Estruturas de Dados

Relatório trabalho final

Alunos:

João Victor Santos Andrade

João Rodrigues de Melo Neto

Objetivo: Responder as 15 questões entregues.

Questões:

1. A classe Vector se baseia em ser um Array como qualquer outro. Contém index e reserva um espaço predeterminado na memória, porém sua mecânica de redimensionamento é feita em 100%, ou seja, o dobro. Esse fato pode ou não ser benéfico dependendo da sua utilização, pois toda a vez que um array é redimensionado ele tem um custo exponencial ao número de espaço ocupados, além de ser “thread–safe”, resumidamente duas ações em um mesmo arquivo sem uma interferir na outra. Vale ressaltar que classes “List” aceitam dados repetidos.

2. A classe LinkedList é uma lista duplamente encadeada, sendo assim temos vantagens clara como: uma inserção rápida visto que é sempre no início, porém sua remoção é lenta visto que o sistema precisa percorrer toda a lista no pior caso e pelo mesmo motivo o método para pegar uma informação. Uma vantagem é a alocação dinâmica visto que a aplicação cria espaços na memória a média em que inserções vão acontecendo.

3. A classe ArrayList é um array assim como o Vector já descrito a cima e exatamente com as mesma características: index para o aceso rápido e espaços previamente definidos pelo programa, porém para não serem exatamente iguais, o Array tem um redimensionamento em 50%, ou seja metade, o que tem uma vantagem em economia de espaços porém caso o array tenha de ser redimensionado várias vezes o processamento tende a ser prejudicado.

4. Primeiramente Classes “Set” tem como restrição não aceitar elementos repetidos. Falando sobre a HashSet é o mais rápido dos três, utiliza uma HashTable e tem como vantagem ter a mesma complexidade para adicionar, remover ou retirar no caso O (1). O que tornar ele extremamente útil para tratamento de milhões de dados em troca ele não se preocupa em organizar os dados. Importante de lembrar que os métodos Set não são Thread – safe, ou seja, o programador deve criar formas que permitam a execução de várias tarefas em um mesmo dado diferente das Classes List.

5. Primeiramente Classes “Set” tem como restrição não aceitar elementos repetidos. Utilizando uma estrutura chamada red – black tree ou árvore rubro – negra. Como uma adição essa classe contém um método chamado SortedSet, ou seja, o armazenamento é ordenado. Como consequência quanto mais a elementos inseridos mais demorada a inserção será devido a reorganização da arvore. Sua complexidade nos métodos de inserção e remoção aumenta para O (log n). Importante de lembrar que os métodos Set não são Thread – safe, ou seja, o programador deve criar formas que permitam a execução de várias tarefas em um mesmo dado diferente das Classes List.

6. Primeiramente Classes “Set” tem como restrição não aceitar elementos repetidos. O LinkedHashSet traz um pouco da performance do HashSet e um pouco da capacidade de ordenação de TreeSet. A classe faz uso de uma HashTable com LinkedList, ou seja, para operações básicas a complexidade é O (1) e a ordenação é garantida. Importante de lembrar que os métodos Set não são Thread – safe, ou seja, o programador deve criar formas que permitam a execução de várias tarefas em um mesmo dado diferente das Classes List.

7. Cabe citar que Classes “Map” utilizam um código Hash para a criação de chaves e em caso de repetições a chave nova sobrescreve a chave antiga. A classe HashMap tem como características: elementos não ordenados e inconstantes, uma busca e inserção rápida de dados e permite a inserção de chaves e valores nulos. Seu comportamento é muito semelhante a HashTable excerto pelas características já citadas. Também não se preocupa com a ordenação dos elementos. Ademais tem uma inserção e uma coleta muito rápida da ordem de O (1) e utiliza uma lista ligada. Também é notável citar que as Classe Map não são sincronizadas, ou seja, não garante a exclusão mútua de duas ações em um mesmo dado.

8. Cabe citar que Classes “Map” utilizam um código Hash para a criação de chaves e em caso de repetições a chave nova sobrescreve a chave antiga. A classe LinkedHashMap utiliza uma lista duplamente ligada, suas chaves e valores podem ser nulos e a ordenação é na ordem de inserção, sua complexidade é da ordem de O (1) nos métodos “insert” e “get”. Também é notável citar que as Classe Map não são sincronizadas, ou seja, não garante a exclusão mútua de duas ações em um mesmo dado.

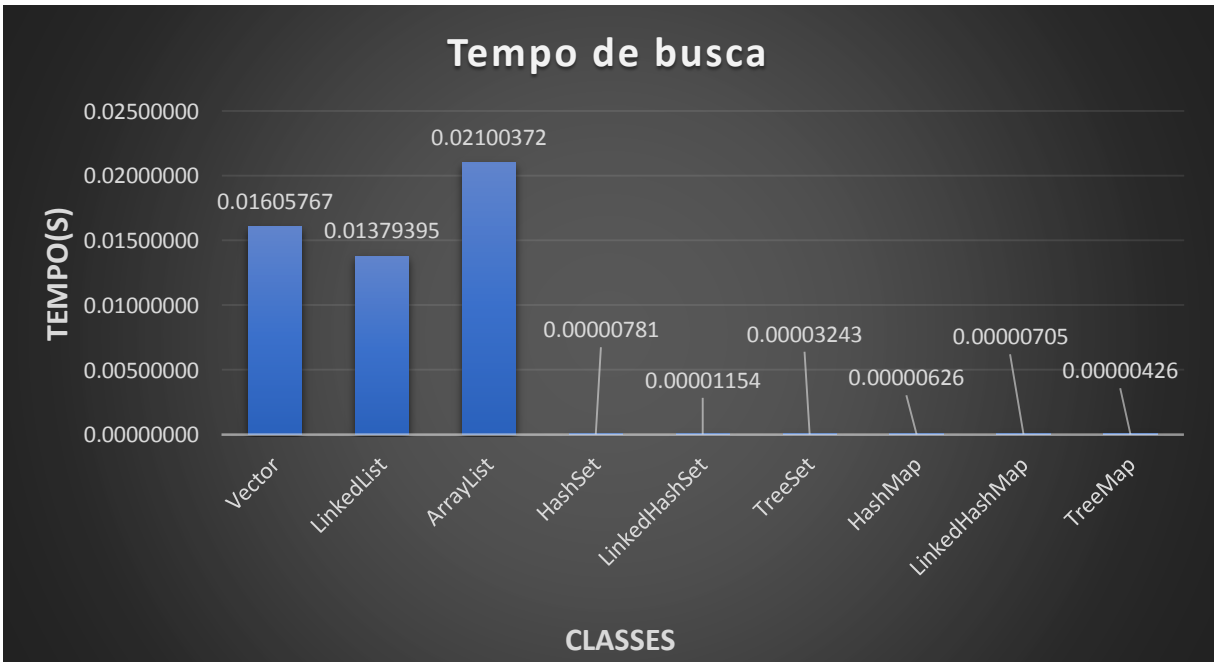
9. Cabe citar que Classes “Map” utilizam um código Hash para a criação de chaves e em caso de repetições a chave nova sobrescreve a chave antiga. A classe TreeMap utiliza uma árvore rubro – negra, tem como restrição a de que apenas valores podem ser nulos e tem a complexidade dos métodos “insert” e

“get” da ordem de O (log n). Sua ordenação é ordenada a isso se deriva o fato de uma inserção mais lenta. Também é notável citar que as Classe Map não são sincronizadas, ou seja, não garante a exclusão mútua de duas ações em um mesmo dado.

10.

11. Feito no código.

12.



13. Feito no código.

14.

15. A Classe Priority queue é um ADT onde cada elemento tem uma prioridade associada. Sua implementação foi feita com Árvores: caso dois elementos tenham a mesma prioridade a ação prioriza a ordem de inserção, esta ordem de inserção depende da classe utilizada para fazer a *PriorityQueue*, a classe precisa implementar o método da classe *Comparable*(*compareTo*) utilizando o atributo escolhido pelo programador para fazer a comparação, assim, conseguir o critério da prioridade. Em se tratando de complexidade das árvores utilizadas a mais lenta é da ordem de O (log n) e na questão de algoritmos de ordenação o melhor dos piores casos fica na Ordem de O (n log n) e o pior dos piores casos na ordem de O (n^2).