Relatório Avaliação de Ordenação e Pesquisa de Dados

Integrantes grupo:

Lucas Fadini Carbonaro
João Pedro Michelim
Abner de Oliveira Magalhães
Nichollas Henrique Lázaro Magalhães

Algoritmos escolhidos:

- Selection Sort, Bubble Sort

Conjunto de Dados:

- **Melhor caso:**
  {1, 10, 13, 14, 29, 37}
- **Caso médio:**
  {29, 10, 14, 37, 13, 1}
- **Pior caso:**
  {37, 29, 14, 13, 10, 1}

Implementação:

- Linguagem de programação de sua preferência (**Python**).

Bubble Sort - DECRESCENTE

```python
def bubble_sort_decrescente(lista):
    n = len(lista)
    trocas = 0  # Contador de trocas

    print(f"Estado inicial: {lista}")

    for i in range(n - 1):
        for j in range(n - 1 - i):
            if lista[j] < lista[j + 1]:
                lista[j], lista[j + 1] = lista[j + 1], lista[j]
                trocas += 1
                print(f"Troca {trocas}: índice {j} <-> {j + 1} -> {lista}")

    print(f"Vetor ordenado (decrescente): {lista}")
    print(f"Total de trocas: {trocas}")
    print("-" * 50)
    return lista

# Casos de teste
melhor_caso = [1, 10, 13, 14, 29, 37]
caso_medio  = [29, 10, 14, 37, 13, 1]
pior_caso   = [37, 29, 14, 13, 10, 1]

# Execução dos casos
print(" 🔹 MELHOR CASO (já decrescente):")
bubble_sort_decrescente(pior_caso[:])

print(" 🔹 CASO MÉDIO (já decrescente):")
bubble_sort_decrescente(caso_medio[:])

print(" 🔹 PIOR CASO (já decrescente):")
bubble sort decrescente(melhor caso[:])
```

```
 🔹 MELHOR CASO (já decrescente):
Estado inicial: [37, 29, 14, 13, 10, 1]
Vetor ordenado (decrescente): [37, 29, 14, 13, 10, 1]
Total de trocas: 0
--------------------------------------------------
```

🔹 CASO MÉDIO (já decrescente):

```
Estado inicial: [29, 10, 14, 37, 13, 1]
Troca 1: índice 1 <-> 2 -> [29, 14, 10, 37, 13, 1]
Troca 2: índice 2 <-> 3 -> [29, 14, 37, 10, 13, 1]
Troca 3: índice 3 <-> 4 -> [29, 14, 37, 13, 10, 1]
Troca 4: índice 1 <-> 2 -> [29, 37, 14, 13, 10, 1]
Troca 5: índice 0 <-> 1 -> [37, 29, 14, 13, 10, 1]
Vetor ordenado (decrescente): [37, 29, 14, 13, 10, 1]
Total de trocas: 5
```

🔹 PIOR CASO (já decrescente):

```
Estado inicial: [1, 10, 13, 14, 29, 37]
Troca 1: índice 0 <-> 1 -> [10, 1, 13, 14, 29, 37]
Troca 2: índice 1 <-> 2 -> [10, 13, 1, 14, 29, 37]
Troca 3: índice 2 <-> 3 -> [10, 13, 14, 1, 29, 37]
Troca 4: índice 3 <-> 4 -> [10, 13, 14, 29, 1, 37]
Troca 5: índice 4 <-> 5 -> [10, 13, 14, 29, 37, 1]
Troca 6: índice 0 <-> 1 -> [13, 10, 14, 29, 37, 1]
Troca 7: índice 1 <-> 2 -> [13, 14, 10, 29, 37, 1]
Troca 8: índice 2 <-> 3 -> [13, 14, 29, 10, 37, 1]
Troca 9: índice 3 <-> 4 -> [13, 14, 29, 37, 10, 1]
Troca 10: índice 0 <-> 1 -> [14, 13, 29, 37, 10, 1]
Troca 11: índice 1 <-> 2 -> [14, 29, 13, 37, 10, 1]
Troca 12: índice 2 <-> 3 -> [14, 29, 37, 13, 10, 1]
Troca 13: índice 0 <-> 1 -> [29, 14, 37, 13, 10, 1]
Troca 14: índice 1 <-> 2 -> [29, 37, 14, 13, 10, 1]
Troca 15: índice 0 <-> 1 -> [37, 29, 14, 13, 10, 1]
Vetor ordenado (decrescente): [37, 29, 14, 13, 10, 1]
Total de trocas: 15
```

Bubble Sort - CRESCENTE

```python
def bubble_sort(lista):
    n = len(lista)
    trocas = 0  # Contador de trocas

    print(f"Estado inicial: {lista}")

    for i in range(n - 1):
        for j in range(n - 1 - i):
            if lista[j] > lista[j + 1]:
                # Troca os elementos
                lista[j], lista[j + 1] = lista[j + 1], lista[j]
                trocas += 1
                print(f"Troca {trocas}: índice {j} <-> {j + 1} -> {lista}")

    print(f"Vetor ordenado crescente: {lista}")
    print(f"Total de trocas: {trocas}")
    print("-" * 50)
    return lista

# Casos de teste
melhor_caso = [1, 10, 13, 14, 29, 37]
caso_medio = [29, 10, 14, 37, 13, 1]
pior_caso   = [37, 29, 14, 13, 10, 1]

print(" 🔹 MELHOR CASO (já ordenado):")
bubble_sort(melhor_caso[:])  # usar [:] para copiar a lista e não modificar a original

print(" 🔹 CASO MÉDIO (já ordenado):")
bubble_sort(caso_medio[:])

print(" 🔹 PIOR CASO (já ordenado):")
bubble_sort(pior_caso[:])
```

🔹 MELHOR CASO (já ordenado):

```
Estado inicial: [1, 10, 13, 14, 29, 37]
Vetor ordenado crescente: [1, 10, 13, 14, 29, 37]
Total de trocas: 0
```

🔹 CASO MÉDIO (já ordenado):

```
Estado inicial: [29, 10, 14, 37, 13, 1]
Troca 1: índice 0 <-> 1 -> [10, 29, 14, 37, 13, 1]
Troca 2: índice 1 <-> 2 -> [10, 14, 29, 37, 13, 1]
Troca 3: índice 3 <-> 4 -> [10, 14, 29, 13, 37, 1]
Troca 4: índice 4 <-> 5 -> [10, 14, 29, 13, 1, 37]
Troca 5: índice 2 <-> 3 -> [10, 14, 13, 29, 1, 37]
Troca 6: índice 3 <-> 4 -> [10, 14, 13, 1, 29, 37]
Troca 7: índice 1 <-> 2 -> [10, 13, 14, 1, 29, 37]
Troca 8: índice 2 <-> 3 -> [10, 13, 1, 14, 29, 37]
Troca 9: índice 1 <-> 2 -> [10, 1, 13, 14, 29, 37]
Troca 10: índice 0 <-> 1 -> [1, 10, 13, 14, 29, 37]
Vetor ordenado crescente: [1, 10, 13, 14, 29, 37]
Total de trocas: 10
```

🔹 PIOR CASO (já ordenado):

```
Estado inicial: [37, 29, 14, 13, 10, 1]
Troca 1: índice 0 <-> 1 -> [29, 37, 14, 13, 10, 1]
Troca 2: índice 1 <-> 2 -> [29, 14, 37, 13, 10, 1]
Troca 3: índice 2 <-> 3 -> [29, 14, 13, 37, 10, 1]
Troca 4: índice 3 <-> 4 -> [29, 14, 13, 10, 37, 1]
Troca 5: índice 4 <-> 5 -> [29, 14, 13, 10, 1, 37]
Troca 6: índice 0 <-> 1 -> [14, 29, 13, 10, 1, 37]
Troca 7: índice 1 <-> 2 -> [14, 13, 29, 10, 1, 37]
Troca 8: índice 2 <-> 3 -> [14, 13, 10, 29, 1, 37]
Troca 9: índice 3 <-> 4 -> [14, 13, 10, 1, 29, 37]
Troca 10: índice 0 <-> 1 -> [13, 14, 10, 1, 29, 37]
Troca 11: índice 1 <-> 2 -> [13, 10, 14, 1, 29, 37]
Troca 12: índice 2 <-> 3 -> [13, 10, 1, 14, 29, 37]
Troca 13: índice 0 <-> 1 -> [10, 13, 1, 14, 29, 37]
Troca 14: índice 1 <-> 2 -> [10, 1, 13, 14, 29, 37]
Troca 15: índice 0 <-> 1 -> [1, 10, 13, 14, 29, 37]
Vetor ordenado crescente: [1, 10, 13, 14, 29, 37]
Total de trocas: 15
```

Selection Sort - DECRESCENTE

```python
def selection_sort_decrescente(lista):
    n = len(lista)
    trocas = 0  # Contador de trocas

    print(f"Estado inicial: {lista}")

    for i in range(n):
        max_idx = i
        for j in range(i + 1, n):
            if lista[j] > lista[max_idx]:  # critério para decrescente
                max_idx = j
        if max_idx != i:
            lista[i], lista[max_idx] = lista[max_idx], lista[i]
            trocas += 1
            print(f"Troca {trocas}: índice {i} <-> {max_idx} -> {lista}")
```

```
🔷 MELHOR CASO (já decrescente):
Estado inicial: [37, 29, 14, 13, 10, 1]
Vetor ordenado (decrescente): [37, 29, 14, 13, 10, 1]
Total de trocas: 0
```

```
🔷 CASO MÉDIO (já decrescente):
Estado inicial: [29, 10, 14, 37, 13, 1]
Troca 1: índice 0 <-> 3 -> [37, 10, 14, 29, 13, 1]
Troca 2: índice 1 <-> 3 -> [37, 29, 14, 10, 13, 1]
Troca 3: índice 3 <-> 4 -> [37, 29, 14, 13, 10, 1]
Vetor ordenado (decrescente): [37, 29, 14, 13, 10, 1]
Total de trocas: 3
```


```
🔷 CASO MÉDIO (já decrescente):
Estado inicial: [29, 10, 14, 37, 13, 1]
Troca 1: índice 0 <-> 3 -> [37, 10, 14, 29, 13, 1]
Troca 2: índice 1 <-> 3 -> [37, 29, 14, 10, 13, 1]
Troca 3: índice 3 <-> 4 -> [37, 29, 14, 13, 10, 1]
Vetor ordenado (decrescente): [37, 29, 14, 13, 10, 1]
Total de trocas: 3
```

Selection Sort - CRESCENTE

```python
def selection_sort_crescente(lista):
    n = len(lista)
    trocas = 0  # Contador de trocas

    print(f"Estado inicial: {lista}")

    for i in range(n):
        min_idx = i
        for j in range(i + 1, n):
            if lista[j] < lista[min_idx]:  # critério para crescente
                min_idx = j
        if min_idx != i:
            lista[i], lista[min_idx] = lista[min_idx], lista[i]
            trocas += 1
            print(f"Troca {trocas}: índice {i} <-> {min_idx} -> {lista}")


    print(f"Vetor ordenado (crescente): {lista}")
    print(f"Total de trocas: {trocas}")
    print("-" * 50)
    return lista
```

```python
# Casos de teste
melhor_caso = [1, 10, 13, 14, 29, 37]  # já em ordem crescente (pior para decrescente)
caso_medio  = [29, 10, 14, 37, 13, 1]
pior_caso   = [37, 29, 14, 13, 10, 1]  # já em ordem decrescente (melhor para decrescente)

# Execução dos casos
print("🔷 MELHOR CASO (já crescente):")
selection_sort_crescente(melhor_caso[:])  # é o melhor caso para ordem decrescente

print("🔷 CASO MÉDIO (já crescente):")
selection_sort_crescente(caso_medio[:])

print("🔷 PIOR CASO (já crescente):")
selection_sort_crescente(pior_caso[:])  # é o pior caso para ordem decrescente
```

```
🔷 MELHOR CASO (já crescente):
Estado inicial: [1, 10, 13, 14, 29, 37]
Vetor ordenado (crescente): [1, 10, 13, 14, 29, 37]
Total de trocas: 0
--------------------------------------------------
🔷 CASO MÉDIO (já crescente):
Estado inicial: [29, 10, 14, 37, 13, 1]
Troca 1: índice 0 <-> 5 -> [1, 10, 14, 37, 13, 29]
Troca 2: índice 2 <-> 4 -> [1, 10, 13, 37, 14, 29]
Troca 3: índice 3 <-> 4 -> [1, 10, 13, 14, 37, 29]
Troca 4: índice 4 <-> 5 -> [1, 10, 13, 14, 29, 37]
Vetor ordenado (crescente): [1, 10, 13, 14, 29, 37]
Total de trocas: 4
--------------------------------------------------
🔷 PIOR CASO (já crescente):
Estado inicial: [37, 29, 14, 13, 10, 1]
Troca 1: índice 0 <-> 5 -> [1, 29, 14, 13, 10, 37]
Troca 2: índice 1 <-> 4 -> [1, 10, 14, 13, 29, 37]
Troca 3: índice 2 <-> 3 -> [1, 10, 13, 14, 29, 37]
Vetor ordenado (crescente): [1, 10, 13, 14, 29, 37]
Total de trocas: 3
```

Análise do grupo

Bubble Sort

Vantagens:

- **Clareza de implementação**: muito simples de entender e programar (troca repetitiva de elementos vizinhos fora de ordem).
- **Detecção de lista ordenada**: se implementado com um flag de troca, pode parar antes do fim no melhor caso (quando já está ordenado).
- **Didático**: bom para aprender lógica de ordenação.

Desvantagens:

- **Número de comparações**: sempre faz $n(n-1)/2$n(n-1)/2$n(n-1)/2$ no pior caso (independente de estar quase ordenado).
- **Número de trocas**: elevado, pois troca muitas vezes os mesmos elementos (ineficiente em prática).
- **Desempenho**: muito lento para listas grandes, sendo praticamente inviável em aplicações reais.

Selection Sort

Vantagens:

- **Número de trocas reduzido:** sempre realiza no máximo $n-1$ trocas, considerando que n não seja um número alto, independentemente da ordem inicial. Isso pode ser útil quando a operação de troca exige um grande custo de processamento.
- **Clareza de implementação:** também simples, escolhe sempre o menor elemento e coloca na posição correta.
- **Previsibilidade**: custo de comparações e trocas é fixo, sem depender da ordem inicial da lista.

Desvantagens:

- **Número de comparações:** elevado e fixo de n² comparações, mesmo em listas quase ordenadas.
- **Não aproveita listas parcialmente ordenadas:** ao contrário do Bubble Sort com flag ou insertion sort que se adapta.
- Um pouco menos intuitivo que Bubble Sort (embora ainda simples).

Resumo:

O Bubble Sort é interessante didaticamente e pode ser ligeiramente melhor em listas já ordenadas. O Selection Sort tem desempenho mais estável, realiza menos trocas e é melhor quando o custo da troca é alto.

Ambos são ineficientes para listas grandes, sendo usados mais em ensino do que em prática real.

Contribuição de cada integrante:

1) Abner e Nichollas foram os responsáveis pela execução do Bubble Sort;

2) Lucas e João foram os responsáveis pela execução Selection Sort ;

3) Dia 17 durante a aula e na presença de todos integrantes do grupo, os códigos foram executados e os resultados foram obtidos. Algumas inconsistências e dúvidas surgiram e foram corrigidas pelo grupo e algumas explicações em relação ao código foram esclarecidas com a professora Stefane.

4) Após observação dos códigos em execução o grupo se reuniu para análises e comparações dos resultados obtidos e elaboração do relatório;

5) Nichollas foi responsável por encaminhar o relatório via Github.