**ADS Módulo II**

**Estrutura de Dados I**

**Professor: José Olímpio**

**Aluno: Bruno Camargo Manso**

# Resumo sobre Listas Ordenadas

# **Listas Lineares**

As Listas Lineares são listas mais simples de se manipular. Agrupa informações referentes a um conjunto de elementos que se relacionam entre si. As operações mais frequentes são as operações de busca, de inclusão e remoção de um determinado item. Outras operações podem ser incluídas, por exemplo a alteração de um elemento, a combinação de duas ou mais listas em uma única, a ordenação dos nós segundo determinado campo a determinação do primeiro e do último nó da lista, a determinação das cardinalidades etc.

Se as inserções e remoções são permitidas apenas na extremidade da lista são chamadas então de deque (double ended queue). Se as remoções são efetivadas apenas em um extremo a lista é chamada de pilha, sendo que nas filas as inserções são realizadas em um extremo e as retiradas no outro.

# **Filas**

Filas podem ser organizadas de diversas formas, a alocação pode ser do tipo sequencial e do tipo encadeada ou dinâmica, as últimas funcionam de forma que as posições de memória forem são alocadas ou desalocadas conforme são necessárias ou dispensáveis. As alocações são definidas pelos ponteiros, indicando a posição do elemento dentro da Fila. No entanto, as alocações encadeadas apesar do gasto de memória ser maior devido a realocação de ponteiros é mais conveniente quando se trata de mais de uma lista. Por outro lado em relação às listas do tipo sequencial, o acesso ao último elemento da lista é imediato enquanto na alocação encadeada é necessário percorrer toda a lista até o elemento necessário.

# **Alocação Sequencial**

Na Alocação Sequencial maneira mais simples de se manter uma lista linear na memória é colocando nós em posições contíguas. Nas linguagens de alto nível ocorre a reserva prévia de memória para cada estrutura utilizada, assim a inserção e remoção dos nós não ocorrem de fato mas utiliza-se uma espécie de simulação para essas operações. O armazenamento sequencial é desejado porque as operações básicas podem ser implementadas de forma bastante eficiente.

# **Listas Lineares em Alocação Sequencial**

Em uma lista linear cada nó é formado por campos que armazenam características diferentes dos elementos da lista. Cada nó da lista possui um identificador chamado ‘chave’. Supõe-se então que todas as chaves sejam diferentes. Os campos podem estar ordenados ou não, dependendo dos valores de suas chaves. Por esse motivos existem listas ordenadas e não ordenadas. Um exemplo de eficiência de um algoritmo é a busca binária, a busca então percorre toda a lista mas abandonando partes do catálogo que pelo tipo de ordem com certeza a busca não haverá sucesso nessas partes. O primeiro nó é buscado na metade da lista, a outra metade, por com certeza não conter o elemento buscado, será abandonada. Assim faz-se uma recursão dessa função no sentido de ir cortando as metades selecionadas até esgotar a tabela.

# **Diferenças básicas entre Fila e Pilha**

Qual é a diferença entre uma Lista e uma Pilha? A diferença está nas operações destas duas estruturas de dados. As operações de uma Pilha são mais restritas do que as de uma Lista. Por exemplo, você pode adicionar ou remover um elemento em qualquer posição de uma Lista mas em uma Pilha você só pode adicionar ou remover do topo.

Então, uma Lista é uma estrutura mais poderosa e mais genérica do que uma Pilha. A Pilha possui apenas um subconjunto de operações da Lista. Então o interessante é que para implementar uma Pilha podemos usar uma Lista.

As Pilhas necessitam de apenas um ponteiro no topo, as Filas precisam de 2 ponteiros indicando o início e o fim da Lista (Estruturas de Dados e Seus Algoritmos - Jayme L. Szwarcfiter e Lilian Markenzon), para a adição de um ponteiro move-se o ponteiro de retirada consequentemente move-se também o ponteiro de Início de Fila. A medida que os ponteiros são incrementados na memória disponível, a Fila se move. Tal situação dá a falsa impressão de que a memória, mais cedo ou mais tarde pode se esgotar. Para resolver isso deve-se considerar que os elementos estivessem em círculo.

# **Lista de Espaço Disponível**

Existem algoritmos que facilitam o funcionamento de uma fila gerenciando a memória de forma mais eficaz, como por exemplo o algoritmo de Lista de Espaço disponível, que segundo Szwarcfiter e Markenzon (1994), que é usada para gerenciar tais posições, contém posições de memórias ainda não utilizadas ou dispensadas após sua utilização. Com esse artifício, pode se fazer um dimensionamento da memória usando um único vetor que simulará a memória total consumida, permite o controle total das posições ocupadas e livres por parte dos usuários.

Após o encadeamento dos campos dos ponteiros na lista de espaços disponíveis um ponteiro vago é inicializado no endereço do primeiro nó da lista encadeada, o ponteiro do último nó recebe um valor que indica o fim dessa lista. Assim em operações de inserção ou remoção, manipula-se tanto a lista original, quanto a lista auxiliar de espaços disponíveis. Quando existe a necessidade de um novo nó para uma execução de inserção, busca-se na lista auxiliar, devolve-se então, à lista de espaços disponíveis, um nó dispensado pelo algoritmo de inserção.

# **Listas Simples Encadeadas**

As Listas Simples Encadeadas também precisam da posição do ponteiro do primeiro nó, o percurso da lista então é feito a partir desse ponteiro. Segue-se então consecutivamente pelos endereços existentes do campo que indica o próximo nó, da mesma forma que na alocação sequencial se acrescentava uma unidade ao índice do percurso. Porém podem existir problemas em relação a esse tipo de lista: considerando que a existência de um ponteiro apontando o primeiro nó que obrigam os algoritmos de inserção e remoção apresentarem testes especiais para o caso de inserção e última remoção. A solução estaria em uma simples variação na estrutura de armazenamento: o uso do nó-cabeça que nunca será removido e que sempre representará o nó de início da lista indicado pelo ponteiro. Tal nó não não tem informações sobre a tabela propriamente dita, muitas vezes podem conter informações sobre o algoritmo implementado.

# **Listas Circulares Encadeadas**

As Listas Circulares Encadeadas tem a característica de obrigar o último nó da lista indicar o nó cabeça. Desta maneira o teste de fim de lista nunca é satisfeito, assim a preocupação passa a ser um critério de parada possa ser incorporado ao teste de busca. Para solucionar isso basta colocar a chave procurada no nó cabeça de forma que uma resposta positiva é sempre encontrada.

# **Listas Duplamente Encadeadas**

Listas Duplamente Encadeadas os campos de ponteiros são nomeados ant (anterior) e post (posterior) que aponta o nó seguinte. Listas não circulares e listas sem nó-cabeça podem também ser duplamente encadeadas. No algoritmo de busca, a função retorna indicando o nó procurado ou, se não for encontrado, o nó que seria seu consecutivo.

# **Orientação Topológica**

A Ordenação Topológica é usada quando o problema abordado remete uma organização parcial dos componentes. A ordem topológica trata de imergir a ordem parcial em uma ordem linear, ou seja, rearrumar os componentes em uma sequência. Assim considerando que um objeto não é precedido por nenhum outro na ordem parcial, ele então é um componente de saída, na ordem final. Quando ele é removido do seu conjunto, seus componentes não removidos vão obedecer uma ordem parcial até que conjunto se reordene por completo.

# **Alocação de Espaço de Tamanho Variável**

O modelo de Alocação de Espaço de Tamanho Variável utiliza-se de um vetor onde cada elemento é associado a uma unidade de memória denominada ‘palavra’. Um conjunto de palavras são chamados de ‘blocos’. O endereço do bloco é o índice da primeira palavra na lista. Quando um programa tem a finalidade de criar blocos e quando tais blocos, que contêm listas, quando extintos deverá ser liberado da memória para possível uso com outra finalidade. O modelo de Alocação de Espaços de tamanho variável devem apresentar blocos reservados e disponíveis um programa. Tal configuração faz com que haja variação devido às diferentes solicitações de reserva e liberação de blocos. Existe uma grande variação do tamanho de blocos dependendo das características individuais das estruturas utilizadas.