

**UNIVERSIDADE ESTADUAL DA PARAIBA**

**CENTRO DE CIÊNCIAS E TECNOLOGIAS**

**RELATÓRIO**

**KEISON RANIERY RIBEIRO TRAVASSOS FILHO**

**MARÇO – 2024**

**CAMPINA GRANDE - PB**

**SUMÁRIO**

1. **OBJETIVO**
2. **EXPLICAÇÃO DE CADA ALGORITIMO DE ORDENAÇÃO**
3. **RESULTADOS**

**1. INTRODUÇÃO**

O objetivo desse relatorio é fazer uma comparação entre os algoritimos de ordenação: InsertionSort, SelectionSort e BobbleSort,dando detalhes de como cada um funciona e mostrando os resultados encontrados no codigo gerado.Dessa forma,foi usada as mesmas entradas para cada um dos algoritimos e analisado as diferenças de cada um,relacionadas ao numero de troca, numero de comparações e tempo de execução.A entrada usada nos códigos para resgate dos resultados consiste em 5 elementos (200,49,4,70,21).

**2.1 INSERTION SORT**

O Insertion Sort é um algoritmo de ordenação mais “básico”, que funciona de maneira semelhante a como as pessoas costumam ordenar cartas em um jogo de baralho. Primeiro, começa com o segundo elemento do array (índice 1), que é o elemento atual, compara este elemento com cada elemento antes dele, e se o elemento atual for menor que o elemento anterior, troca os dois elementos de lugar. Dessa forma continua a fazer isso até que o elemento atual esteja em sua posição correta (ou seja, não haja mais nenhum elemento maior antes dele), e daí é feito isso com todos os elementos até que todo o array esteja ordenado.

Numero de trocas:

Sobre o numero de trocas, Cada vez que o elemento atual é menor que o elemento anterior, é feito uma troca. Portanto, o número total de trocas é a soma de todas as vezes que um elemento teve que ser movido para sua posição correta.

Numero de comparações:

Sobre as comparações, para cada elemento, comparamos com os elementos antes dele.

Tempo de execução:

Depende da quantidade de comparações que são feitas entre os elementos, podendo ser (no pior caso) o numero de elementos ao quadrado, ou apenas numero de elementos (no melhor caso).

**2.2 SELECTION SORT**

O SelectionSort é um algoritmo de ordenação que funciona selecionando o menor (ou maior, dependendo da ordem desejada) elemento do array e colocando-o na posição correta. Começa com o primeiro elemento do array (índice 0), que é o elemento atual, procura o menor elemento no restante do array e troca o menor elemento encontrado com o elemento atual. Dessa forma passa para o próximo elemento e repete esse processo até que todo o array esteja ordenado.

Numero de trocas:

No SelectionSort a quantidade de trocas é ,no maximo, igual a quantidade de elementos de um array. Isso ocorre porque para cada elemto fazemos no máximo uma troca (com o menor elemento do restante do array).

Numero de comparações:

Sobre as comparações, para cada elemento é feita uma comparação com todos os elementos depois dele.

Tempo de execução:

O tempo de execução é sempre a quantidade de elementos ao quadrado, pois o algoritmo não tem como saber se o array já está ordenado sem fazer todas as comparações.

**2.3 BOBBLE SORT**

O Bobble Sort é um algoritmo de ordenação que funciona repetidamente trocando os elementos adjacentes se eles estiverem na ordem errada. Assim, começa com o primeiro elemento do array (índice 0),que é o elemento atual, compara este elemento com o elemento seguinte e se o elemento atual for maior que o próximo elemento, troca os dois elementos de lugar. Dessa forma passa para o próximo par de elementos e repete o processo até que tenha percorrido todo o array. Logo repete todo o processo para o array até que nenhum elemento precise ser trocado.

Numero de trocas:

No BubbleSort a quantidade de trocas depende de como o array está ordenado, no pior caso (array ordenado em ordem decrescente) será necessário fazer todas as trocas.

Numero de comparações:

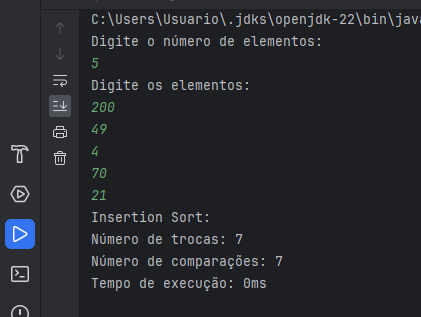
Sobre as comparações, para cada elemento é feita uma comparação com o elemento seguinte, assim independente de como o array está ordenado, é feita uma comparação com todos os elementos.

Tempo de execução:

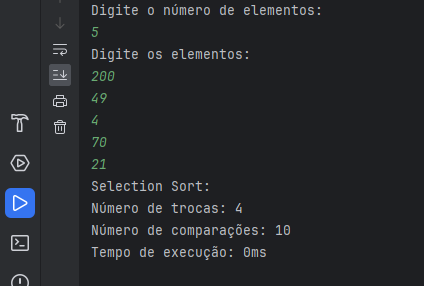
O tempo de execução é sempre a quantidade de elementos ao quadrado, (assim como o SelectionSort) , pois o algoritmo não tem como saber se o array já está ordenado sem fazer todas as comparações.

**3.RESULTADOS**

INSERTION SORT:



SELECTION SORT:



BUBBLE SORT:

