ELEC 278

Lab 6: Sorting

Weeks 11 & 12
November 2019

Contents

1. Objective

The objectives of this lab are:

1- Practice with sorting algorithms,
2- Analyze sorting algorithms.

2. Organization

This lab includes two steps.

In Part 1 (Section 3), you will modify various sorting algorithm to sort in descending order and then count the number of swaps required.

In Part 2 (Section 4), you will study and compare the complexity of 3 sorting algorithms.

3. Step 1: Modifying sorting algorithms

3.1 Tasks

1. Download and extract the code in lab6code.zip
2. Compile and run sorting_step1.c in step1 folder.

```
cc  sorting_step1.c  –o  sorting_step1
```

3. The output should be: (the actual number of swaps will be different than 0.)

```
Before   Sorting:        10 24 5 32 1 84 19
*****
After Insertion Sort:    1 5 10 19 24 32 84
Number of swaps: 0
*****
After Bubble Sort:       1 5 10 19 24 32 84
Number of swaps: 0
*****
After Quick Sort:        1 5 10 19 24 32 84
Number of swaps: 0
*****
After Heap Sort:         1 5 10 19 24 32 84
Number of swaps: 0
*****
After Merge Sort:        1 5 10 19 24 32 84
Number of swaps: 0
*****
After Radix Sort:        1 5 10 19 24 32 84
Number of swaps: 0
```

4. For each sorting algorithm, change the sorting algorithm so that it sorts the data in **descending** order.
5. Count the number of swaps performed in each sorting algorithm and return the count as an output. (Don't add swap counts to the radix sort)
6. Recompile and run **sorting_step1.c**
7. The output should be: (The number of swaps will be some non-zero value.)

```
Before   Sorting:        10 24 5 32 1 84 19
*****
After Insertion Sort:    84 32 24 19 10 5 1
Number of swaps: 0
*****
After Bubble Sort:       84 32 24 19 10 5 1
Number of swaps: 0
*****
After Quick Sort:        84 32 24 19 10 5 1
Number of swaps: 0
*****
After Heap Sort:         84 32 24 19 10 5 1
Number of swaps: 0
*****
After Merge Sort:        84 32 24 19 10 5 1
Number of swaps: 0
*****
After Radix Sort:        84 32 24 19 10 5 1
Number of swaps: 0
```

3.2 Deliverable

Show your code and the resulting output to your TA.

4. Step 2: Analyze the performance of sorting algorithms

In this task you will measure the time taken by a sorting algorithm for specific size of data.

1. Compile and run sorting_step2.c in step2
2. The main function in sorting_step2.c will do the following:
    a. Ask the user of the size of the array to use (n)
    b. Create an array of size n and fill it with random numbers
    c. Take timestamps before and after calling the sorting function
3. Between the start and end timestamps, you will call the required sorting algorithm
4. Run the code for different sizes and fill the following table

n	QuickSort	MergeSort	HeapSort
5,000,000			
10,000,000			
15,000,000			
20,000,000			
25,000,000			

5. Using Excel or any other spreadsheet application, plot the curves of the time taken by each algorithm. Which one has the best performance?
6. Repeat the experiment for random numbers less than 10 (i.e. use %10) and compare the performance of Quick sort and Radix sort

n	QuickSort	RadixSort
5,000,000		
10,000,000		
15,000,000		
20,000,000		
25,000,000		

7. Plot the curves. Which one has better performance?
8. Repeat the experiment for random numbers less than 100 (i.e. use %100) and compare the performance of Quick sort and Radix sort

n	QuickSort	RadixSort
5,000,000		
10,000,000		
15,000,000		
20,000,000		
25,000,000		

4.2 Deliverable

Show your 3 plots to the TA.

5. Stretch Goals

Modify Quicksort so that there is a second array containing indices to the data, not the actual data. Your sort should not move any of the actual data – it should just move the index to the data. That is, you might have the following data:

34
17
62
91
3
37
59
18

You create a second array as follows:

0
1
2
3
4
5
6
7

After the sort, the second array contains:

3
2
6
5
0
7
1
4

Which if you access the first array using the indexes found in the second, you will get a sorted list. Show your code and results to the TA.

ELEC 278 LAB EXERCISE 6 CHECK-OFF SHEET

Print this sheet and bring it to the lab with you. The TA will sign off the steps as you complete them. Make sure you hand in your signed sheet before you leave the lab.

ONE SHEET PER STUDENT PLEASE!

Student Name (optional)	
Student Number (8 digits)	

Part Number	Step Description	Mark	TA Sign Off
Step 1 (Section 3)	Modify sorting code.	5	
Step 2 (Section 4)	Performance evaluation of sorting algorithms	5	
Stretch goal (Section 5)	Rearrange array of indices based on data	4	
	Full marks (complete lab requirements)	10	
	Maximum possible marks	14	