Computer Engineering
CSE 222/505 - Spring 2023
Homework # Report

Name: Ahmet Furkan Ekinci
Student Number: 200104004063

1. Introduction

Purpose of Document

This document is report for this project. The report is going to explain you what stages the homework was going through.

Purpose of Project

The purpose of this assignment is to create a merge sort algorithm using the map data structure. When we do and understand the whole of this assignment, we will actually understand the issues of map and merge sort.

2. Problem Solution Approach

LinkedHashMap <String, info> is storing an object of type info together with a key of type string, and then in class MergeSort I will sort an object of type MyMap according to size of ArrayList.

```
private LinkedHashMap <String, info> map;
private int mapSize;
private String str;
```

mapSize is a variable that holds the size of the map variable. The str variable is a String object formed by converting uppercase letters to lowercase letters and deleting other characters due to what is requested in the pdf.

I converted the string entered in the Main class according to the criteria in the pdf and kept it in the str object, which is the String object in the MyMap object, and I used this object in the printing to screen part.

The info class was required for the map variable inside the MyMap class. We can think of the info class as a specially defined string array. The only variable I keep inside anyway is an ArrayList of type String.

```
private ArrayList<String> words;
```

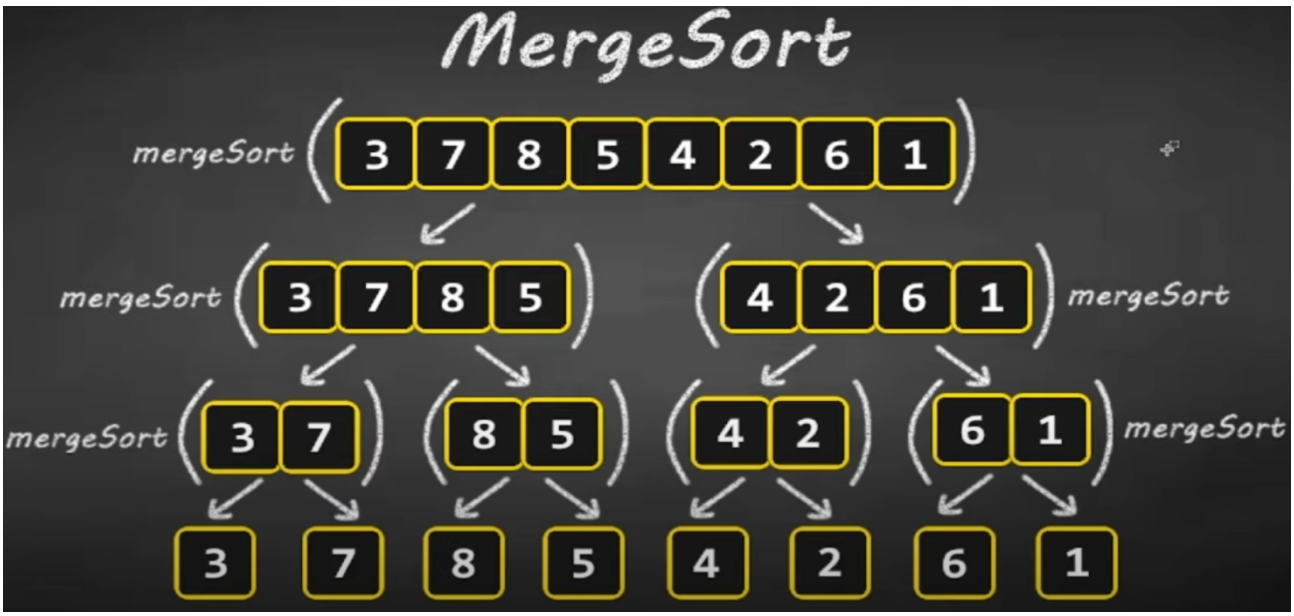
I created a method named getSize in the info class and this method returns the size of the ArrayList type variable. The constructor of this class takes as a parameter a string object and a char variable, which will be the key of the map object. As a result, it adds strings to the object of type ArrayList as desired in the pdf.

In the MergeSort class, I kept 3 private variables.

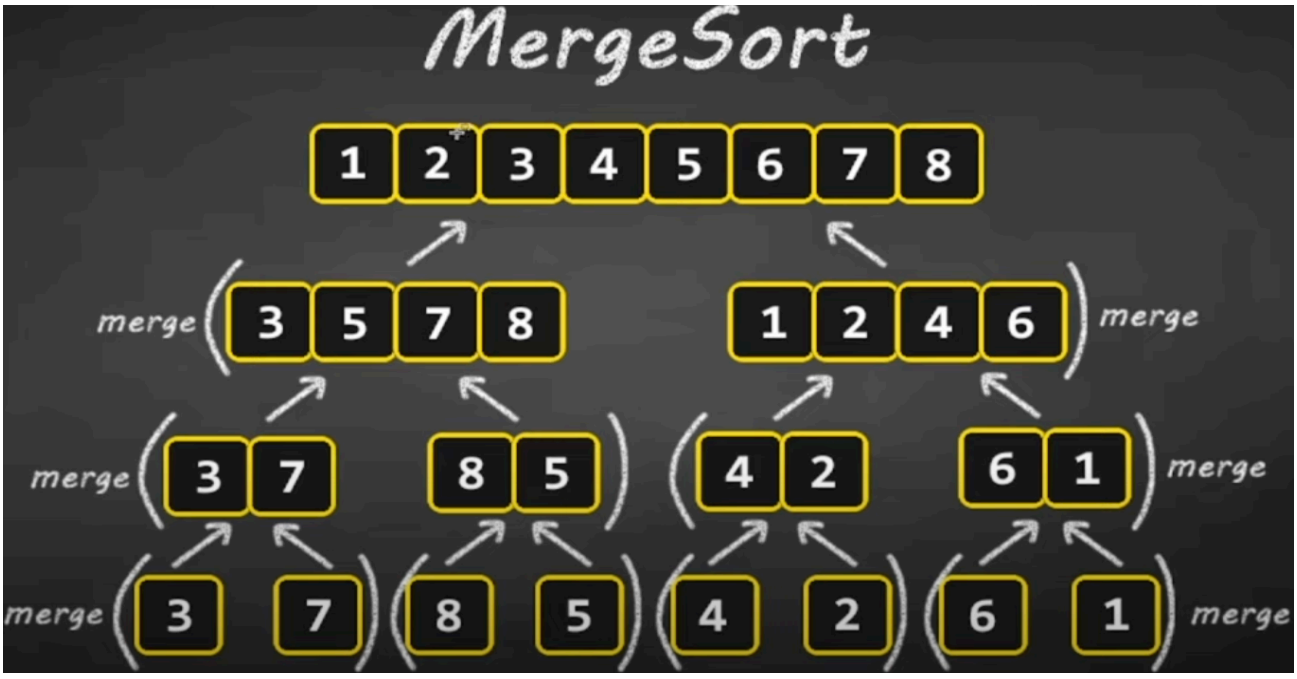
```
private MyMap originalMap;
private MyMap sortedMap;
private String[] aux; //it
```

The variable named originalMap is an object of MyMap class and it is our aim to sort the elements of this object. sortedMap is an object of MyMap class and is the variable that we will keep the map sorted by. aux is a String array. We will run the merge sort algorithm over this String array and then use this aux array to store the sorted map in the sortedMap variable.

I will try to explain the merge sort algorithm with two images because the purpose of this class is to sort using the merge sort algorithm.



Merge sort algorithm uses divide and conquer logic. In general, it divides the array we have into two parts and separates it until it turns the array into arrays with a single element. Because array with single element also means sorted array. in other words, all arrays with a single element are sorted arrays.



After each element of the array creates a single array, the merge operation is applied using the same order as the separation operation, and the arrays with a single element are sorted and created new arrays that are 2 times longer, regaining their old size. Thus, we get a sorted array.

I created a method named sort and I did the sorting process, filling the sortedMap, and also printing the aux variable on the screen. We will also examine the methods by which I do these operations below.

```
private void mergeSort(int left, int right)
{
    if (left < right)
    {
        int mid = left + (right - left) / 2;
        mergeSort(left, mid);
        mergeSort(mid + 1, right);
        merge(left, mid, right);
    }
}
```

The mergeSort method works recursively as long as the right parameter is greater than the left parameter and splits the entire array into single-element arrays. First the left part of the array is split until it becomes an array has a single element. Then, the right part of these arrays starts to divide and when these division processes are completed at one point, the merge method comes into play and sorts the arrays and combines them to create new arrays.

```
private void merge(int low, int mid, int high)
```

The merge method normally completes the sorting process by creating two new arrays inside and sorting those arrays, but I didn't take any risks since it wasn't said to be allowed in the pdf, so by keeping the size of our aux variable to be twice the normal size, I created a new array in a secret sense and indexes this array. I divided it into two part with the help of the indexes and performed the merge operation like this.

```
while (i < number1 && j < number2)
{
    if (originalMap.getMap().get(aux[i]).getSize() <= originalMap.getMap().get(aux[mid - low + 1 + j]).getSize())
    {
        aux[originalMap.getMapSize() + k] = aux[i];
        i++;
    }
    else
    {
        aux[originalMap.getMapSize() + k] = aux[number1 + j];
        j++;
    }
    k++;
}
while (i < number1)
{
    aux[originalMap.getMapSize() + k] = aux[i];
    i++;
    k++;
}
while (j < number2)
{
    aux[originalMap.getMapSize() + k] = aux[number1 + j];
    j++;
    k++;
}
```

The most important part of the merge method is undoubtedly the part where two arrays are sorted and reduced to a single array. When sorting two already sorted arrays, it starts from the first index of both and transfers to the new array until it reaches the end of any of them. Whether the transfer will be made from the left array or the right array to the new array, it is checked whether the value of the element is smaller and the smaller element is transferred. If there is an unfinished sequence when the end of any sequence is reached, it is transferred to the new sequence completely from where it left off and a sequence that both sorts and combines the two sequences is obtained.