Trabalho Pr tico: Lista Encadeada Simples
Nome: Luis Felipe de Andrade Marques
Matr cula: 557238
Curso: Ci ncia da Computa o - Estrutura de Dados

O Trabalho Dirigido 3 sobre criar uma Lista Encadeada Simples de n meros inteiros.
Nele, voc vai trabalhar com fun es como inserir, apagar e acessar elementos da lista.

Principais Dificuldades
Umas das dificuldades foi inserir elemento no final ou no meio da lista. A dificuldade se deu em percorrer todos os elementos da forma certa para fazer a insercao.
Outra dificuldade foi na funcao main, as funcoes de inserir elemento e remover nao estavam funcionando adequadamente, com um pouco de analise indentifiquei o problema e resolvi!
Nas outras funcoes nao obtive nenhum dificuldade, usei o codigo dos outros trabalhos como base para o desenvolvimento do atual.

Nota o Big-O:
criarLista:
Alocar mem ria demanda um tempo constante, ou seja, O(1);
Atribui o de valor para vari vel tem um tempo de execu o de O(1);
Portanto, a complexidade de tempo da fun o criarLista de O(1), pois todas as opera es
s o realizadas em tempo constante.

insereFinal:
 Aloca o de mem ria para o novo n ocorre em tempo constante.O(1)
Atribuicoes de valor pro no ocorre em tempo constante. O(1)
A verifica o da condi o uma opera o de tempo constante.O(1)
Percorrer a lista ate o final. No pior caso, percorre todos os n s da lista para encontrar o ltimo. O(n)
Contudo, a complexidade de tempo da funcao sera O(n)

inserePosicao:
Aloca o de mem ria para o novo n ocorre em tempo constante.O(1)
O if que verifica se a posi o dada pelo usu rio v lida demanda tempo constante O(1);
Inserir o valor na posi o desejada demanda O(1);
Percorrer a lista ate a posicao anterior demanda p-1, equivalende a p. O(p)
A complexidade de tempo no pior caso ser O(p), e no melhor caso O(1);

insere:
Aloca o de mem ria para o novo n ocorre em tempo constante.O(1)
Atribuicao de valores demanda O(1).
Complexidade de tempo da funcao. O(1)

removePosicao:
O if que verifica se a lista est vazia demanda tempo constante O(1);
O if que verifica se a posi o v lida tamb m demanda tempo constante O(1);
Percorrer a lista ate a posicao anterior demenda p-1 no pior caso. O(p)
A complexidade de tempo da fun o ser , no pior caso, O(p) e no melhor caso O(1);

removerValor
O if que verifica se a lista est vazia demanda tempo constante O(1);
O loop que percorre todos os elementos para encontrar o valor, tem tempo de O(n);
A verifica o se o item foi encontrado demanda tempo constante O(1);
A complexidade de tempo da fun o ser , no pior caso, O(n) e no melhor caso O(1);

obterPosicao:

O if que verifica se a lista est vazia demanda tempo constante O(1);
O if que verifica se a posi o v lida tamb m demanda tempo constante O(1)
Percorrer a lista ate a posicao demenda nno pior caso. O(n)
A complexidade de tempo da fun o obterPosicao O(n);

procurar:
O if de verifica o se a lista est 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 verifica o do resultado tem tempo constante O(1);
A impress o e retorno do valor tem tempo de O(1);
Mediante isso, a complexidade de tempo da fun o, no seu pior caso, O(n);

exibir:
O if que verifica se a lista est vazia demanda tempo O(1);
O loop de exibi o de valores da lista tem tempo de O(n);
A impress o final tem tempo de O(1);
Portanto, a fun o ter complexidade O(n) no seu pior caso;

tamanho;
O if que verifica se a lista est vazia demanda tempo O(1);
O loop que percorre a lista incremenando o contador de valores da lista tem tempo de O(n);
O retorno final tem tempo de O(1);
Portanto, a fun o ter complexidade O(n) no seu pior caso;