INSTITUTO FEDERAL DE CIÊNCIA E TECNOLOGIA DO CEARÁ
CAMPUS TIANGUÁ

ÁLEFE MEDEIROS DE OLIVEIRA
DANIEL ALBUQUERQUE CARVALHO
LUCIANA ALVES AMARAL

CONSTRUNÇÃO E ÁNALISE DE ALGORITMOS

TIANGUÁ
2019

ÁLEFE MEDEIROS DE OLIVEIRA
DANIEL ALBUQUERQUE CARVALHO
LUCIANA ALVES AMARAL

ATIVIDADE: ALGORITMOS DE ORDENAÇÃO

Trabalho para obtenção de nota na disciplina Construção e Análise de Algoritmos no bacharelado em Ciência da Computação apresentado no Instituto Federal de Educação, ciência e Tecnologia do Ceará.

Orientador: Prof. Adonias Caetano de Oliveira.

TIANGUÁ
2019

1. QUESTÕES TÉORICAS

1). Defina formalmente o problema de ordenação.

Os algoritmos de ordenação, servem para ordenar/organizar uma lista de números ou palavras de acordo com a sua necessidade. As linguagens de programação já possuem métodos de ordenação, mas é bom saber como funcionam os algoritmos, pois há casos de problemas em que o algoritmo de ordenação genérico não resolve, às vezes é necessário modificá-lo. Um dos problemas mais recorrentes em é o uso da memória e de tempo. Encontrar um algoritmo que consiga resolver problemas grandes em menor tempo e pouco consumo de memória.

2). Defina formalmente o problema de encontrar o menor valor de um vetor.

Se a linguagem utilizada não tiver alguma biblioteca ou função pré-definida o algoritmo terá que fazer comparações com todos números no vetor para encontrar o menor valor, assim gastando tempo de execução, em vetores pequenos não teria muito gasto de tempo, porém em grandes vetores poderia levar problemas e não satisfazendo o programador.

3). Forneça um exemplo de aplicação real que envolva o problema de ordenação e de encontrar o menor valor.

Criando uma variável v contendo um vetor de tamanho 9. Com os valores: 81 23 58 12 98 43 87 34 54 67

0 1 2 3 4 5 6 7 8 9

V = [81, 23, 58, 12, 98, 43, 87, 34, 54, 67]

Pode ordena-lo em ordem crescente, para isso verificando qual o menor valor do vetor colocamos na posição inicial e assim até todos estarem ordenados.

O menor número será o 12, e para isso o algoritmos fará 10 comparações e assim por diante até conseguir colocar os números em ordem crescente.

12, 10 comparações; 23, 9 comparações; 34, 8 comparações; 43, 7 comparações; 54, 6 comparações; 58, 5 comparações; 67, 5 comparações; 81, 3 comparações; 87, 2 comparações; 98, 1 comparações.

Vetor ordenado v=[12, 23, 34, 43, 54, 58, 67, 81, 87, 98], para ordená-lo foi preciso de 55 comparações.

1. QUESTÕES PRÁTICAS

4). Escreva um programa que receba valores em um vetor e imprima ORDENADO se eles estiverem em ordem crescente. O programa não aplica ordenação, mas apenas verifica.

```
def bubbleSort(arr):
    for j in range(1,len(arr)):
        for i in range(0,len(arr)-1):
            if(arr[i]>arr[i+1]):
                aux = arr[i+1]
                arr[i+1]=arr[i]
                arr[i] = aux
    return arr
def ordenado(arr):
    for i in range(1,len(arr)-1):
        if arr[i-1]>=arr[i]>=arr[i+1]:
            return 0
            break
    return 1
a = [0]*10
for i in range(0,10):
    a[i]=int(input())
if ordenado(a)==1:
    print("Já Ordenado")

else:
    print("Vetor Ordenado: ",bubbleSort(a))
```

5). Escreva um programa que receba um vetor ordenado e um número extra e insira esse número na sua posição correta no vetor ordenado, deslocando os outros números se necessário. Esta questão não pode usar nenhum método de ordenação.

```
def acrescenta(v,a):
    j=[0]*(len(v)+1)
    if a <= min(v):
        j[0]=a
        for i in range(1,len(j)):
```

```
                    j[i]=v[i-1]
        elif a >= max(v):
            j[len(j)-1]=a
            for i in range(0,len(v)):
                j[i]=v[i]
        else:
            j[0]=v[0]
            j[len(j)-1]=v[len(v)-1]
            for i in range(1,len(j)-1):
                if v[i-1]<=a<=v[i]:
                    j[i]=a
                    p=i
                else:
                    j[i]=v[i]
            for i in range(p,len(j)-1):
                j[i+1]=v[i]
        return j
v = [1,4,8,12,16,24,27,30,36,39,41,44,48,53]
print(v)
n = int(input("Entre com um número: "))
print(acrescenta(v,n))
```

6). Para cada problema a seguir escolha um método de ordenação distinto e implemente numa linguagem de programação adequada. As linguagens de programação permitidas são: Java, C, C++, Python e Ruby. Não é permitido usar mais de duas vezes a mesma linguagem.

a) PROBLEMA 01: Considere a seguinte estrutura:

struct pessoa{ int Matricula; char Nome[30]; float Nota; };

Faça uma função que dado um array de tamanho N dessa estrutura, ordene o array pelo campo escolhido pelo usuário.

Em Java:

```
import java.util.Scanner;

class Pessoa{
    private int Matricula;
```

```
        private String Nome;
        private double Nota;
        public Pessoa(){
        }
        public Pessoa(int Matricula,String Nome,double Nota){
                this.Matricula=Matricula;
                this.Nome=Nome;
                this.Nota=Nota;
        }
        public void setMatricula(int Matricula) {
                this.Matricula = Matricula;
        }
        public void setNome(String Nome){
                this.Nome=Nome;
        }
        public void setNota(double Nota){
                this.Nota=Nota;
        }
        public int getMatricula() {
                return Matricula;
        }
        public String getNome(){
                return Nome;
        }
        public double getNota(){
                return Nota;
        }

}
class Programa{
   public static void main(String[] args) {
        int n;
        Scanner ler = new Scanner(System.in);
      System.out.print("Entre com o tamanho: ");
```

```
n = ler.nextInt();
Pessoa x[] = new Pessoa[n];
String Nome; int Matricula; double Nota;
for (int i=0; i<n; i++) {
   x[i] = new Pessoa();
}
   for (int i=0; i<n; i++) {
            System.out.print("Entre o numero de Matricula: ");
            Matricula = ler.nextInt();
            x[i].setMatricula(Matricula);
            ler.nextLine();
            System.out.print("Entre o numero de Nome: ");
            Nome = ler.nextLine();
            x[i].setNome(Nome);
            System.out.print("Entre com a Nota: ");
            Nota = ler.nextDouble();
            x[i].setNota(Nota);
            }
            System.out.println();
            for (int i=0; i<n; i++) {
            System.out.println("Nº Matricula: "+x[i].getMatricula());
            System.out.println("Nome: "+x[i].getNome());
            System.out.println("Nota: "+x[i].getNota());
            System.out.println();
            }
            System.out.println("Digite 1 - Matricula");
            System.out.println("Digite 2 - Nome");
            System.out.println("Digite 3 - Nota");
            System.out.print("Como deseja ordenar? ");
            int escolha = ler.nextInt();
            int aux=0;
            String aux1;
            double aux2;
             switch (escolha) {
```

```
   case 1:
            for(int i = 0; i<n; i++){
for(int j = 0; j<n-1; j++){
   if(x[j].getMatricula() > x[j + 1].getMatricula()){
   aux = x[j].getMatricula();
   aux1 = x[j].getNome();
   aux2 = x[j].getNota();
   x[j].setMatricula(x[j+1].getMatricula());
   x[j].setNome(x[j+1].getNome());
   x[j].setNota(x[j+1].getNota());
      x[j+1].setMatricula(aux);
   x[j+1].setNome(aux1);
x[j+1].setNota(aux2);
            }
   }
   }
            break;
   case 2:
   for(int i = 0; i<n; i++){
for(int j = 0; j<n-1; j++){
   if (x[j].getNome().compareTo(x[j + 1].getNome())>0){
   aux = x[j].getMatricula();
   aux1 = x[j].getNome();
   aux2 = x[j].getNota();
   x[j].setMatricula(x[j+1].getMatricula());
   x[j].setNome(x[j+1].getNome());
   x[j].setNota(x[j+1].getNota());
      x[j+1].setMatricula(aux);
   x[j+1].setNome(aux1);
x[j+1].setNota(aux2);
            }
   }
   }
```

```
                    break;
                case 3:
                    for(int i = 0; i<n; i++){
                for(int j = 0; j<n-1; j++){
                if(x[j].getNota() > x[j + 1].getNota()){
                aux = x[j].getMatricula();
                aux1 = x[j].getNome();
                aux2 = x[j].getNota();
                x[j].setMatricula(x[j+1].getMatricula());
                x[j].setNome(x[j+1].getNome());
                x[j].setNota(x[j+1].getNota());
                   x[j+1].setMatricula(aux);
                x[j+1].setNome(aux1);
                x[j+1].setNota(aux2);
                    }
                }
                }
                    break;
            }
        System.out.println();
        for (int i=0; i<n; i++) {
            System.out.println("Nº Matricula: "+x[i].getMatricula());
            System.out.println("Nome: "+x[i].getNome());
            System.out.println("Nota: "+x[i].getNota());
            System.out.println();
            }
        }
}
```

b) PROBLEMA 02: Usando uma linguagem de programação orientada a objetos como Java ou C++, implemente uma classe Funcionário de atributos nome (String) e salario (float ou double). Os atributos são privados e há métodos get/set para cada atributo. Após isso, faça um programa que cadastre o nome e o salário de 5 funcionários armazenados em um array. Usando dois métodos de ordenação diferentes, liste todos os dados dos funcionários das seguintes formas:

I. em ordem crescente de salário; II. em ordem alfabética.

Em C++

```
#include <iostream>
#include <cstring>
using namespace std;

class Funcionario{
private:
	string Nome;
	float Salario;
public:
	Funcionario(){
	}
	void setNome(string _Nome){
		Nome = _Nome;
	}
	void setSalario(float _Salario){
		Salario = _Salario;
	}
	string getNome(){
		return Nome;
	}
	float getSalario(){
		return Salario;
	}
};
int main(){
	Funcionario funcionario[5];
	string nome;
	float salario;
	for (int i=0;i<5;i++){
		cin>>nome;
		funcionario[i].setNome(nome);
		cin>>salario;
```

```
            funcionario[i].setSalario(salario);
        }
        cout<<endl<<"Ordenado por Nome!"<<endl;
        for( int i = 1; i < 5; i++){
      for ( int j = 0; j < 4; j++){
        float aux2 = funcionario[j].getSalario();
                    string aux3 = funcionario[j].getNome();
                    char c1[30];  strcpy(c1, funcionario[j].getNome().c_str() );
                    char c2[30];  strcpy(c2, funcionario[j+1].getNome().c_str() );
        if(strcmp(c1,c2) > 0){
           aux3=funcionario[j].getNome();
           funcionario[j].setNome(funcionario[j+1].getNome());
           funcionario[j+1].setNome(aux3);
           aux2=funcionario[j].getSalario();
           funcionario[j].setSalario(funcionario[j+1].getSalario());
           funcionario[j+1].setSalario(aux2);
        }
     }
}
for (int i=0;i<5;i++){
            cout<<funcionario[i].getNome()<<endl;
            cout<<funcionario[i].getSalario()<<endl;
        }

        cout<<endl<<"Ordenado por Salário!"<<endl;
        /*Insertion Sort*/
        for (int i = 1; i < 5; i++) {
            float escolhido = funcionario[i].getSalario();
            string aux = funcionario[i].getNome();
            int j = i - 1;
            while ((j >= 0) && (funcionario[j].getSalario() > escolhido)) {
                funcionario[j+1].setSalario(funcionario[j].getSalario());
                funcionario[j+1].setNome(funcionario[j].getNome());
                j--;
```

```
            }
            funcionario[j + 1].setSalario(escolhido);
            funcionario[j + 1].setNome(aux);
        }
        for (int i=0;i<5;i++){
            cout<<funcionario[i].getNome()<<endl;
            cout<<funcionario[i].getSalario()<<endl;
        }

return 0;
}
```

c) PROBLEMA 03: Faça um programa que cadastre 10 números, ordene-os e em seguida encontre e mostre:

I. o menor número e quantas vezes ele aparece no vetor; II. o maior número e quantas vezes ele aparece no vetor.

Em Python:

```
def bubbleSort(a):
    if len(a) <= 1:
        sa = a
    else:
        for j in range(0,len(a)):
            for i in range(0,len(a)-1):
                if a[i]>a[i+1]:
                    aux = a[i+1]
                    a[i+1]=a[i]
                    a[i]=aux
        sa=a
    return sa
def maior(a):
    m=0
    for i in range(0,len(a)):
        if(a[9]==a[i]):
            m=m+1
    return m
```

```
def menor(a):
    n=0
    for i in range(0,len(a)):
        if(a[0]==a[i]):
            n=n+1
    return n
v=[0]*10
for i in range(0,10):
    v[i]=int(input())
print(bubbleSort(v))
print("Maior: ",v[9], "aparece ", maior(v), "vezes.")
print("Menor: ",v[0], "aparece ", menor(v), "vezes.")
```

d) PROBLEMA 04: Usando uma linguagem de programação orientada a objetos como Java ou C++, implemente uma classe Aluno de atributos nome (String), nota1 e nota2 (float ou double). Os atributos são privados e há métodos get/set para cada atributo. Depois faça um programa que cadastre 8 alunos em array. Para cada aluno devem ser cadastrados: nome, nota1 e nota2. Usando três métodos de ordenação diferentes, liste todos os dados dos alunos das seguintes formas:

I. Em ordem crescente de média ponderada das notas, tendo a primeira nota peso 2 e a segunda peso 3.

II. Em ordem crescente pela nota 1.

III. Finalmente, considerando que para ser aprovado o aluno dever ter no mínimo média 7 liste, em ordem alfabética, os alunos reprovados.

Em C++

```
#include <iostream>
#include <cstring>
using namespace std;

class Aluno{
private:
    string Nome;
    float Nota1;
    float Nota2;
    float Media;
```

```
public:
        Aluno(){
        }
        void setNome(string _Nome){
                Nome = _Nome;
        }
        void setNota1(float _Nota1){
                Nota1 = _Nota1;
        }
        void setNota2(float _Nota2){
                Nota2 = _Nota2;
        }
        void setNota(float _Nota1,float _Nota2){
                Nota1 = _Nota1;
                Nota2 = _Nota2;
                setMedia();
        }
        void setMedia(){
                Media = (Nota1*2+Nota2*3)/5;
        }
        string getNome(){
                return Nome;
        }
        float getNota1(){
                return Nota1;
        }
        float getNota2(){
                return Nota2;
        }
        float getMedia(){
                return Media;
        }
};
int main(){
```

```
        Aluno aluno[8];
        string nome;
        float salario,nota1,nota2,media;
        for (int i=0;i<8;i++){
                cin>>nome;
                aluno[i].setNome(nome);
                cin>>nota1;
                cin>>nota2;
                aluno[i].setNota(nota1,nota2);
        }
        cout<<endl<<"Insertion Sort - Ordenado por Media!"<<endl;
        int i, j;
        float key;
        Aluno aux;
for (i = 1; i < 8; i++)
{
   key = aluno[i].getMedia();
   aux = aluno[i];
   j = i - 1;
                while (j >= 0 && aluno[j].getMedia() > key)
   {
      aluno[j + 1] = aluno[j];
      j = j - 1;
   }
   aluno[j + 1] = aux;
}
for (int i = 0;i<8;i++){
        cout<<aluno[i].getNome()<<endl;
        cout<<aluno[i].getMedia()<<endl;
}
        cout<<endl<<"Shell Sort - Ordenado por Nota 1!"<<endl;
        for (int gap = 8/2; gap > 0; gap /= 2)
{
   for (int i = gap; i < 8; i += 1)
```

```
   {
      float temp = aluno[i].getNota1();
      Aluno aux = aluno[i];

      for (j = i; j >= gap && aluno[j - gap].getNota1() > temp; j -= gap) {
         aluno[j] = aluno[j - gap];
      }

      aluno[j] = aux;
   }
}
for (int i = 0;i<8;i++){
      cout<<aluno[i].getNome()<<endl;
      cout<<aluno[i].getNota1()<<endl;
}

      cout<<endl<<"Bubble Sort - Ordenado por Nome (Alunos Reprovados)!"<<endl;
      for( int i = 1; i < 8; i++){
    for ( int j = 0; j < 7; j++){
      Aluno aux = aluno[j];
                        char c1[30];  strcpy(c1, aluno[j].getNome().c_str() );
                        char c2[30];  strcpy(c2, aluno[j+1].getNome().c_str() );
      if(strcmp(c1,c2) > 0){
         aux=aluno[j];
         aluno[j]=aluno[j+1];
         aluno[j+1]=aux;
      }
   }
}
for (int i = 0;i<8;i++){
      if (aluno[i].getMedia()<7){
      cout<<aluno[i].getNome()<<endl;
      cout<<aluno[i].getMedia()<<endl;
}
```

```
    }

return 0;
}
```

e) PROBLEMA 05: Crie um programa que dado uma string, coloque as letras dela em ordem crescente.

Em C:

```
#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
#include<string.h>
void merge(char arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 =  r - m;
    int L[n1], R[n2];
    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1+ j];

    i = 0;
    j = 0;
    k = l;
    while (i < n1 && j < n2)
    {
        if (L[i] <= R[j])
        {
            arr[k] = L[i];
            i++;
        }
        else
        {
```

```
            arr[k] = R[j];
            j++;
        }
        k++;
    }
    while (i < n1)
    {
        arr[k] = L[i];
        i++;
        k++;
    }
    while (j < n2)
    {
        arr[k] = R[j];
        j++;
        k++;
    }
}
void mergeSort(char arr[], int l, int r)
{
    if (l < r)
    {
        int m = l+(r-l)/2;
        mergeSort(arr, l, m);
        mergeSort(arr, m+1, r);

        merge(arr, l, m, r);
    }
}

void printArray(char A[], int size)
{
    int i;
    for (i=0; i < size; i++)
```

```
        printf("%c", A[i]);
    printf("\n");
}

int main()
{
    char arr[30];

    scanf("%s",arr);
    int arr_size = strlen(arr);
    printf("Given array is \n");
    printArray(arr, arr_size);

    mergeSort(arr, 0, arr_size - 1);

    printf("\nSorted array is \n");
    printArray(arr, arr_size);
    return 0;
}
```

f) PROBLEMA 06: Faça um programa que leia N nomes e ordene-os pelo tamanho.

Em Java:

```
import java.util.Scanner;
public class Main
{
        public static void main(String[] args) {
            Scanner ler = new Scanner(System.in);
            int n,i,j;
                System.out.print("Entre com o tamanho do Vetor: ");
                n = ler.nextInt();
                int v[] = new int[n];
                String nomes[] = new String[n];
                for (i = 0; i < n; i++){
                    System.out.printf("Entre com a palavra No %d: ",i);
                    nomes[i] = ler.next();
```

```
            v[i]=nomes[i].length();
        }
         int min, temp;
         String aux;

    for (i = 0; i < n-1; i++)
    {
       min = i;
       for (j = i+1; j < n; j++)
       {
          if (v[j] < v[min])
             min = j;
       }
       temp = v[i];
       aux = nomes[i];
       nomes[i]=nomes[min];
       v[i] = v[min];
       v[min] = temp;
       nomes[min] = aux;
    }
          for (i = 0; i < n; i++){
              System.out.printf("%s \n", nomes[i]);
          }
  }
}
```

7). Escolha 4 métodos de ordenação, no qual pelo menos um é linear. Crie um programa que dado uma string, coloque as letras dela em ordem decrescente usando os 4 algoritmos. Os métodos podem ser implementados na mesma linguagem.

```
def countingSort(arr, exp1):
  n = len(arr)
  output = [0] * (n)
  count = [0] * (10)
  for i in range(0, n):
    index = int(arr[i]/exp1)
    count[int((index)%10)] += 1
  for i in range(1,10):
```

```
    count[i] += count[i-1]
  i = n-1
  while i>=0:
    index = (int(arr[i]/exp1))
    output[ count[ int((index)%10) ] - 1] = arr[i]
    count[ (index)%10 ] -= 1
    i -= 1
  i = 0
  for i in range(0,len(arr)):
    arr[i] = output[i]
def radixSort(arr):
    max1 = max(arr)
    exp = 1
    while max1/exp > 0:
      countingSort(arr,exp)
      exp *= 10
    nome= [0]*len(arr)
    for i in range(0,len(arr)):
      nome[i]=chr(arr[i])
    return nome
def bubbleSort(v):
  arr = v
  for j in range(1,len(arr)):
    for i in range(0,len(arr)-1):
      if(arr[i]>arr[i+1]):
        aux = arr[i+1]
        arr[i+1]=arr[i]
        arr[i] = aux
  nome= [0]*len(arr)
  for i in range(0,len(arr)):
    nome[i]=chr(arr[i])
  return nome
def insertionSort(v):
  arr = v
  for i in range(0,len(arr)):
    x = arr[i]
    j = i-1
    while j>=0 and arr[j] >x:
      arr[j+1]=arr[j]
      j=j-1
      arr[j+1]=x
  nome= [0]*len(arr)
  for i in range(0,len(arr)):
    nome[i]=chr(arr[i])
  return nome
def shellSort(v):
    arr = v
    n = len(arr)
    gap = int(n/2)
```

```
    while gap > 0:
        for i in range(gap,n,1):
            temp = arr[i]
            j = i
            while  j >= gap and arr[j-gap] >temp:
                arr[j] = arr[j-gap]
                j -= gap
                arr[j] = temp
        gap = int(gap/2)
    nome= [0]*len(arr)
    for i in range(0,len(arr)):
      nome[i]=chr(arr[i])
    return nome
def main():
  n = str(input("Entre com a palavra: "))
  v= [0]*len(n)
  for i in range(0,len(n)):
    v[i]=ord(n[i])
  iS=bS=sS=rS=v
  print(insertionSort(iS))
  print(bubbleSort(bS))
  print(shellSort(sS))
  print(radixSort(rS))

if __name__ == "__main__":
    main()
```

8). Implementar os algoritmos abaixo para ordenar uma lista encadeada.

I. Selection Sort II. Insertion-Sort III. Bubble-Sort. IV. Shell-Sort V. Quick-Sort.

```
from random import randint
from selection_sort import selectionSort
from insertion_sort import insertionSort
from bubble_sort import bubbleSort
from quick_sort import quickSort
from shell_sort import shellSort
def main():
        n = int(input("Entre com o tamanho de vetor: "))
        v = [0]*n
        for i in range(0,n):
                v[i] = randint(0,100)
        print(v)
```

```
    sS=iS=bS=Ss=qS=rS=v[:]
    print(selectionSort(sS))
    print(insertionSort(iS))
    print(bubbleSort(bS))
    print(quickSort(qS,0,n-1))
    print(shellSort(Ss))

if __name__ == "__main__":
   main()
```

9). Implemente em Python todos os nove algoritmos ensinados em sala de aula, realizando experimentos que avaliem o tempo de execução para ordenar de acordo com as seguintes regras:

I. Serão nove vetores com os seguintes tamanhos para cada um: 1000, 3000, 6000, 9000, 12000, 15000, 18000, 21000, 24000. II. Os valores armazenados nos nove vetores serão números inteiros gerados aleatoriamente.

III. Usar a biblioteca "matplotlib.pyplot" IV. Plotar um gráfico comparando o tempo de execução dos algoritmos de acordo com o tamanho do vetor.

O seguinte código exemplifica a solução da questão.

```
import random
import math
import timeit
import matplotlib.pyplot as plt
from selection_sort import selectionSort
from insertion_sort import insertionSort
from bubble_sort import bubbleSort
from quick_sort import quickSort
from shell_sort import shellSort
from radix_sort import radixSort
from counting_sort import countingSort
from bucket_sort import bucketSort
from merge_sort import mergeSort

timeSelection = []
timeInsertion = []
```

```
timeBubble = []
timeQuick = []
timeShell = []
timeRadix = []
timeCounting = []
timeBucket = []
timeMerge = []

tamanhos = [1000, 3000, 6000, 9000, 12000, 18000, 21000, 24000]
def randArray(length):
        array = []
        tmp = 0
        while tmp < length:
                num = random.randint(1, length*10)
                if num not in array:
                        array.append(num)
                        tmp += 1
        return array

def timePopulate():
        for numTamanhos in tamanhos:
                base = []
                base = randArray(numTamanhos)

                _tmp = list(base)
                timeSelection.append(timeit.timeit("selectionSort({})".format(_tmp), \
                setup="from __main__ import selectionSort", \
                number=1))

                _tmp = list(base)
                timeInsertion.append(timeit.timeit("insertionSort({})".format(_tmp), \
                setup="from __main__ import insertionSort", \
                number=1))
```

```
        _tmp = list(base)
        timeBubble.append(timeit.timeit("bubbleSort({})".format(_tmp), \
        setup="from __main__ import bubbleSort", \
        number=1))

        _tmp = list(base)
        timeQuick.append(timeit.timeit("quickSort({})".format(_tmp), \
        setup="from __main__ import quickSort", \
        number=1))

        _tmp = list(base)
        timeShell.append(timeit.timeit("shellSort({})".format(_tmp), \
setup="from __main__ import shellSort", \
number=1))

        _tmp = list(base)
        timeRadix.append(timeit.timeit("radixSort({})".format(_tmp), \
        setup="from __main__ import radixSort", \
number=1))

        _tmp = list(base)
        timeCounting.append(timeit.timeit("countingSort({})".format(_tmp), \
setup="from __main__ import countingSort", \
number=1))

        _tmp = list(base)
        timeBucket.append(timeit.timeit("bucketSort({})".format(_tmp), \
setup="from __main__ import bucketSort", \
number=1))

        _tmp = list(base)
        timeMerge.append(timeit.timeit("mergeSort({})".format(_tmp), \
        setup="from __main__ import mergeSort", \
        number=1))
```

```
            print("Lista de Tamanho {}".format(numTamanhos),"ordenada")

def axis():
        timePopulate()

        plt.plot(tamanhos, timeSelection, label="Selection Sort")
        plt.plot(tamanhos, timeInsertion, label="Insertion Sort")
        plt.plot(tamanhos, timeBubble, label="Bubble Sort")
        plt.plot(tamanhos, timeQuick, label="QuickSort")
        plt.plot(tamanhos, timeShell, label="ShellSort")
        plt.plot(tamanhos, timeRadix, label="RadixSort")
        plt.plot(tamanhos, timeCounting, label="CountingSort")
        plt.plot(tamanhos, timeBucket, label="BucketSort")
        plt.plot(tamanhos, timeMerge, label="MergeSort")

def graphic():
        plt.legend(loc='upper center', shadow=True).get_frame().set_facecolor('0.90')
        plt.xlabel('Tamanho(int)')
        plt.ylabel('Tempo(s)')
        plt.show()

def main():
        axis()
        graphic()

if __name__ == "__main__":
        main()
```