## **Lab 5 Report**

## **CS303L-L3 Algorithms and Data Structures**

## **Sam Lazrak**

## **Spring Semester 2018**

**Objectives:**

* Learn to come up with novel sorting ideas
* Learn to choose the best sorting algorithm for a given problem

**In-class Assignment:**

1. Consider the following approach:
2. An algorithm sorts an array of n elements by finding the smallest and largest elements and then exchanges those elements with the elements in the first and last positions in the array.
3. Then the size of the array is reduced by two elements after excluding the two elements that are already in the proper positions, and the process is repeated on the remaining part of the array until the entire array is sorted.

2. Write a code to implement the above algorithm.

3.Write a driver program to show the novel sorting algorithm works correctly.

4. Have your work reviewed by the TA or the Instructor before departing the lab. If you do not complete this during the lab period you will have to visit the TA during Office hours

to demonstrate your work.

**Source code for In-class Assignment:**

package lab6;

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Arrays;

import java.util.Scanner;

public class lab6 {

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

File file = new File("input\_100.txt");

Scanner s = new Scanner(file);

int[] array = new int[100];

int i = 0;

while(s.hasNextInt()) {

array[i] = s.nextInt();

i++;

}

novelSort(array, 0, array.length - 1);

System.out.println(Arrays.toString(array));

}

public static void novelSort(int[] a, int min, int max) {

if(min >= max) return;

for(int i = min; i <= max; i++){

if(a[i] < a[min]) {

swap(a, i, min);

}

if(a[i] > a[max]){

swap(a, i, max);

}

}

novelSort(a, min + 1, max - 1);

}

public static void swap(int[] a, int x, int y){

int temp = a[x];

a[x] = a[y];

a[y] = temp;

}

}

**Output for In-class Assignment:**

[0, 0, 0, 1, 1, 3, 4, 5, 7, 8, 10, 12, 12, 13, 14, 15, 15, 16, 18, 22, 22, 22, 23, 24, 24, 25, 25, 27, 29, 29, 30, 34, 36, 36, 37, 38, 38, 39, 40, 40, 41, 43, 45, 47, 47, 49, 50, 50, 52, 52, 53, 54, 54, 54, 54, 57, 60, 61, 63, 64, 64, 67, 67, 68, 69, 69, 70, 72, 73, 73, 74, 74, 75, 75, 75, 76, 77, 77, 78, 80, 81, 81, 81, 83, 83, 84, 85, 85, 86, 87, 88, 89, 92, 93, 94, 94, 94, 95, 96, 98]

**Homework Assignment:**

1. Write up the algorithm from the in-class portion in pseudocode and give a proof outline that shows this algorithm correctly sorts the array. (Your “proof” does not have to be very formal.)

**NOVELSORT**(a, min, max)

**if** min >= max

**return** // if min >= max exit than exit novel sort

**for** min to max // loops from the minimum to the maximum

**if** a[i] < a[min] // if a[i] <= a[min]

**exchange** a[i] with a[min] // exchange a[i] and a[min]

**if** a[i] > a[max] // if a[i] >= a[max]

**exchange** a[i] with a[max] // exchange a[i] and a[min]

**NOVELSORT**(a, min + 1, max - 1) // calls novel sort recursively and reduces the size of the array getting rid of min and max as they have been initialized as the smallest and largest of the array.

This algorithm works because we are reducing the size of the array through each recursive call while continuously sorting as we compare each element and add the smallest and largest to the beginning and the end of the array.

1. Consider an application that logs transactions.
   1. The log includes the location where a transaction originated and the time the transaction occurred and this information is stored such that they are ordered by the time of the transaction (see sample input below).
   2. You are required to sort this log by location while preserving the order of the timefield (see sample output below). CS303 Lab Assignment 1 of 2
   3. Implement an algorithm of your choice to sort this array based on the location while preserving the order of the time field.
   4. Test your algorithm with the input file provided in Canvas named“NovelSortInput.txt”.
   5. Explain in your report why the sorting algorithm you chose is the best for the job.

**Source code for Homework Assignment:**

package lab6;

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Arrays;

import java.util.Scanner;

public class lab6hw {

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

File file = new File("NovelSortInput.txt");

Scanner s = new Scanner(file);

Object[] a = new Object[17];

Object[] t = new Object[17];

System.out.println("Unsorted Array: ");

int i = 0;

while(s.hasNextLine()) {

String l = s.nextLine();

a[i] = l.substring(0, 7);

t[i] = l.substring(9,16);

System.out.format("%s %s\n", a[i], t[i]);

i++;

}

novelSort(a, 0 , a.length -1);

System.out.println("\n" + "Sorted Array: ");

int j = 0;

while(j <= 16) {

System.out.format("%s %s\n", a[j], t[j]);

j++;

}

}

public static void novelSort(Object[] a, int min, int max) {

if(min >= max) return;

for(int i = min; i <= max; i++){

char e = a[i].toString().charAt(0);

char s = a[min].toString().charAt(0);

char m = a[max].toString().charAt(0);

if(e < s) {

swap(a, i, min);

}

if(e > m){

swap(a, i, max);

}

}

novelSort(a, min + 1, max - 1);

}

public static void swap(Object[] a, int x, int y){

Object temp = a[x];

a[x] = a[y];

a[y] = temp;

}

}

**Output for Homework Assignment:**

Unsorted Array:

Chicago 9:00:00

Phoenix 9:00:03

Houston 9:00:13

Chicago 9:00:59

Houston 9:01:10

Chicago 9:03:13

Seattle 9:10:11

Seattle 9:10:25

Phoenix 9:14:25

Chicago 9:19:32

Chicago 9:19:46

Chicago 9:21:05

Seattle 9:22:43

Seattle 9:22:54

Chicago 9:25:52

Chicago 9:36:14

Phoenix 9:37:44

Sorted Array:

Chicago 9:00:00

Chicago 9:00:03

Chicago 9:00:13

Chicago 9:00:59

Chicago 9:01:10

Chicago 9:03:13

Chicago 9:10:11

Chicago 9:10:25

Houston 9:14:25

Houston 9:19:32

Phoenix 9:19:46

Phoenix 9:21:05

Phoenix 9:22:43

Seattle 9:22:54

Seattle 9:25:52

Seattle 9:36:14

Seattle 9:37:44

**Analysis:**

To sort this by location while keeping the order intact I chose to use the novel sort. My approach was to sort by the first character of each element. I separated the data into two arrays; one for location and the other for times. I only sorted the left array with novel sort as the right was already sorted. I think my approach is good because it is efficient.