Texto Complementar

Filas com Listas Encadeadas

Introdução

Neste material, focaremos nas operações de filas utilizando listas encadeadas, um conceito fundamental no universo das estruturas de dados em computação. As filas organizam os elementos seguindo o princípio FIFO (First In, First Out), no qual o primeiro elemento adicionado é o primeiro a ser removido. Este princípio é essencial para compreender como as operações de inserção (*enqueue*) e remoção (*dequeue*) são realizadas em uma fila e como essas operações podem ser implementadas de maneira eficiente utilizando listas encadeadas.

Características Importantes

Filas são amplamente aplicadas em situações que requerem tratamento de dados em ordem específica, como gerenciamento de tarefas em sistemas operacionais, simulação de filas de espera em ambientes reais e controle de fluxo de dados em aplicações de redes. Uma propriedade fundamental das filas é sua capacidade de garantir que todos os elementos sejam processados de maneira justa e sequencial.

Definição dos Componentes da Fila

Uma fila implementada com listas encadeadas é composta por **nós** que contêm os dados armazenados e uma referência ao próximo **nó** na sequência. Existem dois pontos críticos em uma fila: a "cabeça" da fila, que aponta para o primeiro elemento inserido (ainda não removido), e a "cauda" da fila, que aponta para o último elemento inserido, permitindo a adição de novos elementos de forma eficiente.

Operações de Inserção e Remoção

- **Enqueue (Inserir na Fila):** A adição de um novo elemento no final da fila requer a criação de um novo nó e o ajuste da referência do último nó atual para apontar para este novo. Esse processo mantém a ordem de chegada dos elementos, sendo eficiente, pois não necessita percorrer toda a lista.

```
def enqueue(self, valor):
    novo_no = No(valor)
    if self.cauda is not None:
        self.cauda.proximo = novo_no
    self.cauda = novo_no
    if self.cabeca is None:
        self.cabeca = novo_no
```

- **Dequeue (Remover da Fila):** A remoção do elemento na frente da fila implica em alterar a referência da cabeça da fila para o segundo nó, efetivamente removendo o primeiro elemento da fila. É necessário verificar se a fila não está vazia antes de realizar a remoção.

```
def dequeue(self):
    if self.cabeca is not None:
        removido = self.cabeca
        self.cabeca = self.cabeca.proximo
        if self.cabeca is None:
            self.cauda = None
        return removido.valor
    raise Exception("Fila vazia")
```

- **Peek (Visualizar o Primeiro Elemento):** Para acessar o elemento na frente da fila sem removê-lo, simplesmente retornamos o valor do nó apontado pela cabeça da fila, desde que a fila não esteja vazia.

```
def peek(self):
    if self.cabeca is not None:
        return self.cabeca.valor
    raise Exception("Fila vazia")
```

Encerramento

Com este material, espera-se que você tenha adquirido um entendimento claro de como as filas são implementadas usando listas encadeadas e como executar as operações fundamentais de inserção, remoção e acesso. Essas estruturas e operações são a espinha dorsal de muitas aplicações críticas em ciência da computação, sublinhando a importância de compreender suas funcionalidades e eficiências.

Bons estudos!

Equipe da Disciplina de Estrutura de Dados