Using Relational Problems to Teach Property-Based Testing

1. Motivate

Relational problems **have inputs that admit more than one valid output.**

These problems are already common in computing education. Property based testing can handle their uncertainty!

Examples

Sorting By Key
An unstable sort-by-key function admits multiple valid outputs for inputs where two elements have the same key.

Minimum Spanning Tree
A graph may admit multiple minimum spanning trees.

Graph Shortest Path
There may be more than one shortest path between two vertices.

2. Teach

No QuickCheck Required !

Scaffold PBT

Given some specified function:

```
specified-fun   :: Input → Output
```

...implement these two functions:

```
is-valid        :: (Input, Output) → Bool
```
Satisfied if the specification admits the given `Output` as a valid result for `Input`.

```
generate-input  :: Number → Input
```
Produce a random `Input` of a given size for the system-under-test.

PBT-*Focused* Assignments

We teach this scaffold with assignments *focused* on property based testing:

- Students are given a spec.
- Students do *not* implement the spec.
- Instead, students implement `is-valid` and `generate-input`.

3. Evaluate

We apply property-based thinking to evaluate where students struggle.

For instance, an `is-valid` for a function that sorts people by their age must enforce:

Same Size
`Input` and `Output` have the same size.

Same Elements
`Input` and `Output` have the same elements.

Ordered
`Output` is in ascending order.

We discover students' difficulties with PBT by constructing test cases where *all but one* of these properties are satisfied; e.g:

```
is-valid([Prsn("Ash",1), Prsn("Cal",9)],
         [Prsn("Cal",9), Prsn("Ash",1)])
  is false
```

This test case detects *only* when ordering is not correctly enforced by `is-valid`, and doesn't care if *Same Size* or *Same Elements* isn't enforced.

Focused tests like this, for each property, give us fine-grained insight into which properties students failed (or even forgot) to enforce.