Applying sorting algorithms
Report #07

By
Eduardo Castro

CS 303 Algorithms and Data Structures

October 15, 2014

1. Problem Specification

The goal of this assignment was to design a sorting algorithm from scratch with a group of developers and also to apply all the knowledge acquired during this semester in sorting algorithms for sorting data structures more similar to what we could find in the “real world”.

2. Program Design

The first part of this lab requires the creation of a sorting algorithm that search for the smallest and biggest elements from a list, puts its into their right positions and keep doing it until the list is ordered. We did this in group.

The following steps were required to develop this program:

a) write a pseudocode for the sorting algorithm
b) write the SortLab class translating this pseudocode to real code
c) write the SortLabTest class
d) use the driver class to display and measure the results

The following methods were defined within the SortLab and SortLabTest classes:

a) sort(int[] array)
    a. Sorts the array
a) swap(int[] array, int i, int j)
    a. swap elements from positions i and j
b) sortRecursively(int[] array, int first, int last, int size)
    a. Basically do the same previous sorting but in a recursively way
c) getBiggest(int[] array, int first, int last)
    a. Returns the biggest element from the list. The “size" of the list varies according to the progress of the sorting. At the first time we look for the biggest element into all the array, at the second time it will be only between elements first+1 and last-1, etc
d) getSmallest(int[] array, int first, int last)
    a. Returns the smallest element from the list. The “size" of the list varies according to the progress of the sorting. At the first time we look for the smallest element into all the array, at the second time it will be only between elements first+1 and last-1, etc
e) main()
    a. Driver class

The second part requires us to build a Priority Queue using a min heap algorithm. For this part I created two classes: PriorityQueue and PriorityQueueTest.

The following steps were required to develop this program:

a) write the SortAirport class
b) write the SortAirportTest class
c) use the driver class to display and measure the results

The following methods were defined within the SortAirport and SortAirportTest classes:

f) sortAirportList(String[] listOfAirports)
    a. Sort the airport list using insertion sort algorithm
g) getHowManyElementsOnTheInputsFile(String inputFileAddress)
    a. Returns how many lines there are in the inputFile
h) main()
    a. Driver class

The Scanner class provided by Java was used to read in the necessary values within the provided driver program. The println method of the System.out object was used to display the inputs and results for the provided driver program.

3. Testing Plan

For the first part of the lab, the group development part, we first created the pseudocode, evaluated it and started coding in Java. We tested for an array of 10 elements and worked.

For the second part, the SortAirport part, I used the insertion sort algorithm just changing the way to compare elements as integers for strings using the CompareTo method. The output must be the same as oriented by the lab instructions, the elements needs to be sorted only by location.

4. Test Cases

The test cases from the first part of the Lab, the SortLab part:

```
Original array: {5, 2, 6, 9, 10}
Sorted array: {2, 5, 6, 9, 10}
```

The test cases from the second part of the Lab, the SortAirport part:

ATL	10:35:59	DCA	18:51:03	EGLL	16:30:57	LEBL	14:55:43	MSP	19:09:09	SAN	12:47:53	YVR	13:55:29
ATL	10:47:16	DCA	19:05:52	EGLL	18:37:22	LEBL	16:15:38	MSP	19:30:35	SAN	13:39:16	YYC	13:50:18
ATL	10:49:07	DCA	19:14:38	EGLL	20:29:14	LEBL	18:36:36	MSY	14:55:53	SAN	15:11:14	YYZ	14:45:42
ATL	10:50:53	DEN	15:00:58	EGLL	21:00:52	LEMD	13:28:12	MSY	17:06:51	SAN	16:47:32	YYZ	18:03:55
ATL	11:05:24	DEN	18:37:37	EGLL	9:59:34	LEMD	13:52:42	MSY	18:19:33	SAV	21:45:12	YYZ	22:45:27
ATL	11:05:39	DEN	19:05:21	EH	11:05:28	LEMD	13:55:43	MSY	18:35:27	SBGL	13:40:17		
ATL	11:10:31	DFW	13:10:12	EH	11:35:20	LEMD	14:55:34	MSY	19:50:42	SBGR	13:30:41		
ATL	11:33:35	DFW	13:10:54	EH	17:39:03	LEMD	16:20:19	MSY	21:40:16	SDF	15:11:01		
ATL	11:52:15	DFW	13:36:18	EH	17:39:37	LEMD	16:30:37	MTPP	13:43:51	SDF	22:00:50		
ATL	12:19:07	DFW	13:44:23	EH	17:44:03	LEMD	16:44:01	MTPP	14:19:23	SEA	13:33:45		
ATL	14:40:20	DFW	15:40:45	EIDW	15:05:52	LEMD	18:25:00	ORD	13:14:31	SEA	14:15:57		
ATL	15:20:26	DFW	17:00:28	EINN	13:52:32	LEMD	19:14:12	ORD	13:40:35	SEA	15:10:52		
ATL	17:50:56	DFW	3:52:34	EINN	13:53:20	LEMD	19:55:55	ORD	16:58:20	SEA	15:40:49		
ATL	18:00:06	DFW	6:01:34	EINN	15:50:47	LFPG	11:20:21	ORD	18:19:39	SEA	16:30:53		
ATL	18:26:00	DFW	7:00:33	EINN	16:00:02	LFPG	12:53:41	ORD	18:29:13	SEA	17:49:24		
ATL	19:05:42	DFW	7:15:01	FLL	14:55:26	LFPG	13:15:22	ORD	23:50:14	SEA	19:15:26		
ATL	19:14:29	DFW	7:16:00	FLL	15:00:36	LFPG	13:25:51	ORD	4:19:14	SEA	19:55:25		
ATL	22:45:09	DFW	7:40:06	FLL	17:03:44	LFPG	13:55:35	ORD	6:05:39	SEA	23:10:13		
AUS	13:43:11	DFW	8:18:23	FLL	17:03:54	LFPG	14:02:16	ORD	6:15:10	SEA	23:45:37		
AUS	14:19:54	DFW	8:35:20	FLL	17:24:07	LFPG	14:08:38	ORD	6:29:35	SEA	9:49:29		
AUS	16:57:37	DFW	8:43:12	FLL	18:35:32	LFPG	14:12:23	ORD	6:57:11	SFO	14:19:10		
AUS	17:03:52	DTW	10:30:24	FLL	18:37:08	LFPG	14:15:05	ORD	7:21:11	SFO	14:40:36		
AUS	17:22:42	DTW	10:43:28	FLL	19:49:11	LFPG	15:24:31	ORD	7:30:50	SFO	15:00:58		
AUS	17:49:36	DTW	12:17:21	FLL	22:02:12	LFPG	16:20:55	ORD	7:34:04	SFO	15:30:58		
AUS	18:19:39	DTW	12:19:16	IAD	18:45:00	LFPG	16:47:56	ORD	7:59:11	SFO	17:50:36		
AUS	23:59:25	DTW	12:25:39	IAD	19:05:57	LFPO	16:09:28	ORD	8:18:52	SFO	18:50:35		
BOS	13:36:51	DTW	14:48:23	IND	15:20:32	LGAV	14:48:57	ORD	8:38:12	SFO	19:19:19		
BOS	13:39:36	DTW	18:10:26	IND	19:31:13	LGAV	18:15:46	ORF	14:29:38	SFO	20:10:39		
BOS	13:40:02	DTW	20:25:23	JAX	17:50:21	LIMC	13:28:01	ORF	17:52:20	SFO	22:05:01		
BOS	15:00:57	EDDF	17:02:40	JAX	18:50:33	LIMC	16:20:11	ORF	19:18:43	SJU	13:36:59		
BOS	15:18:14	EDDL	13:55:00	JAX	19:05:34	LIRF	12:47:50	ORF	20:04:08	SJU	15:30:26		
BOS	16:15:28	EDDL	13:55:14	JAX	19:29:43	LIRF	13:34:01	OTBD	13:55:15	SJU	15:30:38		
BOS	16:25:38	EDDM	17:00:04	JAX	20:15:04	LIRF	16:15:10	PDX	15:10:39	SJU	16:44:23		
BOS	18:15:39	EDDT	13:55:54	JFK	11:15:14	LSZH	13:39:02	PHL	15:15:09	SJU	17:06:16		
BOS	19:05:01	EGCC	13:40:12	JFK	11:30:26	MCO	10:27:38	PHL	19:29:48	SJU	17:06:25		
BOS	21:00:04	EGCC	16:20:54	JFK	11:59:58	MCO	13:45:50	PHL	20:15:07	SJU	17:25:42		
BUF	15:25:58	EGLL	10:00:46	JFK	12:00:51	MCO	14:50:50	PHL	20:55:37	SJU	19:19:56		
BUF	17:04:24	EGLL	10:06:52	JFK	12:18:00	MCO	16:34:36	PHX	17:52:19	SJU	20:05:27		
BUF	18:25:37	EGLL	12:45:49	JFK	12:55:00	MCO	18:19:57	PHX	22:00:27	SJU	23:37:34		
BUF	19:14:43	EGLL	12:53:37	JFK	13:12:24	MIA	12:55:47	PIT	14:45:40	SLC	11:42:15		
BUF	20:25:50	EGLL	12:55:23	JFK	6:29:40	MIA	12:56:56	PIT	18:10:05	SLC	15:18:12		
BUF	22:00:43	EGLL	12:55:33	JFK	6:57:14	MIA	13:36:57	PIT	18:50:34	SLC	15:20:20		
BWI	15:30:32	EGLL	12:55:58	JFK	7:34:02	MIA	13:38:55	PIT	19:05:01	SLC	21:45:35		
BWI	19:20:56	EGLL	13:37:36	JFK	8:03:45	MIA	13:40:40	PIT	22:00:57	SPIM	13:55:30		
BWI	20:15:23	EGLL	13:38:22	JFK	8:13:49	MIA	13:44:47	RDU	12:42:41	SYR	15:25:21		
CHS	15:19:23	EGLL	13:43:44	LAS	13:15:15	MIA	14:50:23	RDU	12:45:08	SYR	15:38:42		
CHS	15:26:44	EGLL	13:45:37	LAS	13:34:12	MIA	15:00:14	RDU	12:50:10	SYR	17:30:25		
CLE	14:44:09	EGLL	13:45:42	LAS	14:55:55	MIA	16:20:44	RDU	14:50:01	TPA	14:40:25		
CLE	18:01:31	EGLL	13:47:50	LAS	15:11:03	MIA	16:25:06	RDU	17:15:33	TPA	17:03:41		
CLE	21:44:01	EGLL	13:50:11	LAS	15:19:00	MIA	18:19:16	RDU	18:15:49	TPA	17:25:24		
CLT	14:29:10	EGLL	13:50:15	LAS	16:32:24	MIA	18:59:59	RDU	19:15:29	TPA	17:59:11		
CLT	17:52:23	EGLL	13:53:22	LAS	16:53:50	MIA	19:30:06	RDU	19:59:53	TPA	19:18:23		
CLT	19:10:46	EGLL	13:55:18	LAS	20:25:21	MIA	21:15:44	RIC	12:38:25	TPA	20:05:23		
CMH	12:25:30	EGLL	13:55:55	LAS	20:55:36	MIA	6:29:33	RIC	12:47:32	TXKF	11:10:46		
CMH	12:29:14	EGLL	14:56:41	LAX	10:27:08	MIA	7:08:34	RIC	12:51:00	TXKF	13:30:21		
CMH	12:50:54	EGLL	15:05:17	LAX	11:05:33	MIA	7:30:36	RIC	15:05:08	VHHH	13:45:48		
CMH	15:04:02	EGLL	15:42:44	LAX	13:00:23	MIA	8:20:59	RIC	15:12:36	VHHH	13:47:47		
CMH	15:39:41	EGLL	15:42:45	LAX	13:05:57	MIA	8:55:05	RJAA	13:50:16	VHHH	13:53:50		
CMH	19:55:41	EGLL	15:45:45	LAX	13:12:30	MMMX	15:00:45	RJAA	13:50:28	YQB	14:30:09		
CMH	22:00:42	EGLL	15:47:36	LAX	13:45:56	MMMX	18:54:00	RJAA	16:20:00	YQB	17:55:36		
CVG	12:25:18	EGLL	15:50:20	LAX	15:30:58	MMMX	22:35:59	RJAA	9:17:21	YQB	19:18:24		
CVG	12:28:49	EGLL	15:50:27	LAX	19:29:51	MMUN	13:24:00	ROC	14:45:13	YQB	20:05:59		
CVG	14:30:10	EGLL	15:50:31	LAX	6:00:02	MSP	10:29:10	ROC	17:03:34	YQB	21:00:22		
CVG	15:18:16	EGLL	16:00:42	LAX	7:08:52	MSP	11:08:22	ROC	18:04:31	YUL	14:45:28		
CVG	17:59:23	EGLL	16:13:32	LAX	7:22:00	MSP	11:29:24	ROC	18:45:35	YUL	14:55:49		
CVG	21:51:18	EGLL	16:20:14	LAX	9:10:19	MSP	11:59:48	ROC	22:02:32	YUL	18:04:35		
DCA	14:19:31	EGLL	16:25:34	LAX	9:15:25	MSP	15:26:46	SAEZ	13:40:51	YUL	18:25:24		
DCA	17:50:27			LEBL	13:38:58			SAN	10:20:10	YUL	20:51:06		

5. Analysis and Conclusions

At the first part of the lab the group development was interesting for discussing the various possibilities and to help each other to improve the solution. At first we decided to do it in a iterative way and it was pretty easy. Basically we check every element of the array, put the biggest and the smallest in the right positions and keep checking until all the elements are sorted. Not a big deal, then we decided to do the same in a recursive way and the discussion got deeper with each one helping each other to solve the problem. It was great and unfortunately we didn't programmed using pair programming and coding dojo's methodologies, it would be great.

The second part was made alone by me, I decided to use the insertion sort algorithm just because it is easier and stable. The only thing that I had to change at the algorithm was that rather than checking which one was the biggest integer between two numbers I would check the strings lexicographically(following the order of the alphabet), for this I used the comparteTo() method from the String class, it was pretty easy and simple. For sorting the elements just by the location I decided to store each line of the log.txt into a position of the array so I could sort the elements and still have the hour information, at the beginning I was checking only the first three elements of the String with the compareTo() method but then I noticed that the output file showed at the instructions also was showing the locations sorted by time, then I just removed it and compared all the string and got the same output as expected.

6. References

The parameters used was from the homework assignment provided in class, I also used as reference for learning how to work with compareTo the Java documentation and another references from an Algorithms course from Princeton.