Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 6 REPORT

STUDENT NAME
Can BEYAZNAR
STUDENT NUMBER
161044038

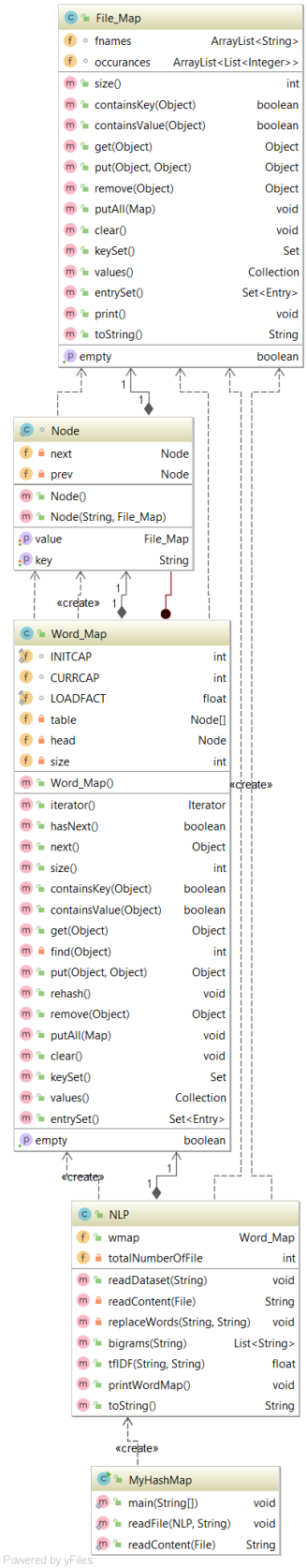
1 INTRODUCTION

1.1 Problem Definition

In this assignment, we are asked to read the texts in the files given to us. Taking the words one by one in these texts, we are asked to find out which files contains these words. We are also asked to find out what words they are in each text. And we want to solve this problem using the Map interface.

2 METHOD

2.1 Class Diagrams



2.2 Problem Solution Approach

2.2.1 File_Map Class

In File_Map class, ArrayList <String> fnames element keeps the names of the files we read. ArrayList <List <Integer >> occurances element lists the index of the words in the file we read. Thus, we will be able to see how often each word we read is in what files. And we need to implement the map class to make this class map. To implement the Map class, we need to override the methods of this map class. While applying these methods, we have to use the methods in ArrayList collection. In addition, we must use TreeSet collection to maintain the sequence of elements in entrySet method.

Time Complexities of File_Map Class:

size()	get()	put()	remove()	containsKey()	remove()
O(1)	O(n)	O(n)	O(n)	O(n)	O(n)
putAll()	clear()	keySet()	values()	containsValue()	entrySet()
O(n)	O(n)	O(n)	O(n)	O(n)	O(n)

2.2.2 Word_Map Class

In the Word_Map class, we need to type the same HashMap class. To implement this class, we need to implement the Map class. And we need to override the methods in the Map class. During the element insertion, the solution of the linear probe is applied to prevent the index of two different elements from being matched. In addition, when using this solution, we use the "find" method to find the position of the element to be added in the element insertion method. We also create a static Node class for each of the words to be linked in this class. Thus, we do not go through the empty elements to reach the words and reduce the time loss. In order to reach these words more easily, we make this class iterable and can easily navigate between words. And also we implement our next(), hasNext() and iterator() methods.

Time Complexities of Word_Map Class:

Method Name	size()	get()	put()	remove()	containsKey()	rehash()	find()
Time Complexity	O(1)	O(1)	O(1)	null	O(1)	O(n)	O(1)
Method Name	**putAll()**	**clear()**	**keySet()**	**values()**	**containsValue()**	**entrySet()**	**isEmpty()**
Time Complexity	O(n)	O(1)	O(n)	O(n)	O(n)	null	O(1)

2.2.3 NLP Class

The NLP class allows you to keep the data of words in all files in the wmap element of the Word_Map type. To do this, we use the readDataSet method. In this method, using the folder name we received as a parameter, we open all the files in the folder. When opening these files, we remove characters such as commas and periods in this text. And we only get a text of words. Then we assign all the words in this text to our "wmap" element one by one. So we keep all of our information in our wmap element. The time complexity of readDataSet method is O(count of files * count of words) so it is O(n). In the Bigram method, the word given to us in the file combined with the word attached to it is requested to be put into a list. To do this, we first get the names of the files in which our word is found and the indices where they are found by using the methods in wmap. Then, we take the text in these files one by one and we look for the word next to the word given to us. And we combine these two words into the list. If these two words are already in the same list, no additions will be made. After all the files have been read, we return the list to the words we keep. The time complexity of this method is O(n). In the "tfIDF" method, you will be given a word and file name that will be found in the file. Then we open this file to get all the information needed for the formula. Then we calculate the TF and IDF formulas according to the information in our wmap element. And to find the TFIDF result, we multiply the two. And we return the result. Time complexity of this method is O(n).

3 RESULT

3.1 Test Cases

```
bigram very
tfidf coffee 0001978
bigram world
bigram costs
bigram is
tfidf Brazil 0000178
```

```
public static void main(String args[]) throws IOException {

    NLP x = new NLP();
    x.readDataset( dir: "dataset");

    readFile(x, FileName: "./src/input.txt");

    System.out.println("-o-o-o-o-o-o-o-o-o-o-o-o-");
    System.out.println("printWordMap Method : ");
    x.printWordMap();
    System.out.println("-o-o-o-o-o-o-o-o-o-o-o-o-");
}
```

The readFile method will read the file and execute the desired commands.

3.2 Running Results

The result is little bit long but we can show a little part of it.

```
[very difficult, very soon, very promising, very rapid, very aggressive, very attractive, very vulnerable]
0.0048781727
[world market, world coffee, world made, world share, world markets, world price, world bank, world as, world cocoa,
[costs have, costs and, costs of, costs Transport]
[is the, is possible, is not, is forecast, is expected, is caused, is depending, is slightly, is projected, is estima
0.0073839487
-o-o-o-o-o-o-o-o-o-o-o-o-
printWordMap Method :
24025 32553 certification 56578 classification pending 192710 261 392845 239 decrease 585794 3 585555 certified Orlea
-o-o-o-o-o-o-o-o-o-o-o-o-
```