Programming Projects

1. Primarily Instructional
2. Instructor Solution Exists
3. Automated Feedback
Feedback Goals

1. Help students arrive at a correct answer
2. Students can help themselves
3. Every missed bug is a missed learning opportunity
<table>
<thead>
<tr>
<th>Targeted</th>
<th>Comprehensive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isolates One Issue</td>
<td>Tests every case</td>
</tr>
<tr>
<td>Guide Student Attention</td>
<td>Hard To Engineer</td>
</tr>
<tr>
<td>Many Targeted Tests</td>
<td>Hard To Compute</td>
</tr>
</tbody>
</table>
Fuzz Testing
Fuzz Testing

Testing the behavior of the program on many random inputs

Complementary to Manual Testing

Historically used for security
Fuzz Testing (Security)

Generate Random Inputs

User Program

Verify Security Constraints
Fuzz Testing (Programming Projects)

Generate Random Inputs

Student Program

Verify Correctness
Generate Random Inputs

Student Program

Student Output

Instructor Program

Expected Output
How to compare output?
How many inputs are required?
How to improve upon Fuzz Tests?
Creating & Distributing Tests

- Input domain is known
- Generating Random Inputs is easy
Generate Inputs

- Random input generation
- All students were provided with an identical set of tests
- Large number of inputs needed
Creating & Distributing Tests

- Input domain is known
- Generating Random Inputs is easy

Generate Inputs

Student Program

Instructor Program

Verify Output
Creating & Distributing Tests

Generate Inputs

- Raw Output Check
- Hashing
- Program Tracing
RQ1: How to Compare Outputs?

Verify Output

- Raw comparison of output
- Compare against precomputed result
RQ1: How to Compare Outputs?

Hashing Output

- Hash combination of outputs
- Compare against precomputed result

Test #100

Student output

Output Vector

Combine & Hash

477587826
In Person CS1 Course with 1400 Students Enrolled
<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Students Completing Project</td>
<td>1,331</td>
</tr>
<tr>
<td>Code Snapshots</td>
<td>486,482</td>
</tr>
<tr>
<td>Average Snapshots per Student</td>
<td>349</td>
</tr>
<tr>
<td>Incorrect Attempts at Target Question</td>
<td>48,079</td>
</tr>
</tbody>
</table>
Output Vectors

Generated Inputs

Correct Student Attempt
Student Attempt
Student Attempt
Student Attempt

Output Vector

Output Vector

Output Vector
Output Vectors

Output Vector

- ✅
- ❌
- ❌
- ❌

[Diagram showing four rows of colored circles, with the first row having a green check mark and the remaining rows having red cross marks.]
RQ2: How many inputs are needed?
R2: How many inputs are needed?

1 Input

- [✔] 1st input
- [✔] 2nd input
- [✔] 3rd input
- [✗] 4th input
RQ2: How many inputs are needed?

2 Inputs

- ✔
- ✔
- ✔
- ✗
- ✗
RQ2: How many inputs are needed?

3 Inputs

✔

✔

❌

❌
RQ2: How many inputs are needed?

4 Inputs

✔

❌

❌

❌
RQ2: How many inputs are needed?
656 Students (48.4%) passed all of the targeted tests but still had an error caught by the Fuzz Tests.
RQ3: Improving Fuzz Tests

The graph shows the cumulative percent fixed for different types of tests over attempts required to fix errors. The x-axis represents the attempts required to fix an error, and the y-axis represents the cumulative percent fixed.

- **Targeted Test** (blue line)
- **Fuzz Test** (green line)

The graph indicates that Targeted Test generally requires fewer attempts to fix errors, showing a higher cumulative percent fixed at each attempt compared to the Fuzz Test.
How to Improve On Fuzz Tests?

As a result of the Fuzz Test:

46% of students reported spending 1 to 4 hours debugging

19% of students reported spending more than 4 hours debugging

Obfuscating output made it harder for instructors to help students
Program Inspection

Incorrect result after playing 1 game(s):

<table>
<thead>
<tr>
<th>Turn</th>
<th>score0</th>
<th>score1</th>
<th>Turn Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turn 0:</td>
<td>0</td>
<td>0</td>
<td>Player 0 rolls 0 dice: +1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Turn 1:</td>
<td>1</td>
<td>0</td>
<td>Player 1 rolls 7 six-sided dice: +37 3, 4, 6, 3, 3, 4, 6</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>37</td>
<td>Dice sum: 29</td>
</tr>
</tbody>
</table>

...  

Incorrect implementation of game at turn 1.  
Please read over the trace to find your error.  
(error_id: 1189294328)
Thank you

sumukh@berkeley.edu
@sumukhsridhara

okpy.org

cs61a.org