MINISTÉRIO DA EDUCAÇÃO
Universidade Federal de Alfenas
UNIFAL-MG
Rua Gabriel Monteiro da Silva, 700 – Centro. Alfenas/MG. CEP: 37130-001

Algoritmos e Estrutura de Dados I-Prática

Nome:Jeann Victor Batista **RA:2024.1.08.014**

Introdução

O objetivo do trabalho proposto é desenvolver um projeto baseado na linguagem C/C++ que ordene vetores com números não-repetidos e dispostos de três formas: crescente, aleatória e decrescente. Onde deveria ser feito uma comparação sobre o desempenho dos três métodos de ordenação estudados (**Bubble Sort, Insertion Sort, Selection Sort**) em diferentes tipos/tamanhos de vetores e também tamanhos de vetores.

Referencial Teórico

O referencial teórico utilizado foi, além do aprendizado durante as aulas práticas ministradas pelos professores **Paulo Bressan e Luiz Eduardo da Silva**, foi necessario algums pesquisas a mais sobre os metodos de ordenção.

O *Bubble Sort*, é um algoritmo de ordenação dos mais simples. Cuja a ideia é percorrer um conjunto de elementos diversas vezes, e a cada passagem fazer flutuar para o topo o maior elemento da sequência. Essa movimentação lembra a forma como as bolhas em um tanque de água procuram seu próprio nível, e disso vem o nome do algoritmo. À complexidade do *Bubble sort* é O(n^2) no pior caso e no caso médio, onde n é o número de elementos na lista.

O Insertion Sort, é um algoritmo de ordenação que, dado uma estrutura (array, lista) constrói uma matriz final com um elemento de cada vez, uma inserção por vez. Pode-se fazer uma comparação do Insertion Sort com o modo como algumas pessoas organizam um baralho num jogo de cartas. À complexidade do *Insertion Sort* é O(n^2) no pior caso e no caso médio, onde n é o número de elementos na lista

O *Selection Sort* é um algoritmo de ordenação baseado em se passar sempre o menor valor do vetor para a primeira posição (ou o maior dependendo da ordem requerida), depois o de segundo de menor valor para a segunda posição,e assim é feito sucessivamente com os (n-1) elementos restantes, até os ultimos dois elementos.A complexidade deste algoritmo será sempre O(n^2).

Materiais Utilizados

Os materiais utilizados para se realizar este projeto foram:

- O material PDF enviado pelo professor Bressan: para fazer os algoritmos de ordenação. https://drive.google.com/file/d/17F8gU5jplJuVCSkHqld2i57pEX_aaQLr/view
- O aplicativo utilizado para se fazer os gráficos requisitados foi por um código em Python .
- O aplicativo para realizar o código foi o NetBeansApache.

Conclusão

O que foi possivel observar é que o método menos eficaz é o Bubble Sort, devido ao seu grande número de operações.Já o Insertion Sort e Selection Sort não apresentam grandes diferenças entre si.

Essas características fazem com que o **Insertion Sort** seja preferido para listas pequenas ou quase ordenadas, enquanto o **Selection Sort** pode ser útil em situações onde a estabilidade não é uma preocupação.

- As diferenças entre Selection Sort e Insertion Sort são que: em listas quase ordenadas o Insertion é mais efetivo e mais estável em todos os tipos de casos, e a principal vantagem do Selection é ter um número de trocas menor (a diferença é muito pouca, mas é menor). Ambos têm complexidade $O(n^2)$ no pior caso e Insertion Sort pode ter O(n) no melhor caso, enquanto Selection Sort é consistentemente $O(n^2)$.

Métodos Implementados

O método implementado para o Bubble Sort foi:

```
algoritmo "bubble sort"
   para i de 9 ate 1 passo -1 faca
      para j de 0 ate i-1 passo 1 faca
         se v[j] > v[j+1] entao
            aux <- v[j]
            v[j] <- v[j+1]
            v[j+1] <- aux
         fimse
      fimpara
   fimpara
fimalgoritmo
```

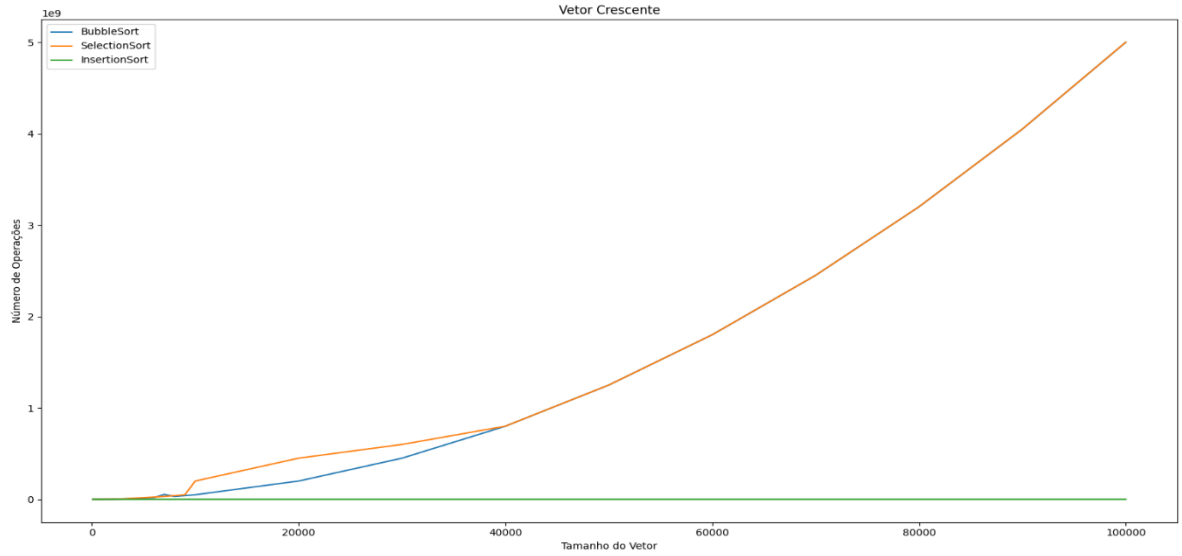
O método implentado para o Insertion Sort foi:

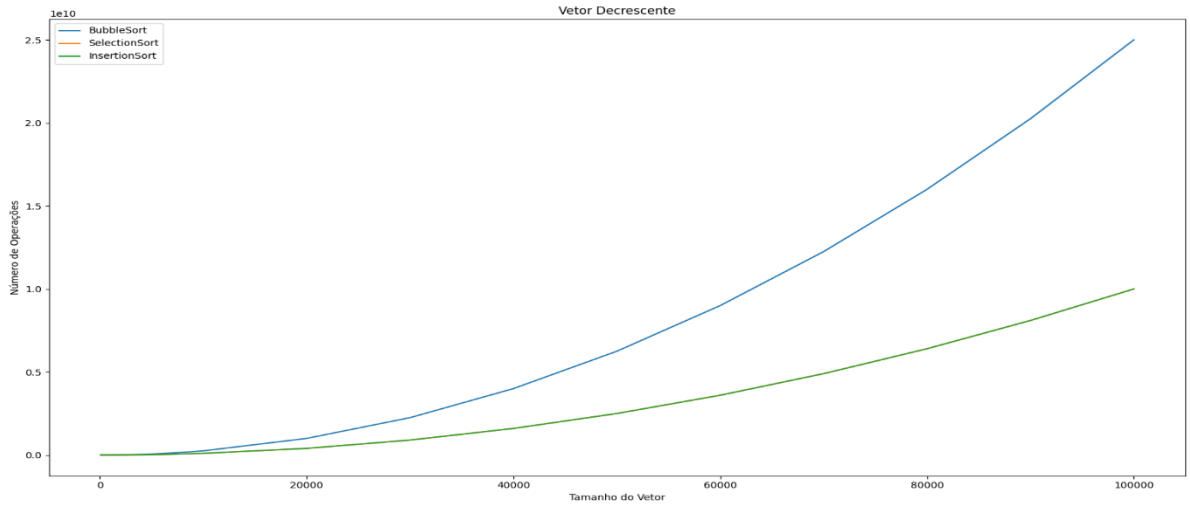
```
algoritmo "insertion sort"
   para i de 1 ate 9 passo 1 faca
      aux <- v[i]
      j <- i - 1
      enquanto ((j >= 0) e (v[j] > aux)) faca
         v[j+1] <- v[j]
         j <- j - 1
      fimenquanto
      v[j+1] <- aux
   fimpara
fimalgoritmo
```

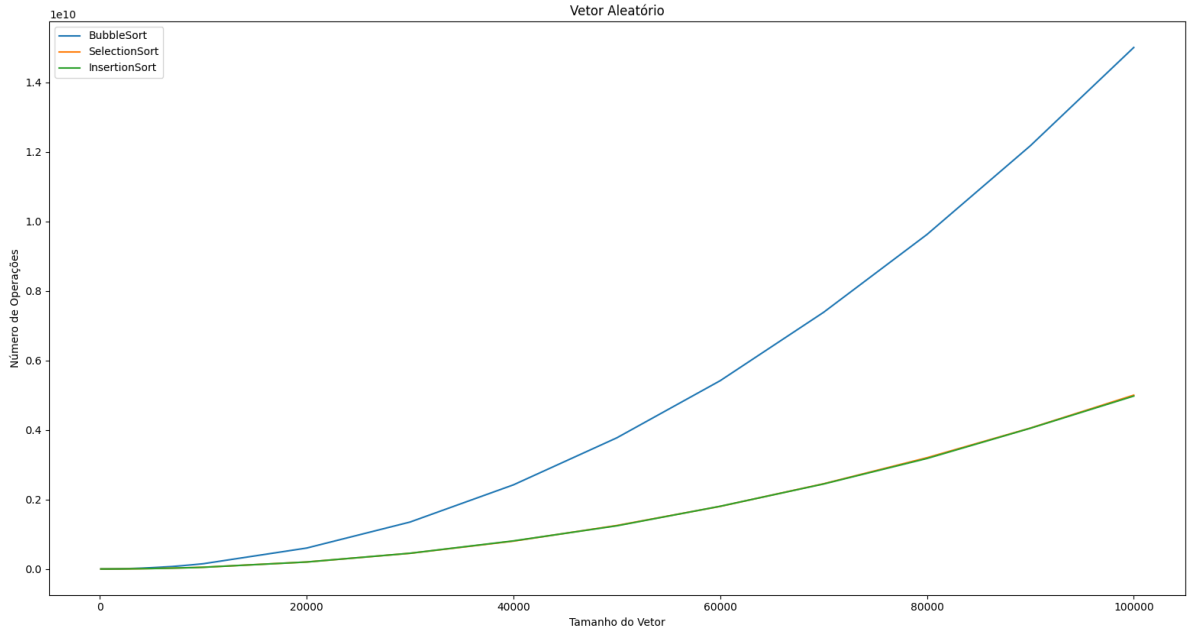
O método implementado para o Selection Sort foi:

```
algoritmo "insertion sort"
   para i de 0 ate 9 passo 1 faca
      min <- i
      aux <- v[i]
      para j de i+1 ate 9 passo 1 faca
         se v[j] < aux entao
            min <- j
            aux <- v[j]
         fimse
      fimpara
      aux <- v[i]
      v[i] <- v[min]
      v[min] <- aux
   fimpara
fimalgoritmo
```

Resultados Obtidos







Observação no gráfico feito pelo Python, devido as diferenças quase ‘’minúsculas’’ devido a escala, ocorreu que na maioria dos casos a linha do Insertion Sort e Selection Sort ficaram muito juntas.