CS1632-Deliverable 4-Property based testing

https://github.com/nda7/CS1632/tree/master/Project4

Nathan Anuskiewicz

Summary

For this project I was required to write a mapping function as well as use property based tests to confirm that the function worked correctly. The function itself was simple. It accepts an array of integers and returns a new array of 1 greater length. This new array is populated by traversing the input array and squaring each value. The final value of the new array is filled with the sum of the now squared values.

When deciding how to test this function, I came up with three basic tests. These tests were able to cover all properties of the returned array. In order to reduce the length of the tests, I created a helper method within the Junit class to generate random length (1 to 100 inclusive) with random integer values (1-100 inclusive).

The first test checks that the length of the laboonified array is always one integer greater than the input array. This makes sure that the function is correctly creating the new array’s structure and allocating one extra position to house the sum.

Next I created a test to go through each index of the input array and the laboonified array. It checks that values in the laboonified array are the squared values of those in the input array at the corresponding index. This test makes sure the function is mapping values properly. It also confirms that the output array is being derived from values passed into the function. Note that this test does not check the final index (the sum).

Finally the last test checks the final index in the returned array. This is used to confirm that the final index in the array is being populated correctly. It will locate any off by 1 errors in the method code that would perhaps miss the first or last value in the array when computing the sum.

When running my tests I decided to run it for 10000 arrays instead of 100. I mainly did this because each test was completing in 0.000 seconds on my machine, and I wanted to make sure it was actually running through each test. Other than this I didn’t run into any complications during this assignment.

I decided not to test any features of the input arrays as these are not properties of the function itself. For example, if the input included a negative number, the test would not fail, but the input is invalid. These input specifications are listed in the requirements of the assignment and are therefore expected to be abided by when using the laboonify function. The function was not created to handle negative numbers, 0’s, null arrays, arrays with length greater than 100, or arrays with values greater than 100. When using the function with any of these inputs there is no guarantee that it will perform correctly.

