Estrutura de Dados - Aula 1

Conceitos Básicos: Tipos Abstratos de Dados, Estruturas, Alocação e Complexidade

Introdução à Disciplina

- A disciplina de Estrutura de Dados é fundamental na formação em Computação.
- Vamos estudar como organizar, armazenar e acessar dados de forma eficiente.

O que são Estruturas de Dados?

- São formas organizadas de armazenar e manipular dados na memória de um computador.
- Permitem acesso eficiente aos dados para diferentes operações.

Tipos Abstratos de Dados (TAD)

- Um TAD define um conjunto de operações válidas sobre um tipo de dado, independentemente da implementação.
- Exemplos: Lista, Pilha, Fila, Dicionário.

Exemplo de TAD: Pilha

- Operações típicas: push (inserir), pop (remover), top (consultar topo), isEmpty (verificar se vazia).

TAD vs Estrutura Física

- TAD define o 'o que'. A estrutura de dados define o 'como'.
- Exemplo: uma lista pode ser implementada com arrays ou listas encadeadas.

Representação Física da Estrutura

- A estrutura física diz respeito à forma como os dados são efetivamente armazenados em memória.

Tipos de Estruturas de Dados

- • Lineares: arrays, listas, pilhas, filas
- • Hierárquicas: árvores
- • Não lineares: grafos

Importância das Estruturas

- Escolher a estrutura correta impacta diretamente na eficiência do algoritmo.

Forma de Alocação de Memória

- • Alocação Estática: feita em tempo de compilação
- • Alocação Dinâmica: feita em tempo de execução

Alocação Estática

- Memória é reservada antes da execução do programa.
- Exemplo: vetores com tamanho fixo.

Alocação Dinâmica

- Memória é alocada conforme a necessidade durante a execução.
- Exemplo: listas encadeadas.

Ponteiros e Referências

- Usados para manipular dados alocados dinamicamente.
- Apontam para o endereço de memória de um dado.

Noções de Complexidade

- Complexidade mede o consumo de tempo e espaço de um algoritmo.

Notação Big-O

- Big-O descreve o pior caso de execução.
- Exemplos: O(1), O(n), O(n^2), O(log n), O(n log n)

Exemplo de Complexidade

- Busca em vetor:
- • Linear: O(n)
- • Binária (em vetor ordenado): O(log n)

Comparação de Algoritmos

- Dois algoritmos podem resolver o mesmo problema, mas com desempenho muito diferente.

Eficiência x Facilidade

- Algoritmos mais eficientes podem ser mais complexos de implementar.
- Equilíbrio é importante.

Importância da Análise

- Evita soluções ineficientes que desperdiçam recursos computacionais.

Aplicações Reais

- • Banco de dados
- • Sistemas operacionais
- • Jogos e simulações
- • Redes e telecomunicações

Estudo de Caso

- Aplicando conceitos em um exemplo de fila de impressão de documentos.

Leitura Recomendada

- • Cormen et al. - Algoritmos
- • Tenenbaum - Estruturas de Dados
- • Ziviani - Projeto de Algoritmos