CESED – CENTRO DE ENSINO SUPERIOR E DESENVOLVIMENTO
CENTRO UNIVERSITÁRIO UNIFACISA

APLICAR ALGORITMOS DE APOIO À PESQUISA OPERACIONAL

PROF. JONHNANTHAN OLIVEIRA

ALGORITMO UTILIZADO PARA ORDENAÇÃO DA BIBLIOTECA BUBBLE SORT

ALUNOS DO GRUPO
GUSTAVO TOMIO MAGALHÃES KUBO
SÉRGIO MAGNO CASTOR PINHEIRO
THIAGO LIMEIRA DE ALENCAR

CAMPINA GRANDE – PB
2024.2

Bubble Sort: Algoritmo utilizado para ordenação

O Bubble Sort é um algoritmo de ordenação simples. Funciona comparando pares de elementos adjacentes e trocando-os se estiverem fora de ordem, fazendo com que o maior "suba" para o final da lista a cada rodada.

Passo a Passo do Funcionamento:

1. **Comparação de Pares**: O algoritmo começa comparando o primeiro com o segundo elemento. Se o primeiro é maior, trocam de lugar. Senão, ficam onde estão.

2. **Movimento Progressivo**: O algoritmo passa para o próximo par (segundo e terceiro, e assim por diante) até o fim da lista.

3. **Repetição**: A lista é percorrida várias vezes, e a cada passagem o maior número da parte desordenada vai para a posição correta no final.

4. **Parada do Algoritmo**: O algoritmo termina quando faz uma passagem completa sem nenhuma troca. Isso indica que todos os elementos estão ordenados.

Exemplo Rápido:

Dado a lista `[5, 3, 8, 4, 2]`, veja a primeira passagem:

- Comparação entre 5 e 3 → trocam → `[3, 5, 8, 4, 2]`
- Comparação entre 5 e 8 → sem troca → `[3, 5, 8, 4, 2]`
- Comparação entre 8 e 4 → trocam → `[3, 5, 4, 8, 2]`
- Comparação entre 8 e 2 → trocam → `[3, 5, 4, 2, 8]`

Aqui, o maior número (8) já está no fim da lista.

Porque utilizamos o do – while e não um for:

No do-while, a execução do bloco ocorre **pelo menos uma vez** antes de verificar a condição de repetição. Isso é importante para o Bubble Sort quando queremos garantir que a lista seja percorrida até que nenhuma troca seja necessária, sem nos preocuparmos com o estado inicial da lista. No do-while, o fluxo é o seguinte:

1. **Executa uma vez**: Realiza todas as comparações e trocas em uma passagem da lista.

2. **Verifica a condição**: Após cada passagem, verifica se houve trocas.
    1. Se houve troca, o do-while repete a passagem para garantir que a lista fique ordenada.
    2. Se não houve trocas, o loop termina, indicando que a lista já está ordenada.

Esse comportamento é conveniente para o Bubble Sort, pois precisamos sempre executar a primeira passagem para verificar se já está ordenado.

Um for também poderia fazer múltiplas passagens pela lista, mas ele não é ideal aqui porque a condição para encerrar o loop no Bubble Sort depende de uma **verificação após a execução** — ou seja, precisamos saber se houve trocas **depois de cada passagem completa**. Para conseguir isso com um for, teríamos que adicionar uma lógica extra para controlar manualmente o número de passagens e a verificação de trocas, o que deixaria o código menos intuitivo.

No caso do Bubble Sort:

- Usar um for exigiria que nós inicializássemos e verificássemos uma variável de controle (troca) fora e dentro do loop para rastrear se o loop deve continuar, o que torna o código mais complexo.

Em resumo, o do-while é mais apropriado aqui porque:

1. Ele permite que o código execute pelo menos uma vez, independentemente do estado da lista.

2. A condição é verificada após a execução da passagem, simplificando a lógica de controle de trocas e repetições.

Esses pontos tornam o do-while uma escolha mais intuitiva e direta para o Bubble Sort, pois ele garante que a lista seja processada de maneira contínua até que não haja mais trocas a serem feitas.

Análise de Eficiência:

O Bubble Sort é útil para listas pequenas ou quase ordenadas, mas lento para listas grandes devido às muitas comparações e trocas. A complexidade de tempo é:

- **Melhor Caso**: $O(n)$, quando já está ordenado e não há trocas.
- **Pior Caso e Médio Caso**: $O(n^2)$, para listas totalmente desordenadas.

Vantagens e Desvantagens:

Vantagens

Simplicidade: fácil de entender e implementar.

Utilidade para listas pequenas ou quase ordenadas.

Desvantagens

Ineficiente para listas grandes, especialmente em comparação com algoritmos mais avançados, como o Merge Sort ou o Quick Sort.

Maior número de comparações e trocas.

Conclusão:

O Bubble Sort foi escolhido pelo projeto ser pequeno e por ser um algoritmo fácil de entender, adequado para a lista de livros do sistema.