Laboratório de Estrutura de Dados

Primeira versão do projeto da disciplina

Estruturas de dados

Arthur Felipe Muniz Sales - 142086649

1. Introdução

Este relatório corresponde ao relato dos resultados obtidos no projeto da disciplina de LEDA que tem como objetivo implementar algumas estruturas de dados, substituindo assim, em algumas partes do código onde era utilizado vetores por tais estruturas.

O projeto inicial consistia em utilizar apenas vetores em sua composição, não sendo permitido quaisquer outras estruturas, o que leva a ser mais verboso relacionado a quantidade de código escrito no projeto. Na entrega do projeto da segunda unidade, deveríamos substituir em algumas partes do código onde se utilizava de vetores por outras estruturas de dados estudadas em sala de aula, além de implementar tais estruturas, deve-se também informar o motivo da escolha das estruturas implementadas.

Foi possível notar uma maior velocidade em manipular os dados, mas também um aumento maior consumo de memória, porém de forma geral, o processamento por ter sido maior, acaba compensando a utilização das estruturas, mas existiu um pouco de dificuldade por conta do ambiente (Computador) utilizado no teste, já que o mesmo não possui um quantidade grande de memória e nem de processamento.

Na parte 2 do documento, será comentado quais estruturas de dados foram utilizadas e características de cada estrutura utilizada. Além disso, irei abordar onde e qual estrutura foi implementada no projeto.

Na descrição geral do ambiente de teste, comentarei qual a configuração do ambiente onde foi realizado o teste e desenvolvimento da ferramenta.

2. Descrição geral sobre o método utilizado

Foram utilizadas três estruturas, seguindo a orientação do professor em utilizar as que foram feitas nas atividades da segunda unidade, além de escolher uma outra que eu mesmo quisesse escolher.

As estruturas escolhidas foram, *listas encadeadas*, *filas* e *HashMap*. Antes de falar o motivo da escolha, falarei um pouco sobre cada estrutura, comentando sobre características, vantagens e desvantagens.

Listas Encadeadas:

As listas encadeadas são estruturas de dados onde os elementos estão armazenados de forma não contígua na memória. Cada elemento, chamado de nó, contém o dado a ser armazenado e um ponteiro (ou referência) para o próximo nó na sequência. A lista é percorrida a partir do primeiro nó (chamado de cabeça) seguindo os ponteiros até chegar ao último nó, que aponta para NULL.

Caracteristicas:

Consiste em nós conectados, onde cada nó contém um valor e um ponteiro para o próximo nó na sequência.

Não requerem espaço contíguo na memória.

Vantagens:

Inserção e remoção eficientes no início (tempo constante) se houver referência à cabeça.

Não requer espaço contíguo na memória, o que pode ser útil quando o espaço é fragmentado.

Desvantagens:

Acesso sequencial aos elementos, não é tão eficiente para acessar elementos aleatórios.

Requer mais espaço devido aos ponteiros de ligação entre os nós.

Filas:

As filas são estruturas de dados que seguem o princípio "primeiro a entrar, primeiro a sair" (FIFO - First-In-First-Out). Novos elementos são adicionados ao final da fila (chamado de "enfileiramento" ou "enqueue"), e os elementos são removidos do início (chamado de "desenfileiramento" ou "dequeue").

Caracteristicas:

Seguem o princípio FIFO (First-In-First-Out) para adição e remoção de elementos.

Operações básicas: enfileirar (adicionar um elemento) e desenfileirar (remover o próximo elemento).

Vantagens:

Estrutura simples e fácil de entender.

Útil para casos onde a ordem de chegada é crucial, como em sistemas de processamento de tarefas.

Desvantagens:

Acesso apenas ao primeiro elemento da fila.

Implementações baseadas em arrays têm tamanho fixo, podendo causar problemas de capacidade.

HashMap:

Um hashmap (ou tabela hash) é uma estrutura de dados que mapeia chaves únicas para valores correspondentes. Funciona através de uma função de hash que transforma a chave em um índice na estrutura de armazenamento (array).

Caracteristicas:

Mapeia chaves únicas para valores correspondentes usando uma função de hash.

Armazena pares chave-valor em uma estrutura que utiliza a função de hash para acesso eficiente.

Vantagens:

Busca rápida de valores usando chaves (geralmente tempo médio O(1)).

Eficiente para acesso aleatório aos elementos.

Desvantagens:

Colisões de hash podem ocorrer, exigindo tratamento especial.

Pode consumir mais memória, especialmente em casos de colisões.

Motivo de escolha de cada estrutura.

As escolhas de cada estrutura tem um motivo simples, por exemplo, decidi escolher a estrutura de lista encadeada para leitura dos arquivos, como não sabemos a quantidade exata de dados, fez mais sentido em utilizar-la, uma vez que outros arquivos podem ser lidos e com tamanhos diferentes, logo escolher essa estrutura me parece ser a escolha mais eficiente.

Já a estrutura de fila, foi utilizada para classificar as senhas, como é preciso ler a senha e verificar o seu grau para depois qualificá-la como ruim ou boa e etc, e como deve ser feito em sequência, nada melhor que utilizar a fila, uma vez que o primeiro a entrar deverá ser processado e também o primeiro a sair, facilitando a verificação de todas elas.

Já a HashMap, foi utilizada para selecionar as senhas Boa e Muito Boa, como a porcentagem em comparação a todo o arquivo, a quantidade de senhas com essas qualificações são pequenas, salvando a linha toda na hash, impossibilitando/dificultando as colisões de hash.

Descrição geral do ambiente de testes

Os testes foram desenvolvidos com 3 arquivos de tamanhos diferentes, de 1000, 2000 e 5000. Com apenas essas três listas, podemos analisar o crescimento de tempo de acordo com o tamanho da lista, para realizar uma análise com todas as informações das listas, demandaria vários dias apenas de testes, pois ordenar todas as informações requer muito tempo.

Devido a esse pequeno problema (tempo), acabei optando por esses tamanhos, desde que os campos sejam respeitados, a ordenação acontecerá!. Porém, é importante ressaltar que, ordenar toda a lista original, em meu caso, é impossível, uma vez que a minha máquina de teste é limitada.

Tendo como máquina com 4GB de ram, HD de 500GB, sistema operacional Windows 10 / 32 bits. Devido a falta de capacidade de testar grandes quantidades de dados, essa situação foi descartada.

O projeto foi desenvolvido com a linguagem Java version “19.0.1” 2022-10-18, foi também utilizada a IDE do Netbeans como editor de código. No projeto foi desenvolvido, além do solicitado, uma interface gráfica, para deixar a ferramenta mais elegante, não só isso, mas também para ser intuitiva e fácil de usar/entender.

A interface consiste de um menu, onde terá apenas informações com relação ao projeto e algoritmos, uma tela onde deverá ser mostrado as informações em tempo de execução, e dois botões, o primeiro para iniciar o processo. Caso o arquivo exigido não esteja no diretório solicitado, um alerta será emitido, e assim impossibilitando a continuação do processo.

Durante a execução dos testes, foi notado um aumento maior de memória, comparada com vetores que antes eram utilizados, mas a velocidade de processamento também aumentou.

3. Resultados e Análise

Como na ferramenta não foi implementado nenhum tipo de monitoramento de memória, fica complicado informar com precisão do uso/consumo de memória, sendo apenas utilizado o gerenciador de tarefas do windows, a parte da memória consumida. Foi notado que em vários casos, que a informação de consumo chegou muito próximo do consumo total, e em alguns momentos chegando de fato a 100% do consumo, o que me leva a acreditar que mesmo sendo mais rápido o processamento em comparação a como estava implementado anteriormente, só não foi mais rápido por conta desse consumo.

De forma geral, as estruturas de dados são super interessantes, e saber como e quando usá-las faz parte da vida de quem quer seguir como desenvolvedor, não só conhecendo a teoria das mesmas, mas principalmente em criá-las, utilizá-las e também, analisá-las.