CS 201, Fall 2020
Homework Assignment 2

Due: 23:59, Nov 30, 2020

In this homework, you will merge two sorted integer arrays. That is, given two sorted integer arrays `arr1` and `arr2` of size `N`, the problem is to create and return a third sorted array `arr3` with the items of `arr1` and `arr2`. It will have size `2N`. Assume that sorting is done in ascending order of the integer items.

Two alternative algorithms that solve this problem are given below in lay terms. Each algorithm has a different time complexity. The goal of this homework is to evaluate the growth rates of both algorithms using different inputs.

Algorithm 1: Append all items in `arr1` in the same order to `arr3`. Then, for all items in `arr2` starting from the beginning, find the right place to insert in `arr3` by shifting the items in `arr3` if needed.

Algorithm 2: Compare pairs of numbers across `arr1` and `arr2 and insert the smallest to arr3`. In this algorithm you are allowed to visit every item only once.

ASSIGNMENT:

1. Implement global functions for these algorithms and write a driver (main) function that calls these functions. Then, create sorted arrays of different sizes. You are expected to try many different input sizes, both small inputs and very large inputs. For each input size, `N`, consider the following ordering cases: (i) all numbers in `arr1` are smaller than `arr2`, (ii) all numbers in `arr2` are smaller than `arr1`, and (iii) there is no such ordering between these arrays. Run each array size and ordering combination and record the execution times. **Do not include the time elapsed to allocate the array and initialize their entries.** If you want, you can use `cstdlib` library's `rand()` function to generate random integers and `algorithm` library's `sort()` function to sort the input arrays `arr1` and `arr2`.

2. Use these results to generate a plot of running time (y-axis) versus the input size `N` (x-axis), per ordering case (i.e., for (i), (ii), and (iii) above). Specifically, you are expected to produce plots similar to Figure 2.3 of the handout chapter on algorithm analysis.

3. Based on your plots indicate the best, average and worst cases for each algorithm and the corresponding worst case time complexity.

4. Provide the specifications (processor, RAM, operating system etc.) of the computer you used to obtain these execution times. **You can use any computer with any operating system for this assignment, but make sure your code compiles and runs on the dijkstra server.**

5. Also, plot the expected worst case growth rates obtained from the theoretical analysis by using the same `N` values that you used in your simulations. That is, using your theoretical analysis in (3), for each `N` value plot the expected number of operations.

6. Finally, compare the expected growth rates obtained in step 5 and the worst case results obtained in step 3, and discuss your observations in a paragraph.

You can use the following code segment to compute the execution time of a code block. For these operations, you must include the `ctime` header file.

```
//Store the starting time
double duration;
clock_t startTime = clock();

//Code block
//...

//Compute the number of seconds that passed since the starting time
```

```
duration = 1000 * double( clock() - startTime ) / CLOCKS_PER_SEC;
cout << "Execution took " << duration << " milliseconds." << endl;
```

An alternative code segment to compute the execution time is as follows. For these operations, you must include the `chrono` header file.

```
//Declare necessary variables
std::chrono::time_point< std::chrono::system_clock > startTime;
std::chrono::duration< double, milli > elapsedTime;

//Store the starting time
startTime = std::chrono::system_clock::now();

//Code block
...

//Compute the number of seconds that passed since the starting time
elapsedTime = std::chrono::system_clock::now() - startTime;
cout << "Execution took " << elapsedTime.count() << " milliseconds." << endl;
```

NOTES:

1. This assignment is due by 23:59 on Nov 30, 2020.

2. This homework will be graded by your **TA Mert Duman (mert.duman at bilkent edu tr). Thus, you may ask your homework related questions directly to him.**

3. Put your code (single cpp file including the implemented global functions and the main function) and PDF report into a folder and create a zip file. The name of this zip file should conform the following name convention: secX---Firstname---Lastname---StudentID.zip where X is your section number.

4. Before the deadline, upload your zip file to Moodle. No hardcopy submission is needed. **Type your report, handwritten reports will not be graded. Note that you might get an FZ grade for this reason.**

5. The standard rules about late homework submissions apply. Please see the course web page for further discussion of the late homework policy as well as academic integrity.