Reflexão (Documentação e Justificativa)

1. Justificativa Algorítmica

O algoritmo escolhido para esse projeto foi o Bubble Sort. Apesar de não ser o mais eficiente em termos de desempenho, ele é um dos mais simples de entender, o que o torna ideal para fins educativos.

Sua lógica é bem direta: ele percorre a lista diversas vezes, sempre comparando dois elementos vizinhos e trocando de posição quando estão fora de ordem. Essa simplicidade ajuda muito na hora de visualizar cada passo da ordenação. Por isso, o Bubble Sort é uma boa escolha quando o objetivo é mostrar o funcionamento do algoritmo de forma visual e didática.

2. Gerenciamento da Animação uso da função after

Para que a animação da ordenação aconteça de forma fluida, usamos a função after do Tkinter. Essa função permite programar a execução de uma determinada ação depois de um intervalo de tempo, sem travar a interface do programa.

Com isso, conseguimos fazer com que o algoritmo não execute tudo de uma vez só. Em vez disso, cada passo do processo (como uma comparação ou uma troca entre dois elementos) é mostrado com uma pequena pausa. Isso cria a sensação de movimento, como se o algoritmo estivesse "pensando" entre uma ação e outra.

A principal vantagem da função after é que ela permite manter a interface gráfica responsiva, mesmo durante a execução do algoritmo. Ou seja, a janela continua funcionando normalmente, sem travamentos, enquanto a ordenação acontece passo a passo.

3. Terminologia Técnica

Durante a implementação do projeto, utilizamos alguns conceitos técnicos importantes:

Canvas: É a área gráfica do Tkinter onde desenhamos os elementos visuais. No nosso caso, usamos o canvas para representar os valores da lista como barras verticais, que vão mudando de posição conforme a ordenação avança.

Lista: Utilizamos uma lista (ou array, no contexto da programação) para armazenar os números que serão ordenados. Cada elemento da lista representa uma barra na tela.

Indexação: Durante a ordenação, acessamos os elementos da lista por meio dos seus índices. Por exemplo, usamos dados[j] e dados[j+1] para comparar dois valores vizinhos e, se necessário, fazer a troca entre eles.

Fase 3 – Ação (Proposição de Solução)

O Bubble Sort funciona bem para poucas barras, mas fica lento com muitas, como 500. Para melhorar a performance, seria necessário trocar o algoritmo por um mais eficiente, como o Quick Sort ou o Merge Sort.

Como esses algoritmos são recursivos, a lógica de visualização precisaria ser adaptada. Em vez de executar diretamente cada passo, o algoritmo poderia registrar as comparações e trocas em uma lista de ações. Depois, essas ações seriam animadas uma por uma usando a função `after`, sem travar a interface.

Também seria possível otimizar o visual, por exemplo, afinando as barras ou agrupando algumas delas, para manter a fluidez da animação mesmo com muitos dados.