CS 1632 – DELIVERABLE 4:

Property-Based Testing

Brandon S. Hang

https://github.com/brandonhang/Pitt-Projects/tree/master/cs1632/Deliverable%204

For this deliverable, I chose to do the property-based testing option. I chose this project as it allowed me to practice using JUnit more and, for me at least, I believe a programmer should always work on their skills when given the opportunity. As such, I wrote a number of JUnit test cases that test the properties listed in the lecture slides regarding a sorted array.

To test the Arrays.sort() function built in to Java, I made an effort to test every property listed in the lecture slides concerning a sorted array. Additionally, I used the @BeforeClass annotation to create a function that sets up the arrays to be tested. This function generates a number of randomly sized arrays and fills them with integers ranging from a pre-defined size. I defined these values as *NUMTESTS*, the number of tests to run; *MAXSIZE*, the maximum size of any randomly-sized array; and *INTRANGE*, the range of the integers that fill the array (from *–INTRANGE* to *+INTRANGE* – 1). All three of these values can be defined by the user should they choose to run a different number of tests or test a different size-range of arrays.

Testing some properties were simpler than others; testing that sorted and unsorted arrays were the same size simply asserted that the two arrays were of the same length. Likewise, testing for idempotence and purity simply asserted that two sorted arrays (in accordance with the rules of idempotence or purity) were equal. Even testing that a sorted array's values are always increasing or staying the same was manageable; these tests simply have a potentially lengthy runtime of $\Theta(nm)$ where n is the number of tests and m is the size of the array.

However, testing that a sorted and unsorted array have the same contents was a bit trickier. To test this, I decided to convert each unsorted array and its associated unsorted array into linked lists. I then popped a value from one list (similar to a stack) and removed it from the other list using Java's built in LinkedList functions, asserting that the removal was true.

I found that testing the properties of something to be somewhat more challenging than previous JUnit tests. I believe this is because it requires knowledge about how something, a sorted array in this case, must be rather than directly asserting values. I also found that testing certain properties intrinsically tested another related property. For example, testing that a sorted integer is always greater than or equal to its previous neighbor also tests that sorted values are never decreasing.

Repo: https://github.com/brandonhang/Pitt-Projects/tree/master/cs1632/Deliverable%204

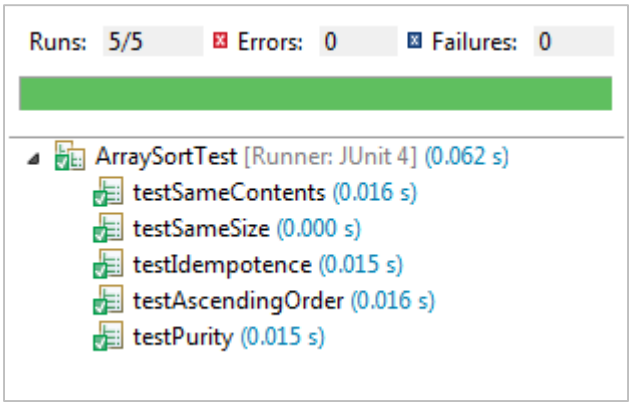

Figure 1. JUnit test run of the java.util.Arrays.sort() test cases.