Novel Sort Using Java

Homework #6

By:

Pratyusha Thundena

CS 303L – L1A

October 9, 2018

### Problem Specification

The objectives included learning to come up with a novel sorting idea and then implementing the best sorting algorithm for the given problem. To meet the objectives, the assignment involved implementing a method that would sort an input file using a modified version of selection sort, and then creating a driver program to test that algorithm.

### Program Design

This program required the following:” NovelSortInput.txt”, Driver program(NovelDriver.java), test program(SortArray.java), BufferedReader class, FileReader class, IO Exception class, ArrayList class, Arrays class, and List class.

The following steps were required to develop these 2 programs:

1. create a modified version of selection sort algorithm
2. test the sorting algorithm with an unsorted array in SortArray.java
3. iterate over the array
4. find the minimum element and maximum element index in unsorted array
5. swap the minimum element with the first element and the maximum with the last element
6. print the unsorted and sorted array in SortArray.java
7. read the file “NovelSortInput.txt” in driver program in NovelDriver.java
8. create the list to use with novel sort
9. print out the sorted list in NovelDriver.java

The following constructors and methods were defined within the class:

1. sortarray ()

Basic method that finds the minimum and maximum element in unsorted array in

SortArray.java.

b) printer ()

Basic method that prints the array in SortArray.java.

c) s ()

Constructor created to help print unsorted and sorted array in SortArray.java.

d) main ()

Driver method that tests the sorting algorithm in SortArray.java.

d) in ()

Constructor created to help with file reading in NovelDriver.java.

e) a ()

Constructor used with BufferedReader in NovelDriver.java.

f) List<String>list ()

Constructor used in creating a list in NovelDriver.java.

g) main ()

Driver method that tests and prints out the sorted version of NovelSortInput.txt in NovelDriver.java.

The println method of the System.out object was used to display the inputs and results for the driver program.

### Testing Plan

The test plan included a list of unsorted elements from “NovelSortInput.txt” in NovelDriver.java that were eventually sorted.

### Results

|  |  |
| --- | --- |
| Input | Output |
| Chicago 09:00:00  Phoenix 09:00:03  Houston 09:00:13  Chicago 09:00:59  Houston 09:01:10  Chicago 09:03:13  Seattle 09:10:11  Seattle 09:10:25  Phoenix 09:14:25  Chicago 09:19:32  Chicago 09:19:46  Chicago 09:21:05  Seattle 09:22:43  Seattle 09:22:54  Chicago 09:25:52  Chicago 09:36:14  Phoenix 09:37:44 | **Chicago 09:00:00**  **Chicago 09:00:59**  **Chicago 09:03:13**  **Chicago 09:19:32**  **Chicago 09:19:46**  **Chicago 09:21:05**  **Chicago 09:25:52**  **Chicago 09:36:14**  **Houston 09:00:13**  **Houston 09:01:10**  **Phoenix 09:00:03**  **Phoenix 09:14:25**  **Phoenix 09:37:44**  **Seattle 09:10:11**  **Seattle 09:10:25**  **Seattle 09:22:43**  **Seattle 09:22:54** |

### Analysis and Conclusions

In this program, “NovelSortInput.txt” contained a log that included the location of transactions and the time that the transactions had occurred. It was required that the log was sorted by location while still preserving the order of the time field. In order to complete the requirements, a modified version of the selection sort algorithm was implemented. Selection sort was chosen because the input file already had the times ordered, but it was out of order in terms of location. In the best-case scenario for selection sort, the array has already been sorted and the time complexity would be O(n2). In the worst-case scenario for selection sort, the array has been reversed, the body of the if statement has been invoked on every occasion, which would result in doubling the number of steps in the inner loop. However, the time complexity would still be O(n2). The results demonstrated that selection sort was the best algorithm since the array was sorted, such that, it was already ordered by the time of the transaction. It was just the location that needed rearranging. So, this was a best-case scenario for selection sort.

### References

The parameters and input file was provided in the homework assignment (by Dr. Bangalore) and Introduction to Algorithms (3rd ed.) was used to do the lab report.