# ITCS_6114_8114: Algorithms and Data Structures

Fall 2015

# Programming Project 2: Quick Sort

**Due Date**

The project is due on September 29nd, 2015 at 11 PM. And late submissions are not allowed. No submission via email will be considered. This is individual project and MUST be done individually.

**Project Description**

Implement the quick sort algorithm. To be specific, call your program "Qsort". Your program takes "test.txt as an input file containing the numbers to sort and outputs the result to "answer.txt" file.

Your program should compile and run from the command line. For example, if you implement your program in Java, it should run using the following command assuming running in the default folder - the location of your source files:

*javac Qsort.java*

*java Qsort test.txt*

The input file "test.txt" should be plain text file and contains sequence of numbers separated by semicolon. Note that the name of the input file can be different so DO NOT hardcode its name in your program. It has to be read from the command line as input argument to your program (this means you will not prompt for a user to enter this input file name)! When grading, different input file name will be used.

The output of your program should be a plain text file called "answer.txt" recording the right sorted sequence of all the numbers in the input file (the answer.txt file should be saved in the same folder as your default folder).

For example, for input file:

*test1.txt: 12; -12; 10; 5; 100; -9; 1; 10*

the answer.txt will look like:

-12; -9; 1; 5; 10; 10; 12; 100

**Algorithm Performance Analysis**

Test your program with sample arrays of different sizes, such as arrays with $10^2$, $10^4$, $10^6$ numbers, and collect the performance. You can compute the performance by collecting the system time right before and after you run the quick sort algorithm. Google "how to get system time" for this purpose. For example, Java programmers can use the following command:

System.currentTimeMillis();

Your results should include two parts:

1. The sorted result of input file numbers

2. A table like format listing the sizes of the arrays and the algorithm performances for each one.

For example, if we consider the example input file given in "project description" section, the final answer.txt should look like this:

*-12; -9; 1; 5; 10; 10; 12; 100*

*Performance analysis:*
*Size Sorting Time(in milliseconds)*
*8 2*

**Programming Languages and IDE**

The program MUST be written in Java or C++.

We highly recommend that you use jGrasp environment for your coding. If you are using Mac or Unix system computer, you must test your code on windows environment and provide a detail step by step instruction of how it should be run. If you are using C++ language, use Cygwin or MinGW to compile and run your program. You can also use a plain text editor to write your program.

We will use *jGrasp* to grade all the submissions. As this programming project is easy, the main purpose is to get you familiar with the requirements on programming, documentation, and submission. Please stick to the project description.

**Grading Criteria**

The total of 100 points for this project is broken up into:

- 20 points for proper construction of data structures required in the program.
- 20 points for correctly handing the input and output files.
- 40 points for efficiently implementing the quick sort algorithm.
- 20 points for compilation, structure, and documentation (a readme file and comment in the code).

Within these criteria, your grade will be based on program structure, efficiency, and correct execution. The structure of your program will be judged for quality of the comments, quality of the data structure design, and especially the logic of the implementation. The comments need not be extremely long: just explain clearly the purpose of each class and each function within each class.

**Submission Guidelines**

Your submission should include all your source code files and a brief report as a README file. Do NOT include any IDE-specific project files, any compiled files, or any executable files. Every file should have your name in a comment line at the top. Your README file should have a brief description of your program design, the breakdown of the algorithm, the compiler you used, the platform you used, a summary of what you think works and fails in your program, and a short description of your data structure design.

You will submit your project on Moodle. You should submit a single zip file containing your source code files and README file. You can submit your project multiple times; only the most recent project submission will be graded. No late submission is allowed for this project and no submission through email will be considered. The submission link will automatically cut off at the end of the due date and time.

**Final Warning**

A project that does not follow the submission guidelines will receive a 10 points deduction. Proper submission is entirely your responsibility. Contact the TA if you have any doubts whatsoever about your submission. Do NOT submit your project via email. Please observe the academic integrity guidelines in the syllabus, and submit your own work.

Some useful links to setup your development environment:

- For Cygwin: https://cygwin.com/install.html
- For MinGW: https://users.cs.jmu.edu/bernstdh/web/common/help/cpp_mingw-setup.php
- For jGrasp: http://www.jgrasp.org/