CS 202 Fundamental Structures of Computer Science II
Assignment 1 – Algorithm Efficiency and Sorting

Assigned on: 7 July 2020 (Tuesday)
Due Date: 14 July 2020 (Tuesday) by 23:55

Important Note

Please do not start the assignment before reading the notes in HAND IN section (last page):

Question 1 (30 points)

Trace the following sorting algorithms to sort the array `[4, 8, 3, 7, 6, 2, 1, 5]` into **ascending** order. Use the array implementation of the algorithms as described in the textbook.

a) Insertion sort.
b) Selection sort.
c) Bubble sort.
d) Merge sort; also list the calls to **`mergesort`** and **`merge`** in the order they occur.
e) Quick sort; also list the calls to **`quicksort`** and **`partition`** in the order they occur. Assume that the first item is chosen as pivot.

Question 2 (10 points)

Write the recurrence equation for the time requirements of **`mergesort`** and **`quicksort`** algorithms in the worst case, and solve them using ***repeated substitutions*** method. Show all the steps clearly.

Question 3: (50 points)

Programming Assignment -- You are asked to implement the **`insertion sort`**, **`merge sort`** and **`quick sort`** algorithms for *an array of integers* and then perform the measurements as detailed below.

1. For each algorithm, implement the functions that take an array of integers and the size of the array and then sort it in ***ascending order***. Add counters to count the number of key comparisons and the number of data moves during sorting.
2. For the quick sort algorithm, you are supposed to take the first element of the array as the pivot.
3. Write a main function to measure the time required by each sorting algorithm. To this end, use the library function `clock()`, which is defined in time.h. Invoke the `clock()` library function before and after each sorting algorithm to measure the elapsed time in milliseconds.
4. Although you will write your own main function to get the experimental results, we will also write our own main function to test whether or not your algorithms work correctly. In our main function, we will call your sorting algorithms with the following prototypes.

```
void insertionSort( int* arr, const int size, int& compCount, int& moveCount );
void mergeSort( int* arr, const int size, int& compCount, int& moveCount );
void quickSort( int* arr, const int size, int& compCount, int& moveCount );
```

In all of these prototypes, **`arr`** is the array that the algorithm will sort, **`size`** is the array size, **`compCount`** is the number of key comparisons in sorting, and **`moveCount`** is the number of data moves in sorting. After returning this function, **`arr`** should become sorted.

For counting key comparisons, you should count each comparison like “a < b” as 1 comparison. For counting number of moves, you should count each assignment as 1 move, for example, swap method has three moves:

```
void swap( DataType& x, DataType& y ) {
   DataType temp = x;
   x = y;
   y = temp;
}
```

5. Put the implementations of these functions in **`sorting.cpp`**, and their interfaces in **`sorting.h`**. Do not include your main function in these files. Submit your main function inside a separate file, called **`main.cpp`**.

6. ***You will lose a significant amount of points if you do not comply with these naming conventions.***

After implementing the sorting algorithms,

1. Create three identical arrays with ***random 20,000 integers*** using the random number generator function `rand`. Use one of the arrays for the insertion sort, another one for the merge sort, and the last one for the quick sort algorithm. Output the number of key comparisons, the number of data moves, and the elapsed time to sort these integers using each of these algorithms. Repeat this experiment for at least 5 different input sizes that are greater than 20,000 (for instance, 20 000, 40 000, 60 000, 80 000, 100 000). With the help of a graphical plotting tool, present your experimental results graphically. Note that you should plot the number of key comparisons, the number of data moves, and the elapsed time in different figures.
2. Then, create three identical copies of an array ***with 20,000 integers that are sorted in descending order***. Use each array for each algorithm. Repeat all of the experiments and present your experimental results graphically. (That is, for each algorithm, create arrays with at least 5 different input sizes and output the number of key comparisons, the number of data moves, and the elapsed time to sort these arrays and present your results graphically.)
3. Lastly, create three identical copies of an array ***with 20,000 integers that are sorted in ascending order***. Use each array for each algorithm. Repeat all of the experiments and present your experimental results graphically.

Question 4: Interpretation (10 points)

Interpret your experimental results that you obtained in Question 3. Compare these results with the theoretical ones for each sorting algorithm. Explain any differences between the experimental and theoretical results.

HAND-IN

- You should prepare the answers of Questions 1, 2, 3 and 4 using a word processor (in other words, do not submit images of handwritten answers).
- This homework will be graded by your TA, Süleyman Aslan (suleyman.aslan at bilkent edu tr). Thus, you may ask your homework related questions directly to him.
- Before 23:55 of July 1st, upload your solutions to Moodle. You should upload a single **zip** file that contains
  - **hw1.pdf**, the file containing the answers to questions 1, 2, the sample output of the program, and the graphical findings of the experiments for Question 3 and the answer to question 4.
  - **sorting.cpp**, **sorting.h**, and **main.cpp**, the files containing the C++ source code
  - **readme.txt**, the file containing anything important on the compilation and execution of your program in question 2.
  - Do not forget to put your name and student id in all of these files. Well comment your implementation. Add a header comment to the beginning of each file as follows:

```
/**
* Title: Algorithm Efficiency and Sorting
* Author: Firstname LastName
* ID: 21000000
* Assignment: 1
* Description: description of your code
*/
```

  - Do not put any unnecessary files such as the auxiliary files generated from your favorite IDE. Be careful to avoid using any OS dependent utilities (for example to measure the time).
  - Name your zip file as follows: **NAME_SURNAME_hw1.zip**; any violation of these will cause a significant point deduction from your grade.
  - Keep all the files before you receive your grade.

IMPORTANT:

- Although you may use any platform or any operating system to implement your algorithms and obtain your experimental results, your code should work on the **dijkstra** server (dijkstra.ug.bcc.bilkent.edu.tr). We will compile and test your programs on that server. If your C++ code does not compile or execute in that server, you will lose points.
- **Attention:** For this assignment, you are allowed to use the codes given in our textbook and/or our lecture slides. However, you ARE NOT ALLOWED to use any codes from other sources (including the codes given in other textbooks, found on the Internet, belonging to your classmates, etc.). Furthermore, you ARE NOT ALLOWED to use any data structure or algorithm related function from the C++ standard template library (STL).

Do not forget that plagiarism and cheating will be heavily punished. Please do the homework yourself.