3. Este item diz respeito à escolha da melhor estrutura de dados para resolver os problemas listados abaixo. Para cada um dos itens abaixo, implemente uma solução utilizando alguma das estruturas de dados vistas em aula: vetores, vetores ordenados, arraylists, listas ligadas, tabelas hash, ou árvores. Utilize Collections e as implementações de maps e arraylist disponíveis no JDK ou fastutils. Faça um texto (relatório em .pdf), justificando a escolha das estruturas de dados escolhidas para cada um dos itens abaixo. Inclua no relatório a resposta das perguntas abaixo.

Cada arquivo ".txt" foi lido, filtrado os caracteres especiais, quebrado por palavras e armazenado em um ArrayList do tipo String. Foi utilizado ArrayList pelas suas facilidades de manuseio, porém armazenar arquivos grandes se torna uma operação muito custosa sendo *"n^2"*

(a) Crie um método toString para Livro que apresente os atributos na tela (exceto conteúdo).

R: Foi Criado um método dentro do objeto Livro chamado "toString()" que retorna as informações retiradas da primeira linha do livro e armazenadas em suas respectivas variáveis.
Sendo uma operação em *"n"*

(b) Quantas palavras distintas existem em cada livro?

R: Usou-se a estrutura HashMap para realizar tal operação, pois a estrutura armazena somente chaves distintas umas das outras. Assim quando adicionamos cada palavra do livro como uma chave do HashMap as palavras iguais apontavam para a mesma chave e a chave anterior era substituída pela nova, não deixando haver repetições.

Usando a função "size()" pode-se saber o tamanho do Hash e determinar quantas palavras distintas existiam.

Operação "size()" está em "*n".*

(c) Quantas palavras distintas existem em toda biblioteca?

R: Foi Criado um TreeMap de Strings para essa função, para podermos realizar algumas buscas específicas posteriormente. Dentro dessa Tree foi adicionado o conteúdo do HashMap da questão anterior. Fazendo isso todas as vezes que um novo arquivo TXT era lido e armazenado no HashMap.

Usando a função "size()" pode-se saber o tamanho do TreeMap e determinar quantas palavras distintas existiam.

Operação "size()" está em "*n".*

(d) Quantas vezes a palavra "anyone" (ou qualquer outra) aparece em cada livro?

R: A operação foi realizada através da função "get([chave])" na estrutura HashMap.

A operação é direta, sendo de custo "1".

(e) Quantas vezes a palavra "anyone" (ou qualquer outra) aparece em toda a biblioteca?

R: Essa operação foi realizada através do armazenamento do resultado da busca "get([chave])" na estrutura HashMap em uma variável tipo "int" e somando cada novo valor com o anterior.

A operação é do tipo "*n".*

(f) Qual a palavra mais frequente em cada livro?

R: A resposta corresponde com a resposta da questão (h).

(g) Qual a palavra menos frequente em cada livro?

R: A resposta corresponde com a resposta da questão (i).

(h) Qual as cinco (ou qualquer outro número) palavras mais frequentes em cada livro?

R: Primeiramente foi criado um List com um Map.Entry com parâmetro, onde será guardado as diferentes palavras do livro e sua quantidade. Após serem armazenadas, o List é ordenado utilizando o Collections.sort.

Foi criado um outro List com o parâmento Map.Entry, mas agora recebendo o List principal com o método subList com os parâmetros (0, 5). Pegando assim, as primeiras posições do List ordenado, ou seja, as cinco palavras com maior frequência no livro.

(i) Qual as cinco (ou qualquer outro número) palavras menos frequente em cada livro?

R: Foi utilizado o mesmo List ordenado da questão anterior.

Foi criado um outro List com o parâmetro Map.Entry, recebendo o List principal com o método subList, mas agora com os parâmetros (statistics.size() - 5, statistics.size()).

O primeiro parâmetro, tanto na questão anterior quanto nesta indica a posição inicial que está sendo pego o elemento, neste caso, foi utilizado o statistics.size() para saber o tamanho do List, sabendo assim a última posição, ou seja, foi utilizado o -5 para começar 5 posições antes da final, pegando assim os elementos com menor quantidade do livro.

Operação “size()” está em “*n”.*

(j) Quais as palavras com frequência acima de 1000 (ou qualquer outro número) na biblioteca?

R: Foi realizada a leitura de todos os (TreeMaps palavrasIguais) e adicionados em outro TreeMap de forma ordenada, invertendo conteúdo e chave. Esta operação está em “*n”.*

Operação “tailMap([chave])” está em *“Log n”.*

(k) Retorne a lista de palavras que tenham ordem lexicográfica acima de “warm” (ou qualquer outra).

R: Foi utilizada a função “tailMap([chave])” presente na estrutura TreeMap, que retorna a chave e todas as chaves acima dela.

Usou-se TreeMap pois as buscas nela são de “*~LogN”*

(l) Retorne a lista de palavras que tenham ordem lexicográfica abaixo de “asked” (ou qualquer outra).

R: Foi utilizada a função “headMap([chave])” presente na estrutura TreeMap, que retorna a chave e todas as chaves acima dela.

Usou-se TreeMap pois as buscas nela são de “*~LogN”*