Seminário: Algoritmos de Ordenação

1. Objetivo

Analisar e comparar a complexidade assintótica, a estratégia algorítmica e a aplicabilidade de métodos de ordenação eficientes, com foco em algoritmos sub-quadráticos e híbridos.

2. Formato

- **Estrutura:** Três (3) grupos de apresentação.
- **Tempo Alocado:** 45 minutos de apresentação e 15 min Q&A.

3. Tópicos de Apresentação

- **Grupo 1: Shell Sort**
    - **Objeto de Análise:** Método de ordenação por comparação baseado em incrementos (gaps) decrescentes.
    - **Pontos Críticos:**
        - Análise do impacto da sequência de gaps (ex: Knuth, Sedgewick) na performance.
        - Comparativo de complexidade e performance com o Insertion Sort base.
        - Justificativa da sua classe de complexidade, que varia entre $O(n \log^2 n)$ e O(n^(3/2)).
- **Grupo 2: Heap Sort**
    - **Objeto de Análise:** Método de ordenação por seleção baseado na estrutura de dados Max-Heap.
    - **Pontos Críticos:**
        - Implementação e análise de custo das operações heapify e sift-down.
        - Prova da sua natureza *in-place* (complexidade de espaço O(1)).
        - Análise da complexidade de tempo $\Theta(n \log n)$ garantida em todos os casos.

- **Grupo 3: TimSort**
    - **Objeto de Análise:** Algoritmo de ordenação híbrido, adaptativo e estável.
    - **Pontos Críticos:**
        - Arquitetura híbrida: sinergia entre Insertion Sort e Merge Sort.
        - Estratégia adaptativa: identificação e otimização de "runs" (sequências pré-ordenadas).
        - Análise de performance em cenários de dados reais e justificativa para sua adoção como padrão em linguagens como Python e Java.

4. Roteiro Técnico Obrigatório por Apresentação

1. **Fundamentação Teórica (5 min)**
    - Contextualização e motivação para o desenvolvimento do algoritmo.
    - Classe de complexidade e posicionamento teórico frente a outros algoritmos.
2. **Especificação Formal do Algoritmo (15 min)**
    - Descrição da lógica operacional e invariantes de laço.
    - Apresentação do pseudocódigo canônico.
    - Execução de um *trace* do algoritmo com um conjunto de dados definido, demonstrando o estado da estrutura de dados a cada passo crítico.
3. **Implementação e Análise Prática (10 min)**
    - Apresentação de uma implementação funcional em Python.
    - (Opcional) Breve análise de *profiling* ou benchmark comparando com outro método.
4. **Análise de Complexidade Assintótica (10 min)**
    - **Tempo:** Pior Caso (O), Melhor Caso (Ω), Caso Médio (Θ). A análise deve ser justificada.
    - **Espaço:** Complexidade de espaço auxiliar (O).
    - **Estabilidade:** Classificação como estável ou instável, com prova ou contraexemplo.

5. **Análise Comparativa e Conclusão (5 min)**
    - Análise de *trade-offs* (tempo vs. espaço, complexidade de implementação, etc.).
    - Discussão sobre cenários de aplicabilidade ideais.