INSTITUTO FEDERAL
Ceará
Campus Tianguá

TRABALHO COMPUTACIONAL 01 – ORDENAÇÃO E BUSCA	
CURSO	Bacharelado em Ciência da Computação
DISCIPLINA	Construção e Análise de Algoritmos
PROFESSOR	Adonias Caetano de Oliveira
EQUIPE	**NO MÁXIMO 04 PESSOAS**

INSTRUÇÕES:

I. O não cumprimento sobre a quantidade máxima de alunos por equipe resulta em -2 pontos na atividade;
II. Todos os códigos-fontes devem ser compactados em arquivo ".zip" ou ".rar".
III. A entrega do trabalho pode ser por anexo no classroom ou no e-mail ou mesmo compartilhado pelo Google Drive;
IV. Não compartilhe arquivos executáveis;
V. Proibido uso de bibliotecas ou APIs que implementam os métodos de ordenação e busca;
VI. Envie a imagem do gráfico gerado;
VII. Comente as linhas importantes dos códigos e seus métodos;
VIII. Não será aceito nenhuma resposta das questões anotadas em folhas de caderno ou A4 ou impressa ou digitalizadas ou outro tipo não autorizado nestas instruções;
IX. Compartilhar ou enviar ao e-mail: adonias.oliveira@ifce.edu.br
X. Será atribuída a mesma nota para cada membro de uma mesma equipe. Caso o líder da equipe discorde disto deve executar as seguintes ações:
   - Enviar um e-mail com cópias para todos os membros da equipe;
   - No corpo do e-mail deve atribuir valores de 0% até 100% como nível de contribuição que cada membro deu no trabalho;
   - Descrever a metodologia de divisão das tarefas
   - Relatar os problemas que teve com outro membro que prejudicou o trabalho;
XI. Após a correção desta atividade, em caso de discordância da nota E negativa do professor em atender suas solicitações, é preciso seguir as orientações do artigo 96 do ROD do IFCE.

MÉTODOS DE ORDENAÇÃO

1) 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. Espera-se ao final da questão que você tenha utilizado 09 métodos de ordenação diferentes.

a) **PROBLEMA 01:** Considere a seguinte estrutura na linguagem C:

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

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

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.

c) **PROBLEMA 03:** Faça um programa que cadastre 10 números, ordene-os e em seguida encontre e mostre (Dica: Use C ou Python ou Ruby):

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

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.

e) **PROBLEMA 05:** Crie um programa que dado uma string, coloque as letras dela em ordem crescente (Dica: Use C ou Python ou Ruby).

f) **PROBLEMA 06:** Faça um programa que leia N nomes e ordene-os pelo tamanho (Dica: Use C ou Python ou Ruby).

2) 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.

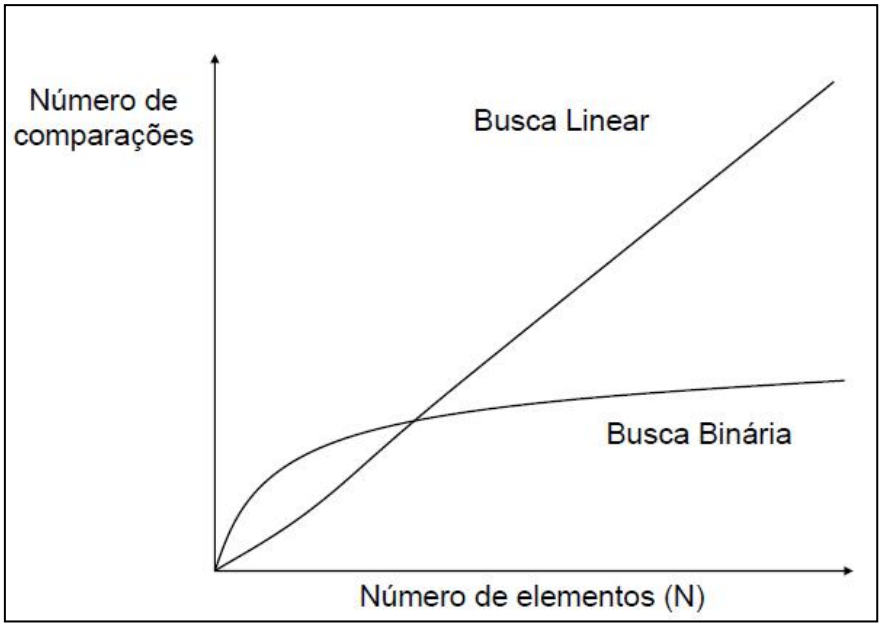
O seguinte código exemplifica a solução da questão.

```
import matplotlib.pyplot as plt
import random
import timeit

#Metodo da bolha
def bubbleSort(lista):
    i = 0
    while i < len(lista) :
        j = 0
        while j < len(lista) - 1:
            if lista[j] > lista[j + 1]:
                temp = lista[j]
                lista[j] = lista[j + 1]
                lista[j + 1] = temp
            j += 1
        i += 1


# gera uma lista de inteiros de modo aleatorio
def geraLista(tam):
    random.seed()
    i = 0
    lista = []
    while i < tam:
        lista.append(random.randint(1, tam))
        i += 1

    return lista

tamanhos = [1000, 3000, 6000, 9000, 12000, 15000, 18000, 21000, 24000]

temposBurbbleSort = []

for tamanho in tamanhos :
    lista = geraLista(tamanho)

    lista_teste = list(lista) #copia a lista
    temposBurbbleSort.append(timeit.timeit("bubbleSort({})".format(lista_teste),setup="from __main__ import bubbleSort", number=1))
    print( "Lista de tamanho {}".format(tamanho), " ordenada")


fig, ax = plt.subplots()
#ax.semilogx(tamanhos, temposBurbbleSort, label="Bubble Sort")
#ax.loglog(tamanhos, temposBurbbleSort, label="Bubble Sort")
ax.plot(tamanhos, temposBurbbleSort, label="Bubble Sort")
```

BUSCA LINEAR E BINÁRIA

3) Implemente em Python a busca linear convencional e com sentinela, busca binária convencional e rápida realizando experimentos que avaliem o tempo de execução para encontrar uma chave de acordo com as seguintes regras:

Figura 1 – Exemplo do gráfico



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. Aplique um método de ordenação linear nas buscas do tipo binária;
IV. O elemento chave a ser buscado pode ser informado pelo usuário ou gerado aleatoriamente;
V. Usar a biblioteca “matplotlib.pyplot”
VI. Conforme a Figura 1, plote um gráfico comparando o tempo de execução dos algoritmos de acordo com o tamanho do vetor
VII. Faça comentários objetivos e sucintos sobre os gráficos.