**UNIVERSIDADE FEDERAL DE ALFENAS**

**UNIDADE SANTA CLARA**

**CIÊNCIA DA COMPUTAÇÃO**

**IAGO SACHSIDA TEIXEIRA RASSILAN BRAGA 2024.1.08.045**

**ATIVIDADE - MÉTODOS DE ORDENAÇÃO**

**RELATÓRIO DO DESEMPENHO**

# 1- Introdução

Este projeto tem como foco o estudo e comparação de diferentes algoritmos de ordenação não recursivos usando a linguagem C/C++. A avaliação será feita em três cenários específicos: ordenação de vetores em ordem crescente, aleatória e decrescente. A análise comparativa dos métodos levará em conta o uso dos vetores, registrando todas as operações de leitura e escrita, e realizando um gráfico como produto final. O relatório final incluirá uma introdução ao problema, uma revisão teórica dos algoritmos de ordenação, uma descrição dos materiais utilizados, detalhes sobre a implementação dos métodos, uma análise dos resultados e conclusões baseadas nesses resultados.

# 2- Referencial Teórico

A ordenação de dados, é o processo de organizar elementos em uma determinada sequência, seja crescente, decrescente ou conforme algum critério específico. Este processo é essencial porque torna a busca, a recuperação e a manipulação de dados mais eficientes. Existem vários algoritmos de ordenação, cada um com suas próprias características e vantagens e eles são fundamentais em muitas áreas, como bancos de dados, onde ajuda na organização e recuperação rápida de registros, e em algoritmos de busca e otimização. Em sistemas de informação, uma ordenação eficiente melhora o desempenho geral, especialmente quando se lida com grandes volumes de dados, como em motores de busca e transações financeiras. Neste tópico será abordado 3 algoritmos exemplificados em sala de aula.

### 2.1 Bubble Sort

Bubble Sort é um algoritmo de ordenação simples que organiza os elementos de uma lista em ordem crescente ou decrescente, comparando repetidamente pares de elementos adjacentes e trocando-os se estiverem na ordem errada. É conhecido por sua simplicidade, mas também por sua ineficiência em grandes conjuntos de dados.

### 2.2 Insertion Sort

O Insertion Sort é um algoritmo de ordenação simples e intuitivo, frequentemente usado para ordenar pequenos conjuntos de dados. Em seu funcionamento, ele retira um elemento e reorganiza ele para sua posição correta. Em listas grandes, porém, é menos eficiente que algoritmos mais avançados como Merge Sort ou Quick Sort.

### 2.3 Selection Sort

O Selection Sort é um algoritmo de ordenação que funciona repetidamente selecionando o menor (ou maior, dependendo da ordem desejada) elemento de uma lista e movendo-o para a posição correta. É geralmente considerado mais simples e intuitivo de entender e implementar do que o Insertion Sort, porque embora também ele seja simples, envolve mais manipulação de elementos e pode ser um pouco mais difícil de seguir, especialmente para iniciantes.

## 3- Métodos Utilizados

O objetivo da atividade era desenvolver um projeto baseado na linguagem C/C++ que ordene vetores com números não-repetidos e dispostos de três formas: crescente, aleatória e decrescente. E por seguinte implementar os códigos de ordenação, onde deviam ser inseridos contadores de utilização do vetor, e sua comparação, onde é feita entre os três métodos implementados e variando o tamanho do vetor, e logo em seguida realizando o gráfico correspondente. Nesse tópico, é apresentado um breve resumo de como o código foi elaborado.

O código é inicializado com a construção de 3 funções, onde são elaborado os elementos Crescente, decrescente e aleatório. Segue abaixo um exemplo da função “ criarVetorAleatorio”.

```c
// Função para criar um vetor aleatório
void criarVetorAleatorio(int vetor[], int tamanho) {
    for (int i = 0; i < tamanho; i++) {
        vetor[i] = i + 1;
    }

    // Embaralhar o vetor
    srand(time(NULL));
    for (int i = tamanho - 1; i > 0; i--) {
        int j = rand() % (i + 1);
        int temp = vetor[i];
        vetor[i] = vetor[j];
        vetor[j] = temp;
    }
}
```

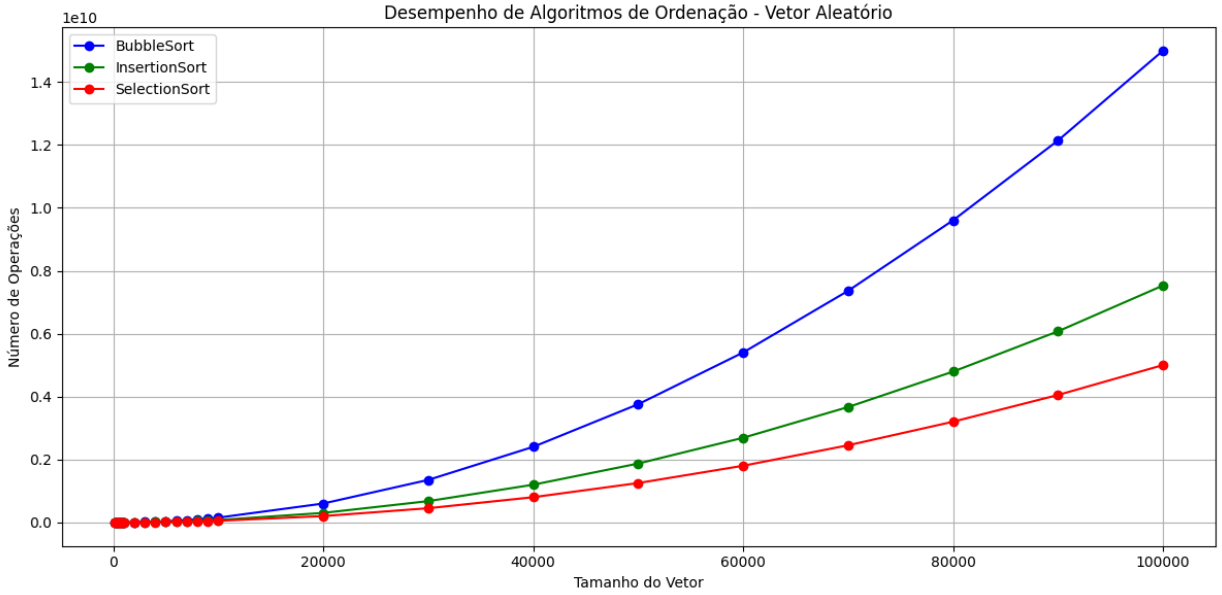
Em seguida, são feitas as funções dos algoritmos de ordenação, *bubbleSort*, *insertSort* e *selectionSort*. Segue abaixo um exemplo da função “bubbleSort “.

```c
void bubbleSort(int vetor[], int tamanho, ContadorUso *contador) {
    int i, j;
    for (i = 0; i < tamanho - 1; i++) {
        for (j = 0; j < tamanho - i - 1; j++) {
            contador->leituras += 2;
            if (vetor[j] > vetor[j + 1]) {
                int temp = vetor[j];
                vetor[j] = vetor[j + 1];
                vetor[j + 1] = temp;
                contador->escritas += 3;
```

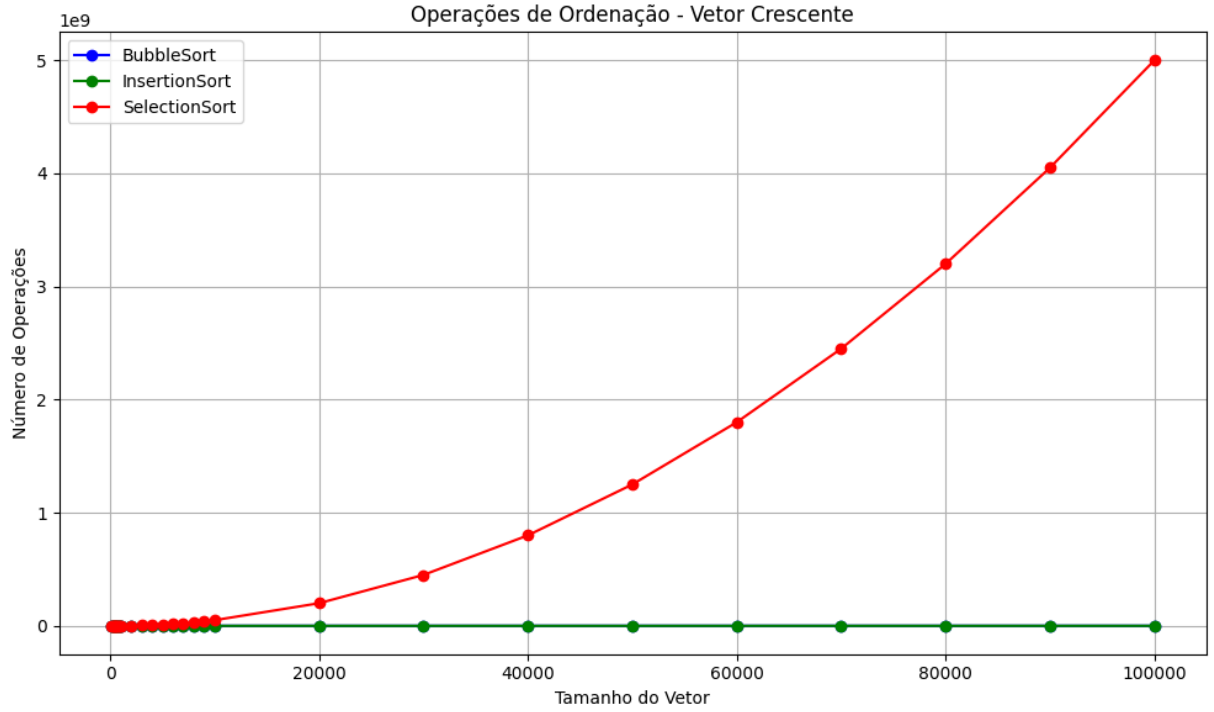
Em suma, no final do programa é usado um sistema básico de “call“ de funções, onde é apresentado o valor de cada algoritmo de ordenação.

# 4-Resultado Obtidos

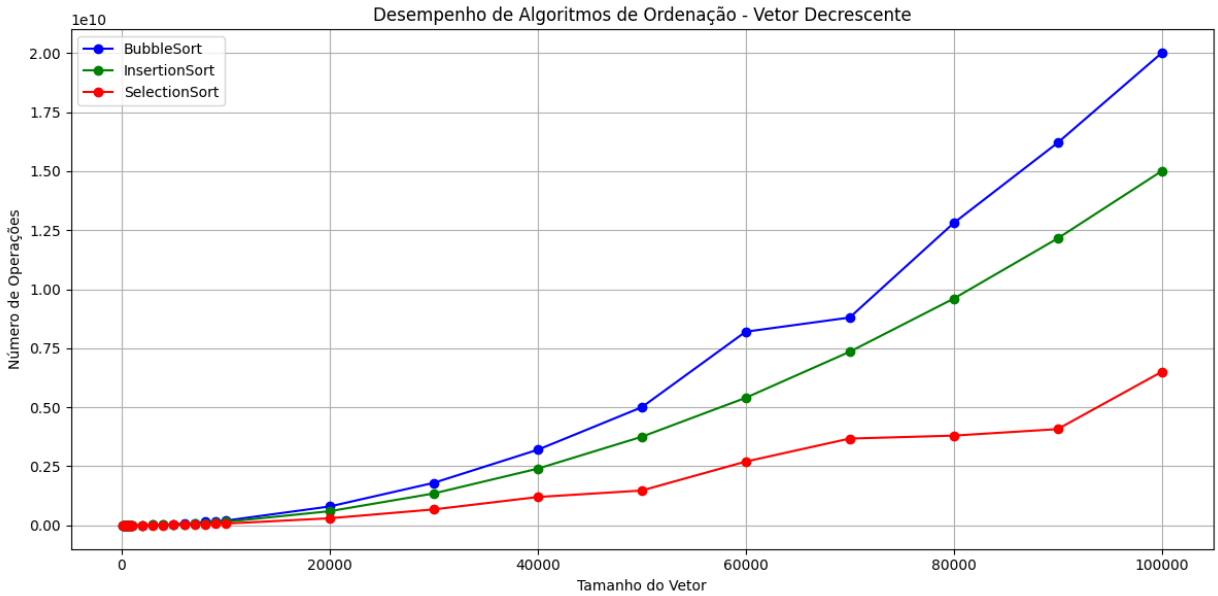
## 4.1 Vetores aleatórios




## 4.2 Vetores crescente




## 4.3 Vetores decrescentes




## 5- Conclusão

Com os dados apresentados, é nítido o mal desempenho do Bubble Sort, nos gráficos decrescentes e aleatório, com muitas leituras e mal otimização de tempo, sendo o pior algoritmo de ordenação para se escolher, porém uma boa porta de entrada para quem deseja aprender sobre o assunto, sendo o mais fácil de manipular.

Insertion Sort se mostra mais eficiente do que o Bubble Sort, tendo o mesmo resultado no gráfico crescente, e se mostrando superior em outros. Tendo sua complexidade ainda de nível baixo e maior estabilidade, é uma melhor alternativa.

Embora o Selection Sort seja muito similar ao Insertion Sort, ele tende a realizar um maior número de trocas, o que pode ser menos eficiente em seu projeto, como é verificado no gráfico crescente, onde teve um resultado ruim, porém, ganhando de seus companheiros nos outros gráficos.

Em suma, o Insertion Sort e Selection Sort são as melhores escolhas em relação ao Bubble Sort. No entanto, é importante observar que a escolha entre os dois, depende do contexto específico, incluindo o tamanho e o estado dos dados a serem ordenados, assim como as preferências de implementação e os requisitos de estabilidade.

## Referências

**1.** Goodrich, Michael T., Roberto Tamassia, and Michael H. Goldwasser. *Data Structures and Algorithms in Python*. Hoboken, NJ: Wiley, 2013.

MENEZES, Nilo Ney Coutinho. *Introdução à Programação com Python: Algoritmos e Lógica de Programação Para Iniciantes*. São Paulo: Novatec, 2014.

Algoritmos de ordenação: análise e comparação - https://www.devmedia.com.br/algoritmos-de-ordenacao-analise-e-comparacao/28261 [Acesso em 15-06-2024].