CS1632 – DELIVERABLE 4

Property-Based Testing

Created by Philip Ni

CS 1632 Software Testing

University of Pittsburgh

https://github.com/phil-nye/CS1632/tree/master/Deliverable4

# Summary

For Deliverable 4, I decided to perform the JUnit-based property-based testing on Java’s Arrays.sort() method. The primary factor in selecting this option rather than the combinatorial test was that I was having difficulties accessing the NIST webpage and was not able to get the NIST ACTS software prepared in time. In addition, I felt that testing the Arrays.sort() method would allow me to gain some helpful insight into the mechanisms and behaviors of the Arrays class. Rather than trying to spend time learning a new tool – like ACTS – I was able focus on testing.

I tested four Arrays.sort() properties: 1) testing sorted and unsorted lengths, 2) testing sorting consistency, 3) testing for the same content in the sorted and unsorted arrays, and 5) testing for idempotency. Each test was conducted over one hundred (primitive) integer arrays of random length and random content. The content was limited to integers between 1 and 100,000. For the first property, I expect an unsorted arrays to have the same length as the sorted arrays. There has be no addition or removal of entries; only the ordering has changed. Second, sorting two identical, yet independent, unsorted arrays should yield the same results after sorting. In, other words, the arrays should follow the same sorting procedure every time. In the third test, I checked to see if all of the content in the arrays were the same. In order to compare the arrays, I counted the number of occurrences of each unique value in the unsorted array, and then compared those counts to that of the sorted array. If the sorting method is functioning properly, then I would expect that the entries should be identical in both the sorted and unsorted array, without any unique values missing or added to either array, and with the same number of occurrences of unique values. In the penultimate test, I checked to make sure that the sorting method ordered every value by ascending order. So, I expect every value to be greater than its preceding entry. The final test checks for idempotency. That is, it checks to ensure that sorting a sorted array yields the same result as the sorted array. If I sort an array, and then sort it again, the order should not have changed between the final array and the sorted array.

I encountered very little complications during the process of testing. The greatest difficulty was determining what limits I should set for the random number generators. I found that I would get errors whenever the length of my arrays approached close to the Java MAX\_INT value. So, I set my upper limits to more reasonable values. The length of each array was limited to between 1 and 100,000 and each entry in the array was set to the same limits. The most difficult part of this process was determining how to test each property in a nontrivial way. I wanted to be sure that my tests left no doubt that the property being tested was functioning properly. Debugging was a pain when problems were encountered because of the high volume of values that were tested.

