INTRODUÇÃO À PROGRAMAÇÃO ESTRUTURADA EM LINGUAGEM C

Conteúdo desta aula:

- Revisão sobre tópicos da aula anterior e solução de dúvidas

- Ordenação de vetores
    - Quick Sort
    - Buble Sort
    - Selection Sort
    - Insertion Sort

O que é a ordenação de vetores?

A ordenação é um procedimento ou uma sequência de passos finita, projetada para **organizar os elementos** de uma lista ou array **em** uma determinada **ordem**, geralmente **crescente ou decrescente**.

Esse tipo de algoritmo é **fundamental** para melhorar a eficiência de outros algoritmos que requerem **dados organizados**, como buscas e mesclagens de dados.

Dentre diversos tipos, aqui veremos alguns dos mais conhecidos:
- Quick Sort
- Buble Sort
- Selection Sort
- Insertion Sort

Ordenação de vetores: Quick Sort

O algoritmo de ordenação Quick Sort é um dos métodos mais eficientes para ordenar uma lista de elementos. Ele utiliza a técnica de "dividir para conquistar" (divide and conquer), que envolve dividir o problema em subproblemas menores e resolver cada um de forma recursiva. Estes são os passos utilizados nesta ordenação:

1. Escolha de um pivô: Um elemento da lista é escolhido como pivô. Existem várias estratégias para escolher o pivô, como selecionar o primeiro elemento, o último, o elemento do meio ou um elemento aleatório.

2. Particionamento: A lista é reorganizada de forma que todos os elementos menores que o pivô fiquem à esquerda e todos os elementos maiores que o pivô fiquem à direita. O pivô está agora na posição correta em relação à ordenação final.

3. Recursão: A mesma operação é aplicada recursivamente às sublistas à esquerda e à direita do pivô até que a base da recursão seja atingida (listas de tamanho zero ou um, que já estão ordenadas).

Ordenação de vetores: Quick Sort

Complexidade

Melhor Caso: O Quick Sort tem uma complexidade de tempo de O(n log n) quando o pivô divide a lista em duas partes aproximadamente iguais.

Pior Caso: A complexidade piora para O(n^2) quando o pivô escolhido é o menor ou o maior elemento repetidamente, resultando em partições desequilibradas.

Complexidade Média: Apesar do pior caso, o Quick Sort tem uma complexidade média de O(n log n), tornando-o eficiente na prática para muitas aplicações.

Ordenação de vetores: Quick Sort

Vantagens:

- Geralmente rápido na prática, com bom desempenho em uma variedade de dados.

- Utiliza memória de forma eficiente (in-place, a ordenação é feita diretamente nos dados alocados).

Desvantagens:

- Pode ter um desempenho ruim em listas quase ordenadas ou com muitos elementos repetidos.

- Sensível à escolha do pivô, que pode afetar o desempenho.

Ordenação de vetores: Buble Sort

O algoritmo de ordenação Bubble Sort é um dos métodos mais simples e básicos para ordenar uma lista de elementos. Ele funciona comparando repetidamente pares adjacentes de elementos e trocando-os se estiverem na ordem errada. Esse processo é repetido até que a lista esteja ordenada. Estes são os passos utilizados nesta ordenação:

1. Comparação e Troca:
1.1. Inicie pelo primeiro elemento da lista.
1.2. Compare o primeiro elemento com o segundo. Se o primeiro é maior que o segundo, troque-os.
1.3. Mova para o próximo par de elementos e repita a comparação e troca, se necessário.
1.4. Continue esse processo até o final da lista.

2. Repetição do Processo:
2.1. Após a primeira passagem, o maior elemento está no final da lista.
2.2. Repita o processo para os elementos restantes, excluindo o último, que já está em sua posição correta.

3. Verificação de Ordenação:
3.1. Repita os itens 1 e 2 até que nenhuma troca seja necessária, indicando que a lista está ordenada.

Ordenação de vetores: Buble Sort

Complexidade

Pior Caso e Caso Médio: O Bubble Sort tem uma complexidade de tempo de O(n^2), onde n é o número de elementos na lista. Isso ocorre porque, em cada passagem, todos os pares adjacentes são comparados e possivelmente trocados.

Melhor Caso: O melhor caso ocorre quando a lista já está ordenada, com complexidade de tempo O(n). Uma otimização pode ser feita para parar o algoritmo se durante uma passagem completa não houver trocas, indicando que a lista está ordenada.

Ordenação de vetores: Buble Sort

Vantagens:

- Simplicidade: Fácil de entender e implementar.

- Pouca memória adicional: É um algoritmo in-place, ou seja, não requer memória extra significativa.

Desvantagens:

- Ineficiência: Muito lento para listas grandes devido à sua complexidade de tempo O(n^2).

- Ineficaz para grandes conjuntos de dados em comparação com algoritmos mais avançados, como Quick Sort ou Merge Sort.

Ordenação de vetores: Selection Sort

O Selection Sort é um método simples de ordenação que trabalha dividindo a lista em duas partes: a parte ordenada à esquerda e a parte não ordenada à direita. Ele seleciona repetidamente o menor (ou maior, dependendo da ordem desejada) elemento da parte não ordenada e o move para o final da parte ordenada. Este processo é repetido até que toda a lista esteja ordenada.

1. Encontrar o Menor Elemento:
1.1. Comece no início da lista.
1.2. Encontre o menor elemento na lista não ordenada.
1.3. Troque este menor elemento com o primeiro elemento da lista não ordenada.

2. Repetir para o Restante da Lista:
2.1. Movimente o limite entre a parte ordenada e a não ordenada um elemento para a direita.
2.2. Repita o processo de encontrar o menor elemento na lista não ordenada e trocá-lo com o primeiro elemento da parte não ordenada.
2.3. Continue até que toda a lista esteja ordenada.

Ordenação de vetores: Selection Sort

Complexidade

Pior Caso, Melhor Caso e Caso Médio: A complexidade do Selection Sort é O(n^2), onde n é o número de elementos na lista. Isso ocorre porque existem dois loops aninhados: um para iterar sobre os elementos e outro para encontrar o menor elemento na parte não ordenada.

Memória: O Selection Sort é um algoritmo in-place, ou seja, requer memória adicional constante O(1) além da lista original.

Ordenação de vetores: Selection Sort

Vantagens:

- Simplicidade: Fácil de entender e implementar.

- Estável em termos de número de trocas: Realiza no máximo n-1 trocas, o que é mínimo entre os algoritmos de ordenação O(n^2).

Desvantagens:

- Ineficiência: Muito lento para listas grandes devido à sua complexidade quadrática.

- Não é estável: A estabilidade (preservação da ordem relativa dos elementos iguais) pode ser um problema a menos que modificações adicionais sejam feitas.

Ordenação de vetores: Insertion Sort

O Insertion Sort é um método simples e intuitivo de ordenação que funciona da mesma forma que se ordenaria cartas na mão em um jogo de cartas. Ele constrói a lista ordenada elemento por elemento, colocando cada novo elemento na posição correta em relação aos elementos já ordenados.

1. Divisão da Lista:
1.1. A lista é dividida em duas partes: a parte ordenada à esquerda e a parte não ordenada à direita.
1.2. Inicialmente, a parte ordenada contém apenas o primeiro elemento da lista e a parte não ordenada contém o resto dos elementos.

2. Inserção de Elementos:
2.1. Iterativamente, cada elemento da parte não ordenada é retirado e inserido na posição correta da parte ordenada.
2.2. Para encontrar a posição correta, o algoritmo percorre a parte ordenada de trás para frente até encontrar o ponto onde o elemento deve ser inserido.

Ordenação de vetores: Insertion Sort

Complexidade

Melhor Caso: O(n), ocorre quando a lista já está ordenada. O algoritmo apenas percorre a lista uma vez sem fazer trocas.

Pior Caso e Caso Médio: O(n^2), ocorre quando a lista está ordenada em ordem inversa. Cada inserção percorre toda a parte ordenada.

Memória: O Insertion Sort é um algoritmo in-place, ou seja, requer memória adicional constante O(1) além da lista original.

Ordenação de vetores: Insertion Sort

Vantagens:

- Simplicidade: Fácil de entender e implementar.
- Eficiência em listas pequenas ou quase ordenadas.
- Estável: Preserva a ordem relativa dos elementos iguais.
- In-place: Não requer espaço adicional significativo.

Desvantagens:

- Ineficiente em listas grandes devido à complexidade quadrática no pior caso.
- Não é adequado para grandes conjuntos de dados em comparação com algoritmos mais avançados, como Quick Sort ou Merge Sort.