SMU | LYLE SCHOOL OF ENGINEERING

CSE 3353 ALGORITHMS: LAB 1

Overview: In this lab you will use the Strategy Design Patter to create a single interface for implementing and analyzing the performance of 3 sorting algorithms (bubble, merge and insertion). This framework will be used as a base in future labs as we continue to grow your algorithm integrations. The project requires no GUI or terminal interface, it should execute completely without human interaction.

Due: September 16, 2019 @ 11:59pm.

Code Requirements:

- Use a Strategy Pattern with a base class named Algorithm that has the following properties
  - Load [Takes a filename in and can read input data file, can be implemented differently as long as it loads file]
  - Execute [Executes the search algorithm]
  - Display [Prints solution to screen]
  - Stats [Prints algorithm name, execution time and number of records analyzed to screen in a readable format]
  - Select [enum , int, id or string passed as input and loads corresponding algorithm to interface]
  - Save [Saves solution to file path given as input]
  - Configure [Future expandability]
- Create the following sorting algorithms
  - Bubble, merge and insertion
- All timing stats should be collected using the C++ std::chrono::high_resolution_clock class
- Create the Class Sort which inherits from Algorithm and overrides the base functionality of abstract class
- Sort should use the concrete implementations of merge, bubble and insertion through composition to complete the Strategy Pattern implementation
- All classes should be created using both .h and .cpp files (when possible).
- Create the following data sets of int values with the following size and characteristic and save to file(s)
  - Data Set Sizes: 10; 1,000; 10,000; 100,000
  - Characteristics:
    - Random: values are place in complete random order
    - Reversed Sorted Order: values are sorted in reverse order
    - 20 % unique values Data should duplicate numbers, only 20% of values are unique, remaining 80% are duplicates of those and in random order
    - Approximately 30% randomized: This is a semi-sorted list, approximately 30% of file is randomized and out of order

    - You will have 16 total data sets, 1 of characteristic for each data set size
- In main.cpp, loop through all algorithms and data sets to collect the performance timing. Your main.cpp should look similar to (when and how your load data can be different). Overall execution loop can be different, but it should be similar in practice to one loop that executes across different algorithms. Main.cpp should have minimal code, and only need to include your interface class to access the algorithms.

```
Algorithm Sort;

for (all Algo Sort Types)
    for(all Data Types)
        Sort.Load(input file type); //Load can be called in previous loop
        Sort.Execute();
        Sort.Stats();
```

- You should have files similar to these in your final submission (more are possible and you can adjust naming convention to fit your project/style). Each instantiated non templated class should have a .h/.cpp when relevant. Final code should utilize the Strategy pattern.
    - Algorithm.h, Sort.h/cpp, Bubble.h/cpp, Insertion.h/cpp, Merge.h/cpp, main.cpp
- You may modify the make file as needed to insure you code passes the Git Action, if it does not compile it receives a 0.

Report Requirements:

1. Collect all execution stats for each data set size and place into a summary table similar:

		Algorithms		
		Bubble	Insertion	Merge
Data Types	Random			
	Reversed			
	Unique			
	Partial Sort			

2. Create plot for timing performance for each data type (Random, Reversed, Unique, Semi-Sorted) where the x axis is the data set size (10, 1K, 10K, 100K), and y axis is the execution time for each algorithm. Plot all 3 algorithms on the same graph. There should be a total of 4 graphs, 1 for each data set type (Random, Reversed, Unique, Partial Sort).
3. Submit all tables and graphs in nicely formatted report in word. Also provide the raw excel files used to create the charts and tables (or whatever program used to generate graphs with raw data).
4. Explain results and compare and contrast the algorithms with various input data types. Be sure to give insight on why there are similarities, differences between the different algorithms timing and the corresponding data sets, or any code/data structures that might have caused differences.

All code should be in the "Code" folder, raw data files generated should be in "Data" and your final report should be in "Report" folder