CS1632: Lecture 19

Property-Based Testing

Bill Laboon/Dustin Iser

So far...

First we learned how to build and execute manual tests.

Then we learned how to instruct a computer to execute our manual tests for us.

We still have to design the tests for the computer to execute. What if we didn't have to do this?

A sorting algorithm

Possible test cases

- Null
- []
- [1]
- [-1]
- [1, 2, 3, 4, 5]
- [5, 4, 3, 2, 1]
- [-9, 7, 2, 0, -14]
- [1, 1, 1, 1, 1, 1]
- [1, 2, 3, 4, ..., 99999, 100000, 100001]

Property-based testing

Instead of writing lots of boilerplate tests, let's describe the output as properties.

- Output array same size as passed-in array.
- Values in output array always increasing or staying the same.
- Every element in input array is in output array.
- No element not in input array is in output array.
- [2, 1, 3].sort() == [2, 1, 3].sort().sort()
- Given the same input, always returns the same output.

The input

Create random input.

Random input is generated by “monkeys”.

- Dumb monkeys
- Smart monkeys
- Evil monkeys
- Chaos monkeys

What happens when a test fails?

Since we are using random input, it is important to record the exact input that caused the failure.

We can go one step further, shrinking.

Shrinking

```
[9, 0, -6, -5, 14] -> [0, -6, -5, 9, 14] FAIL
[9, 0, -6] -> [0, -6, 9] FAIL
[-6, -5, 14] -> [-6, -5, 14] OK
[9, 0] -> [0, 9] OK
[0, -6] -> [0, -6] FAIL
[0] -> [0] OK
[-6] -> [-6] OK

Shrunk Failure: [0, -6] -> [0, -6]
```

Shrinking

- Finds the smallest possible failure.
- Helps track down actual issues.
- A “toy” failure is a great thing to add to a defect report.