Campus: IFMS-JD
Curso: Licenciatura em Computação
Disciplina: Programação com Estrutura de Dados
Docente: Profa. Raiane
Aluna: Talita Mendonça Marquês

1. **Bubble Sort (Ordenação por Bolha)**:
    - **Nome**: O nome "Bubble Sort" vem da ideia de como os elementos "borbulham" para suas posições corretas à medida que o algoritmo percorre a lista.
    - **Funcionamento**: O Bubble Sort compara pares adjacentes de elementos e troca suas posições se estiverem fora de ordem. Esse processo é repetido até que a lista esteja completamente ordenada.
    - **Desempenho**: O Bubble Sort é simples, mas ineficiente para listas grandes. Sua complexidade média é O(n^2), onde n é o número de elementos na lista.

2. **Insertion Sort (Ordenação por Inserção)**:
    - **Nome**: O nome "Insertion Sort" reflete como o algoritmo insere cada elemento na posição correta na lista ordenada.
    - **Funcionamento**: O Insertion Sort constrói a lista ordenada um elemento de cada vez, inserindo cada novo elemento na posição correta em relação aos elementos já ordenados.
    - **Desempenho**: O Insertion Sort também tem complexidade média O(n^2), mas é mais eficiente que o Bubble Sort em muitos casos.

3. **Selection Sort (Ordenação por Seleção)**:
    - **Nome**: O nome "Selection Sort" deriva do processo de selecionar o menor elemento e colocá-lo na posição correta.
    - **Funcionamento**: O Selection Sort encontra o menor elemento na lista e o coloca na primeira posição. Em seguida, encontra o próximo menor elemento e o coloca na segunda posição, e assim por diante.
    - **Desempenho**: O Selection Sort também tem complexidade média O(n^2), mas é menos eficiente que o Insertion Sort e o Bubble Sort.

4. **Merge Sort**:
    - **Nome**: O Merge Sort divide a lista em partes menores, ordena cada parte e, em seguida, mescla essas partes ordenadas para obter a lista final.
    - **Funcionamento**: Divide a lista em metades recursivamente, ordena cada metade e, em seguida, combina as metades ordenadas.
    - **Desempenho**: O Merge Sort tem complexidade média O(n log n) e é mais eficiente para listas grandes.

5. **Quick Sort**:

   - **Nome**: O Quick Sort é rápido porque faz particionamento rápido da lista.
   - **Funcionamento**: Divide a lista em torno de um "pivô", colocando elementos menores à esquerda e maiores à direita. Repete o processo recursivamente para as sub-listas.
   - **Desempenho**: O Quick Sort também tem complexidade média O(n log n) e é amplamente usado devido à sua eficiência.