UNIVERSIDADE FEDERAL DE MATO GROSSO
Campus Universitário do Araguaia - CUA
Instituto de Ciências Exatas e da Terra - ICET
Curso de Bacharelado em Ciência da Computação

Relatório de análise das listas estática e dinâmicas da disciplina de Estrutura de Dados 1

Discentes: Fernanda Lima de Souza, Guilherme da Silva Ferraz

Docente: Dr. Ivairton Monteiro Santos

Barra do Garças – MT

2024

Neste trabalho realiza-se operações de inserção em três tipos diferentes de listas: estática, encadeada e duplamente encadeada. O usuário pode escolher o tipo de lista e o programa insere elementos de forma aleatória (no início, no final ou em uma posição aleatória) na lista escolhida. Além disso, o tempo de execução de cada operação é medido.

No arquivo MAIN.C, além de inicializar as listas, também é inicializada a "semente" para gerar os valores pseudoaleatórios - ***srand(time(NULL))*** da biblioteca ***"time.h"***. Em seguida temos um Switch Case, no qual o usuário poderá escolher em qual das listas serão inseridos os valores aleatórios, sendo as listas: **estática, encadeada ou duplamente encadeada.** Caso a escolha seja a lista estática, foi feito um laço for para que seja inserido o total de valores pedido pelo trabalho e para que ocorra de maneira correta a inserção por início, fim e para a inserção por posição foi criado um *if-else* para que enquanto a condição do *if* estiver sendo satisfeita, os valores serão inseridos no início, fim ou por posição.

Ademais, foram criados 3 co*ntador*es (ins_inicio, ins_fim e ins_pos) que contatarão a quantidade de entradas na lista ocorreu por cada método de inserção e com isso fará impressão, tendo o mesmo ocorrido com as demais. Além disso, o programa computa o tempo de execução das operações de inserção usando a função ***clock()***. Essa medição é importante para avaliar o desempenho do programa e comparar diferentes abordagens de implementação.

A função ***contaNosEnc*** conta o número de nós em uma lista encadeada. Ela percorre a lista - do mesmo modo, o ***contaNosDu*** só que em uma lista duplamente encadeada, incrementando um contador a cada nó encontrado, até que alcance o final da lista (quando o ponteiro *lst* se torna NULL). O valor do contador é então retornado, representando o número total de nós na lista encadeada.

Estrutura	Qtd. Ins. Início	Qtd. Ins. Fim	Qtd. Ins. Pos	Menor tempo	Maior tempo	Tempo médio
Lista estática com capacidade de 10.000	3333	3333	3334	0.040725	0.046585	0.0430396
Lista dinâmica com encadeamento (simples)	3333	3333	3334	0.089987	0.92586	0.0909906
Lista dinâmica duplamente encadeada	3333	3333	3334	0.11598	0.123865	0.055519

Em relação à avaliação da estrutura de dados mais eficiente, isso dependerá do contexto e dos requisitos específicos do programa. Cada tipo de lista tem suas próprias vantagens e desvantagens em termos de desempenho e uso de memória.

Neste caso, A lista estática é definitivamente mais eficaz que a dinâmica em questão de velocidade, no entanto há um tamanho máximo predefinido. Já a lista encadeada simples é mais eficaz em relação ao consumo da memória além de ser menos complexa em quando comparada com a lista duplamente encadeada, todavia, só é possível percorrer por um lado já que os nós apontam apenas para o nó seguinte.

A lista duplamente encadeada pode ser percorrida em ambos os lados, tendo em vista que há a possibilidade de começar pelo "início" ou pelo "fim" e navegar nó por nó, podendo assim escolher o caminho mais eficiente, apresentando um consumo de memória relativamente alto.

Em termos de eficiência de memória, as listas estáticas podem ser mais eficientes se o número de elementos for conhecido antecipadamente e se for relativamente estável. No entanto, caso apenas uma pequena parte dela for utilizada, isso pode levar ao desperdício de memória. Visto que o seu tamanho não varia com a quantidade de dados dentro dela.

Em listas encadeadas simples, cada nó contém o valor e um ponteiro para o próximo nó. Isso permite que a lista encolha ou cresça dinamicamente durante a execução do programa. Contudo, cada nó requer memória extra para armazenar o ponteiro, o que resultar em maior uso da memória se a lista contiver muitos elementos. Já em listas duplamente encadeadas, cada nó contém um valor e dois ponteiros (Um ponteiro para o próximo nó e para o nó anterior. Dessa forma, permitindo a navegação em ambas as direções na lista, mas também requer mais memória para armazenar

- Lista estática: 4 allocs, 4 frees, 6,616 bytes allocated.
- Lista encadeada: 10,004 allocs, 5 frees, 166,616 bytes allocated.
- Lista Duplamente encadeada: 10,004 allocs, 5 frees, 246,616 bytes allocated.

Foi utilizada a ferramenta Valgrind para a análise de desempenho no contexto da memória. A suíte precisa ser instalada na em sua máquina, é compilado normalmente o programa, logo após, usando a ferramenta com a opção $ valgrind --leak-check=full. */nomedoprog*, será produzirá uma saída detalhada no terminal, indicando possíveis vazamentos de memória, alocações e desalocações.