Sorting

CSCE 221

Due: November 15th, 2019

1 Description

For this programming assignment, you will implement several different sorting algorithms and study their performance. You will implement

i Bubble Sort

ii Heap Sort

iii Merge Sort

iv (deterministic) Quick Sort

1.1 Input

We will test your implementations using a set of numbers in a text file named *"numbers.txt"*. Each line of the file will contain a number. The first number will be which sorting algorithm to use (1=Bubble Sort, 2=Heap Sort, 3=Merge Sort, 4=Quick Sort). The second number will be the number of remaining elements in the file, n. The first two numbers of the file (sorting algorithm and n) will NOT be sorted. However, the remaining n numbers will be sorted. A sample numbers.txt file has been provided to you. **Your main program should take an input filename and act appropriately.** You should output the sorted numbers to a file titled *output.txt*. Only output the sorted numbers and not the type of sort or the number of elements n.

1.2 Heap

You should already have a heap working from the previous assignment. However, if you do not, there is a heap implementation on eCampus.

1.3 Timing

Using your four implementations, you will also time how long it takes to sort n numbers. You should test your sorts on three different types of inputs: sorted, the reverse of sorted order, and random. You may use a random number generator as opposed to the file input above. Time how long sorting using the StopWatch.h class we provided (you don't need to print the numbers when timing). You should measure the total time at different values of n. You will then show a graph of the sort time versus the number of elements for each implementation and type of input.

Note: Stopwatch.h only works in the Windows environment. If working (or compiling) in Linux or Mac, use the time.h library.

2 Coding Portion(50 Points)

– Create the four implementations of the different sorting algorithms.

– Be sure to test the correctness of your algorithms and implementations (we will).

- Your code will be graded based on whether or not it compiles, runs, produces the expected output, produces correct output, whether or not your experimental setup is correct, and your coding style (does the code follow proper indentation/style and comments).

Create an array of numbers from the input file and pass this into your sort function. Your sort function should not return a new array; the original array should be modified.

3 Report (50 Points)

You will write a brief report that includes theoretical analysis, a description of your experiments, and discussion of your results. At a minimum, your report should include the following sections:

3.1 Introduction

In this section, you should describe the objective of this assignment.

3.2 Theoretical Analysis

In this section, you should provide an analysis of the complexity of each sorting algorithm on the different input types. What do you expect the complexity should be for each algorithm and input type?

3.3 Experimental Setup

In this section, you should provide a description of your experimental setup, which includes but is not limited to

a Machine specification

b How did you generate the test inputs? What input sizes did you test? Why? Did you use something other than an array for your data structure? If so, why?

c How many times did you repeat each experiment?

3.4 Experimental Results

In this section, you should compare the performance (running time) of the sort operation for the four different sorting algorithms to one another and to their theoretical complexity.

a Make a plot showing the running time (y-axis) vs. the number of elements (x-axis). You must use some electronic tool (matlab, gnuplot, excel, . . .) to create the plot – hand-written plots will NOT be accepted.

b Provide a discussion of your results, which includes but is not limited to:

  i Which of the sorting algorithms performs the best and why? Does it depend on the input? Why or why not?

  ii To what extent does the theoretical analysis agree with the experimental results? Attempt to understand and explain any discrepancies you note.

4 Submission

Submit the following in a zip file named in the format *LastName_FirstName_UIN_PA4.zip*:

- bubblesort.h
- heapsort.h
- heap.h

- mergesort.h
- quicksort.h
- main.cpp
- (optional) Makefile
- Your report as a PDF

Note: we should be able to compile and run your program out of the box so if there are other dependencies, include those. The .h files are listed under the assumption that it contains implementation. If, however, you seperated it into .h and .cpp, then also include the .cpp.

4.1 Visual Studio

If you created a Visual Studio solution, do the following. Delete the hidden .vs folder. Delete all x64 and Debug folders. Maintain the directory hierarchy created by Visual Studio. Your submission should now include *.sln, *.cpp, *.h, *.vcxproj* files, your report, and nothing more. Once again, we should be able to run your solution out of the box.