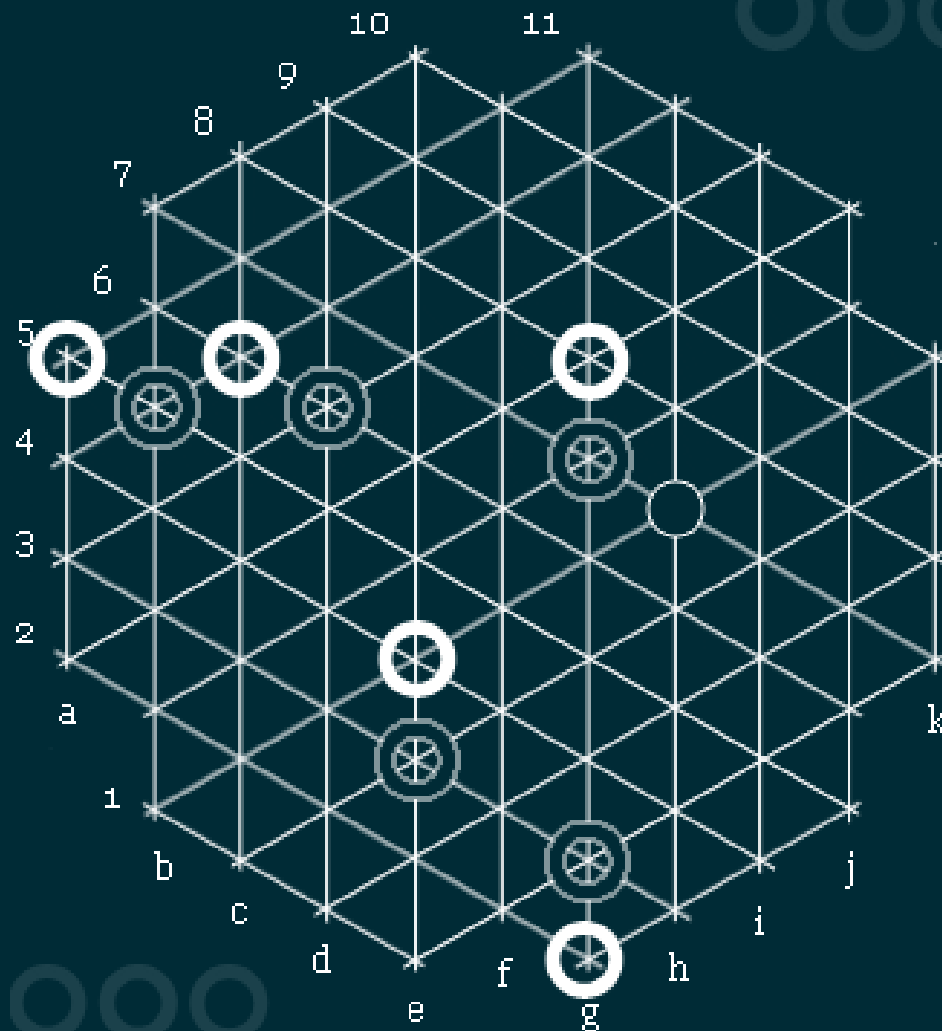


# Randomness in Testing

Paul Grayson  
2013-09-25

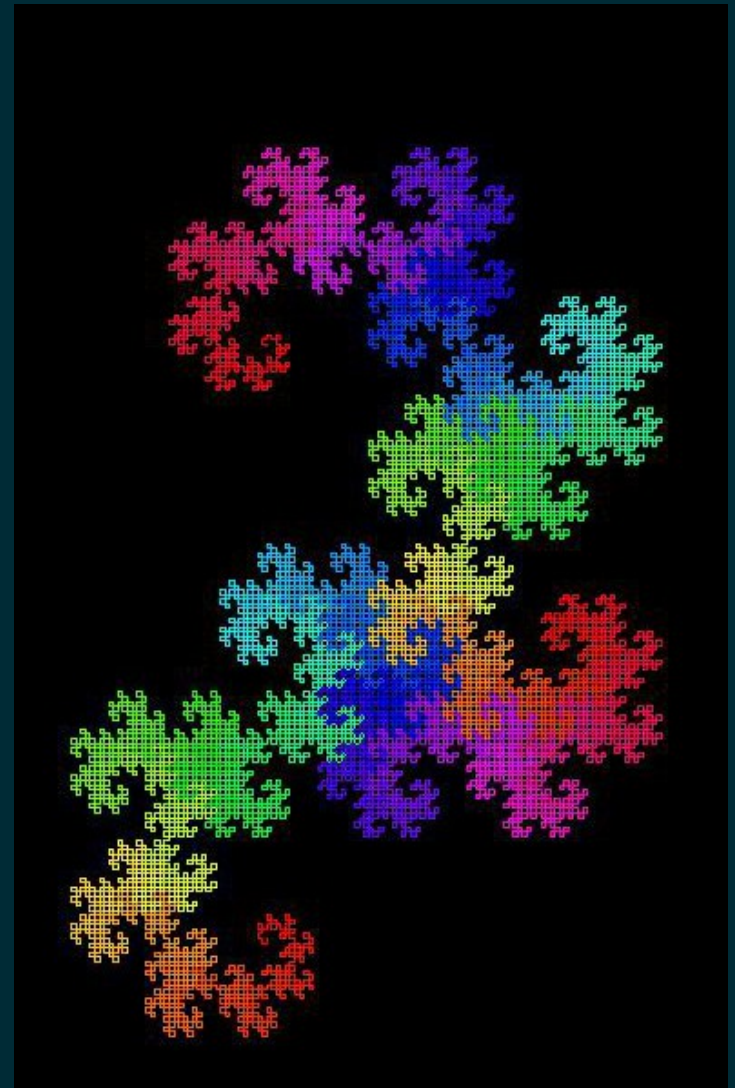
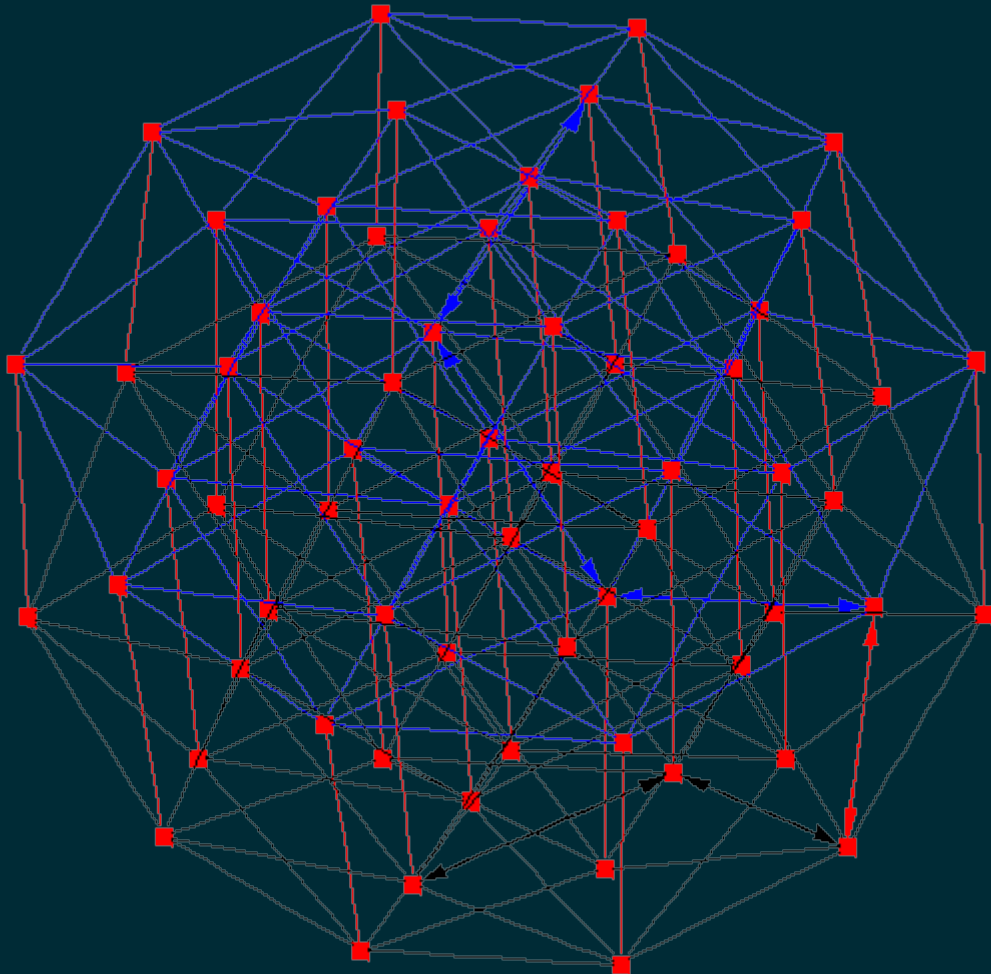
# Normal testing: try edge cases

- Easy when you have a geometric picture:



# “Fuzz testing” for difficult edges

- What are the “edge cases” here?



# Fuzz testing kata example

Goal: find a program that can find bugs in other programs

Goal: find a program that can find bugs in other programs

Goal: find a program that can find bugs in other programs

Goal: find a program that can find bugs in other programs

Goal: find a program that can find bugs in other programs

Goal: find a program that can find bugs in other programs

Goal: find a program that can find bugs in other programs

Goal: find a program that can find bugs in other programs

# Note: fuzz tests generally more simple-minded

Things to Fuzz test for:

- exceptions or other bad behavior
- outputs consistent with inputs
- exact answer for a simplified domain (hard!)

# Fuzz Testing: NOT part of RSpec

- Just use `rand` in your tests.

# Test order dependence

- Ideal: tests are independent
  - DO NOT use objects created out of the context
  - DO NOT break when other objects exist

# Test order kata example

1. Write a test that fails

2. Write the simplest code that passes the test

3. Refactor the code to make it more readable

4. Repeat steps 1-3 until the code is complete

5. Repeat steps 1-3 until the code is complete

6. Repeat steps 1-3 until the code is complete

7. Repeat steps 1-3 until the code is complete

8. Repeat steps 1-3 until the code is complete



# When a test fails...

- Write a test that always fails, or fix the tests
- Re-use the seed to debug

```
rspec --seed 1234
```

# Using RSpec seed with rand

```
 srand RSpec.configuration.seed + 1
```

# Putting it together

- Automatically re-use the seed

# Other random examples

- Code mutation
- Unavoidable randomness (network, threads)
- Testing random code