CS590 homework 1 – C++ & running times

The due date for this assignment is Tuesday, February 18th, at 11.59pm. This assignment is worth 10% of your final grade.

Any sign of collaboration will result in a 0 and being reported to the Graduate Academic Integrity Board. The programming assignment will be done individually. No collaboration is allowed between students. No code from online resources is allowed to be used besides the code that I will share with you. Late submission policy described in the syllabus will be applied.

You are given an integer vector which is represented by *int** an array of integers and its dimension *n* as a separate parameter. We are interested in sorting arrays of integer vectors according to a pre-defined notion of vector length. You therefore are given the function *ivector_length(v, n)* that computes and returns the length of vector *v* with dimension *n* as $\sum_{i=1}^{n} |v_i|$.

You are given a naive (and very inefficient) implementation of insertion sort for arrays of integer vectors.

Questions (100 points)

1. Develop an improved implementation of insertion sort for integer vector (*insertion_sort_im*) that precomputes the length of each vector before the sorting. Keep in mind that the vectors are sorted according to their length (see *ivector_length* function). You can test the correctness of your sorting algorithm using the provided *check_sorted* function.

2. Implement a merge sort for an array of integer vectors. As for the improved insertion sort algorithm, you should precompute the length of the vectors before the sorting and the sorting is done according to the vector length. Test the correctness of your merge sort implementation using the provided *check_sorted* function.

3. Measure the runtime performance of insertion sort (naive and improved) and merge sort for random, sorted, and inverse sorted inputs of size *m* = 10000; 25000; 50000; 100000; 250000; 500000; 1000000; 2500000 and vector dimension *n* = 10; 25; 50. You can use the provided functions *create_random_ivector*, *create_sorted_ivector, create reverse_sorted_ivector.*
Repeat each test a number of times (usually at least 10 times) and compute the average running time for each combination of algorithm, input, size *m*, and vector dimension *n*. Report and comment on your results.

Remarks:

- You are not allowed to use code from online resources. Your submission will be tested against that, and will receive a 0, and a report to the Graduate Academic Integrity Board if it is detected.
- Your report has to be typed, and submitted in a pdf file. You should provide figures that plot the running times in relation to the input size parameters.
- You might have to adjust the value for *n* depending on your computers speed, but allow each test to take up to a couple of minutes.
- Start with smaller values of *n* and *m* and stop if one instance of the algorithm takes more than 5 min to complete (the insertion sort implementations will hit that limit early on).

- Report and comment means that you have to analyze and interpret your findings properly. What do the experiments tell you?
- The programming, testing and the experimentation will take some time. Start early.
- Feel free to use the provided source code for your implementation. You have to document your code.
- A Makefile is provided to build the code in the Virtual Box provided.
- Your code has to compile, and run at the Virtual box shared with you in order to be graded.
- Your methods should be added in the sort.h/sort.cpp files. You must keep the same file structure, makefile that is provided in the skeleton code.