|  |
| --- |
|  |
| CS 1632 – DELIVERABLE 4 |
| PROPERTY-BASED TESTING |

|  |
| --- |
|  |

Github URL: https://github.com/swanc12/deliv4.git

Colin Swan – CRS110@pitt.edu

March 24th, 2016

# Why and How

For this project I chose to create property based tests for the java “Arrays.sort(int[])” method. The primary motivation for this decision was admittedly shallow; I was unsure that I would be able to acquire the ACTS software from NIST quickly enough to be able to finish the project on time, and even if I were to have the software I did not have a project ready to go that I could easily create tests for. I also felt that property based testing would be more immediately applicable to my testing practices; it seemed more like something I would actually implement in my current tests, whereas combinatorial testing seemed more advanced, and like something that I may not be able to make immediate use of.

Implementation of the tests was relatively straight forward. I created three tests, one for each property, and in each test I created an array of random size between 0 (inclusive) and 10,000 (exclusive). I then populated the array with random integers between 0 (inclusive) and Integer.MAX\_VALUE (exclusive). I then proceeded to use this array to test a property, and repeated this one hundred times in each test. In other words, each test contains a loop in which an array is randomly generated and tested, so that one hundred arrays of random length with random elements are tested for each property.

The only minor snag that came up while developing these tests was with the length of the arrays. I had hoped to create an array of random size between 0 and Integer.MAX\_VALUE, but this resulted in a heap overflow, so I had to make due with a smaller range of possible array sizes. I do not feel this is a major detriment to the tests, as none of them assert anything special in regards to the maximum size of an array supported by the Arrays.sort(int[]) method.

The main thing I gleaned from this exercise was how property based testing could fit into a testing framework. There were a few times when I was confused as to how these tests were adequately testing the properties of the arrays, as they did not seemed to account for edge cases. For example, the case where every element is the same seemed like a case that would not be consistently covered by the property tests, which seemed troublesome to me at first. As I thought about it though I realized that testing such edge cases was more in the realm of strict unit tests, and that realization gave me a better feel about how these property tests would be made alongside unit tests, and the role they serve for quality assurance.

# Tests

