Métodos elementares de ordenação - Estudo de eficiência

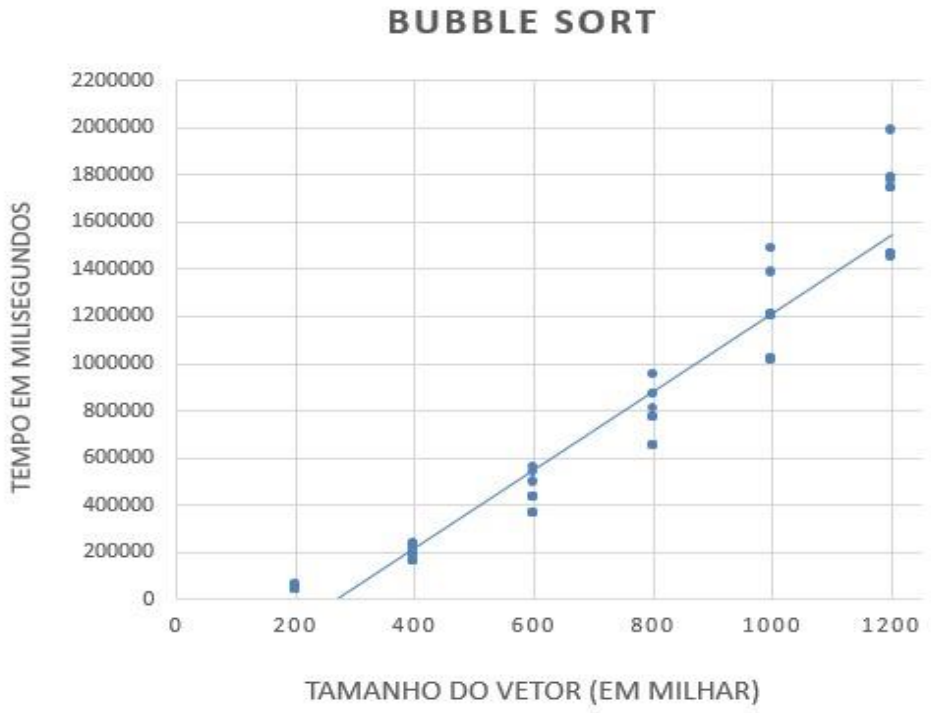
Alunos: Lucas Kazuhiro Okokama RA: 2040482312022

Pedro de Oliveira Ramos Trovo RA: 2040482312006

1) Tabela de médias

Médias de Tempo (ms)			
Valor	Bubble Sort	Insertion Sort	Selection Sort
200000	46898	10476	7425
400000	187645	40430	28618
600000	422559	91579	64594
800000	752866	161869	114824
1000000	1174255	252966	180508
1200000	1708817	366663	260429
Média Geral			
	715507	153997	109400

2) Gráfico Bubblesort



3) Gráfico Insertion sort

4) Gráfico Selection sort

5) Gráfico Geral

6) Análise dos métodos

6.1) Método bubble sort

O método bubble sort se demonstrou como o menos eficiente. Ele percorre a lista várias vezes, comparando elementos próximos e realizando trocas se estiverem na ordem errada.

6.2) Método insertion sort

O método insertion sort se demonstrou mais eficiente do que o bubble sort, mas menos eficiente do que o selection sort. Ele percorre a lista, examina cada elemento e o insere na posição correta, deslocando os elementos maiores para a direita até que os elementos estejam ordenados.

6.3) Método selection sort

O método selection sort se demonstrou como o mais eficiente dentre os três métodos utilizados. Ele percorre a lista, selecionando repetidamente o menor elemento da lista não ordenada e movendo-o para o início.

7) Análise dos resultados

Segundo os resultados obtidos, é possível verificar que o método de ordenação selection sort foi o mais eficiente dentre os três métodos utilizados na pesquisa. Além disso, é possível concluir que o método bubble sort se demonstrou como o menos eficiente dentre os três métodos utilizados. O experimento foi realizado em duas máquinas com diferentes configurações: a primeira com um processador Intel Core i5-1135G7 e 8GB de memória RAM, e a segunda possui um processador Intel Core i5-10210U e 8GB de memória RAM

8) Aplicação

8.1) Código da aplicação do bubble sort

```
import java.util.Date;

public class BubbleSort {

    public static void main(String[] args) {

        int limite[]={200000,400000,600000,800000,1000000,1200000};
        int i;


        VetorDinamico v;
        long ini, fim;
        for(i=0; i<10; i++){
            v = new VetorDinamico(limite[0]);
            v.geraElementos();
            ini = new Date().getTime();
            v.bubbleSort();
            fim = new Date().getTime();
            System.out.printf("%d, ", (fim-ini));
            v.resetVetor();

        }
        for(i=0; i<10; i++){
            v = new VetorDinamico(limite[1]);
            v.geraElementos();
            ini = new Date().getTime();
            v.bubbleSort();
            fim = new Date().getTime();
            System.out.printf("%d, ", (fim-ini));
            v.resetVetor();

        }
        for(i=0; i<10; i++){
            v = new VetorDinamico(limite[2]);
            v.geraElementos();
```

```
            ini = new Date().getTime();
            v.bubbleSort();
            fim = new Date().getTime();
            System.out.printf("%d, ", (fim-ini));
            v.resetVetor();

        }
        for(i=0; i<10; i++){
            v = new VetorDinamico(limite[3]);
            v.geraElementos();
            ini = new Date().getTime();
            v.bubbleSort();
            fim = new Date().getTime();
            System.out.printf("%d, ", (fim-ini));
            v.resetVetor();

        }
        for(i=0; i<10; i++){
            v = new VetorDinamico(limite[4]);
            v.geraElementos();
            ini = new Date().getTime();
            v.bubbleSort();
            fim = new Date().getTime();
            System.out.printf("%d, ", (fim-ini));
            v.resetVetor();

        }

    }
}
```

8.2) Código do método bubble sort

```
public void bubbleSort(){
        int j;
        for(int i=1; i<qtde; i++){
            for(j=0; j<qtde-i;j++){
                if (elementos[j]>elementos[j+1]){
                    int aux = elementos[j];
                    elementos[j] = elementos[j+1];
                    elementos[j+1] = aux;
                }
            }
        }
    }
```

8.3) Código da aplicação do insertion sort

```
import java.util.Date;

public class Insertion {

    public static void main(String[] args) {

        int limite[]={200000,400000,600000,800000,1000000,1200000};
        int i;


        VetorDinamico v;
        long ini, fim;
        for(i=0; i<10; i++){
            v = new VetorDinamico(limite[0]);
            v.geraElementos();
            ini = new Date().getTime();
            v.insertionSort();
            fim = new Date().getTime();
            System.out.printf("%d, ", (fim-ini));
            v.resetVetor();

        }
        for(i=0; i<10; i++){
            v = new VetorDinamico(limite[1]);
            v.geraElementos();
            ini = new Date().getTime();
            v.insertionSort();
            fim = new Date().getTime();
            System.out.printf("%d, ", (fim-ini));
            v.resetVetor();

        }
        for(i=0; i<10; i++){
            v = new VetorDinamico(limite[2]);
            v.geraElementos();
            ini = new Date().getTime();
            v.insertionSort();
            fim = new Date().getTime();
            System.out.printf("%d, ", (fim-ini));
            v.resetVetor();

        }
        for(i=0; i<10; i++){
            v = new VetorDinamico(limite[3]);
            v.geraElementos();
            ini = new Date().getTime();
            v.insertionSort();
```

```
            fim = new Date().getTime();
            System.out.printf("%d, ", (fim-ini));
            v.resetVetor();

        }
        for(i=0; i<10; i++){
            v = new VetorDinamico(limite[4]);
            v.geraElementos();
            ini = new Date().getTime();
            v.insertionSort();
            fim = new Date().getTime();
            System.out.printf("%d, ", (fim-ini));
            v.resetVetor();

        }

    }
}
```

8.4) Código do método insertion sort

```
public void insertionSort(){
        for (int j = 1; j< qtde; j++){
            int x = elementos[j];
            int i;
            for(i = j-1; i>=0 && elementos[i] > x; --i)
                elementos[i+1] = elementos[i];
            elementos[i+1] = x;


        }
    }
```

8.5) Código da aplicação do selection sort

```
import java.util.Date;

public class Selection {

    public static void main(String[] args) {

        int limite[]={20,400000,600000,800000,1000000,1200000};
        int i;
```

```
        VetorDinamico v;
        long ini, fim;
        for(i=0; i<10; i++){
            v = new VetorDinamico(limite[0]);
            v.geraElementos();
            ini = new Date().getTime();
            v.selectionSort();
            fim = new Date().getTime();
            System.out.printf("%d, ", (fim-ini));
            v.resetVetor();

        }
        for(i=0; i<10; i++){
            v = new VetorDinamico(limite[1]);
            v.geraElementos();
            ini = new Date().getTime();
            v.selectionSort();
            fim = new Date().getTime();
            System.out.printf("%d, ", (fim-ini));
            v.resetVetor();

        }
        for(i=0; i<10; i++){
            v = new VetorDinamico(limite[2]);
            v.geraElementos();
            ini = new Date().getTime();
            v.selectionSort();
            fim = new Date().getTime();
            System.out.printf("%d, ", (fim-ini));
            v.resetVetor();

        }
        for(i=0; i<10; i++){
            v = new VetorDinamico(limite[3]);
            v.geraElementos();
            ini = new Date().getTime();
            v.selectionSort();
            fim = new Date().getTime();
            System.out.printf("%d, ", (fim-ini));
            v.resetVetor();

        }
        for(i=0; i<10; i++){
            v = new VetorDinamico(limite[4]);
            v.geraElementos();
            ini = new Date().getTime();
            v.selectionSort();
            fim = new Date().getTime();
```

```
            System.out.printf("%d, ", (fim-ini));
            v.resetVetor();

        }

    }
}
```

8.6) Código do método selection sort

```
public void selectionSort (){
        int i, j, pos_menor, aux,  menor;
        for(i=0; i< qtde-1; i++){
            pos_menor = i;
            menor = elementos[i];
            for (j=i+1;j<qtde;j++){
                if(elementos[j]<menor){
                    menor = elementos[j];
                    pos_menor = j;
                }
            }
            aux = elementos[pos_menor];
            elementos[pos_menor] = elementos[i];
            elementos[i] = menor;
        }
    }
```