GIT Department of Computer Engineering

CSE 222/505 - Spring 2023 Homework #6 Report

Pelin Erdoğan

210104004266

1. SYSTEM REQUIREMENTS

When the program executes you should press 1 to input string and 0 to exit.

2. PROBLEM SOLUTION APPROACH

Problem solution of arrange of string:

I have used toLowerCase and replaceAll function to qrrange string for creation of map.

Problem solution of Creation of mymap:

Mymap class has string str,linkedHashMap map and integer mapsize as a property of class. Constructor that takes string as a parameter arranges string and creates map. It splits string by its spaces and adds letters in words one by one. If a letter is already existing in map, it just pushes the word into info's arraylist of words.

Mymap methods:

public boolean isEmpty(): This method returns true if map is empty. İt is public because mymap should be usable for other projects to.

public info get (Character key): this method returns the info with given key. It uses an iterator to iterate around map.

protected Character getkeybyindex(int index): This method returns the key in given index and throws IndexOutOfBoundsException if index is out of bounds. Method is protected because it is only meant to be used by mergesort class. Method creates an iterator and iterates in map as much as index and returns the key in index.

public void put(Character key, info value): puts new element to end of map and increase mapSize

public void remove(Character key): removes element with the key given in parameter. Uses iterator.

public void putAll(mymap m): Deep copies the map to map given as parameter.

protected void addtoindex(int index,Character key, info value): This method adds new element to given index and if an element with new key already exists in another index swaps that one with the one in the index we want to add our new element to. Method creates another map and put values in it until index then adds the new element and puts rest of map. Then changes old map to new map. MEthod is protected because it is only meant to used by mergesort class.

public void clear(): Clears the map. Changes mapSize to 0;

public void print(): Prints the map

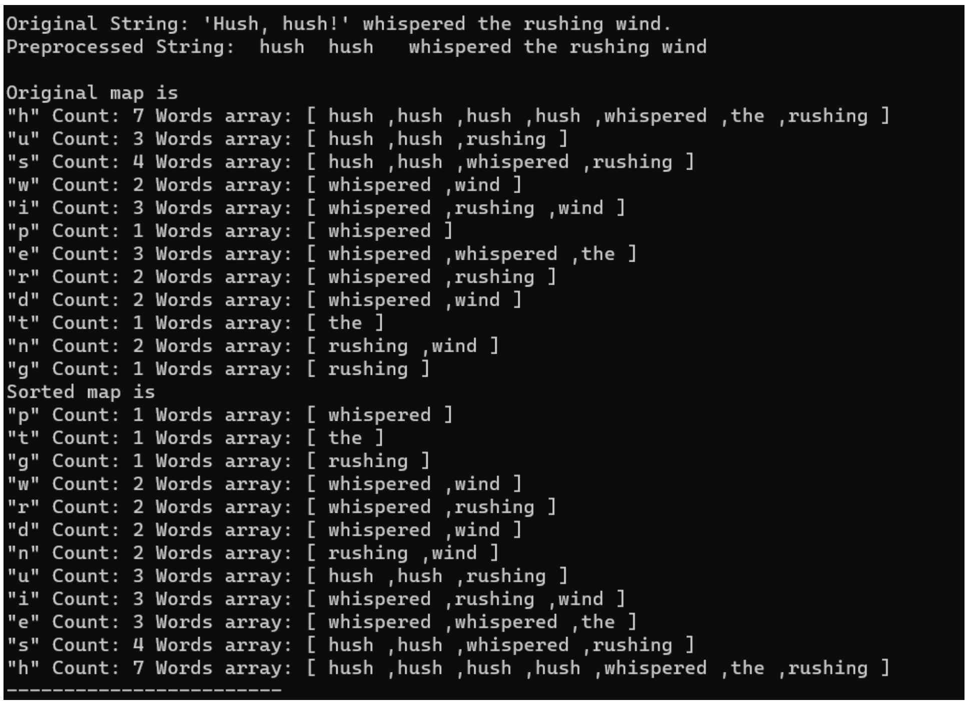
Problem solution of Merge sort:

This class has 2 mymap objects one to keep original map and one to sort. Constructor of class initilazes mymap objects and copies the map we to sort to one of them. Then assignsother map to map we want to sort.

MergeSort methods:

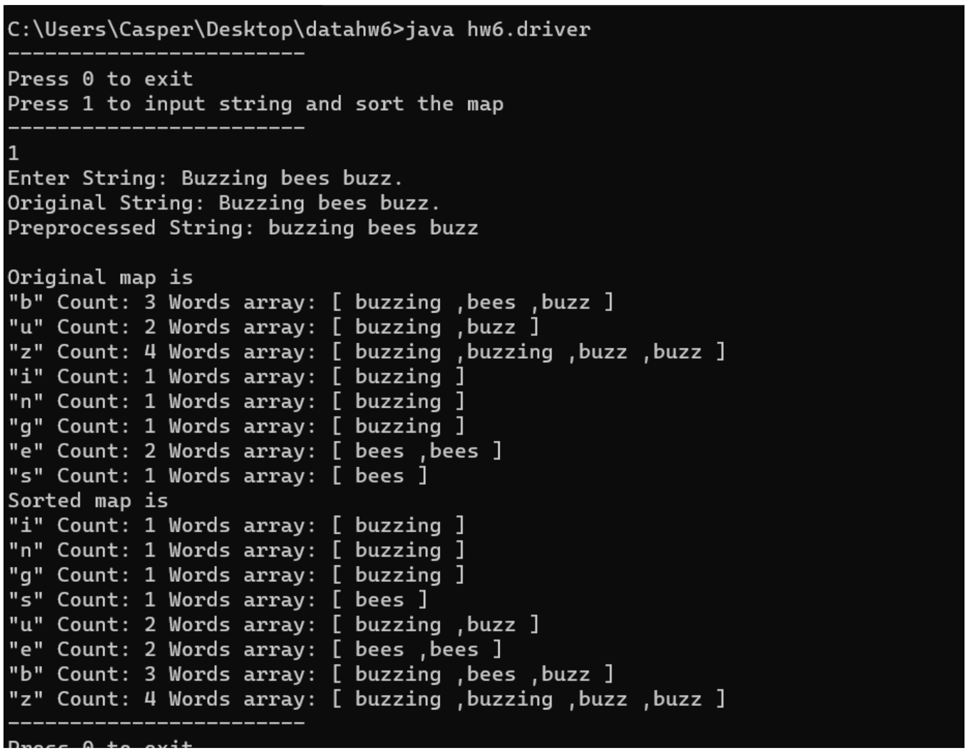
protected void divideandmerge(int left,int right): This method is only suitable for mymap so it is protected. It divides the map into little pieces with than calls merge function and it is a recursive function. It calls itself until left is bigger or equal to right.

private void merge(int left, int mid, int right): Method is private because I wanted to be only called by divideandmerge method. Method creates 2 other maps for left and right sides to merge. Puts the elements we want to merge inside them and then compares left ones with right ones and whichever is smaller adds that one in to original map first. This is why I have addtoindex and getkeybyindex. after comparison if there are any elements in right or left map that is not putted in place in original map. Puts them after the ones added before.



```
Original String: 'Hush, hush!' whispered the rushing wind.
Preprocessed String:  hush  hush   whispered the rushing wind

Original map is
"h" Count: 7 Words array: [ hush ,hush ,hush ,hush ,whispered ,the ,rushing ]
"u" Count: 3 Words array: [ hush ,hush ,rushing ]
"s" Count: 4 Words array: [ hush ,hush ,whispered ,rushing ]
"w" Count: 2 Words array: [ whispered ,wind ]
"i" Count: 3 Words array: [ whispered ,rushing ,wind ]
"p" Count: 1 Words array: [ whispered ]
"e" Count: 3 Words array: [ whispered ,whispered ,the ]
"r" Count: 2 Words array: [ whispered ,rushing ]
"d" Count: 2 Words array: [ whispered ,wind ]
"t" Count: 1 Words array: [ the ]
"n" Count: 2 Words array: [ rushing ,wind ]
"g" Count: 1 Words array: [ rushing ]
Sorted map is
"p" Count: 1 Words array: [ whispered ]
"t" Count: 1 Words array: [ the ]
"g" Count: 1 Words array: [ rushing ]
"w" Count: 2 Words array: [ whispered ,wind ]
"r" Count: 2 Words array: [ whispered ,rushing ]
"d" Count: 2 Words array: [ whispered ,wind ]
"n" Count: 2 Words array: [ rushing ,wind ]
"u" Count: 3 Words array: [ hush ,hush ,rushing ]
"i" Count: 3 Words array: [ whispered ,rushing ,wind ]
"e" Count: 3 Words array: [ whispered ,whispered ,the ]
"s" Count: 4 Words array: [ hush ,hush ,whispered ,rushing ]
"h" Count: 7 Words array: [ hush ,hush ,hush ,hush ,whispered ,the ,rushing ]
------------------------
```



```
C:\Users\Casper\Desktop\datahw6>java hw6.driver
------------------------
Press 0 to exit
Press 1 to input string and sort the map
------------------------
1
Enter String: Buzzing bees buzz.
Original String: Buzzing bees buzz.
Preprocessed String: buzzing bees buzz

Original map is
"b" Count: 3 Words array: [ buzzing ,bees ,buzz ]
"u" Count: 2 Words array: [ buzzing ,buzz ]
"z" Count: 4 Words array: [ buzzing ,buzzing ,buzz ,buzz ]
"i" Count: 1 Words array: [ buzzing ]
"n" Count: 1 Words array: [ buzzing ]
"g" Count: 1 Words array: [ buzzing ]
"e" Count: 2 Words array: [ bees ,bees ]
"s" Count: 1 Words array: [ bees ]
Sorted map is
"i" Count: 1 Words array: [ buzzing ]
"n" Count: 1 Words array: [ buzzing ]
"g" Count: 1 Words array: [ buzzing ]
"s" Count: 1 Words array: [ bees ]
"u" Count: 2 Words array: [ buzzing ,buzz ]
"e" Count: 2 Words array: [ bees ,bees ]
"b" Count: 3 Words array: [ buzzing ,bees ,buzz ]
"z" Count: 4 Words array: [ buzzing ,buzzing ,buzz ,buzz ]
------------------------
Press 0 to exit
```