Relatório – Comparação de Algoritmos de Ordenação (Selection Sort vs Merge Sort)

Objetivo Geral

O objetivo principal do projeto é **comparar o desempenho de dois algoritmos de ordenação clássicos** — *Selection Sort* e *Merge Sort* — utilizando listas aleatórias de diferentes tamanhos. A comparação se baseia no **número de comparações realizadas** e no **tempo de execução** necessário para ordenar essas listas.

Estrutura do Projeto

O sistema é dividido em quatro arquivos Python, cada um com uma responsabilidade específica:

🔹 `main.py` **– Arquivo principal**

- Controla o fluxo da aplicação.
- Gera listas aleatórias com tamanhos crescentes (10, 100, 500, 1000).
- Executa os testes usando `Selection Sort` e `Merge Sort`.
- Mede o tempo de execução e o número de comparações de cada algoritmo.
- Exibe os resultados no terminal de forma clara.

🔹 `selection.py` **– Algoritmo Selection Sort**

- Implementa o algoritmo de ordenação por **seleção**.
- Percorre toda a lista e, para cada posição, encontra o menor elemento restante.
- Realiza a troca se necessário.
- Retorna a lista ordenada e o número total de comparações feitas.

Características:

- Algoritmo **simples** e **intuitivo**.
- Complexidade de tempo: **O(n²)**.

- Pouco eficiente para listas grandes.

🔹 `merge.py` – **Algoritmo Merge Sort**

- Implementa o algoritmo **Merge Sort**, que usa a abordagem de **divisão e conquista**.
- A lista é recursivamente dividida ao meio até que cada sublista tenha no máximo um elemento.
- Em seguida, as sublistas são combinadas (merge) de forma ordenada.
- Retorna a lista final ordenada e o número total de comparações.

Características:

- Algoritmo **eficiente**, estável e adequado para listas maiores.
- Complexidade de tempo: **O(n log n)**.

🔹 `utils.py` – **Funções auxiliares**

- `criar_lista_aleatoria(tamanho)`: Gera listas com valores inteiros aleatórios.
- `medir_tempo_execucao(funcao, lista)`: Mede o tempo e número de comparações de qualquer função de ordenação passada como parâmetro.

Metodologia dos Testes

O programa realiza testes com 4 tamanhos de lista:

- **10 elementos** (pequeno)
- **100 elementos** (médio)
- **500 elementos** (grande)
- **1000 elementos** (maior)

Para cada tamanho:

1. Uma lista aleatória é gerada.

2. A lista é ordenada primeiro com Selection Sort, depois com Merge Sort.
3. O número de comparações e o tempo de execução são medidos e exibidos.

Análise Esperada dos Resultados

Lista Tamanho	Selection Sort	Merge Sort
Pequena (10)	Razoável, porém lento	Muito rápido e eficiente
Média (100)	Lento, muitas comparações	Rápido, com desempenho estável
Grande (500+)	Muito ineficiente	Mantém boa performance

- **Selection Sort** sofre com o aumento do tamanho da lista, pois sempre compara todos os elementos restantes em cada passo.
- **Merge Sort** se mostra escalável e mais adequado para grandes volumes de dados.