similar a ordenação de cartas de baralho com as mãos;

-Paga-se um elemento por vez e coloca em seu devido lugar;

Performance:

-Melhor Caso: O(N)

-Pior Caso: O(N^2)

-Eficiente para pequenos conjuntos de dados;

-Estável: não altera a ordem de números iguais;

-Capaz de ordenar dados a medida que os recebe;

**SelectionSort**:

-A cada passo, procura o menor valor do array e o coloca na primeira posição do array;

-Repete-se o passo acima desconsiderando a posição inicial;

Repete~~-~~se os passos acima para todas as posições do array;

Performance

-Melhor Caso: O(N^2)

-Pior Caso: O(N^2)

-Ineficiente para grandes conjuntos de dados

**MergeSort:**

**-**Estratégia: dividir e conquistar;

-Divide, recursivamente, o conjunto de dados até que cada subconjunto possua 1 elemento;

-Combina 2 subconjuntos de forma a obter 1 conjunto maior e ordenado;

-Esse processo se repete até que exista apenas 1 conjunto;

Performance:

-Melhor caso: O(N log N)

-Pior Caso O(N log N)

-Estável: não altera a ordem de dados iguais;

-Desvantagem: recursivo e usa um vetor auxiliar durante a ordenação;

**QuickSort**:

-Estratégia: dividir e conquistar;

-Um elemento qualquer é escolhido como pivô;

-Particionar: os dados são organizados em valores menores que o pivô e maiores que o mesmo;

-Recursivamente ordena as duas partições;

Performance:

-Melhor caso: O(N log N);

-Pior caso (raro): O(N^2);

-Desvantagens: escolha do pivô afeta o desempenho;