Basic Analysis of Algorithms

CS 221 Homework Assignment

*"Though this be madness, yet there is method in't."*

– William Shakespeare, *Hamlet*, Act II, Scene 2

# Objectives

* Analyze growth function and order of Java methods

# Tasks

For this assignment, you will analyze the runtime growth rate of three methods defined in the [MethodsToAnalyze.java](https://raw.githubusercontent.com/BoiseState/CS221-Public/master/assignments/AlgorithmAnalysis/files/MethodsToAnalyze.java) file under various input scenarios. *Do not analyze any test code.* You will also compare your predicted growth functions against output data from test classes given for each method, running on a pre-compiled copy of the MethodsToAnalyze class with statement counting code.

* Start by reading through the sample analysis, below, to see what is expected.
* Answer all questions in file [AnalysisHW.docx](https://github.com/BoiseState/CS221-Public/raw/master/assignments/AlgorithmAnalysis/files/AnalysisHW.docx). These questions walk you through the analysis of each method. Answers are expected to be in well-written, correct English. These are not one-word or one-sentence answers. Again - See the sample analysis!
* Make sure to support your analysis with specific references to the code.
* You are *strongly* encouraged to walk through the algorithms in the debugger to get a clear understanding of how they work.

# Sample Analysis: DoSomething - A Poor Attempt at Ordering an Array of ints

* Given the following class files:
  + [DoSomething.java](https://raw.githubusercontent.com/BoiseState/CS221-Public/master/assignments/AlgorithmAnalysis/files/DoSomething.java) - contains the method being analyzed
  + [DoSomethingTester.java](https://raw.githubusercontent.com/BoiseState/CS221-Public/master/assignments/AlgorithmAnalysis/files/DoSomethingTester.java) - driver class to collect data
  + [ArrayOfInts.java](https://raw.githubusercontent.com/BoiseState/CS221-Public/master/assignments/AlgorithmAnalysis/files/ArrayOfInts.java) - utility class for getting various arrays for testing
* You will see that the *DoSomething.java* file has been modified so that the number of statements executed in a call to *doSomething()* can be collected for different sized arrays in the *DoSomethingTest.java* driver class.
* The [AnalysisHWSample.docx](https://github.com/BoiseState/CS221-Public/raw/master/assignments/AlgorithmAnalysis/files/AnalysisHWSample.docx) file contains analysis of the doSomething method.

# Files

[This ZIP file](https://github.com/BoiseState/CS221-Public/raw/master/assignments/AlgorithmAnalysis/files/AnalysisOfAlgorithms.zip) contains all handout files for this project including source files, tester files, and ODT documents.

* AnalysisHW.docx - analysis document
* MethodsToAnalyze.java - source file being analyzed
* MethodsToAnalyze\_Data.class - pre-compiled class used by testers
* FindTester.java - source file
* ReplaceAllTester.java - source file
* SortItTester.java - source file
* AnalysisHWSample.docx - example analysis document
* DoSomething.java - example source file
* DoSomethingTester.java - example source file
* ArrayOfInts.java - utility source file

The recommended way to run the testers is to copy all source files (and the .class file) to a directory on onyx and compile and run testers from the command line. If you absolutely must try to run them through your IDE, be warned that the IDE’s project management and build paths are likely to fight you. If you can’t get it to work in the IDE, don’t waste time fighting it. Go to onyx.

Specify "Use project folder as root for sources and class files" when creating an Eclipse project and **import** all .java source files and the MethodsToAnalyze\_Data.class file into the project. Do NOT create the project with separate src/ and bin/ directories or the testers will not pick up the .class file automatically. Note that the .class file will most likely not appear in the project but should still be accessed by the testers when they are compiled and run.

To run tester classes with command line arguments from within Eclipse, you will need to modify the run configurations for each tester class. In the right-click menu for each tester class, choose Run As -> Run Configurations. In the window that opens, make sure you are editing the correct Run Configuration for the correct driver class and project and then choose the Arguments tab. You can directly type in the command line arguments you want in the Program Arguments box, but then you would have to go back and update those arguments every time you want something different. If, instead, you click the Variables button and choose string\_prompt, Eclipse will provide a pop-up dialog box every time you run the tester and you can enter your command line arguments there.

# Grading

Points will be awarded according to the following breakdown:

| **Tasks** | **Points** |
| --- | --- |
| Analysis of *find()* | 5 |
| Analysis of *replaceAll()* | 5 |
| Analysis of *sortIt()* | 5 |
| Overall quality and completeness. | 5 |

# Required Files

Submit copies of the following files (names must match):

* AnalysisHW.docx
* MethodsToAnalyze.java (including any modifications you made to collect data)
* *Any other custom test code used in your analysis*

# Submission

Submit all files from the same directory. Do not include any unnecessary files.

Use the submission command given on your section's class web page from the directory containing your files.