**CURSO TÉCNICO DE INFORMÁTICA PARA INTERNET**
**INTEGRADO AO ENSINO MÉDIO**
**DISCIPLINA DE PROGRAMAÇÃO II**
**PROFESSORA ALESSANDRA VILARINHO**

# APNP 6

**LARA AGUILAR DE AMORIM**

**27 DE JULHO DE 2021**

# RELATÓRIO

O presente trabalho tem como objetivo a experimentação empírica de medição de tempo de execução e comparação de resultados via gráficos de barras e tabelas quanto aos métodos de ordenação e busca.

## EXPERIMENTO DE ORDENAÇÃO:

Objetivo: Medir o tempo de execução dos algoritmos de ordenação em memória primária.

### Ordenação por Seleção:

A ordenação por seleção identifica o menor elemento do vetor e o movimenta para a esquerda, e percorre a lista assim sucessivamente.

```
'''Função ordena uma lista através do método de Seleção
RECEBE uma lista e RETORNA a lista ordenada'''
def selectionSort(lista):
    #processamento
    for i in range(len(lista) - 1):
        posMenor = i

        for k, elemento_analisado in enumerate(lista[i+1:], start=i+1):
            if elemento_analisado < lista[posMenor]:
                posMenor = k

        lista[posMenor], lista[i] = lista[i], lista[posMenor]
    #fim do processamento
    return lista
    #fim da funcao
```

### Ordenação por Inserção:

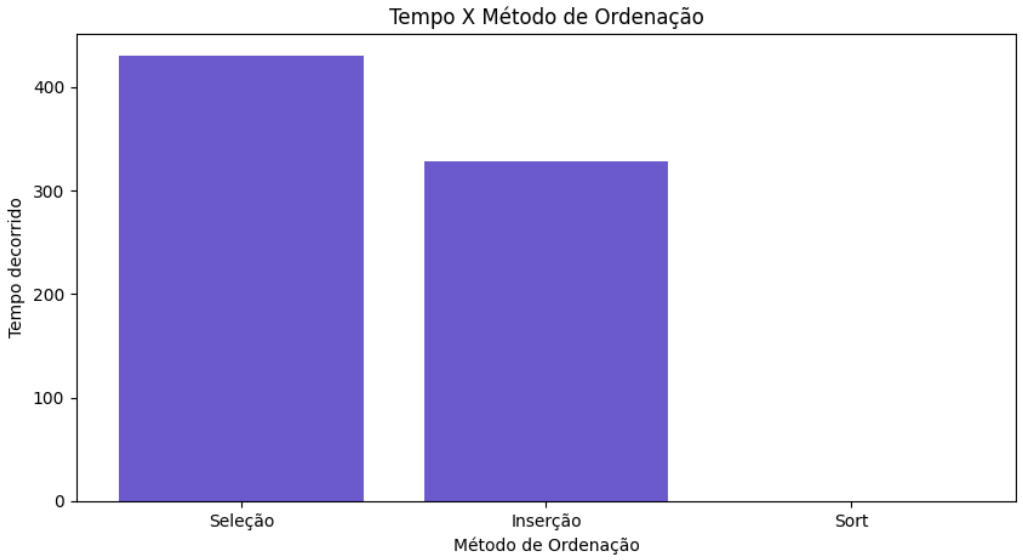
A ordenação por inserção percorre o vetor posicionando os menores itens à esquerda.

```
'''Função ordena uma lista através do método de Inserção
RECEBE uma lista e RETORNA a lista ordenada'''
def insertionSort(lista):
    #processamento
    for i, elemento in enumerate(lista[:], start=0):
        j = i - 1
        while j >= 0 and lista[j] > elemento:
            lista[j+1] = lista[j]
            j = j - 1
        lista[j+1] = elemento
    #fim processamento
```

```
    return lista
    #fim da funcao
```

**Gráfico e tabela obtidos:**

Gráfico: Tempo X Método de Ordenação, para a lista de tamanho 100.000.




Dicionário e tabela: Tempo de processamento de acordo com o método de ordenação e o tamanho da lista.

```
{'Seleção': [0.0, 0.0, 0.03125, 3.75, 430.046875], 'Inserção': [0.0, 0.0, 0.046875, 2.765625, 328.9375], 'Sort': [0.0, 0.0, 0.0, 0.0, 0.03125]}
```

| | 10 | 100 | 1000 | 10000 | 100000 |
|---|---|---|---|---|---|
| Seleção | 0.00000 | 0.00000 | 0.03125 | 3.75000 | 430.04688 |
| Inserção | 0.00000 | 0.00000 | 0.04688 | 2.76562 | 328.93750 |
| Sort | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.03125 |

## EXPERIMENTO DE BUSCA:

Tarefa: Medir o tempo de execução dos algoritmos de busca sequencial e busca binária para buscar um valor existente na lista e buscar um valor não existente na lista.

**Busca Sequencial:**

A busca sequencial percorre a lista do início ao fim para encontrar o elemento informado.

Isso ocorre através do loop while que repete a condicional até encontrar a posição (pos) do elemento ou, caso o item não esteja na lista ainda, até percorrer toda a lista.

```
'''A função faz a busca sequencial de um elemento dentro de uma lista
RECEBE uma lista e o elemento a ser procurado
```

```
RETORNA a posicao do elemento na lista'''
def buscaSequencial(lista, elemento):
    #declaração de variavel
    pos = int(0)
    i = int(0)

    # processamento
    pos = -1
    while i < len(lista) and pos == -1:
        if lista[i] == elemento:
            pos = i
        i += 1
    # fim processamento
    return pos
    # fim da funcao
```

**Busca Binária:**
A busca binária divide uma lista ordenada ao meio e verifica se o elemento na posição central é menor, maior ou igual ao elemento a ser procurado. Caso for menor, o script busca a posição do elemento na primeira parte da lista, se for maior, busca na segunda parte da lista e se for igual, a posição central é a posição do elemento procurado.

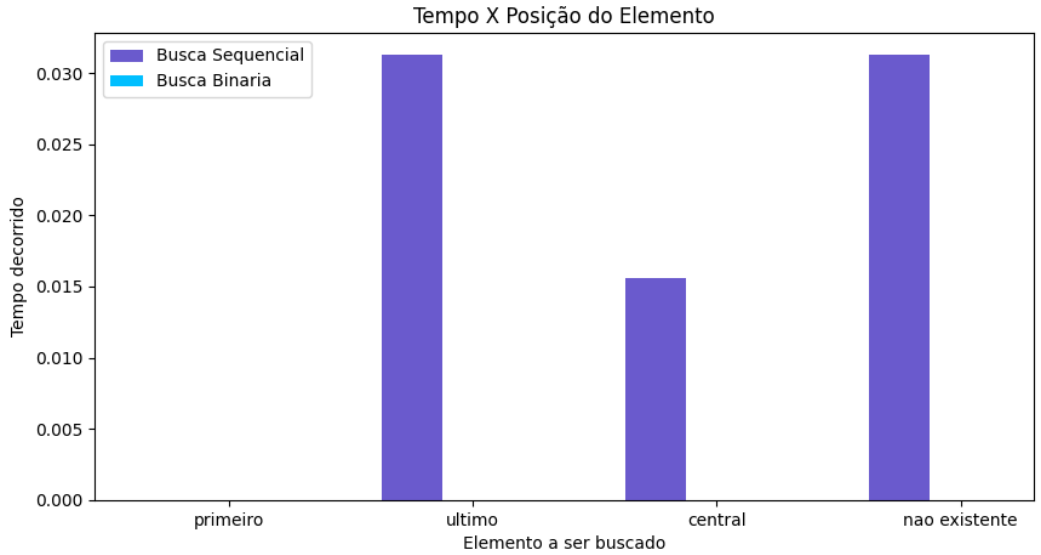
```
'''A função faz a busca binária de um elemento dentro de uma lista ordenada
RECEBE uma lista ordenada e o elemento a ser procurado
RETORNA a posicao do elemento na lista'''
def buscaBinaria(lista, elemento):
    # declaracao de variavel
    pos = int(0)
    inicio = int(0)
    meio = int(0)
    fim = len(lista) - 1
    pos = -1

    #processamento
    while inicio <= fim and pos == -1:
        meio = (inicio + fim) // 2

        if elemento == lista[meio]:
            pos = meio
        elif elemento < lista[meio]:
            fim = meio - 1
        else:
            inicio = meio + 1
    # fim do processamento
    return pos
    #fim da funcao
```

**Gráfico e tabela obtidos:**

Gráfico: Tempo X Método de Ordenação, para a lista de tamanho 100.000.




Dicionário e tabela: Tempo de processamento de acordo com o método de busca e a posição do elemento na lista não ordenada.

```
{'sequencial': [0.0, 0.03125, 0.015625, 0.03125], 'binária': [0.0, 0.0, 0.0, 0.0]}
```

| | primeiro | ultimo | central | nao existente |
|---|---|---|---|---|
| sequencial | 0.00000 | 0.03125 | 0.01562 | 0.03125 |
| binária | 0.00000 | 0.00000 | 0.00000 | 0.00000 |