CS xyz3-001 Foundations of Programming and Data Structures

Instructor Dr. Turgay Korkmaz

Homework 08

Due date: check BB

!!!! NO LATE HOMEWORK WILL BE ACCEPTED !!!

In this hw, you are asked to measure the execution time of two sorting algorithms, namely selection sort and merge sort, as discussed in ch07-AlgorithmicAnalysis.ppt

To help you with measuring the execution time of a function, we provide a sample program hw08.c. So, first get hw08-exec-time-sorting.zip and unzip it. Then familiarize yourself with the provided code hw08.c, which gets a seed number and an integer number (n) from command line. The program currently just initialize random number seed using srand(seed) and demonstrates how to measure the execution time of a function.

Now you are asked to extend this program as follows:

- implement selection_ sort(double arr[], int n); and merge_sort(double arr[], int n);
- in main(int argc, char *argv[])
    - dynamically create a 1D array (arrA) with n double values, and randomly set each value to a real number between (1.0, 1000.0)
    - dynamically create a copy of the same array (arrB)
    - call selection_sort(arrA, n) and measure its execution time
    - call merge_sort(arrB, n) and measure its execution time
    - print the followings when executed as

```
> hw08 seed   n
Execution time of selection sort n#: ....
Execution time of merge sort n#: ....
```

- Run your program with n=10, 100, 1000, 10000, 100000,
- For each n, repeat the run with ten different seed (trail 1, 2, ..., 10)
- So you can generate the following table for each sorting algorithm

N	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 5	Trial 6	Trial 7	Trial 8	Trial 9	Trial 10	AVG of 10 trails
10												
100												
1000												
10000												
100000												

- After generating the tables, compute the average execution times for each n.
- Now compare both algorithms and explain how/why you see significant differences in execution times.

Calculating Execution Times

There are several ways to calculate execution time. The easiest is to use the prompt 'time' before the name of your executable, i.e. ">> time ./a.out". This is useful to measure the execution of an entire program, but not for our purposes here.

A second way to measure time is by using the clock( ) function in time.h. This function returns the number of clock ticks from the start of your program to when the function was called. It uses a variable type 'clock_t'. Since the clock( ) function returns time in clock ticks, not seconds, you must divide by CLOCKS_PER_SEC to calculate the time in seconds. See general example below:

```
#include <time.h>

int main( ) {
        clock_t start, end;
        double seconds;

        start = clock( );
        //call your sorting function
        end = clock( );

        seconds = (double) ( (end – start) / CLOCKS_PER_SEC );
```

You will learn more about how the processor executes multiple programs at the same time (it really switches between them quickly) in future classes. For now, know that running the same code can take slightly different amounts of time, depending on what else your processor is working on. That is why we run several trials with several sizes of input and take the average. Your individual trials and averages will all be different, but the proportions should be similar.

What to do and return: !!!! NO LATE HOMEWORK WILL BE ACCEPTED !!!

1. Create a directory `abc123-hw08`, using your own abc123. Do all your work under that directory.

2. Follow the problem-solving methodology, and solve the problem(s). Then convert your solution(s) to a C program. You can name your program here as `hw08.c`

```
/*
 * Don't forget to include comments about the
 * problem, yourself and each major step in your
 * program! so that we can understand your
 * solution(s).
 */
```

3. Compile and run your program, and create the tables as described above and write a report about your observations (it might be better to use .docx for this report)

4. Zip the whole directory `abc123-hw08 as abc123-hw08.zip`
5. Go to BB Learn (http://learn.utsa.edu/) , login using your abc123
6. Submit your `abc123-hw08.zip` for hw08 under Assignments

You must submit your work using Blackboard Learn and respect the following rules:

1) All assignments must be submitted as either a zip or tar archive file unless it is a single pdf file.
2) Assignments must include all source code.
3) Assignments must include an output.txt file which demonstrates the final test output run by the student.
4) If your assignment does not run/compile, the output.txt file should include an explanation of what was accomplished, what the error message was that prevented the student from finishing the assignment and what the student BELIEVES to be the underlying cause of the error.
