TRABALHO AV2:

Faça uma comparação entre 4 métodos de ordenação estudados em aula com relação a estabilidade (preservar ordem lexicográfica), ordem de complexidade levando em consideração comparações e movimentações. Traga exemplos dos códigos em C e C++. Entrega deverá ser feita pelo link do seu github.

Aluno: Renan Aderne Freitas

GitHub: https://github.com/RenanAderneF

MÉTODOS DE ORDENAÇÃO:

**Bubble Sort:** Esse método compara pares de elementos adjacentes e os troca de lugar se estiverem na ordem errada. Esse processo se repete até mais nenhuma troca for necessária, ou seja, se todo os elementos estiverem ordenados.

**-Performance:** Considerado o algoritmo mais simples, porém mais ineficiente por ter que percorrer o vetor várias vezes dependendo do número de trocas necessárias, o bubble sort possui número de comparações de C(n) = O(n) para melhor caso e C(n) = O(n²) para pior caso. Possui um número muito grande de movimentação de elementos, não sendo recomendado para grandes conjuntos de dados.

Exemplo de bubble sort em C:

Tela de computador com fundo preto

Descrição gerada automaticamente

**Insertion Sort:** Esse método percorre um vetor de elementos da esquerda para a direita e à medida que avança vai ordenando os elementos à esquerda, dividindo os elementos em substruturas para ordenados e elementos para ainda ordenar.

**-Performance:** Seu número de movimentações é um dos menores entre os algoritmos de ordenação, porém ele possui um número de comparações fixo tanto para melhor quanto para pior caso, que seria C(n) = O(n²), além de não ser considerado um algoritmo estável, levando em conta sua implementação.

Exemplo de código em C++:

Texto

Descrição gerada automaticamente

Uma imagem contendo Forma

Descrição gerada automaticamente

**Heap Sort:** Esse método utiliza um vetor que simula uma árvore binária completa, com excessão do último nível. Todo elemento “pai” do vetor possui dois elementos gerados do mesmo. É verificado se um elemento filho é maior que seu pai e, caso ele seja, realiza-se a troca de posição entre os elementos.

**- Performance:**  Sua complexidade de operações é de O(n log n).

Código em C:

Texto

Descrição gerada automaticamente

(Continuação do código na página seguinte)

Texto

Descrição gerada automaticamente

**Quick Sort:** Método que segue a metodologia “dividir e conquistar”, dividindo sua lista de entrada em duas sublistas a partir de um elemento escolhido como para em seguida realizar o mesmo procedimento nas duas listas menores até uma única lista.

**- Performance:** Considerado o método de ordenação interna mais rápido existente. Possui complexidade de C(n) = O(n log n) para melhor caso e C(n) = O(n²) para pior caso.

Código em C:

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente