CMPT 435 - Spring 2023 - Prof. Gormanly

Lab Two

Reece Schenck
Reece.Schenck@Marist.edu

March 10, 2023

1 Program

1.1 KnuthShuffle

I used ChatGPT to help me trouble shoot some errors I had using the random import in the shuffle method. The error turned out to only be a minor issue related to working with string arrays that was easily fixed. This class takes in a string list and shuffles it in the Knuth Shuffle style. Once the array has been shuffled it can be returned with the getArray() method.

See Lines 8-26 of the KnuthShuffle class.(Find under Appendix/Code/KnuthShuffle).

1.2 InsertionSort

I used the psudeo code from class as a skeleton version of my code. I then converted it to java and added the necessary code and requirements outlined by the assignment(adding counters, timers, etc.) and my created methods.

I start with two pointers, i and j. i starts as 1. While traversing the list, j=i+. The strings in the array indexes of j and j-1 are compared while j is greater than 0 and array[j-1] is greater than array[j]. The strings are swapped and the pointer j is decremented by 1 until at least one of the two previous conditions is false, in which i increments up 1. This continues until the entire array is sorted. Once sorted, the time is calculated and the number of comparisons has been recorded. Both are printed out.

I used ChatGPT To help me trouble shoot errors I was having starting and ending the timer.

I used '.compareToIgnoreCase' from:
https://www.w3schools.com/java/ref$_{s}tring_{c}omparetoignorecase.asp$
to compare two strings alphabetically.

See Lines 12-47 of the InsertionSort class.(Find under Appendix/Code/InsertionSort)

1.3 SelectionSort

I used the psudeo code from class as a skeleton version of my code. I then converted it to java and added the necessary code and requirements outlined by the assignment(adding counters, timers, etc.) and my created methods.

I start with three pointers, i, j, and jMin. i starts as 0. While traversing the list, jmin=i and j=i+1. The strings in the array indexes of j and jMin are compared. The pointer j is incremented until array[j] is less than array[jMin], in which the strings are swapped and i increments up 1. This continues until the entire array is sorted. Once sorted, the time is calculated and the number of comparisons has been recorded. Both are printed out.

I used ChatGPT To help me trouble shoot errors I was having starting and ending the timer.

I used '.compareToIgnoreCase' from:
https://www.w3schools.com/java/ref$_string_comparetoignorecase.asp$
to compare two strings alphabetically.

See Lines 12-55 of the SelectionSort class.(Find under Appendix/Code/SelectionSort)

1.4 MergeSort

I used the pseudo code from class as a skeleton version of my code. I then converted it to java and added the necessary code and requirements outlined by the assignment(adding counters, timers, etc.) and my created methods.

I started by getting the length and midpoint of the array. I then made two arrays, left and right that hold the strings from indexes 0-mid-1 and mid-length respectively. I then recursively call the sort method again on the left array this continues until the arrays are size of one, then it goes through all the right arrays and does the same. After all arrays are size 1, they are all merged through the merge method. In the merge method, I first create the ints lenL and lenR which are the lengths of their respective arrays, as well as pointers i, j, and k which all = 0. while i and j are less than the lengths of the arrays, right[i] and left[i] are compared, and array[k] is set to the correct string. After this is done, there is a check to make sure no values were missed, and any that have been are added to the sorted array.

I used ChatGPT To help me create the left and right arrays, specifically through the Arrays.copyOfRange command.

I used '.compareToIgnoreCase' from:
https://www.w3schools.com/java/ref$_string_comparetoignorecase.asp$
to compare two strings alphabetically.

See Lines 13-76 of the MergeSort class.(Find under Appendix/Code/MergeSort)

1.5 QuickSort

I used the pseudo code from class as a skeleton version of my code. I then converted it to java and added the necessary code and requirements outlined by the assignment(adding counters, timers, etc.) and my created methods.

I started by getting the pivot, which is the string at the high index of the parameter array. I then create two pointers leftP and rightP which = the low index and high index respectively. while the left pointer is less than the right pointer, array[leftP] is compared to the pivot until a string out of order is found, then the same is done for rightP. after both loops end, the strings are swapped. Once leftP>=rightP, the indexes of leftP and the high index are swapped. sort is then recursively called with the parameters (array, lowI, lefP-1) until sorted, then it is recursively called again with the parameters (array, leftP+1, highI) until sorted.

I used '.compareToIgnoreCase' from:
https://www.w3schools.com/java/ref$_string_comparetoignorecase.asp$
to compare two strings alphabetically.

See Lines 13-56 of the QuickSort class.(Find under Appendix/Code/QuickSort)

1.6 Main

1.6.1 Imports

These are the imports I needed for my code in Main:
import java.util.ArrayList;
import java.util.Arrays;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Random;

These Are used for reading/writing from the text file into and Array, getting a random index used to shuffle a list, and basic array / array list manipulation.

See Lines 1-6 of the Main class.(Find under Appendix/Code/Main)

1.6.2 Read/Write Text File

I used the same code from lab 1 for this lab, the only change I mad was removing the hard coded file name

To read/Write the text file into an array I first got the filename of the text file and made an array list. I then use the file reader to to read the file line by line adding them to the array list. I used ChatGPT to help me with this part.

See Lines 9-37 of the Main class.(Find under Appendix/Code/Main)

1.6.3 Sorting and Shuffling

After writing the text file into the array, I then instantiate my shuffler using that array as the parameter. I then shuffle the array and store the shuffled array in shuffledArray. I then use this shuffled array and run it through the first sorting algorithm that I implemented. Once that returns the number of comparisons and time of computation, I call shuffler again to reshuffle the array to be used for the next sorting algorithm. I repeat this process until I have used all the sorting algorithms and have all the number of comparisons and times of comparisons. For merge and quick sort I had to add the time counter outside of the class and methods otherwise I would get thousands of numbers printed due to the class's recursive natures. I also had to use their specific getComparisons() methods for this same reason.

See Lines 39-105 of the Main class.(Find under Appendix/Code/Main)

2 Unresolved Issues and Errors

After spending hours repeatedly combing over my code, I ended up with a few errors that I could not fix. The central error being that that my quick sort and merge sort algorithms do not seem to properly sort the arrays. They are always very close but never 100 percent accurate. Another issue I came across and could not figure out was that my merge sort was far more efficient than my quick sort. This could be a result of implementation and the previously mentioned errors, yet I could not fix this issue. The final issue I had was a more minor one, and that was that I could not print the nanosecond symbol(), latex won't let me use it either and that's why there is a empty set of parentheses. It would always appear as a question mark, and all the solutions I looked up resulted in the same issue.

3 Results

results may vary

Algorithm	Number of Comparisons	Time in Nanoseconds
Insertion Sort	115987	7392700
Selection Sort	221445	19951100
Merge Sort	682	3234700
Quick Sort	3464	11292700

4 Appendix

4.1 Code

4.1.1 KnuthShuffle

```
import java.util.Random;
import java.util.Arrays;

public class KnuthShuffle {
    String[] array;
    Random index;

    public KnuthShuffle(String[] array){
        this.array = array;
        index = new Random();
    }

    //used ChatGPT to help me trouble shoot some errors with this section of code
    public void shuffle(){
        for(int i=array.length-1;i>0;i--){
            int j = index.nextInt(i+1);
            String temp = array[i];
            array[i] = array[j];
            array[j] = temp;
        }
    }

    //used for testing if array was actually shuffled
    public String[] getArray(){
        return array;
    }
}
```

4.1.2 InsertionSort

```java
import java.util.Arrays;

public class InsertionSort {
    String[] array;

    public InsertionSort(String[] array){
        this.array = array;
    }

    //used class psudo code as a template, then modified into java with my unique classes and methods
    //used ChatGPT to help fix errors with timer
    public void sort(){
        long startTime = System.nanoTime();
        int comparisons = 0;
        int i = 1;
        int j;

        while(i < array.length){
            j = i;
            /*
            reference code(what the comparison looks like in psudeo code):
            while(j>0 && array[j-1]>array[j]){

            .compareTo gives a positive or negative int depending on whether
            array[j] is less than or greater than array[jMin] respectivly
            gives 0 if they are equal

            found this solution of comparing strings through the link below:
            https://www.w3schools.com/java/ref_string_comparetoignorecase.asp
            */
            while(j>0 && array[j-1].compareToIgnoreCase(array[j]) > 0){
                comparisons++;
                String temp = array[j];
                array[j] = array[j-1];
                array[j-1] = temp;
                j = j - 1;
            }
            i = i + 1;
        }
        //prints the time(in nanoseconds aka  s ) & number of comparisons
        long endTime = System.nanoTime();
        long elapsedTime = endTime - startTime;
        System.out.println("Insertion Sort:");
        System.out.println("Number of comparisons: " + comparisons);
        //The symbol '  ' woldn't show up so I removed it
        System.out.println("Sorting time in nanoseconds: " + elapsedTime);
    }

    //used for testing if array was actually sorted
    public String[] getArray(){
        return array;
    }
}
```

4.1.3 SelectionSort

```java
import java.util.Arrays;

public class SelectionSort {
    String[] array;

    public SelectionSort(String[] array){
        this.array = array;
    }

    //used class psudo code as a template, then modified into java with my unique classes and methods
    //used ChatGPT to help fix errors with timer
    public void sort(){
        long startTime = System.nanoTime();
        int comparisons = 0;
        int i = 0;
        int jMin;
        int j;

        while(i<array.length){
            jMin = i;
            j = i + 1;
            while(j<array.length){
                /*
                reference code(what the comparison looks like in psudeo code):
                if(array[j] < array[jMin]){


                .compareTo gives a positive or negative int depending on whether
                array[j] is less than or greater than array[jMin] respectivly
                gives 0 if they are equal

                found this solution of comparing strings through the link below:
                https://www.w3schools.com/java/ref_string_comparetoignorecase.asp
                */
                comparisons++;
                if(array[j].compareToIgnoreCase(array[jMin]) < 0){
                    jMin = j;
                }
                j = j + 1;
            }
            if(jMin!=i){
                String temp = array[i];
                array[i] = array[jMin];
                array[jMin] = temp;
            }
            i = i + 1;
        }
        //prints the time(in nanoseconds aka  s ) & number of comparisons
        long endTime = System.nanoTime();
        long elapsedTime = endTime - startTime;
        System.out.println("Selection Sort:");
        System.out.println("Number of comparisons: " + comparisons);
        //The symbol '  ' woldn't show up so I removed it
        System.out.println("Sorting time in nanoseconds: " + elapsedTime);
    }

    //used for testing if array was actually sorted
    public String[] getArray(){
        return array;
    }
}
```

4.1.4 MergeSort

```java
import java.util.Arrays;

public class MergeSort {
    String[] array;
    int comparisons;

    public MergeSort(String[] array){
        this.array = array;
        comparisons = 0;
    }

    //used class psudo code as a template, then modified into java with my unique classes and methods
    //used ChatGPT to help fix errors with timer
    public void sort(String[] array){
        comparisons = 0;
        int len = array.length;
        if(len<2){
            return;
        }
        int mid = len / 2;
        /*
        reference code(what the comparison looks like in psudeo code):
        String[] left = array[0, mid-1];
        String[] right = array[mid, len];

        I used ChatGPT to help me create the left and right arrays
        specifically through the Arrays.copyOfRange command
        */
        //.copyOfRange(array, inclusive, exclusive)
        String[] left = Arrays.copyOfRange(array, 0, mid);
        String[] right = Arrays.copyOfRange(array, mid, len);

        sort(left);
        sort(right);

        merge(array, left, right);
    }

    public void merge(String[] array, String[] left, String[] right){
        int lenL = left.length;
        int lenR = right.length;
        int i = 0;
        int j = 0;
        int k = 0;
        while(i<lenL && j<lenR){
            /*
            reference code(what the comparison looks like in psudeo code):
            if(left[i] <= right[i]){

            .compareTo gives a positive or negative int depending on whether
            array[j] is less than or greater than array[jMin] respectivly
            gives 0 if they are equal

            found this solution of comparing strings through the link below:
            https://www.w3schools.com/java/ref_string_comparetoignorecase.asp
            */
            comparisons++;
            if(left[i].compareToIgnoreCase(right[i]) <= 0){
                array[k] = left[i];
                i++;
            }else{
                array[k] = right[j];
                j++;
            }
            k++;
        }
        while(i<lenL){
            array[k] = left[i];
            i++;
            k++;
        }
        while(j<lenR){
            array[k] = right[j];
            j++;
            k++;
        }
    }

    //used for testing if array was actually sorted
    public String[] getArray(){
        return array;
    }

    //used for getting the number of comparisons
    public int getComparisons(){
        return comparisons;
    }
}
```

4.1.5 QuickSort

```
import java.util.Arrays;

public class QuickSort {
    String[] array;
    int comparisons;

    public QuickSort(String[] array){
        this.array = array;
        comparisons = 0;
    }

    //used class psudo code as a template, then modified into java with my unique classes and methods
    //used ChatGPT to help fix errors with timer
    public void sort(String[] array, int lowI, int highI){
        long startTime = System.nanoTime();
        String temp;
        int tempIndex;
        if(lowI>=highI){
            return;
        }
        String pivot = array[highI];
        int leftP = lowI;
        int rightP = highI;
        while(leftP<rightP){
            /*
            reference codes(what the comparison looks like in psudeo code):
            while(array[leftP]<=pivot && leftP<rightP){
            while(array[rightP]>=pivot && leftP<rightP){

            .compareTo gives a positive or negative int depending on whether
            array[j] is less than or greater than array[jMin] respectivly
            gives 0 if they are equal

            found this solution of comparing strings through the link below:
            https://www.w3schools.com/java/ref_string_comparetoignorecase.asp
            */
            comparisons++;
            while(array[leftP].compareToIgnoreCase(pivot)<=0 && leftP<rightP){
                //comparisons++;
                leftP = leftP + 1;
            }
            comparisons++;
            while(array[rightP].compareToIgnoreCase(pivot)>=0 && leftP<rightP){
                //comparisons++;
                rightP = rightP - 1;
            }
            temp = array[leftP];
            array[leftP] = array[rightP];
            array[rightP] = temp;
        }
        tempIndex = leftP;
        leftP = highI;
        highI = tempIndex;

        sort(array, lowI, leftP-1);
        sort(array, leftP+1, highI);
    }

    //used for testing if array was actually sorted
    public String[] getArray(){
        return array;
    }

    //used for getting the number of comparisons
    public int getComparisons(){
        return comparisons;
    }
}
```

4.1.6 Main

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Random;
import java.util.Arrays;

public class Main {
    public static void main(String[] args) {
        //Read and Write text file into an array

        //This wasn't working for me:
        String filename = "magicitems.txt";

        //It only worked when hard coded, so this is what I used for testing:
        //String filename = "C:\\Users\\goldh\\OneDrive\\Documents\\GitHub\\RSchenck-435\\Lab 2\\magici

        ArrayList<String> lines = new ArrayList<String>();

        try {
            BufferedReader reader = new BufferedReader(new FileReader(filename));
            String line = reader.readLine();
            while (line != null) {
                lines.add(line);
                line = reader.readLine();
            }
            reader.close();
        } catch (IOException e) {
            e.printStackTrace();
        }

        String[] linesArray = lines.toArray(new String[lines.size()]);
```

```java
        //prints out the read lines, used for testing
        //System.out.println("Lines read from file:");
        //for (String l : linesArray) {
        //    System.out.println(l);
        //}

        //initializes shuffler and shuffles
        KnuthShuffle shuffler = new KnuthShuffle(linesArray);
        shuffler.shuffle();
        String[] shuffledArray = shuffler.getArray();
        //System.out.println(Arrays.toString(shuffledArray));

        //selection sort
        SelectionSort select = new SelectionSort(shuffledArray);
        select.sort();
        String[] sorted1 = select.getArray();
        //System.out.println(Arrays.toString(sorted1));

        System.out.println(" ");

        //shuffle again
        shuffler.shuffle();
        shuffledArray = shuffler.getArray();
        //System.out.println(Arrays.toString(shuffledArray));

        //insertion sort
        InsertionSort insert = new InsertionSort(shuffledArray);
        insert.sort();
        String[] sorted2 = insert.getArray();
        //System.out.println(Arrays.toString(sorted2));

        System.out.println(" ");

        //shuffle again
        shuffler.shuffle();
        shuffledArray = shuffler.getArray();
        //System.out.println(Arrays.toString(shuffledArray));

        //merge sort
        //used ChatGPT to help fix errors with timer
        long startTime = System.nanoTime();
        MergeSort merge = new MergeSort(shuffledArray);
        merge.sort(shuffledArray);
        String[] sorted3 = merge.getArray();
        //System.out.println(Arrays.toString(sorted3));
        long endTime = System.nanoTime();
        long elapsedTime = endTime - startTime;
        System.out.println("Merge Sort:");
        System.out.println("Number of comparisons: " + merge.getComparisons());
        //The symbol '  ' woldn't show up so I removed it
        System.out.println("Sorting time in nanoseconds: " + elapsedTime);

        System.out.println(" ");

        //shuffle again
        shuffler.shuffle();
        shuffledArray = shuffler.getArray();
        //System.out.println(Arrays.toString(shuffledArray));

        //quick sort
        //used ChatGPT to help fix errors with timer
        long startTime2 = System.nanoTime();
        QuickSort quick = new QuickSort(shuffledArray);
        quick.sort(shuffledArray, 0, shuffledArray.length-1);
        String[] sorted4 = quick.getArray();
        //System.out.println(Arrays.toString(sorted4));
        long endTime2 = System.nanoTime();
        long elapsedTime2 = endTime2 - startTime2;
        System.out.println("Quick Sort:");
        System.out.println("Number of comparisons: " + quick.getComparisons());
        //The symbol '  ' woldn't show up so I removed it
        System.out.println("Sorting time in nanoseconds: " + elapsedTime2);
    }
}
```

4.1.7 Text File (magicitems)

I'm not putting all of the words here because its 666 lines of text/code