BLG 335E

Homework 1

October 18, 2019

Due date is **28th October 2019, 09:00!**

Homework Policy

- Use comments whenever necessary to explain your code. Also, **write your name on top of each file you are sending in comment!**

- Do not forget to handle exceptions and print error messages!

- Your code must be written in C++, and should be able to be compiled and run using g++ compiler. ITU servers provide g++ compiler, you can test your program by connecting to the servers using ssh(If you don't know how to do it, click here). Using object oriented approach is a bonus. You can not use external libraries such as STL.

- IMPORTANT: This is an individual assignment! You are expected to act according to Student Code of Conduct, which forbids all ways of cheating and plagiarism. It is okay to discuss the homework with others, but it is strictly forbidden to use all or parts of code from other students' codes or online sources and let others do all or part of your homework. **Your codes will be checked using plagiarism detection software. In case of cheating occurs, you will be subject to disciplinary actions.**

- If you have any questions, you can contact me at kamard@itu.edu.tr. Do NOT hesitate to send an e-mail if you are confused, in fact I encourage you to ask questions!

Problem Description

In this project, you are expected to implement Bubble Sort and Merge Sort. You are also required to prepare a report including their analyses and a solution to the given question below.

Implementation Details(40 Points)

Two input files (sorted.txt and unsorted.txt) that includes integers to be sorted is provided. Read N numbers from the input file, sort them using Bubble Sort or Merge Sort and report the time that the algorithm takes to sort. You should refer to the lecture materials for Merge Sort algorithm. Additionally, pseudo code for Bubble Sort on an array of numbers **A** is given as follows:

```
i <- length[A]
sorted <- False
while i is greater than 1 AND sorted is False
    sorted <- True
    do for j <- 1 to i - 1
        do if A[j] < A[j - 1]
            then exchange A[j] <-> A[j - 1] and sorted <- False
    i <- i-1
```

Your program should be run from the command line with the following format:

```
./program algorithmType N file
```

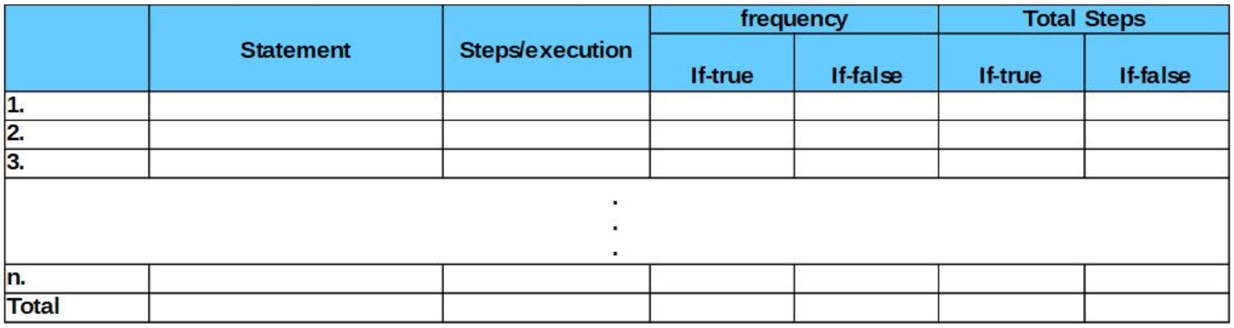
N: Total number of integers to be sorted
algorithmType: Method to be used to sort(’b’ for Bubblesort or ’m’ for Merge Sort)
file: Input file to be read ("sorted.txt" or "unsorted.txt").

An example execution command is given as follows:

```
./program m 1000 sorted.txt
```

This command executes the program using Merge sort with the first 1000 elements of the file "sorted.txt".

After the execution of your program, an output file should be created (output.txt) with the sorted integers in ascending order and your program should report the time used for the algorithm.



	Statement	Steps/execution	frequency		Total Steps	
			If-true	If-false	If-true	If-false
1.						
2.						
3.						
⋮						
n.						
Total						

Figure 1: An example table. This is similar to the ones used in the lectures, create a similar one for yourself and use it for asymptotic bound analysis questions below.

Report(60 points)

In your report, you are expected to analyze and compare the running times of algorithms you implemented with respect to their computational complexity. In addition, you are going to analyze the time complexity of the 'Mystery' function given below as well. Therefore, you need to complete the following tasks and include the results in your report.

(a) **10 points** Give the asymptotic lower bound **AND** upper bound on the running time for Bubble Sort and Merge Sort with the methods you covered in the lectures using the table above and show that your implementation of these algorithms fit these values.

(b) **10 points** Run each search methods for each different value of **N** as1000, 10000, 100000,1000000 with both of the given input files "sorted.txt" and "unsorted.txt". Calculate the average time of execution for each value of **N** for each file.
Note: You can use the **clock()** function under **ctime** library to calculate time of execution for the sort functions. Click here for more details.

(c) **20 points** After calculating execution times for both of the files, you will prepare two line plots (in Excel or Matlab) for each file, in order to visualize the runtime complexity of Bubble Sort and Merge Sort for different values of **N**. Then you are expected to interpret the results with respect to the asymptotic bounds you have given in **a**. Indicate in which cases you would choose which algorithm. Why?

(d) **20 points** Examine the following **Mystery** function:

```
Algorithm Mystery(n)
r <- 0
for i <- 1 to n do
   for j <- i+1 to n do
      for k <- 1 to j do
         r <- r+1;
return r
```

What value is returned by the algorithm? Express your answer as a function of n. Compute its time complexity with the same method you used in **a**.

Submission

Submit your homework files through Ninova. Please zip and upload all your files. You are going to submit the following files:

(a) All your .h and .cpp files

(b) a .pdf file as your report. Your report should be clear and detailed. If your solutions are unclear, it may result in a grade loss for you.

(c) a .txt file explaining how to compile and run your code (in a 1 or 2 line)