GTU Department of Computer Engineering

CSE 222/505 - Spring 2023

Homework #6 Report

ÖMER SARIÇAM

200104004009

1)My Solution

1) To transform the String input into the desired format, I first used the replaceAll method to remove all non-letter and non-space characters, and then converted all characters in the string to lowercase using the toLowerCase() method. I performed these operations in the constructor of myMap class.

2) To create a LinkedHashMap map in the desired format, all the necessary operations were performed within the constructor of the myMap class. The constructor takes the string, which was prepared as mentioned above, as an input. In the constructor, I first initialized the class variables. I initialized the "str" variable to have the same content as the input. Then, I split the "str" string by spaces to obtain a String array called "arr" because the info class requires me to provide which word each letter belongs to. This way, I obtained word information. Next, I wrote a for loop that iterates as many times as the length of the "str" string. Here, each element of "str" is first obtained as a char, then converted to a String called "letter" containing a single letter using the toString() method. I convert it to a String because I need to take advantage of the methods of the String class, and the key is of type String. Then, I check if the "letter" variable is equal to a space character. If it is equal, I do not add it to the map, but simply increment the "word_number" variable indicating that we have moved on to the next word. If the "letter" variable is not equal to a space character, I then check if a key that is the same as that character exists in the map. If it does not exist, I add a new element to the map. If it exists, I simply increment its value. Additionally, the myMap class has a printMap() method that prints the map to the screen.

3) To apply the mergeSort operation, first an object of the mergeSort class needs to be created with a myMap object that needs to be sorted passed as a parameter. I assign the sent object to the originalMap variable in the mergeSort class, as I will not make any changes to this object. In the constructor, I also obtain the key values of the originalMap's map variable using the keySet() method and then convert this object, which is of the Set data type, to a String array using the toArray() method. I assign this String array to the arr variable in the mergeSort class. Once the mergeSort object is created in this way, the sort() method needs to be called to perform the merge sort operation. This method also calls the sort_helper method within it. The sort_helper method takes 2 parameters as left_index and right_index, which are the indexes of the array segment to be sorted.

In sorted_helper function we checks if left_index is less than right_index, if yes, then we finds the middle point of the array by calculating the average of the two indices. We then recursively sorts the left half of the array from left_index to middle_index and the right half of the array from middle_index+1 to right_index. The sort_helper method returns when right_index is not greater than left_index. After that, it calls the merge() function to merge the two halves of the array. The merge() function takes in three indices l, m, and r. l and r are the starting and ending indices of the array to be merged, and m is the middle index. The function creates two temporary arrays left_array and right_array to hold the left and right halves of the array to be merged. The function copies the elements from the arr array to the temporary arrays. After that, the function merges the two subarrays by comparing the values of the info class count of the respective elements in the originalMap. The smaller value is copied to the arr array, and the corresponding index is

incremented. Finally, the function copies the remaining elements of the two subarrays to the arr array.

After the sort_helper method finishes its job, we continue from where we left off in the sort() method. Now, the arr array contains the sorted originalMap keys. It's time to initialize the sortedMap. For this, we use the constructor of myMap that accepts a string and an ArrayList<info> as parameters. To create the string that will be sent as a parameter, the "arr" array is converted to a string called str using the join() method. To create the ArrayList<info> variable to be sent as a parameter, an ArrayList called valuesList is created. The values from the originalMap's map are added to this list in the order of the keys in arr. Then, the myMap constructor is called, and a LinkedHashMap sorted by the count value in the info object is obtained. Finally, the sortedMap is printed to the screen. While the elements in the sortedMap are arranged in order, there is no change in the originalMap.

2)How to run?

If you are in the hw6_200104004009_omer_saricam directory,

To compile:

$ javac *.java

To run:

$ java TestClass

- You can check the code by entering the string you want in the input variable in the main method in the TestClass.java file.

```java
import java.util.LinkedHashMap;

public class TestClass {



    Run | Debug
    public static void main(String[] args){

        String input = "'Hush, hush!' whispered the rushing wind.";

        myMap map_obj1 = new myMap(input);

        System.out.printf(format:"\nThe original (unsorted) map: \n");
        map_obj1.printMap();


        System.out.printf(format:"\n\n\nThe sorted map: \n");

        mergeSort sort_obj = new mergeSort(map_obj1);
        sort_obj.sort();

    }
}
```