UNIVERSIDADE FEDERAL DO PIAUÍ
CENTRO DE CIÊNCIAS DA NATUREZA
DEPARTAMENTO DE COMPUTAÇÃO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Grupo 08

KAUÃ MACHADO DA SILVA
JEOVANDO DA SILVA REIS JUNIOR
WENCYO RAFAEL LIMA DE SOUSA

Relatório Técnico:
Atividade de Participação 07
Java Collections API

Teresina-PI

2023

Relatório Técnico: Resolução das questões sobre a Java Collections API

1. Introdução.

Segue o relatório detalhado da Atividade de Participação 07 dos alunos Kauã Machado Da Silva, Jeovando Da Silva Reis Junior e Wencyo Rafael Lima De Sousa, realizada na disciplina de Estrutura de Dados do Departamento de Computação - CCN da Universidade Federal do Piauí (UFPI), sob a orientação do professor Raimundo Moura

1.1. Descrição do Problema

O problema consiste em explicar as classes do Java Collectins API, além de apresentar uma análise de desempenho das operações realizadas em diferentes estruturas de dados, incluindo ArrayList, LinkedList, HashSet, LinkedHashSet, TreeSet, HashMap, LinkedHashMap e TreeMap. Foram realizadas operações de adição, busca e exclusão em cada uma das estruturas de dados

1.2. Decisões de Implementação

Foi optado pelo uso de cinco package, a List, Map,Set onde estão os algoritmos das classes; App onde fica a Main para a execução dos algoritmos, a File onde estão os arquivos para ser testados a classes Arq que possui os métodos de leitura, escrita dos arquivos além de uma função auxiliar que lê o arquivo principal e gera um novo de forma mais organizada e sem palavras repetidas e sinais de pontuações, tornando o codigo mais otimizado. StopwatchCPU que faz o monitoramento e o medir o desempenho dos algoritmos em segundos de execução.

Foi desenvolvido ainda um programa em Python para geração dos gráficos utilizando da biblioteca MatplotLib.

.

2. Questões

1. Classes Vector, LinkedList e ArrayList.

As classes Vector, LinkedList e ArrayList são três implementações da interface List em Java, que representa uma coleção de elementos que podem ser acessados por um índice.

Vector e ArrayList são muito parecidos. O vector é anterior a inclusão da API Collection, eles utilizam um array interno para armazenar os elementos, e aumentam o tamanho desse array conforme a necessidade. A principal diferença entre eles é que o Vector é sincronizado, assim, considerada obsoleta e menos eficiente do que a classe ArrayList, que não é sincronizada por padrão.

A principal vantagem do vector consiste dele ser sincronizado, pode ser usado como uma pilha, tem métodos para obter a capacidade e o fator de incremento do array interno. Quanta a principal desvantagem é mais lento do que o ArrayList, tem métodos obsoletos que podem causar confusão, não é recomendado para uso em novas aplicações.

Tendo o ArrayList como principais vantagens: a eficiência para operações de acesso aleatório, pode ser facilmente acessado por qualquer método que implemente a interface List, pode ser facilmente convertido para um array. Por outro lado, as desvantagens, não é

tão eficiente quanto LinkedList para operações de inserção e remoção, pode usar mais espaço do que LinkedList.

LinkedList é uma estrutura de dados que mantém uma lista de objetos. Os objetos são armazenados em uma lista ligada, que é uma lista de nós, onde cada nó contém um objeto e um ponteiro para o próximo nó e para o nó anterior.

As principais vantagens, é eficiente para operações de inserção e remoção, não usa tanto espaço quanto ArrayList ou Vector. Enquanto as desvantagem, não é sincronizado, o que significa que não é seguro para uso em threads concorrentes, não pode ser facilmente acessado por qualquer método que implemente a interface List,não pode ser facilmente convertido para um array.

2. Classe HashSet, LinkedHashSet e TreeSet

HashSet, LinkedHashSet e TreeSet são conjuntos em Java. Todos eles implementam a interface Set. A diferença entre eles é a forma como os elementos são armazenados e recuperados.

O HashSet usa uma tabela de hash para armazenar seus elementos. Isso significa que ele não mantém a ordem dos elementos e não permite elementos duplicados. É rápido para operações de inserção, remoção e busca; não guarda a ordem de inserção dos elementos e não permite elementos duplicados. Entretanto, pode usar mais memória do que outros tipos de conjuntos e pode ser difícil pesquisar elementos em um HashSet se você não souber o valor exato do elemento.

O LinkedHashSet é semelhante ao HashSet, mas mantém a ordem dos elementos em que foram adicionados. Ele usa uma lista vinculada para manter a ordem dos elementos. Porém, diferentemente dos outros conjuntos pode ser difícil pesquisar elementos em um LinkedHashSet se você não souber a ordem exata em que os elementos foram inseridos.

O TreeSet armazena seus elementos em uma árvore binária. Isso significa que ele mantém os elementos em ordem classificada e pode ser usado para classificar conjuntos. No entanto, ele pode ser mais lento do que outras implementações.

3. Classe HashMap, LinkedHashMap e TreeMap

Os três tipos de mapas em Java são HashMap, LinkedHashMap e TreeMap. Todos eles implementam a interface Map e têm métodos semelhantes.

HashMaps usam uma tabela hash para armazenar pares de chave-valor. A chave é usada para calcular o índice na tabela hash, onde o valor é armazenado. HashMaps são rápidos para operações de inserção, remoção e busca. Eles também permitem elementos duplicados. Ademais, HashMaps não mantêm a ordem de inserção dos elementos. Eles também podem usar mais memória do que outros tipos de mapas.

LinkedHashMaps são semelhantes a HashMaps, mas eles mantêm uma lista ligada das chaves. Isso permite que você itere sobre os elementos em ordem de inserção. Pode ser destacado que o LinkedHashMaps são rápidos para operações de inserção, remoção e busca.

Eles também permitem elementos duplicados. Eles também mantêm a ordem de inserção dos elementos. Todavia, LinkedHashMaps podem usar mais memória do que HashMaps.

TreeMaps são implementados como árvores binárias balanceadas. Isso permite que você itere sobre os elementos em ordem crescente ou decrescente. O TreeMaps são rápidos para operações de busca também não permitem elementos duplicados e sempre mantêm os elementos em ordem. Em contraponto podem ser mais lentos do que outros tipos de mapas para operações de inserção e remoção de elementos em sequência. Eles também podem usar mais memória do que outros tipos de mapas.

4. Gráfico Tempo de execução: Inserção

As implementações da interface List tiveram em média os melhores desempenhos. E as herdeiras da Map tiveram os piores desempalho. A TreeMaps que são implementados como árvores binárias balanceadas obteve o pior desempenho. Enquanto que, as implementações da interface Set tiveram resultados semelhantes.

Os dados do tempo de exceção para função solicitada nesta questão, assim como, para as questões subsequentes, foram obtidos com o StopwatchCPU. Também, destacamos que o arquivo do tipo txt indicado para usado foi tratado previamente e apenas as palavras (Strings) sem repetições foram incluídas uma estrutura tipo lista. Tal procedimento foi adotado especialmente para que as métricas de tempo das funções não capturassem o tratamento das repetições.

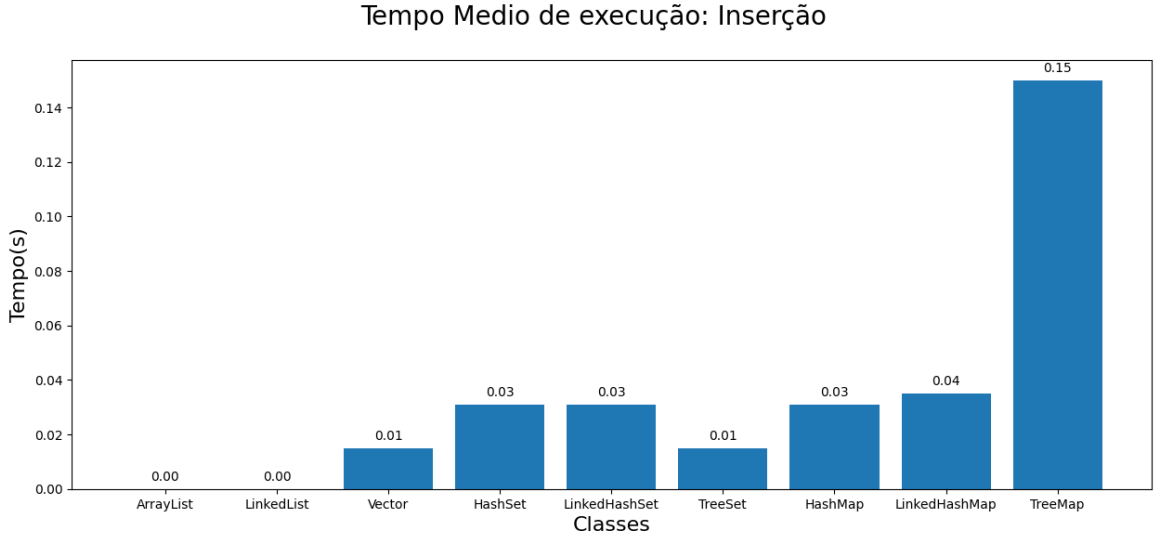

Imagem 1: Gráfico do tempo de inserção

5. Implementação do código para consultar 10 palavras

As palavras a serem consultados foram incluídas um ArrayList onde um método pesquisar busca cada palavra na estrutura analisada, sendo que cada estrutura tem seu método. Como por exemplo a o método buscar do linkedList.

```java
public String buscar(String[] palavras) {
        stopwatch = new StopwatchCPU();
        for (String palavra : palavras) {
            if (!buscar(palavra)) {
                return "False";
            }
        }
        double timeb = stopwatch.elapsedTime();
        return "HashMap: "+ timeb;
    }
```

Imagem 2: Exemplo de método de buscar.

6. Gráfico Tempo de execução: Busca

Quanto a busca as implantações do List e Set auferiram números próximos. Nesta operação os Map tiveram os melhores resultados com tempo de execução muito próximo de zero.

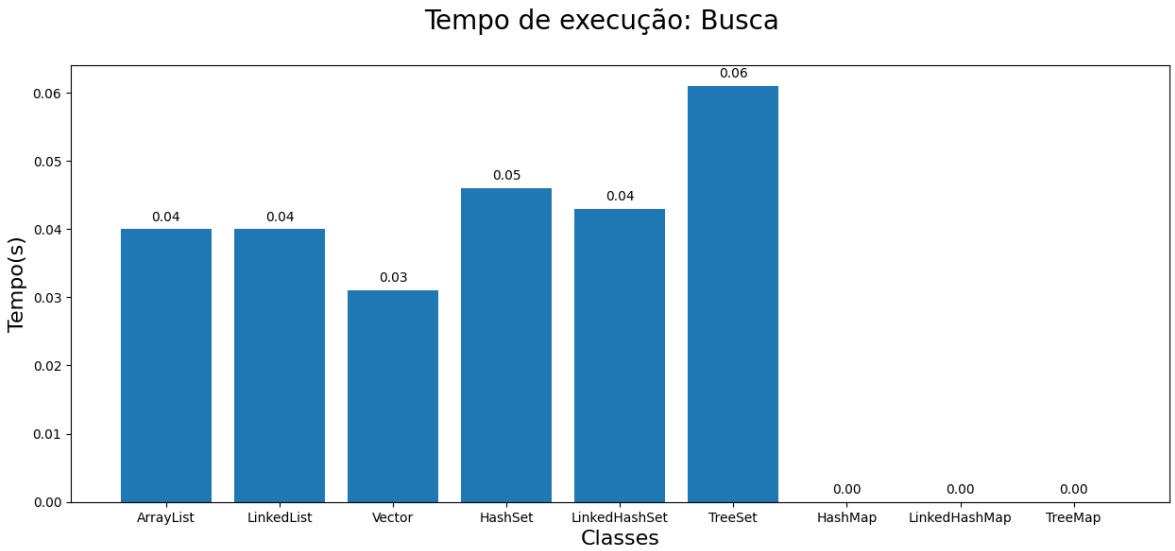

Imagem 3: Gráfico do tempo de pesquisa

7. Implementar do código para excluir 10 palavras

Dado o fato que as palavras a serem excluídas e buscada serem as mesmas, foi usado a lista criada para intromete. A função receber a lesta supracitada usado o método removeAll como no exemplo de uma das classes trabalhada. Assim como pesquisar cada uma das 9 classes trabalhas tem seu método apagar.

```java
 private boolean deletar (String palavra) {

        return hashMap.remove(palavra) !=
null}
    public String deletar(String[] palavras) {
        stopwatch = new StopwatchCPU();
        for (String palavra : palavras) {
            if (!deletar(palavra)) {
                return "False";
            }
        }
        double timed = stopwatch.elapsedTime();
        return "HashMap: "+ timed;
    }
```

Imagem 4: Código para exclusão de palavras da estrutura.

8. Gráfico Tempo de execução: Exclusão

Na exclusão todos os métodos tiveram tempo de execução próximos com exceção das implementações da List. Ainda, é valido destacar que o vector foi o que obteve o pior desempenho, que entre todas as estruturas de armamento testada é a mais antiga e até considerada obsoleta.

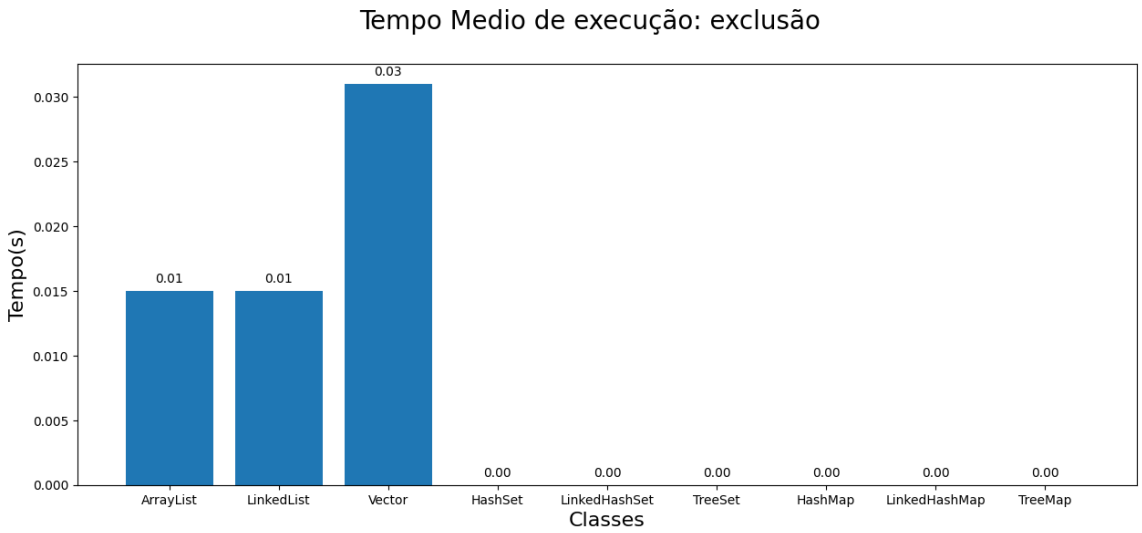

Imagem 5: Gráfico do tempo de pesquisa

3. Conclusão

Com base nos resultados obtidos, é possível observar que o desempenho das estruturas de dados varia dependendo do tipo de operação realizada e do volume de dados.