Trabalho Prático: Lista Linear Sequencial
Nome:** Luis Felipe de Andrade Marques
Matrícula: 557238
Curso: Ciência da Computação - Estrutura de Dados

O Trabalho Dirigido 2 é sobre criar uma Lista Linear Sequencial de números inteiros. Nele, você vai trabalhar com funções como inserir, apagar e acessar elementos da lista.

Principais Dificuldades

Alocação de memória para criação da lista

A princípio, uma das dificuldades foi a implementação da função `criarLista`, mais precisamente no momento de alocar memória para a lista e para os dados. Busquei ajuda nos slides disponibilizados pelo professor Rafael e outros professores da mesma matéria.

Inserir valor em posição específica e mover elementos:

Outra dificuldade foi na função de inserir um valor em uma posição específica e mover os demais valores para o lado. Definitivamente, foi a parte que mais demorei a fazer, porém, com a ajuda dos slides disponibilizados por outros professores na internet, foi possível solucionar.

As demais funções não apresentaram grandes dificuldades. Utilizei as funções que já havia desenvolvido como base, facilitando meu trabalho.

Notação Big-O:
criarLista:
Alocar memória demanda um tempo constante, ou seja, O(1);
Como também foi alocada memória para os dados, então O(1);
Atribuição de valor para variável tem um tempo de execução de O(1);
Portanto, a complexidade de tempo da função criarLista é de O(1), pois todas as operações são realizadas em tempo constante.

tamanho
Uma operação de leitura direta, portanto a notação será O(1);

inserirFim:
O if verifica se a lista está cheia. A comparação demanda um tempo constante, O(1);
A atribuição de valor ao array tem um tempo constante, O(1)
A incrementação no tamanho da lista também contém tempo constante, O(1)
A complexidade de tempo da função será O(1);

inserirPosicao:
O if verifica se a lista está cheia. A comparação demanda um tempo constante, O(1);
O if que verifica se a posição dada pelo usuário é válida demanda tempo constante O(1);
A complexidade de tempo do deslocamento de elementos a partir da posição desejada depende de onde será a inserção, no pior caso O(n)
Inserir o valor na posição desejada demanda O(1);
A incrementação no tamanho da lista também contém tempo constante, O(1);
A complexidade de tempo no pior caso será O(n), e no melhor caso O(1);

removerPosicao:
O if que verifica se a lista está vazia demanda tempo constante O(1);
O if que verifica se a posição é válida também demanda tempo constante O(1);
O deslocamento de elementos após a remoção terá um tempo definido a partir da posição que a remoção foi feita, tendo como pior caso O(n);
O decremento demanda tempo O(1);
A complexidade de tempo da função será, no pior caso, O(n) e no melhor caso O(1);

removerValor
O if que verifica se a lista está vazia demanda tempo constante O(1);
O loop que percorre todos os elementos para encontrar o valor, tem tempo de O(n);
A verificação se o item foi encontrado demanda tempo constante O(1);
O deslocamento de elementos após a remoção terá um tempo definido a partir da posição que a remoção foi feita, tendo como pior caso O(n);

O decremento demanda tempo O(1);
A complexidade de tempo da função será, no pior caso, O(n) e no melhor caso O(1);

obterPosicao:
O if de verificação se a posição é válida demanda O(1);
O acesso ao elemento na posição demanda O(1);
A impressão do valor na posição tem tempo de O(1);
O retorno do valor tem tempo de O(1);
A complexidade de tempo da função obterPosicao é O(1);

obterValor:
O if de verificação se a lista está vazia demanda tempo de O(1);
O loop percorre todos os elementos para encontrar o valor tem tempo de, no pior caso, O(n);
O if de verificação do resultado tem tempo constante O(1);
A impressão e retorno do valor tem tempo de O(1);
Mediante isso, a complexidade de tempo da função, no seu pior caso, é O(n);

exibir:
O if que verifica se a lista está vazia demanda tempo O(1);
O loop de exibição de valores da lista tem tempo de O(n);
O if para quebra de linha após 5 elementos tem tempo constante O(1);
A impressão final tem tempo de O(1);
Portanto, a função terá complexidade O(n) no seu pior caso;